Optical Switching and Networking 8 (2011) 12–22

Contents lists available at ScienceDirect

Optical Switching and Networking

journal homepage: www.elsevier.com/locate/osn

Buffer scaling for optical packet switching networks with shared 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

ARTICLE INFO

Article history:
Received 28 September 2009
Received in revised form 27 June 2010
Accepted 9 July 2010
Available online 17 July 2010

Keywords:
Small buffer
OPS
TCP
Pacing
Shared buffer
XCP

ABSTRACT

According to a historical rule of thumb, which is widely used in routers, the buffer size of each output link of a router should be set to the product of the bandwidth and the average round-trip time. However, it is very difficult to satisfy this buffer requirement for ultra-high-speed dense wavelength division multiplexing (DWDM) networks with the current technology. Recently, many researchers have challenged the rule of thumb and have proposed various buffer sizing strategies requiring less buffer. Most of them were proposed for electronic routers with input and output buffering. However, shared buffering is a strong candidate for future DWDM optical packet switching (OPS) networks because of its high efficiency. As all links use the same buffer space, the wavelength count and nodal degree have a big impact on the size requirements of shared buffering. In this paper, we present a new buffer scaling rule showing the relationship between the number of wavelengths, nodal degree, and the required shared buffer size. By an extensive simulation study, we show that the buffer requirement increases with $O(N^{0.85}W^{0.85})$ for both standard TCP and paced TCP, while XCP-paced TCP's buffer requirement increases with $O(N^{1}W^{0.85})$ for a wide range of N and W, where N is the nodal degree and W is the number of wavelengths.

© 2010 Elsevier B.V. All rights reserved.

1. Introduction

Recent advances in optical networks such as dense wavelength division multiplexing (DWDM) networks have allowed us to achieve ultra-high data-transmission rates in optical networks. However, the optical and electronic buffer technology has not been able to keep pace with the optical data-transmission rates, which have become a bottleneck in routers. There are three main buffer types proposed for optical packet switching networks in the literature: electronic RAM, optical RAM, and fiber delay line (FDL)-based buffering. Electronic RAM allows $O(1)$ reading operation when the output port is free. However, electronic buffering requires the conversion of optical packets to the electronic domain. As the operating speed of the electronic components and electromagnetic interference limit the packet bit rate to about 10 Gbps [1], an electronic approach to direct opto–electro–opto ($O/E/O$) conversion is not a feasible solution. Recently, Takahashi et al. [1] showed that an optoelectronic approach using parallelized all-optical convertors could achieve fast $O/E/O$ conversion. However, the size and the speed of electronic RAM are still bottlenecks as router manufacturers must use large, slow, off-chip DRAMs to satisfy large buffer requirements [2].

Currently, the only available solution to all-optical buffering is switching the contended packets to long fiber delay lines (FDLs). However, FDLs have severe limitations such as signal attenuation and high space requirements in routers, due to the very long fiber lines. It is very difficult to achieve RAM-like $O(1)$ reading operation of variable-length packets in FDL buffers, so FDLs may limit the achievable utilization of links and increase the rate of packet drops. On the other hand, all-optical RAM is still being researched [3]. Optical RAM has several advantages over FDLs and electronic RAM. It solves the problems with FDLs such as lack of

* Corresponding author. Tel.: +81 6 6879 4542; fax: +81 6 6879 4544.
E-mail addresses: a-onur@ist.osaka-u.ac.jp (O. Alparslan), arakawa@ist.osaka-u.ac.jp (S. Arakawa), murata@ist.osaka-u.ac.jp (M. Murata).

1573-4277/$ – see front matter © 2010 Elsevier B.V. All rights reserved.
doi:10.1016/j.osn.2010.07.002

Table 1
Buffer scaling rules.

Author	Traffic	Utilization (%)	Buffer size per link
Rule of thumb [5]	TCP	100	$B \times T$
Appenzeller et al. [2]	TCP	near 100	$(B \times T)/\sqrt{F}$
Avrachenkov et al. [6]	TCP	near 100	$(B \times T)^2/(32 \times F^3)$
Gorinsky et al. [7]	TCP	over 75	$2 \times N \times M$
Enachescu et al. [8]	Paced TCP	80–90	$O(\log S)$

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. [4] demonstrated a photonic crystal-based all-optical bit memory operating at very low power. However, each photonic cell can buffer only a single bit, so all wavelengths must share the same buffer space, like in electronic RAM buffering. Moreover, optical RAM is not expected to have a large capacity soon. Therefore, even a small decrease in the buffer requirements may have a high impact on the realization of RAM-buffered high-speed WDM OPS networks.

According to a historical rule of thumb [5], which is widely used in routers, the buffer size of each output link of a router should be set to the product of the bandwidth (BW) and the average round-trip time (RTT). DWDM is capable of ultra-high data rates in excess of 1 Pbit (petabit) per fiber, and many network operators and router manufacturers currently follow a guideline of 250 ms of buffer size per link, which would require 250 Tbits of ultra-high-speed buffer per fiber. Recently, many researchers have challenged the rule of thumb and proposed new rules requiring less buffer. Some of them are listed in Table 1, where B is the bandwidth of the link, T is the round-trip time (RTT), F is the number of TCP flows on the link, N is the nodal degree, M is the maximum segment size (MSS), and S is the TCP congestion window size. All of these rules and guidelines were proposed for output RAM buffered routers. A detailed comparison of these proposals is available in [9].

The well-known bursty behavior of TCP [10] is the main problem limiting the decrease in buffer requirements, because the bursty behavior of TCP results in a 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 criterion that decreases the short-term burstiness and hence smooths the network traffic. It is well known that applying pacing at TCP senders (paced TCP) dramatically decreases the buffer requirements [11]. However, this method requires changing the TCP sender or receivers. Another possible method for pacing is shaping the traffic at the edge or core nodes. We recently proposed using an explicit congestion control protocol (XCP)-based architecture [12] for pacing at the edge nodes without changing the TCP [13]. We found that our architecture could achieve high utilization and a low packet drop ratio with TCP flows in very small RAM-buffered OPS networks, even better than those with TCP pacing [14].

As shared buffering allows more efficient use of a RAM buffer than output buffering, it decreases the total buffer requirement in the router [15]. There is ongoing research on realization of a WDM OPS router with such a RAM-based buffer, which is fully shared by all links and wavelengths [16]. However, the required shared buffer capacity is still unclear. Most of the papers on the performance of shared buffers in the literature present only the packet drop rate of different shared buffer architectures. They do not propose a guideline for sizing shared buffers for TCP traffic. Moreover, most of the papers on OPS networks with shared buffers use only FDL-based buffering with limitations like fixed packet or slot size. Unlike output buffering, which is a single-output queue, shared buffering is a multiple-output queue, so the nodal degree (N) has a big impact on the buffer requirements. Moreover, all the wavelengths use the same buffer space in optoelectronic and optical RAM, so the wavelength number (W) must be taken into account, too. Additional N and W dimensions make the analytical analysis of RAM-based shared buffering optical routers much more complex than that of the RAM-based output buffering of electronic routers. In this paper, we present a new buffer scaling rule showing the relationship between the number of wavelengths, nodal degree, and the required shared buffer size. We estimate the buffer scaling parameters for TCP, paced TCP, and XCP-based edge node pacing by an extensive simulation study.

