Performance of Paced and Non-Paced Transmission Control Algorithms in Small Buffered Networks

Onur Alparslan *†

Graduate School of Information Science and Technology,

Osaka University, 1-3, Yamadagaoka, Suita, Osaka 560-0871, Japan

a-onur@ist.osaka-u.ac.jp

Shin'ichi Arakawa

Graduate School of Economics,

Osaka University, 1-7 Machikaneyama, Toyonaka, Osaka 560-0043, Japan

arakawa@econ.osaka-u.ac.jp

Masayuki Murata

Graduate School of Information Science and Technology,

Osaka University, 1-3, Yamadagaoka, Suita, Osaka 560-0871, Japan

murata@ist.osaka-u.ac.jp

Abstract

Famous rule-of-thumb states that a buffer sized at $B = RTT \times BW$, where RTT is the average round trip time and BW is the bandwidth of output link is necessary in order to achieve high utilization with TCP flows. However, as the link speeds continue increasing with technological advances, this buffer requirement starts becoming an important cost factor on routers of electronic networks. On the other hand, bursty nature of TCP limits further decreasing the buffer requirements, because it brings a very high packet drop rate in small buffered networks. In this paper, we evaluate several transmission control algorithms in small buffered networks. The algorithms include TCP Reno, TCP NewReno, Highspeed TCP with SACK, and

*Corresponding Author. Phone +81-6-6879-4542. FAX +81-6-6879-4544.

†O. Alparslan is supported by Ministry of Education, Culture, Sports, Science and Technology, Japan (Monbukagakusho)

XCP. Simulation results show that all non-paced TCP and XCP versions perform poorly. Furthermore, the results show that even rule-of-thumb buffers are not enough for XCP in some cases because of high burstiness of XCP. We therefore introduce pacing method at sender side for making transmission control algorithms suitable for very small buffered networks. We show that pacing alone is not enough for XCP to make suitable for small buffered networks, so we introduce a suitable parameter set. Simulation results show that buffer requirements on routers are greatly reduced by paced XCP with suitable parameter settings, while keeping the best fairness, faster convergence, and high utilization.

1 Introduction

TCP was designed to run over a wide range of networks. However, the stable operation of TCP in the Internet imposes some important requirements. One of them is the buffer size requirement of routers. Famous rule-of-thumb [1] states that an output link of a router should provide a buffer sized at $B = RTT \times BW$, where RTT is the average round trip time and BW is the bandwidth of output link, in order to achieve high utilization with TCP flows. Recently, Appenzeller et al. [2] showed 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 flows passing through the link, is enough for achieving high utilization. This brings a significant decrease in required buffer size when there are many flows passing through the link. However, it requires a large number of flows for a drastic decrease in buffer requirements. For example, it is very hard to satisfy even this decreased buffer requirement in high-speed optical packet-switched networks where electronic buffering is not feasible.

TCP is well-known to behave bursty [3]. Bursty nature of TCP limits further decreasing the buffer requirements, because it brings a high packet drop rate in small buffered networks. One possible solution for solving this problem is TCP Pacing. Pacing is defined as transmitting ACK (data) packets according to a special criteria, instead of transmitting immediately upon arrival of a data (ACK) packet [4]. Pacing is initially proposed as a solution for ACK-compression [4]. Kulik et al, proposed using paced TCP for solving the problem of queuing bottlenecks by preventing bursty behavior of TCP [5]. In [6], results of extensive simulations show that there are many cases where paced TCP gives worse performance than TCP without pacing, so pacing must be applied carefully. [7] argues that when pacing is used, $O(\log W)$ buffers are sufficient where W is the maximum congestion window size of each flow. However, architecture in [7] imposes a limitation on the congestion window size of all flows, because the buffer size in their proposal depends on the maximum window size. This brings an important limitation. When there are not enough number of flows, these flows can utilize only a small amount of bandwidth because of the window size restriction. It gives very high utilization only when the network is slightly over-provisioned and there are enough number of flows. In this paper, we do not limit the window size of flows, so it is possible to achieve high utilization with small number of flows.

In this paper, we evaluate Paced (P.) and Non-Paced (N.P.) versions of TCP Reno, TCP NewReno, HSTCP with SACK [8]

and XCP [9]. TCP Reno and NewReno are selected for showing the effect of packet recovery mechanisms when packet losses occur. HSTCP is selected because it can achieve high utilization in high-bandwidth product links much faster than Reno and NewReno without requiring router-assistance, so it is a possible alternative to XCP. In [9], it is shown that XCP significantly improves the overall performance when compared with TCP in terms of drop rate, utilization, queue build-up, delay and fairness. Therefore, we selected and focused on XCP. However XCP requires router-assistance. The income of router-assistance will be shown by comparing the performance of XCP with other TCP versions. In optical packet switched networks, router-assisted transmission control is a possible candidate.

