Improving XCP to Achieve Max-Min Fair Bandwidth Allocation

Lei Zan and Xiaowei Yang
University of California, Irvine
{lzan,xwy}@ics.uci.edu

***Abstract*— TCP is shown to be inefficient and instable in high speed and long latency networks. The eXplicit Control Protocol (XCP) is a new and promising protocol that outperforms TCP in terms of efficiency, stability, queue size, and convergence speed. However, Low et al. recently discovered a weakness of XCP. In a multi-bottleneck environment, XCP may achieve as low as 80% utilization at a bottleneck link and consequently some flows may only receive a small fraction of their max-min fair rates.**

This paper proposes iXCP, an improved version of XCP. Extensive simulations show that iXCP overcomes the weakness of XCP, and achieves efficient and fair bandwidth utilization in both single- and multi- bottleneck environments. In addition, we prove that iXCP is max-min fair in steady state. This result implies that iXCP is able to fully utilize bottleneck bandwidth. Simulations also show that iXCP preserves the good properties of XCP, including negligible queue lengths, near-zero packet loss rates, scalability, and fast convergence.

I. Introduction

It is well known that TCP's congestion control mechanism is inefficient and instable in high bandwidth-delay-product environments [1]–[4]. TCP treats a packet loss as an implicit congestion signal, and reacts to congestion by cutting its congestion window size by half, and then gradually increases its window size by one every round trip time (RTT). This saw-toothed behavior of TCP leads to throughput oscillation and link under-utilization, especially in a high bandwidth-delay-product environment.

The eXplicit Control Protocol (XCP) [5] overcomes the limitations of TCP by sending explicit window adjustment information from routers to end hosts. It is a promising candidate to replace TCP in a future Internet architecture [6]. Unlike TCP, an XCP flow does not implicitly probe available bandwidth. Instead, a router computes the spare bandwidth of an output link, and fairly allocates the bandwidth among all flows that share the same link. The router then writes the amount of window adjustment in the congestion header of an XCP flow. This explicit control mechanism allows a flow to quickly utilize the available bandwidth of a link. Early results have shown that XCP is highly efficient, stable, and scalable [5].

However, Low et al. [7] recently revealed a weakness of XCP. In a multi-bottleneck environment, XCP's congestion control mechanism may cause some bottleneck link to be under-utilized, and a flow may only receive a small fraction of its max-min fair allocation [8].[1] Low et al. demonstrated this result with both a theoretical model and simulations. With the chosen parameters, XCP may utilize only 80% of bottleneck bandwidth in the worst case. In this paper, we refer to this theoretical model as *Low's model*.

Intuitively, this problem occurs because XCP-enabled routers independently compute bandwidth allocation. For instance, if a flow is bottlenecked at a downstream link, an upstream router would still attempt to allocate bandwidth to that flow to ensure local fairness. This leads to link under-utilization, which in turn causes some flow to receive only a fraction of its max-min fair allocation.

In this paper, we propose a simple improvement to XCP that overcomes this limitation. We add an additional bottleneck identifier field into XCP's congestion header. If a flow is bottlenecked at an outgoing link of a router, the router writes the link identifier into this field. Other routers are then aware that the flow is not bottlenecked at their links. We further modify XCP's control algorithm not to waste bandwidth on flows that are bottlenecked at other routers.

We use extensive simulations to show that our improved XCP (iXCP) is able to achieve nearly 100% link utilization and max-min fairness in steady state. We use a theoretical model to show that iXCP is max-min fair. This result also implies that iXCP is able to fully utilize bottleneck bandwidth. In addition, our simulation results show that iXCP preserves the desirable features of XCP. It converges fast to max-min bandwidth allocation with a negligible queue length; it is efficient and fair in high speed and long latency networks as well as conventional networks; it also remains as a scalable stateless solution.

This paper has three key contributions. The first is our analysis on the root cause of XCP's under-utilization problem. Although this problem has been observed in [7], [9], to the best of our knowledge, we are the first to pinpoint the particular protocol mechanism that causes the problem. The second is our improvement to XCP. This improvement makes XCP achieve its full potential: iXCP is highly efficient and fair in all types of topologies. The third is the theoretical analysis

[1]A max-min fair allocation maximizes the bandwidth allocated to the flow with a minimum allocation. Max-min fairness satisfies the following conditions: 1) the allocation is feasible, i.e., the sum of the allocated rates does not exceed a link's capacity; 2) a flow's rate allocation cannot be increased without violating the feasibility constraint or without decreasing the rate of some flow that has an equal or smaller bandwidth share.

that shows iXCP is max-min fair. This analysis provides us the confidence that iXCP will continue to operate efficiently and fairly in scenarios that we did not simulate. We note that our theoretical analysis builds on Low's model [7].