The rest of the paper is organized as follows. Section 2 describes the related work on TCP and XCP pacing. Section 3 describes the XCP pacing and switch architecture. Section 4 describes the simulation methodology and presents the simulation results. Finally, we conclude the paper in Section 5.

2. Related work

This section describes TCP and XCP pacing architectures.

2.1. TCP pacing

TCP pacing is defined as transmitting ACK (data) packets according to a special criterion, instead of immediately transmitting when data (ACK) packets arrive [17]. TCP pacing was initially proposed as a solution to ACK compression [17]. Kulik et al. [18] proposed using paced TCP to solve the problems with queuing bottlenecks by smoothing the bursty behavior of TCP traffic. It is well known that bursty traffic produces high packet losses, and low throughput, especially in small buffered networks [11]. Applying pacing at TCP senders (paced TCP) by evenly spacing the TCP data packets sent into the network over an entire round-trip time was shown to decrease the burstiness and thus the buffer requirements, dramatically [11]. Enachescu et al. [8] recently proposed that $O(\log S)$ buffers

are sufficient, where S 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 $O(\log S)$ buffer size depends on the maximum congestion window size of TCP flows, which may change. Moreover, using slow access links 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, which are not standard TCP, can be difficult. Furthermore, while paced TCP improves performance in small-buffered networks, it is shown to have significantly worse performance than standard TCP in general [11], so paced TCP may not be good choice for the heterogeneous Internet environment. It may be better to apply pacing only where the pacing is necessary.

2.2. Edge node pacing

The main authors of the Stanford tiny-buffer model recently commented that when their initial conditions (using slow access links or modifying the TCP) [8] 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 [19]. This can be a more practical solution than changing the TCP or limiting the link capacity. Applying pacing to aggregate traffic at the edge nodes can decrease the traffic burstiness and hence reduce the buffer requirements. Moreover, it can also smooth the UDP traffic and protect a small buffered network from the burstiness of any misbehaving flows. We recently proposed using an explicit congestion control protocol (XCP)-based architecture for pacing at the edge nodes [13]. The edge nodes calculate the pacing rate by exchanging probe packets with other edge nodes. We found that our architecture could achieve high utilization and a low packet drop rate with TCP flows in very small optical RAM-buffered OPS networks, which were even better than those with TCP pacing [14].

3. Architecture

This section describes the XCP pacing and switch architecture in detail.

3.1. XCP basics

XCP [12] is a congestion control algorithm specifically designed for high-bandwidth and large-delay networks. XCP core routers periodically update their link control parameters calculated by an Efficiency Controller (EC) and Fairness Controller (FC) according to link utilization, spare bandwidth and buffer occupancy. The XCP sender agent at the ingress edge node sends its traffic rate information to the XCP receiver at the egress edge node in the header of data packets or a probe packet. On the way to the XCP receiver, XCP core routers read this information and calculate a feedback that shows the desired traffic rate change for this XCP flow, and update the feedback header of the packet, if necessary. The XCP receiver sends back the final feedback to the XCP sender, which updates its sending rate accordingly.

3.1.1. Efficiency Controller (EC)

The Efficiency Controller is responsible for maximizing the link utilization by controlling the input aggregate traffic. Every router calculates the amount of desired increase or decrease on the aggregate traffic of each output port by the formula $\Phi = \alpha \cdot S - \beta \cdot Q/d$, where Φ is the total amount of desired change in input traffic, α and β are the spare bandwidth control and queue control parameters respectively, 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, and Q is the persistent queue size.

3.1.2. Fairness Controller (FC)

After calculating the aggregate feedback Φ, the Fairness Controller is responsible for fairly distributing the Φ to all flows according to an AIMD-based control. When Φ is positive, the transmission rate of all flows is increased by the same amount. When Φ is negative, the transmission rate of each flow is decreased proportionally to the flow's current transmission rate. Bandwidth shuffling, which redistributes a small amount of traffic among flows, is used for achieving fairness faster when Φ is small. Shuffled traffic is calculated by $h = \max(0, \gamma \cdot u - |\Phi|)$, where γ is the shuffling parameter and u is the aggregate input traffic rate in the last control interval.

When a router receives a packet containing feedback, if its own calculated feedback is smaller than the one in the header, it updates the feedback in the header with its own feedback. Otherwise, it does not change the feedback available in the header. When an XCP source agent receives an XCP feedback, it updates its congestion window size according to the formula cwnd $=$ max(cwnd $+$ $H_$feedback, s), where s is the packet size and $H_$feedback is the feedback in the ACK packet.

3.2. Rate-based paced XCP

In [13], we showed that XCP-based edge node pacing can be used as an intra-domain traffic shaping (pacing) and congestion control protocol in a small buffered OPS network domain. In this architecture, when there is traffic between two edge nodes, first a macro flow (like an LSP in GMPLS) is established between the edge nodes, as shown in Fig. 1. The XCP sender agent on the ingress edge node multiplexes the incoming TCP and UDP flows and applies leaky bucket pacing to the macro flow according to the rate calculated by the XCP. The receiver XCP agent on the egress edge node demultiplexes the macro flow and forwards the TCP and UDP packets to their destinations.

Each macro flow sends its feedback in a separate probe packet once in every control period like in Simplified XCP-based Core Stateless Fair Queuing [12] and TeXCP [20], instead of writing feedback to packet headers, so there is no need for calculating a per-packet feedback. The probe packets are carried on a separate single control channel wavelength with low transmission rate, which allows applying an electronic conversion for updating the XCP feedback in packet headers and buffering the probe packets in a slow electronic RAM in case of a contention.

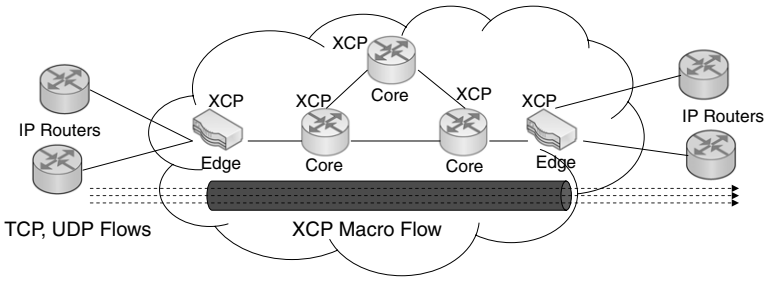

Fig. 1. XCP-based edge node pacing architecture.