First, we show that even rule-of-thumb buffers are not enough to N.P. XCP in some cases because of high burstiness and show that pacing is a solution. Then, we introduce a methodology based on pacing and careful selection of parameters for decreasing the buffer requirements of P. XCP for very small buffered networks. We show the required buffer size for P. XCP on different traffic and network settings. Then, we compare the performance of other P. TCP versions with P. XCP.

The rest of the paper is organized as follows. Section 2 describes the basics of XCP algorithm. Section 3 describes the methods for decreasing buffer usage. Section 4 describes the simulation methodology and presents the simulation results. Finally, we conclude in section 5.

2 XCP Basics

XCP is a new transmission control protocol using a strong control theory framework [9]. It makes use of explicit feedbacks received from the network. It decouples the utilization control from the fairness control.

Routers calculate flow-specific feedbacks by using the information provided by the flows. They do not require maintaining per-flow state information. XCP routers maintain a per-link control-decision timer. When a timeout occurs, a router updates its control decisions calculated by Efficiency Controller (EC) and Fairness Controller (FC).

EC is responsible for maximizing link utilization by controlling aggregate traffic. Every router calculates a desired increase or decrease in aggregate traffic for each output port by using the equation $\phi = \alpha \cdot d \cdot S - \beta \cdot Q$. In this equation, ϕ is the total amount of desired change in input traffic. α and β are the spare bandwidth control parameter and queue control parameter respectively 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.

FC is responsible for fairly distributing the feedback to flows according to ϕ. When ϕ is small, it may take a long time to converge to fairness. Bandwidth shuffling, which redistributes a small amount of traffic among flows, is used in order to prevent this. Shuffled traffic is calculated by h=max(0,$\gamma \cdot y - |\phi|$), where γ is the shuffling parameter and y is the aggregate input traffic rate in the last control interval.

When a router receives a packet containing feedback, it calculates and compares its own feedback with the feedback available in the packet header. If its own 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 a XCP source agent receives an ACK containing 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 Decreasing Buffer Usage for Small Buffered Networks

3.1 Parameter Settings

Routers make use of buffers when there is over-utilization. Long-term traffic over-utilization occurs when total traffic sent by sources has a rate higher than link capacity. It can be prevented by making sure that average input traffic sent by sources is less than bottleneck link speed. TCP sources continue increasing window size until congestion window size limit of operating system is reached or packet loss occurs. Packet loss is assumed as signal of congestion. Guaranteing under-utilization of links with TCP flows is very hard. However, it is possible to guarantee under-utilized operation of the links by carefully setting a parameter in XCP. As explained above, ϕ parameter of the efficiency controller of XCP is calculated according to the equation $\phi = \alpha \cdot d \cdot S - \beta \cdot Q$ where S is the spare bandwidth that is the difference between the link capacity and input traffic in the last control interval. Link capacity is given as a parameter to XCP. Therefore, a router needs to know the capacity (link speed) of its output link when calculating feedbacks to the packets passing through this link. Giving a capacity value less than real link-speed causes under-utilization and a value bigger than real link-speed causes over-utilization. It is possible to use this parameter for under-utilizing the links. When a link is under-utilized, the spare capacity prevents over-utilization caused by oscillations and overshoots in utilization and makes sure that link is always under-utilized.

3.2 Pacing

Even when long-term average input traffic rate is less than link capacity, it is possible to see over-utilization on short-term scales when traffic sources are bursty. When two or more packets contend, it causes a temporary over-utilization and buffer is used. In [9], it is shown that XCP significantly improves the overall performance when compared with TCP in terms of drop rate, utilization, queue build-up, delay and fairness. However, XCP is not suitable for small buffered networks, because it can behave even much more burstier than TCP. XCP is affected by all possible sources of burstiness (except slow start burstiness [3]) of TCP. However, there is one more important reason for burstiness in XCP. TCP increases its congestion window at most by one MSS when it receives an ACK. Unlike TCP, when XCP receives an ACK, it updates its congestion window according to the formula $cwnd = max(cwnd + H\_feedback, s)$ as explained in Section 2. When the congestion window is updated, the increase in window size can be much bigger than a single packet, which may cause injecting a big burst of packets back-to-back into the network. When there is two-way traffic, the joint effect of burstiness of XCP and

ACK-compression [4] can make XCP unstable as it will be shown in the simulations.

Pacing is a possible solution for minimizing burstiness of both XCP and TCP. Pacing can be applied on data packets or ACK packets or both of them. There are different methods for determining the pacing interval. The most common and easiest one is applying an interval between sending times of packets, calculated by $\frac{RTT}{CWND}$, where $CWND$ is the current congestion window and RTT is the estimated round-trip time. There are also some other proposals like using the Bandwidth-Share-Estimate(BSE) of TCP-Westwood [10], or 4-hop propagation delay in wireless networks [11], or randomizing the interval [12].