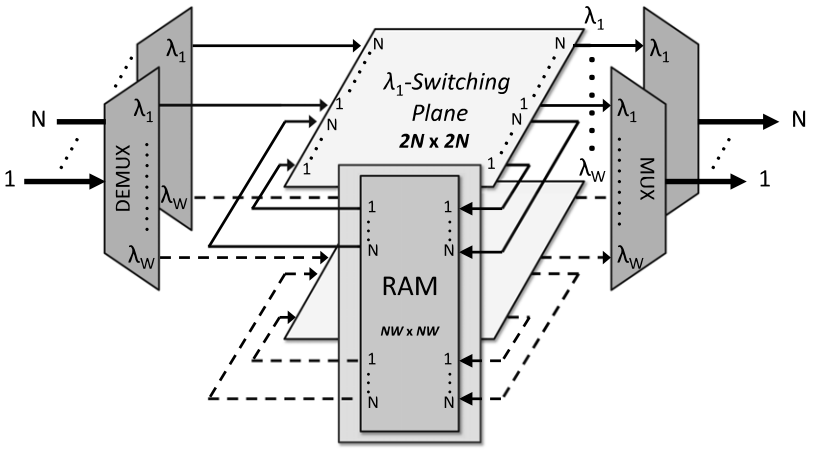

Fig. 2. Switch architecture.

3.3. Switch architecture

We simulated a shared buffered and wavelength partitioned OXC (optical cross-connect) switch architecture, as seen in Fig. 2. There are W wavelengths and the nodal degree is N. There is a single RAM-based shared buffer with a size of $NW \times NW$ that is connected to all switching planes. The shared buffer can be either electronic or optical as long as it operates as a RAM. There are W wavelength switching planes connected to the input/output links and the shared buffer. Therefore, each wavelength switching plane has a size $2N \times 2N$. The simulated switch architecture is almost the same as the switch architecture currently being researched in NTT's Photonic Packet Router Project [16]. One difference is that, in NTT's architecture, each switching plane is connected to the shared buffer by a single port, which makes the switch a blocking switch, while decreasing the number of switching layer and shared buffer ports. When multiple packets collide simultaneously on a wavelength, only a single packet can be forwarded to the buffer. Deflection routing was proposed for blocked contending packets. Moreover, the buffer can forward only a single packet at a time to the output links at each wavelength, which decreases the overall performance. For the sake of simplicity, we used a strictly non-blocking switch that does not require deflection routing, so each switching plane is connected to the shared buffer by N ports.

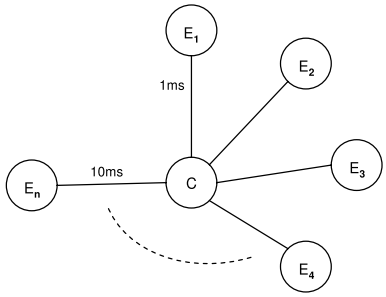

Fig. 3. Star topology.

4. Evaluation

This section discusses our evaluation of the buffer scaling rule of a shared buffered switch, showing the relationship between the number of wavelengths, nodal degree, and the required shared buffer size.

4.1. Simulation settings

The shared buffered WDM switch architecture and algorithms were implemented over *ns* version 2.32 [21]. A star topology shown in Fig. 3 was used in the simulations. There was a single core node for switching the packets. The propagation delay of the links were uniformly distributed between 1 ms to 10 ms. Non-paced standard TCP Reno and paced TCP Reno were used as the traffic sources. The TCP

flows started randomly and sent traffic between randomly selected edge node pairs according to a uniform traffic matrix. The average number of TCP flows per wavelength on each single-way link was 150. The total simulation time was 40 s. The IP datagram for TCP data (ACK) packets was 1500 Bytes (40 Bytes), so a TCP data (ACK) segment was 1480 Bytes (20 Bytes). The maximum congestion window size (S) of the TCP was set to 20 packets. All the data wavelengths had a 1 Gbps capacity. There were no wavelength convertors, so the wavelength continuity was preserved. XCP's α, β and γ parameters were chosen as 0.2, 0.056, and 0.05, respectively, as explained and used in [13]. TCP data and ACK packets are carried in separate XCP macroflows in order to prevent ACK compression. We simulated using $N = 4, 8, 16, 32, 64$ and $W = 1, 2, 4, 8, 16, 32$ to test a wide range of switch sizes.

4.2. Methodology

Our aim was to estimate the effect of nodal degree (N) and wavelength count (W) on the buffer size requirements to achieve the same performance in terms of average flow goodput and wavelength utilization. Therefore, we tried to keep the other simulation settings constant as much as possible. We changed the average number of TCP flows per edge node pair depending on the N value according to the formula $150/(N-1)$ in order to keep the average number of TCP flows per wavelength on a link constant in all simulations. We assumed that there was no self-destined input traffic (namely, a packet coming from input port k is switched to return back to the same link via output port k). As there were N edge nodes each sending traffic to all other $N-1$ edge nodes, there were around $150N$ TCP flows on each wavelength layer in the network. When a new wavelength is added to the network, the same traffic matrix as the first wavelength was applied by creating new TCP flows for the new wavelength layer in order to keep the average number of flows per wavelength on each link the same. Therefore a core router with nodal degree N and wavelength count W carried a total of around $150NW$ TCP flows.

As the average number of flows per wavelength on a link is the same in all simulations, we can expect to achieve the same per flow goodput under different wavelength counts and nodal degrees as long as the packet drop rate and buffering delay in the router is controlled by changing the capacity of shared buffer. Buffering delay may be negligible when the buffer is small, but it can affect the goodput as the buffer gets larger. By applying a curve fitting based on the simulation results, we estimated an approximate buffer scaling rule, which shows the effect of wavelength count and nodal degree, with the formula

$$B = C(N-c)^a W^b, \tag{1}$$

where B is the shared buffer size, C is the buffer scaling variable, N is the nodal degree, c is the nodal degree normalization constant, a is the order of the nodal degree constant, W is the wavelength count, and b is the order of the wavelength count constant. The variable C may be further decomposed into other parameters such as maximum congestion window size, number of flows per wavelength, bandwidth, MSS, etc. As there would be no traffic passing through the switch, N being equal to 1 is meaningless. Furthermore, when N is equal to 2, the buffer is unnecessary, because there would be no contention or overutilization in the core switch as the traffic would just pass through the router without any multiplexing. As the minimum meaningful value of N is 3, the constant c is necessary for normalizing N in the formula.

4.3. Standard TCP results

First, we simulated the star topology with standard TCP traffic without any pacing and calculated the average flow goodput on the first wavelength. Plotting the goodput in all simulations in a single figure does not give much insight, so we divided the results into subfigures based on the N value, as seen in Fig. 4. The x-axis plots the core link buffer size on a log scale and the y-axis plots the average flow goodput in terms of Bytes/s. It is clear that the buffer requirements increase as we increase N and W. Some simulation results with $W = 16$ are missing in Fig. 4(f), because a very long simulation time (over 7 weeks) is required for each point in that region. However, the results in the figure are more than enough for estimating and checking the buffer scaling parameters. First, we estimated the order of dependency on W. The dependency on W can be written as $B = C_T^W W^{b_T}$, where C_T^W is the buffer scaling variable for W and b_T is the order of wavelength count, all for standard TCP. After a simple curve fitting by trial and error, we obtained $b_T = 0.85$. As seen in Fig. 5, which plots C_T^W versus the average flow goodput, all lines overlap in each subplot. We get almost the same goodput at the same C_T^W and N value, independent of W. This clearly shows that the shared buffer requirement of standard TCP increases with $O(W^{0.85})$.

As a next step, we estimated the dependency on N, which can be done by expanding C_T^W into $C_T^W = C_T(N - c_T)^{a_T}$, where C_T is the general buffer scaling variable for W and N, c_T is the normalizing constant, and a_T is the order of nodal degree for standard TCP. Fig. 5 shows that, at the same goodput level, C_T^W is a positive function of N. After some curve fitting by trial and error (see Fig. 5), we estimated $a_T = 0.85$ and $c_T = 1.3$. Putting the estimated parameters into Eq. (1) gives

$$B = C_T(N - 1.3)^{0.85} W^{0.85}, \tag{2}$$

which means that the buffer requirement of standard TCP increases with $O(N^{0.85}W^{0.85})$. Fig. 6, which plots C_T versus the average flow goodput, shows that, with the approximation in Eq. (3), the average flow goodput behaves like a positive function of C_T independent of N and W, especially when the link is underutilized. There is some deviation when $N = 3$ and $C_T < 10\,000$. However, the deviation completely disappears when $N \geq 4$, so it may be better to use minimum $N = 4$ for scaling. There is a goodput error margin of around 10% at $C_T > 100\,000$, where the flow goodput and wavelength utilization are almost maximized. The reason is that as the number of flows and flow arrival times on the first wavelength are random variables that change with N, so there is some deviation at the average maximum flow goodput at full utilization for different N values.

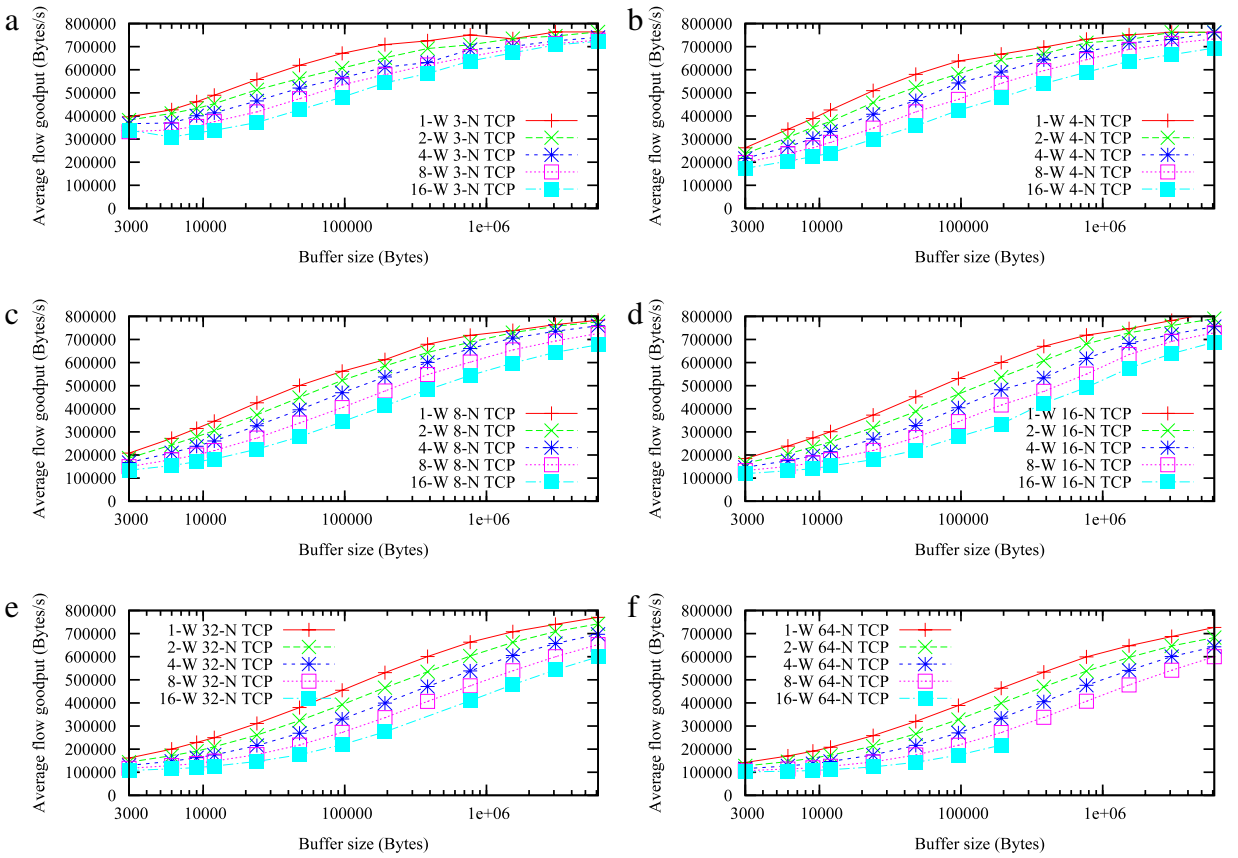

Fig. 4. The average flow goodput of standard TCP for N between 3 and 64.

4.4. Paced TCP results

We repeated the same procedure as that in Section 4.3 in order to estimate the buffer scaling parameters in $B = C_P(N-c_P)^{a_P}W^{b_P}$, where C_P, c_P, a_P, and b_P are the paced TCP counterparts of the parameters in Section 4.3. Fig. 7 plots the core link buffer size versus the average flow goodput of paced TCP. First, we estimated the order of dependency on W in the formula $B = C_P^W W^{b_P}$, where C_P^W is the buffer scaling variable for W and b_P is the order of wavelength count for paced TCP. After a simple curve fitting by trial and error, we obtained $b_P = 0.85$, where we got almost the same goodput at the same C_P^W and N value, independent of W, as in Fig. 5. As a next step, we estimated the dependency on N, which can be done by expanding C_P^W into $C_P^W = C_P(N-c_P)^{a_P}$, where C_P is the general buffer scaling variable, c_P is the normalizing constant, and a_P is the order of nodal degree for paced TCP. After some curve fitting by trial and error, we obtained $a_P = 0.85$ and $c_P = 1.3$. Putting the estimated parameters into Eq. (1) gives

$$B = C_P(N - 1.3)^{0.85}W^{0.85}, \tag{3}$$

which means that the buffer requirement increases with $O(N^{0.85}W^{0.85})$, which is the same as for standard TCP in Section 4.3. Fig. 8, which plots C_P versus the average flow goodput, shows that, with the approximation in Eq. (3), there is some deviation when $N = 3$ and $C_P < 1000$ as in Section 4.3. The deviation completely disappears when $N \geq 4$. Also, there is a relatively high goodput variation at $C_P > 10\,000$. The reason is that paced TCP is plagued by the synchronized packet drops in large buffers [11], which decreases the achievable goodput as the synchronization level increases.