Pacing is implemented by using a variant of token-based leaky bucket algorithm. In this paper, only data packets are paced since data packet pacing has a much bigger impact on the performance than ACK pacing. A packet inside the congestion window is sent to the output link, if there is a token inside the token buffer. After the packet is sent, a token is removed from the token buffer. If there is no token inside the token buffer, packet must wait until a token arrives. Token buffer is filled with a rate of $\frac{RTT}{CWND}$. Changing the token buffer size affects the burstiness of the flow. Using a token buffer size of only one token gives the least bursty output traffic. Token buffer fill rate must be updated each time $CWND$ or RTT changes. Also, pacing algorithms require fine course timers as discussed in [5].

Even when the traffic is perfectly smooth and link is under-utilized, it is possible that two or more packets arrive from different input links at the same time. Again, buffer is used for storing contending packets. There are methods like slot-based reservation for preventing packet contention, but they require complex control mechanisms, so they are not investigated in this paper.

4 Evaluation

4.1 Simulation Settings

We evaluate paced versions of TCP Reno, TCP NewReno, HSTCP with SACK and XCP over ns version 2.28 [13]. Their paced (P.) and non-paced (N.P.) versions are simulated and compared. Original XCP code in ns-2 does not take the arrival of ACK packets into account when calculating the utilization of links used in XCP's efficiency controller. It is modified so that ACKs are counted in the utilization for a more realistic simulation. In all simulations, time-stamps option is used and agents have fine course timers. There is no limit on congestion window size of TCP and XCP flows. Slow start threshold of TCP is 64 packets. Data packet size is 1000Bytes including the headers in both TCP and XCP simulations. A dumbbell topology and a parking-lot topology are used for computer simulations. Simple drop-tail queuing is used. Queue limits are set in terms of Bytes instead of packets for a more realistic simulation.

XCP is simulated with two different parameter sets. One of them is the original parameter set (O.P.) used in [9]. They are α=0.4, γ=0.1 and β=0.226. Capacity of the output links are given as their real link-speed to the XCP algorithm in routers.

The second set is the conservative parameter set (C.P.) used for minimizing the buffer usage. Their selection is as follows:

- XCP parameter α=0.4 used in [9] sometimes causes utilization overshoots and oscillations. Therefore, we selected a more conservative α=0.2, which gives a slower but more stable link utilization and decreases utilization overshoots. Even though it has slower utilization convergence than O.P., it is still much faster than TCP versions on networks with high bandwidth-delay product.
- When α parameter is decreased, it is also necessary to decrease γ parameter that is responsible for bandwidth shuffling. Otherwise, there can be too much under-utilization in some links in case they have flows that are bandwidth throttled in other bottleneck links as explained in [9]. Therefore, γ=0.05 is used instead of γ=0.1 in [9].
- β must be selected according to the formula $\beta = \alpha^2\sqrt{2}$ as proved in [9], so β=0.056 is straightforward.
- Capacity of the output links are given as 90% of their real link-speed to the XCP routers. The spare 10% capacity gives a safety margin against possible oscillations in utilization.

Extensive simulations on mixed flow environments with competing P. and N.P. TCP versions and performance of P. TCP flows in large buffered networks are already available in [6] and [12], so their results are not presented here.

4.2 Dumbbell Topology Simulations with Long Flows

A dumbbell topology is used for simulations with static flows. The capacity of the bottleneck link is 622Mbps (OC-12). Both edges of bottleneck link have extension links. Extension links are 2.4Gbps (OC-48). RTT distribution between source-destination node pairs ranges from 64ms to 100ms for preventing ACK-clocking. Average RTT is 82ms. Bottleneck delay is 30ms.

A limited number of FTP flows start at random times between [0-10]s and continue until the simulation stops. It is assumed that flows always have data to send. Total simulation duration is 100s. Only the data between [40-100]s is used in average utilization calculations. Two-way traffic is created by applying reverse traffic with same properties as the forward traffic for showing the possible effects of ACK-compression problem [4].

4.2.1 Required Buffer Size

XCP tries to minimize the buffer occupancy and not to drop any packets. First, we find the maximum queue occupancy of bottleneck link by P. and N.P. XCP with O.P. and C.P. when there are different number of long flows ranging from 2 to 800. Size of all buffers are limited to rule-of-thumb bandwidth-delay product.

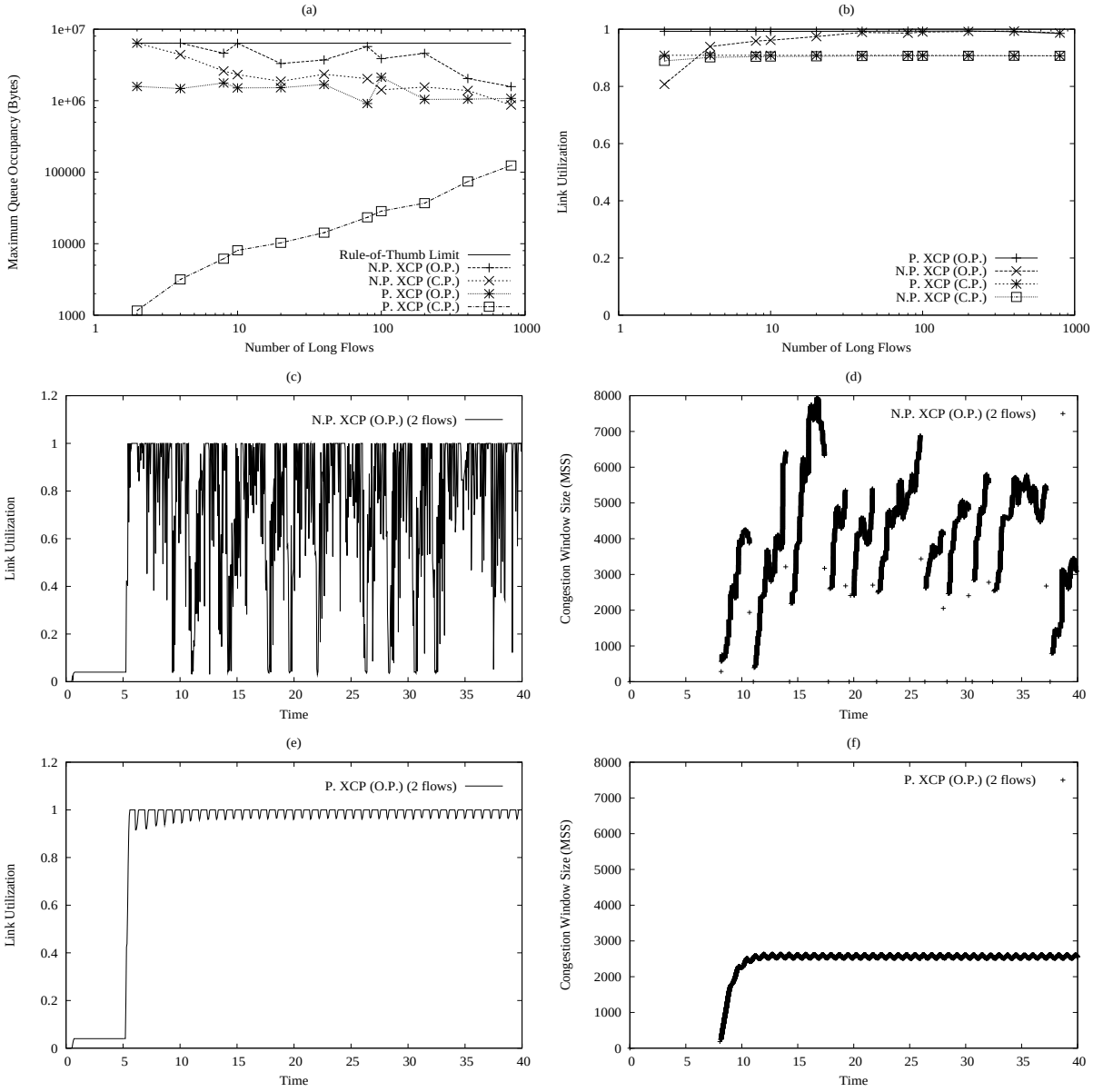

Figure 1. (a) Maximum buffer occupancy and (b) bottleneck utilization of P. and N.P. XCP versions with C.P. and O.P. with rule-of-thumb buffer size. (c) Transient bottleneck utilization and (d) congestion window size of N.P. XCP with O.P.. (e) Transient bottleneck utilization and (f) congestion window size of P. XCP with O.P..

In Fig. 1(a), x-axis shows the number of long flows and y-axis shows the maximum queue occupancy of bottleneck link, both in log scale. In Fig. 1(b), x-axis shows the number of long flows and y-axis shows the average utilization of bottleneck link. X-axis is in log scale. Y-axis is between [0-1] where one means 100% utilization. When the buffer occupancy of P. XCP with O.P. and C.P. are compared, Fig. 1(a) shows that C.P. can decrease the buffer requirement greatly, depending on the number of flows. As an example, when there are 100 flows, the required buffer size is around 6.4MBytes according to the rule-of-thumb, and around 638KBytes according to Appenzeller's $\frac{RTT \times BW}{\sqrt{N}}$ formula. As seen in Fig. 1(a), the maximum buffer occupancy of P. XCP with C.P. was only 23,360Bytes. The maximum buffer occupancy decreases as the number of flows decrease. If we extend the simulation time, it is possible to see a buffer occupancy bigger than these values due to randomness of packet arrivals to the bottleneck link, but its probability is low.