4.5. XCP-paced TCP results

As a last step, we repeated the same procedure as that in Section 4.3 in order to estimate the XCP-paced TCP buffer scaling parameters C_X^W, C_X, c_X, a_X, and b_X. Fig. 9 plots the core link buffer size versus the average flow goodput of XCP-paced TCP. First, we estimated the order of dependency on W by curve fitting; we obtained $b_X = 0.85$. After some curve fitting by trial and error, we obtained $a_X = 0.85$ and $c_X = 2.2$. Putting the estimated parameters into Eq. (1) gives

$$B = C_X(N - 2.2)^1W^{0.85}, \tag{4}$$

which means that the buffer requirement increases with $O(N^1W^{0.85})$. Fig. 10, which plots C_X versus the average flow goodput, shows that the approximation in Eq. (4) fits almost perfectly, especially when link is underutilized. As the average maximum flow goodput changes with N, there is some very small deviation at $C_X > 3000$.

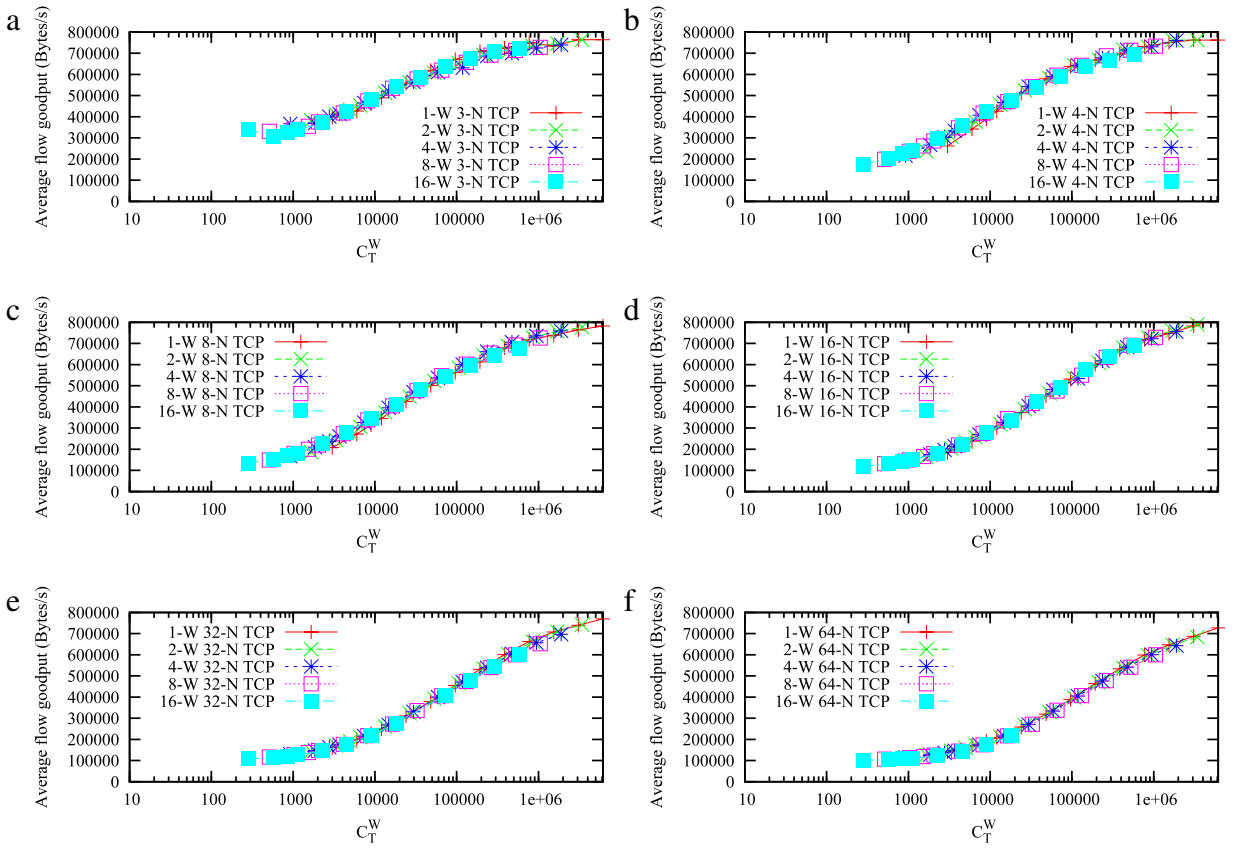

Fig. 5. The average flow goodput of standard TCP versus C_T^W for N between 3 and 64.

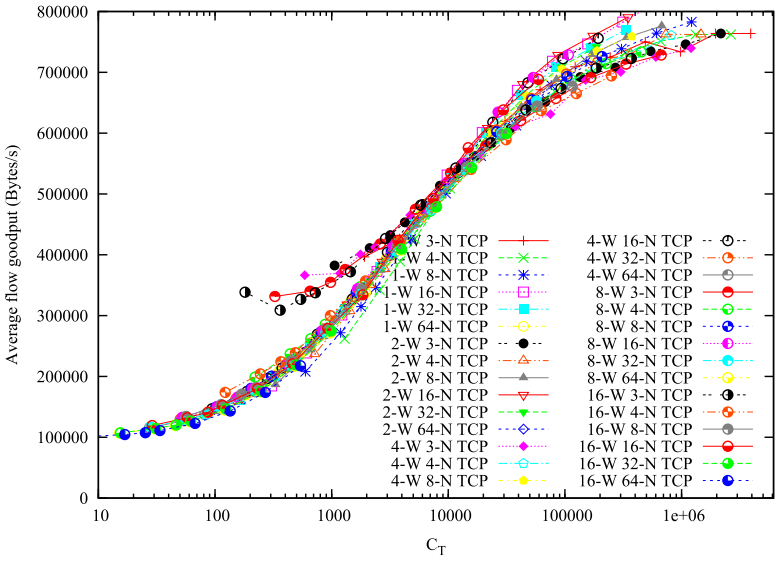

Fig. 6. The average flow goodput of standard TCP versus C_T.