In the Fig. 1(a), it is seen that rule-of-thumb buffer (6.4MBytes) size was not enough for N.P. XCP when the number of flows is very low. N.P. XCP became very oscillatory and started to lose many packets. Therefore N.P. XCP gave low average utilization when the number of flows is low as seen in Fig. 1(b). The reason of this behavior is the burstiness of XCP and ACK compression problem. XCP is a very bursty protocol, because it may increase window size in big amounts as described in the previous chapter, so sometimes it sends too many data packets back to back and the ACK packets are compressed by bottleneck link along the data packets of reverse traffic. Therefore the arrival of ACK packets to the sender is delayed. When the receiver receives only a burst of data packets back-to-back with no ACK packets between the data packets, ACK-starvation occurs and sender can not send new data packets in this period, which may cause under-utilization of the bottleneck link if the number of sources is low. Then, the receiver gets a burst of ACK packets compressed to the end of a data packet and starts sending a burst data packets, which over-utilizes the bottleneck link. Therefore, there is a big oscillation in the spare utilization variable of ϕ calculation. At the same time, queue size starts oscillating between a very low and a high value due to high burstiness of input traffic. XCP's efficiency controller uses only the persistent (minimum) queue size without taking the maximum buffer occupancy into account for calculating the feedback, so including the queue size with $\beta \cdot Q$ in the efficiency controller algorithm does not help stabilizing the system. Therefore, router feedbacks show big oscillations, which further increases the burstiness. It requires a buffer that is much larger than rule-of-thumb so that it can carry big oscillations in the queue size.

In the simulations, the buffer limit was set to rule-of-thumb, so it was not enough for N.P. XCP when the number of flows is low. Therefore packet losses occurred. Current XCP implementation in ns-2 drops its window size to half and applies TCP Reno's recovery algorithm. This can interfere with XCP's own window control algorithm and further increase the oscillations. Figures 1(c),(d),(e),(f) show the transient bottleneck link utilization and congestion window size of one of the P. and N.P. XCP flows with O.P. when there are two flows (four flows including the reverse flows). As seen in Fig. 1(c),(d), utilization and congestion window size of N.P. XCP shows unstable behavior and cause many packet drops and timeouts. Even when an unlimited buffer is provided, N.P. XCP shows a very oscillatory behavior. Unlimited buffer simulation results

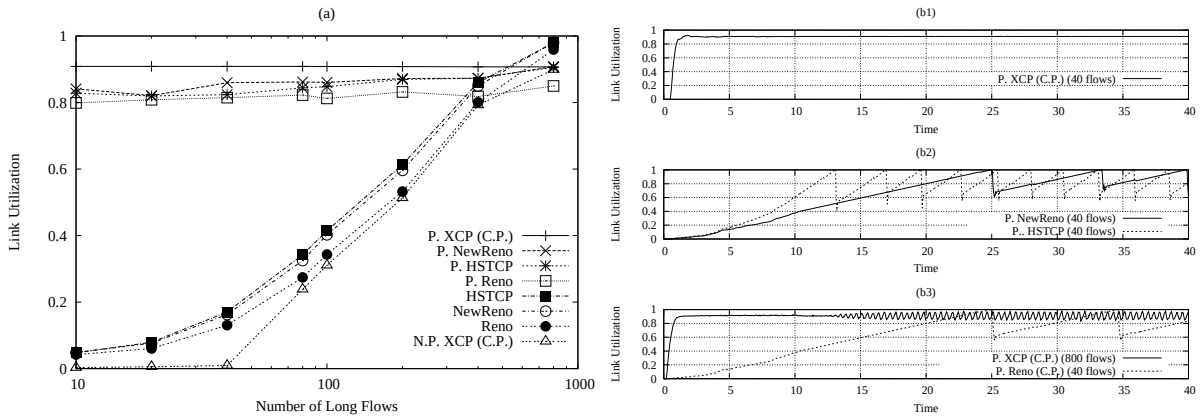

Figure 2. (a) Average bottleneck utilization of P. and N.P. XCP with C.P. and TCP versions with small buffers. (b) Transient bottleneck utilization of P. XCP with C.P. with 40 and 80 flows and P. TCP versions with 40 flows.

are not presented in this paper due to lack of space. On the other hand, as seen in Fig. 1(f), window size of P. XCP reaches its fair-share in a short time and stays almost constant with very small oscillations throughout the simulation.

As the number of flows increase, the oscillations and therefore the required buffer size decrease, because of the decrease in the duration of periods with low input traffic to bottleneck link due to traffic multiplexing. Also, C.P. decreased the degree oscillations, but could not solve it fully.

4.2.2 Behavior in Small-buffered Network

In Fig. 1(a), it is seen that maximum buffer occupancy values of P. XCP with C.P. is very small. We set the buffer limit of bottleneck link to those maximum buffer occupancy values in simulations with corresponding number of flows and evaluate the performance of other TCP versions and XCP with C.P..

In Fig. 2(a), all P. TCP versions achieved over 80% utilization when the buffer size is small. P. XCP with C.P. always gave a utilization around the value of capacity parameter 90%. Among the P. TCP versions, P. NewReno gave the second best utilization that is marginally higher than P. HSTCP. P. Reno is the last one. The reason of why P. TCP versions could not achieve 100% utilization is synchronization of congestion window of flows. When the total input rate exceeds the link-speed, queue starts dropping packets and many flows lose packets due to uniform arrival of packets, so many flows decrease their window size at the same time, which brings synchronization of flows. Also many flows lose multiple packets inside a congestion window. Reno is known to have low performance when there are multiple losses inside a window, so its paced version gives the lowest performance.