4.6. Comparison of XCP-paced TCP and paced TCP goodput

Fig. 11 plots the goodput improvement ratio of XCP-paced standard TCP over paced TCP in terms of percentage versus shared buffer size. It shows that XCP-paced TCP always has better or the same goodput as paced TCP. Up to 50% goodput improvement is achieved when N is small, because C_X is smaller than C_P and c_X is larger than c_P. As we increase N, the goodput improvement diminishes as expected, because XCP-paced TCP has a buffer requirement

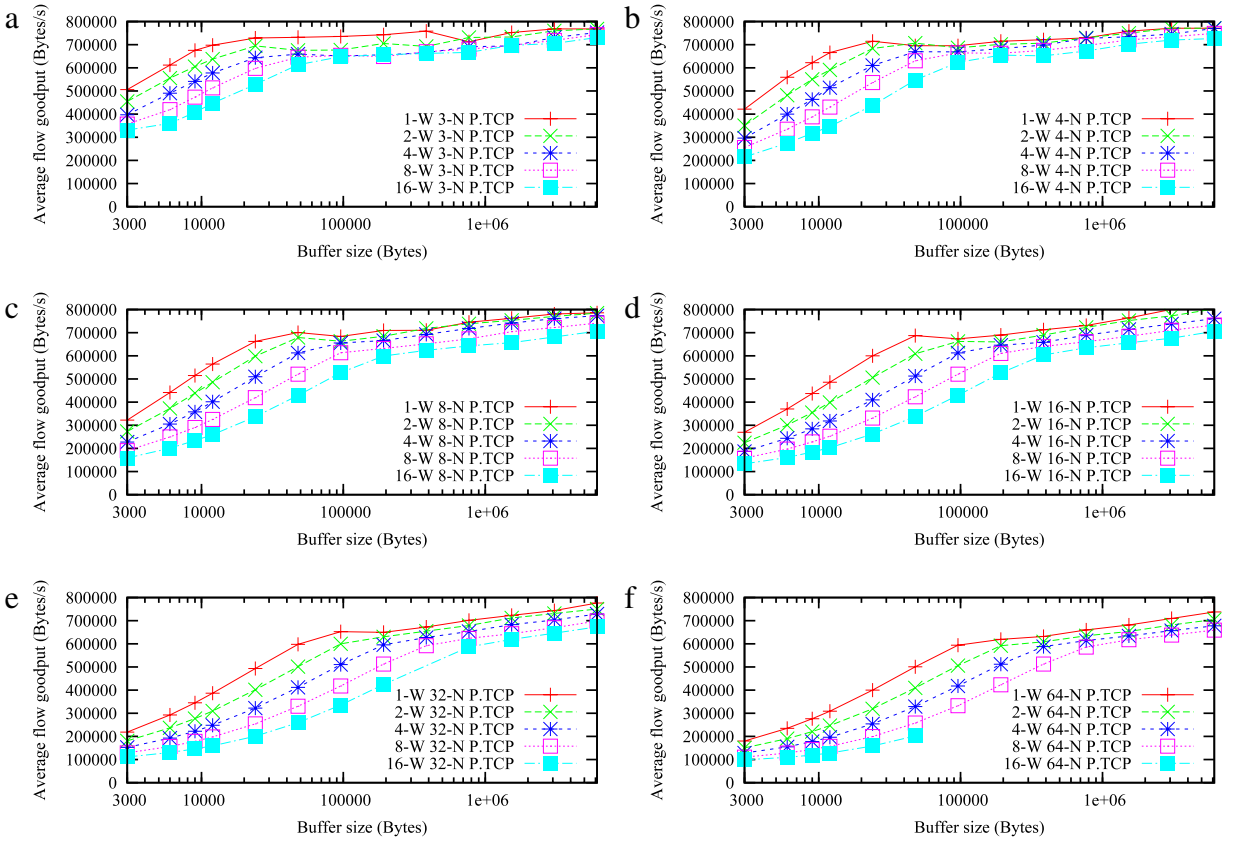

Fig. 7. The average flow goodput of paced TCP for N between 3 and 64.

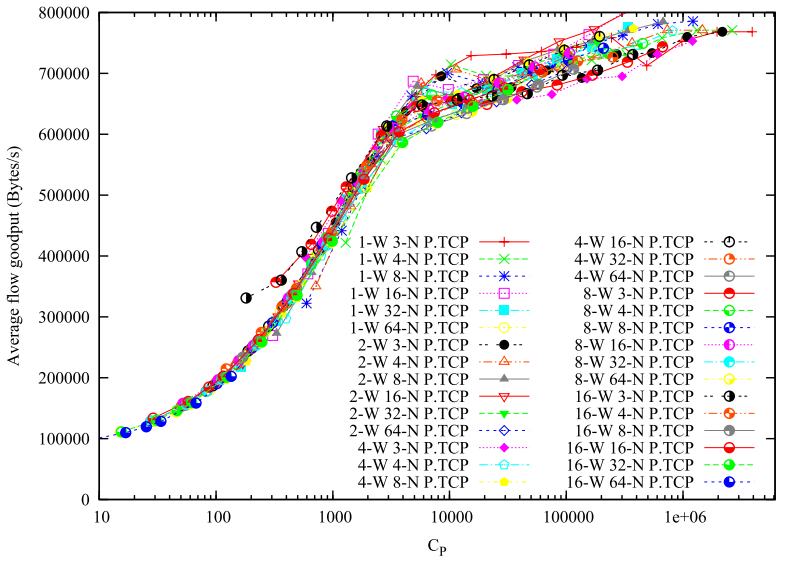

Fig. 8. The average flow goodput of paced TCP versus C_P.

of $O(N^1)$ while paced TCP has a buffer requirement of $O(N^{0.85})$. One can also think that the buffer requirement for XCP-paced TCP may become higher than that for paced TCP, when N is very high. However, Fig. 11 shows that XCP-paced TCP and paced TCP converge to the same goodput as N is increased. The reason is that XCP-paced TCP paces macro flows instead of pacing individual TCP flows. Enachescu et al. [8] proved that multiplexing many paced TCP (not standard TCP) flows causes Poisson-like traffic. While Poisson traffic is less bursty than standard TCP