When there is synchronization, it can be expected to have higher utilization with P. HSTCP than P. NewReno especially when average window size is large. This is because, unlike NewReno, which decreases its window to half, HSTCP decreases

its window size to a value higher than half size when window size is large. However, HSTCP increases its window size at steps larger than one MSS per RTT when window size is large. As a result, P. HSTCP creates a bigger congestion and drops more packets than P. NewReno. Therefore, the number of flows losing packets and getting synchronized is more than P. NewReno. Higher level of synchronization among P. HSTCP flows brings lower utilization than P. NewReno.

It is seen that N.P. TCP versions give very low utilization unless there are large number of flows, due to very high packet loss rate caused by their high burstiness and small-buffer size of the bottleneck link. However, when there are many flows, the aggregated throughput of flows becomes high enough to achieve high utilization. Also simulated buffer size increases with the number of flows according to the maximum buffer occupancy of Paced XCP with C.P. shown in Fig. 1(a). When there are 800 flows, the simulated bottleneck buffer size is 124,040Bytes and simulations show that utilization of N.P. TCP versions are over 90% percent. It is important to note that according to Appenzeller's formula, the required buffer size is only around 225KBytes that is close to the simulated buffer size.

When we compare the average utilization performance of N.P. TCP flows, we see that HSTCP gives the best performance. NewReno gives a marginally lower performance and Reno gives the worst performance. Average window size of flows is very low, so HSTCP's congestion control algorithm operates with the same window increase and decrease factors as TCP. However, HSTCP gives a much better performance than Reno and a marginally better performance than NewReno due to performance improvement of SACK in environments with high loss rate.

When we check the transient utilization of first 40 seconds of P. TCP protocols with 40 flows and P. XCP with 40 and 800 flows in Fig. 2(b1) and Fig. 2 (b3), it is seen that P. XCP with C.P. achieves its target utilization in a very short time. P. HSTCP in Fig. 2(b2) achieves the second fastest conversation time in terms of fully utilizing the link, because of its fast window increase. P. NewReno in Fig. 2(b2) and P. Reno in Fig. 2(b3) give the worst convergence time. P. TCP versions show a saw-tooth behavior due to high synchronization.

When transient utilization graphs of P. XCP with 40 and 800 flows are compared in Fig. 2(b1) and (b3), it is seen that utilization of 800 flows simulation is stable in the first 15 seconds and then utilization starts to oscillate. In the first 15 seconds, window size of flows converge to their fair share. After 15 seconds, all the flows achieve their fair share. Then, small changes in the utilization cause all flows to increase or decrease their window size around their fair share simultaneously. XCP and TCP's congestion window is a real number. However, effective window size is an integer number of MSS. Therefore, the flow speed changes in discrete steps. If the average window size is small like in 800 flows simulation, simultaneous oscillation of congestion window size of many flows causes a big change in input traffic rate. This oscillatory behavior seen when the average window size is small, is also explained in [9] and stated that buffer is used for absorbing this oscillation. However, absorbing oscillations in the buffer is not possible in small buffered networks, so link-speed parameter is given as a lower value than real speed for making sure that the highest value of the oscillatory input traffic rate is less than link capacity as in Fig. 2(b3).

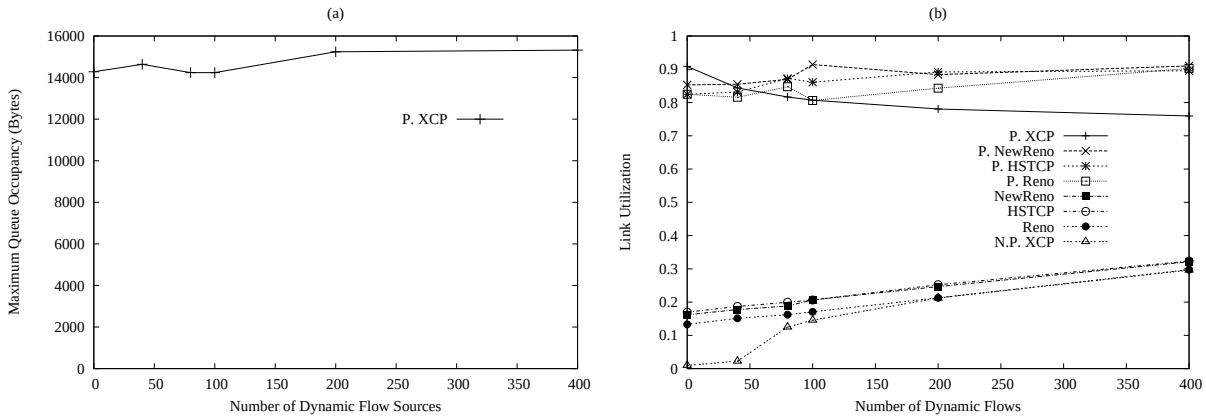

Figure 3. (a) Maximum queue occupancy of P. XCP with C.P.. (b) Average bottleneck utilization of P. and N.P. XCP with C.P. and TCP versions with small buffers.