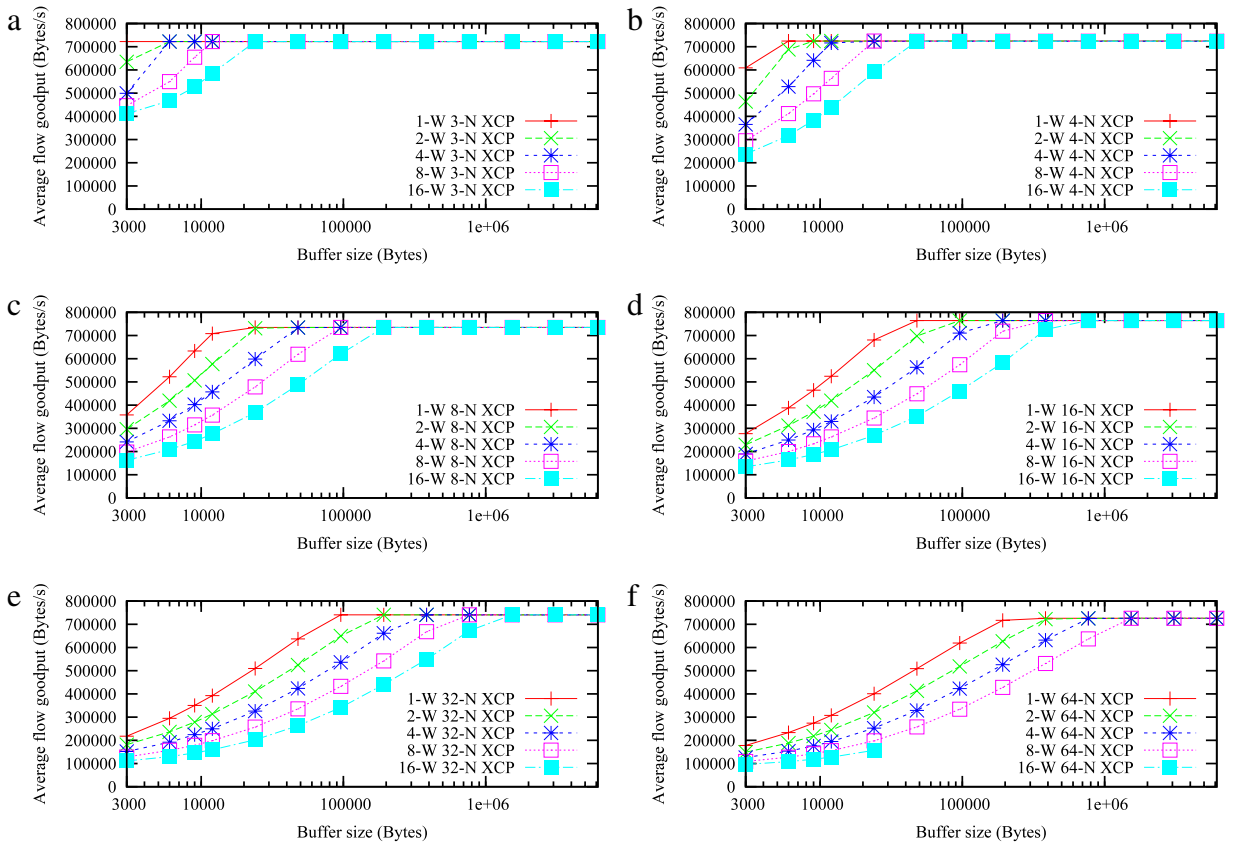

Fig. 9. The average flow goodput of XCP-paced standard TCP for N between 3 and 64.

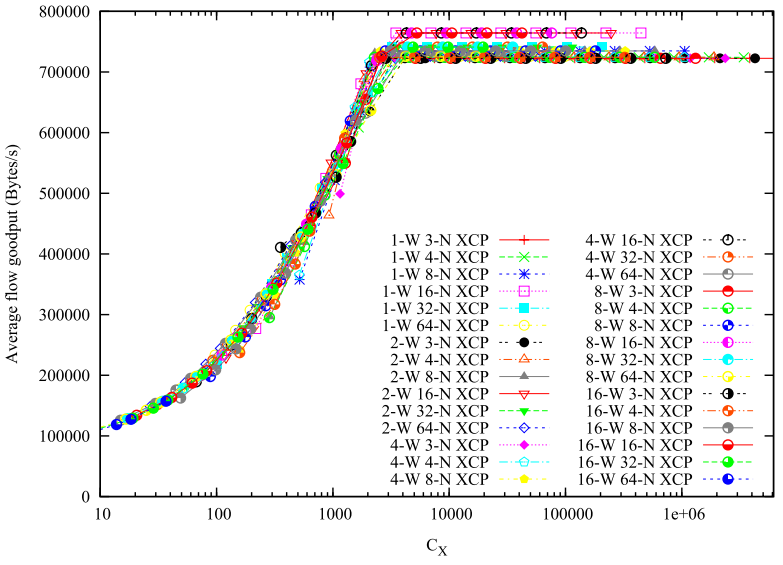

Fig. 10. The average flow goodput of XCP-paced standard TCP versus C_X.

traffic, it is considerably burstier than a single paced TCP flow. As the number of multiplexing paced flows increases, the aggregate traffic becomes more and more Poisson-like and thus burstier. As XCP-paced TCP paces macro flows, which aggregates TCP flows, it allows multiplexing a much smaller number of paced flows at a time. Therefore XCP-paced TCP traffic is less bursty than paced TCP when there are many TCP flows per macro flow and the number of macro flows is low, which was the case when N was small in the simulations. For example, the mean number

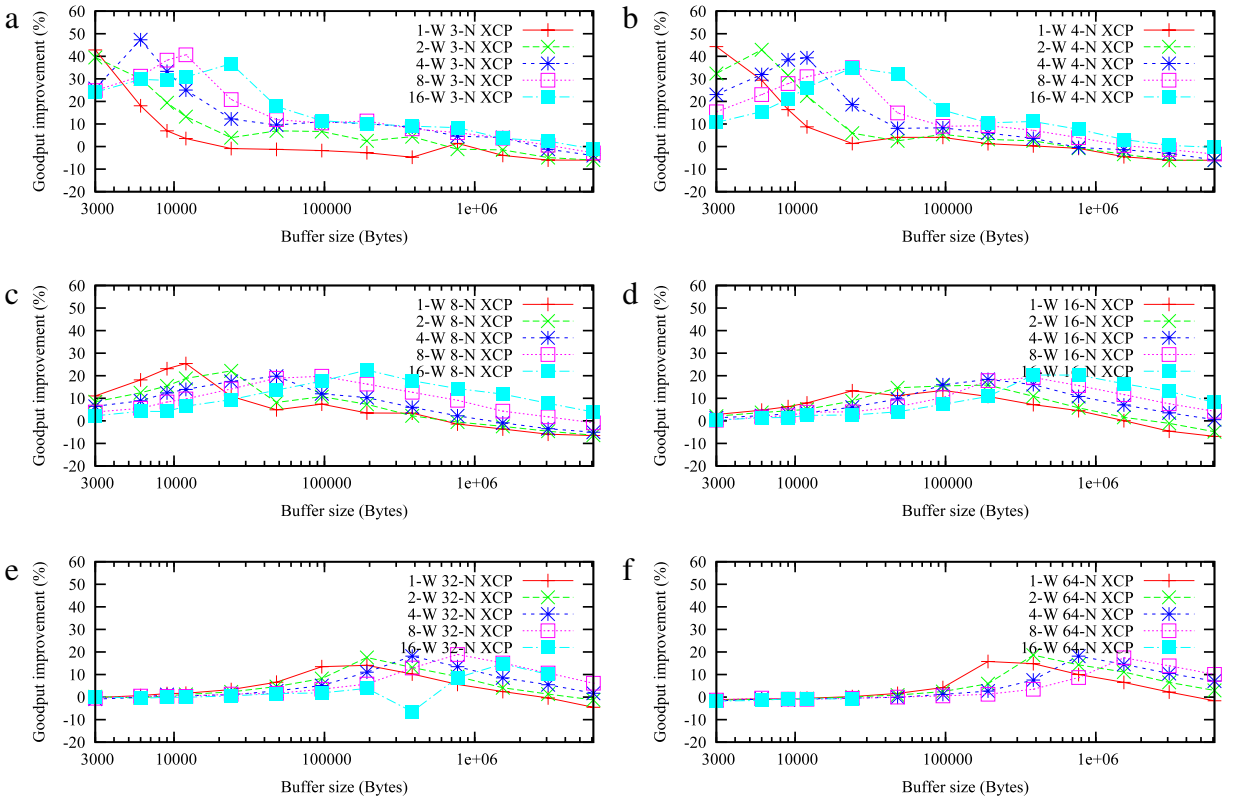

Fig. 11. Goodput improvement ratio of XCP-paced standard TCP over paced TCP.

of one-way TCP flows per wavelength on each link is 150 (excluding the ACK packets of returning flows). TCP data and ACK packets are carried in separate macroflows, which means that there are two macro flows from each ingress edge node to egress edge node. However, ACK traffic is much lower than data traffic in the simulations, so ACK macroflows can be neglected. When N is 3, as in Fig. 11(a), there are only two macro flows for data packets per wavelength on each link, which means that each data macro flow carries around 75 TCP flows. Multiplexing of only two paced macro flows creates a much smoother traffic than multiplexing of 150 paced TCP flows, so XCP-paced TCP has much lower buffer requirements than paced TCP. However, when N is 64, as in Fig. 11(f), there are 63 macro flows for data packets per wavelength on each link, which means that each macro flow only carries an average of 2.3 TCP flows. As the numbers of macro flows and TCP flows are close to each other, the traffic characteristics of paced TCP and XCP-paced TCP become similar. As each macroflow should carry at least one TCP flow, we can expect the XCP-paced TCP to have almost the same buffer requirements under the worst case scenario.

5. Conclusions

In this paper, we have presented a new buffer scaling rule showing the relationship between the number of wavelengths, nodal degree, and the required shared buffer size. Through an extensive simulation study, we showed that the buffer requirement increases with $O(N^{0.85}W^{0.85})$ for both standard TCP and paced TCP, while XCP-paced TCP's buffer requirement increases with $O(N^{1}W^{0.85})$, under a wide range of N and W values. We evaluated the parameters for an approximate buffer scaling rule with the formula $B = C(N - c)^{a}W^{b}$. We showed that the scaling rule applies when the wavelengths are both underutilized and fully utilized. These guidelines can be very useful in estimating the buffer requirements of routers with different sizes easily. Simply by doing a single simulation or experiment to calculate the shared buffer requirement of a very small router with only four links and a single wavelength, we can estimate the buffer requirement of a very large router with many links and wavelengths under the same traffic load, without doing very long simulations or building a big real router for testing. Moreover, when there are routers with different W and N in the network, we can easily estimate the buffer requirement for each router in order to provide an adequate packet drop rate and buffering delay to flows at all routers along the network. Even if the buffer scaling parameters differ due to a different TCP variant, MSS, network configuration, etc., it is likely that the new scaling parameters can be estimated by applying the same procedure as that used in this paper.

As future work, we will try to estimate the buffer scaling parameters for non-uniform traffic. We will simulate mesh architectures to evaluate the applicability on multihop networks. We will try to further decompose and formulate C in terms of other network parameters such as maximum congestion window size, number of TCP flows and

macroflows per wavelength, bandwidth, MSS, etc. Also, we will calculate the buffer requirements of a blocking switch where each switching plane is connected to the shared buffer by fewer than N ports. Such a switch may further decrease the router cost by decreasing the switching fabric size.

Acknowledgement

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

References

[1] R. Takahashi, T. Nakahara, K. Takahata, H. Takenouchi, T. Yasui, N. Kondo, H. Suzuki, Photonic random access memory for 40-GB/s 16-b burst optical packets, IEEE Photonics Technology Letters 16 (2004) 1185–1187.
[2] G. Appenzeller, J. Sommers, N. McKeown, Sizing router buffers, in: Proceedings of ACM SIGCOMM, 2004, pp. 281–292.
[3] T. Aoyama, New generation network (NWGN) beyond NGN in Japan, 2007. Web page: http://akari-project.nict.go.jp/document/INFOCOM2007.pdf.
[4] A. Shinya, S. Matsuo, Yosia, T. Tanabe, E. Kuramochi, T. Sato, T. Kakitsuka, M. Notomi, All-optical on-chip bit memory based on ultra high Q InGaAsP photonic crystal, Optics Express 16 (23) (2008) 19382–19387.
[5] C. Villamizar, C. Song, High performance TCP in ANSNET, Computer Communication Review 24 (5) (1994) 45–60.
[6] K. Avrachenkov, U. Ayesta, A. Piunovskiy, Optimal choice of the buffer size in the internet routers, in: Proceedings of IEEE Conference on Decision and Control, 2005, pp. 1143–1148.
[7] S. Gorinsky, A. Kantawala, J. Turner, Link buffer sizing: a new look at the old problem, in: Proceedings of ISCC, 2005, pp. 507–514.
[8] M. Enachescu, Y. Ganjali, A. Goel, N. McKeown, T. Roughgarden, Part III: routers with very small buffers, ACM SIGCOMM Computer Communication Review 35 (2005) 83–90.
[9] A. Vishwanath, V. Sivaraman, M. Thottan, Perspectives on router buffer sizing: recent results and open problems, ACM SIGCOMM Computer Communication Review 39 (2) (2009) 34–39.
[10] H. Jiang, C. Dovrolis, Source-level IP packet bursts: causes and effects, in: Proceedings of ACM SIGCOMM/Usenix Internet Measurement Conference, 2003, pp. 301–306.
[11] A. Aggarwal, S. Savage, T. Anderson, Understanding the performance of TCP pacing, in: Proceedings of INFOCOM 2000, 2000, pp. 1157–1165.
[12] D. Katabi, M. Handley, C. Rohrs, Congestion control for high bandwidth-delay product networks, in: Proceedings of ACM SIGCOMM, 2002, pp. 42–49.
[13] O. Alparslan, S. Arakawa, M. Murata, Rate-based pacing for small buffered optical packet-switched networks, Journal of Optical Networking 6 (9) (2007) 1116–1128.
[14] O. Alparslan, S. Arakawa, M. Murata, XCP-based transmission control mechanism for optical packet switched networks with very small optical RAM, Photonic Network Communications 18 (2) (2009) 237–243.
[15] O. Alparslan, S. Arakawa, M. Murata, Packet switch architectures for very small optical RAM, in: Proceedings of Internet 2009, 2009, pp. 106–112.
[16] H. Takeuchi, Photonic packet router project, 2008. Web page: http://www.eurojapan-ict.org/ppts_forum_march/HiroakiTakeuchi.pdf.
[17] L. Zhang, S. Shenker, D.D. Clark, Observations on the dynamics of a congestion control algorithm: the effects of two-way traffic, in: Proceedings of ACM SIGCOMM, 1991, pp. 133–147.
[18] J. Kulik, R. Coulter, D. Rockwell, C. Partridge, A simulation study of paced TCP, Technical Report, BBB, 1999.
[19] N. Beheshti, Y. Ganjali, A. Goel, N. McKeown, Obtaining high throughput in networks with tiny buffers, in: Proceedings of IWQoS, 2008.
[20] S. Kandula, D. Katabi, B. Davie, A. Charny, Walking the tightrope: Responsive yet stable traffic engineering, in: Proceedings of ACM SIGCOMM, 2005, pp. 253–264.
[21] S. McCanne, S. Floyd, NS Network simulator, 2002, http://www.isi.edu/nsnam/ns/.