4.3 Dumbbell Topology Simulations with Web-like Traffic

A web-like traffic is simulated by applying dynamic short flows and long FTP flows on a dumbbell topology for showing the performance of pacing algorithms under a more realistic input traffic. Link speeds of bottleneck and extension links are as given in the previous simulations. There are N nodes for short flows and 40 nodes for long flows giving a total of N+40 nodes on each side of the bottleneck link, where N is between [0-400]. RTT distribution of both long and short flows range from 64ms to 100ms. Average RTT is 82ms. Bottleneck delay is 30ms. All sources start at random times between [0-10]s. Long flows continue until the simulation ends. File size of short flows is Pareto distributed with shape 1.35 and average 30 packets as used in [9]. After sending a file, a node stays OFF for an exponential distribution with average 0.2s. A node reserved for short flows carries only one flow at a time. There is two-way traffic by applying reverse traffic with same properties as the forward traffic. The file size distribution of short flows is small, so small flows usually finish before reaching a high window size. Therefore, the total traffic injected by the short flows is low. Simulation duration is 50s and only data from [20-50]s is used in average utilization calculations.

First, we find the maximum queue occupancy of P. XCP with C.P., by setting the size of buffers to rule-of-thumb and changing the number of short flows. As seen in the Fig. 3(a), even when there are a high number of flows, maximum buffer occupancy do not show a big change because of the low rate traffic injection (less than 30% of bottleneck link utilization) of short flows and decreased utilization of bottleneck link as seen in Fig. 3(b). Lower bottleneck utilization decreases the maximum buffer occupancy due to decreased probability of contention.

We simulate N.P. XCP with C.P. and other TCP versions by applying these buffer occupancy values with corresponding number of short flows . As seen in Fig. 3(b), P. XCP gives lower utilization as the number of dynamic flows increase. It is mainly because of the fairness control algorithm of XCP. XCP tries to fairly distribute current utilization among all

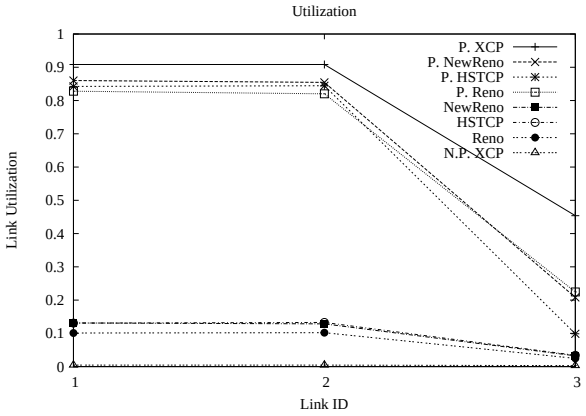

Figure 4. Average link utilizations of P. and N.P. XCP with C.P. and TCP versions with small buffers.

flows. However, the dynamic flows are very short, so they usually finish before using their fair-share. Their share becomes unused and the average bottleneck utilization of P. XCP decreases. Dynamic arrival and departure of short flows decreases the synchronization of long flows. Therefore, P. HSTCP gives higher utilization than P. NewReno at some points. Again, P. Reno generally gives worse performance than P. NewReno and P. HSTCP. Utilization of P. TCP versions increase as the number of dynamic flows increase due to decrease in the level of synchronization.

When we check the average utilization of N.P. TCP versions, we see that NewReno and HSTCP have almost the same performance. Again, Reno gives the worst performance. Utilization of N.P. TCP versions increase as the number of dynamic flows increase due to aggregated bandwidth of long and short flows.

4.4 Parking Lot Topology Simulations

A parking lot topology with 3 middle links is used for fairness and utilization simulations. Link speeds of bottleneck and extension links are as given in the previous simulations. Each middle link has 10ms propagation delay. There are 10 long flows passing through all three middle links. Also there are a set of 10 short flows passing through first middle link and another set of 10 short flows passing through second middle link. Third middle link has only long flows. Therefore, there are two bottlenecks at the same time. RTT distribution long flows (short flows) range from 64ms (24ms) to 100ms (60ms) with average RTT of 82ms (42ms), respectively. Two-way traffic is created by applying reverse traffic with same properties as the forward traffic. Simulation duration is 100s and only data from [40-100]s is used in average utilization calculations.

First, we find the maximum queue occupancy of P. XCP with C.P., by setting the size of buffers to rule-of-thumb. The maximum queue occupancy is found as 10,200Bytes. Then, we simulate N.P. XCP with C.P. and other P. and N.P. TCP versions by applying this buffer size to all middle links. In Fig. 4 shows the average utilization of three middle links. X-axis shows the middle link number. When we compare the P. XCP and P. TCP versions on the first two links, P. XCP gives the

	P. XCP	P. Reno	P. NewReno	P. HSTCP
Fairness	1	0.78	0.76	0.44
	N.P. XCP	N.P. Reno	N.P. NewReno	N.P. HSTCP
Fairness	0.76	0.72	0.74	0.73

Table 1. Fairness index of transmission control algorithms

highest utilization. P. NewReno is the second, P. HSTCP is the third and P. Reno is the last one. This result is the same as the dumbbell topology simulations. However, results are very different on the third link, which shows the average utilization of long flows. P. XCP has the highest utilization, which points to a very good fairness among long and short flows. Long flows are not penalized by high RTT or multiple bottlenecks. P. Reno has the second and P. NewReno has the third highest utilization. P. HSTCP has a very low utilization pointing to a high discrimination against long flows with multiple bottlenecks. When we compare the average utilization of N.P. TCP versions, we see that NewReno and HSTCP have a similar utilization and Reno has the worst utilization at all links. N.P. XCP with C.P. has less than 0.5% utilization, which is the lowest among all tested TCP and XCP versions, so it is hard to see utilization of N.P. XCP in the figure.

Fairness of goodput among long and short flows is evaluated by using Jain's fairness index $f = \frac{(\sum_{i=1}^{i=n} x_i)^2}{n \sum_{i=1}^{i=n} x_i^2}$, where n is the number of flows and x_i is the goodput of flow i [14]. Table 1 shows that P. XCP has perfect fairness. P. Reno and P. NewReno have similar fairness, which is lower than P. XCP. However, P. HSTCP has a very low fairness. N.P. TCP versions have similar fairness. Fairness value of N.P. XCP is meaningless, because it achieved less than 0.5% utilization due to totally unstable behavior.

4.5 Conclusions

TCP and XCP are well-known to behave bursty, so it is not possible to use them in a very small buffered high bandwidth-delay network. We showed that even rule-of-thumb sized buffers are not enough for XCP in some cases due to high burstiness of XCP. We showed that XCP can be adapted to small buffered networks by applying pacing and a careful selection of parameters. We compared P. XCP and different TCP versions and showed that P. XCP is a strong candidate for achieving high performance in small buffered networks.

A big disadvantage of XCP based algorithms is that they require deployment of XCP capable senders, receivers and routers. On the other hand, it is possible to use P. TCP algorithms by updating only senders.

Even though paced sources minimize the burstiness of traffic they send into the network, queues can change the intervals between the packets and make the traffic burstier in a fashion similar to ACK-compression and increase the buffer requirements of bottleneck link. Simulations on more realistic traffic models and bigger topologies are required for better understanding in the buffer requirements of paced algorithms.

References

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

[2] G. Appenzeller, N. McKeown, J. Sommers, and P. Barford, "Recent results on sizing router buffers," in *Proceedings of The Network Systems Design Conference*, 2004.

[3] H. Jiang and C. Dovrolis, "Source-level IP packet bursts: Causes and effects," in *Proceedings of ACM SIGCOMM/Usenix Internet Measurement Conference*, 2003, pp. 301–306.

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

[5] J. Kulik, R. Coulter, D. Rockwell, and C. Partridge, "A simulation study of paced TCP," BBN, Tech. Rep., 1999.

[6] A. Aggarwal, S. Savage, and T. Anderson, "Understanding the performance of TCP pacing," in *Proceedings of INFOCOM 2000 Conference on Computer Communications*, 2000, pp. 1157–1165.

[7] 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.

[8] S. Floyd, "Highspeed TCP for large congestion windows," RFC 3649, 2003.

[9] D. Katabi, M. Handley, and C. Rohrs, "Internet congestion control for future high bandwidth-delay product environments," in *Proceedings of ACM SIGCOMM*, 2002.

[10] A. Razdan, A. Nandan, R. Wang, M. Y. Sanadidi, and M. Gerla, "Enhancing TCP performance in networks with small buffers," in *Proceedings of 11th International Conference on Computer Communications and Networks*, 2002.

[11] S. ElRakabawy, A. Klemm, and C. Lindemann, "TCP with adaptive pacing for multihop wireless networks," in *Proceedings of 6th ACM international symposium on Mobile ad hoc networking and computing*, 2005, pp. 288–299.

[12] K. Chandrayana, S. Ramakrishnan, B. Sikdar, S. Kalyanaraman, O. Tickoo, and A. Balan, "On randomizing the sending times in TCP and other window based algorithms," RPI ECSE Networks Laboratory, Tech. Rep., 2001.

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

[14] R. Jain, *The Art of Computer Systems Performance Analysis: Techniques for Experimental Design, Measurement, Simulation, and Modeling*. John Wiley and Sons, Inc., 1991.