The rest of the paper is organized as follows. In section II, we briefly summarize how XCP works and its weakness. Section III describes iXCP. We use extensive simulations to evaluate the performance of iXCP in Section IV. Section V presents our theoretical analysis that proves iXCP is max-min fair. We compare our work with related work in section VI. Section VII concludes the paper.

II. Understanding the Weakness of XCP

Understanding what causes a problem is the first step to solve the problem. In this section, we describe how XCP works and analyze what makes XCP under-utilize link bandwidth.

A. How XCP works

XCP uses explicit window adjustment for a flow to increase or decrease its congestion window. Each packet carries a congestion header that contains three fields: the sender's current congestion window size: *cwnd*, the estimated round trip time: *rtt*, and the router feedback field: *feedback*. Each sender fills its current *cwnd* and *rtt* values in the congestion header on packet departures, and initializes the *feedback* field to its desired window size.

The core control mechanism of XCP is implemented at routers. Each router has two logical controllers: efficiency controller and fairness controller. In each control interval, the efficiency controller computes spare bandwidth as follows:

$$\phi = \alpha d(c - y) - \beta b \tag{1}$$

where d is the average round-trip time, c is the link capacity, y is the input traffic rate, and b is the persistent queue length. α and β are two control parameters, with default value 0.4 and 0.226 respectively.

The fairness controller is responsible for fairly allocating bandwidth to each flow. When a link is in high-utilization region, XCP's fairness controller performs a "bandwidth shuffling" operation to ensure local fairness. This operation simultaneously allocates and deallocates the shuffled bandwidth among flows. The shuffled bandwidth is computed as follows:

$$h = \max\{0, \gamma y - |\phi|\} \tag{2}$$

where γ is a control parameter with default value 10%.

In each control interval, the spare bandwidth computed from Equation (1) if it is positive and the shuffled bandwidth computed from Equation (2) are allocated to each flow additively, i.e., each flow gets an equal amount of increment. At the mean time, the spare bandwidth if it is negative and the shuffled bandwidth is deallocated multiplicatively, i.e., each flow gets a rate decrement proportional to its current rate. A router writes the net window adjustment (the increment minus the decrement) information in the *feedback* field of a packet. A more congested downstream router may overwrite this field with a smaller increment or a larger decrement. The receiver echoes back the *feedback* to the sender, and the sender adjusts its window size accordingly.

B. The Weakness Revealed

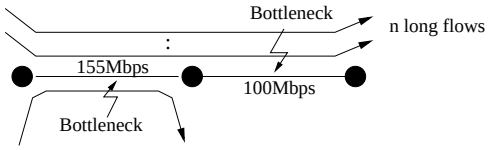


Fig. 1. **In this topology, XCP under-utilizes the 155Mb/s link. The n long flows are bottlenecked at the 100Mb/s link. So the short flow should be able to send at 55Mb/s, but with XCP, it may get a bandwidth allocation anywhere between 24Mb/s to 55Mb/s, depending on the number of long flows.**

Low et al. rigorously modeled XCP's control algorithm in [7]. Both their analysis and simulations revealed that XCP may under-utilize a bottleneck link. In the worst case, XCP may only achieve 80% link utilization (with XCP's default parameters). We describe this problem using the network topology shown in Figure 1. This is an example used by Low et al. in [7]. In this example, there are n ($n \geq 2$) long flows that cross both links in the figure and a short flow that only goes through the 155Mb/s link. Since each long flow is bottlenecked at the 100Mb/s link, in a max-min fair allocation, each of them gets a bandwidth allocation of $100/n$ Mb/s. The short flow should fully utilize the residual bandwidth of the 155Mb/s link, obtaining a 55Mb/s bandwidth allocation. However, both simulations and analysis show that XCP allocates a rate $r(n) < 55$ Mb/s bandwidth to the short flow in steady state. The function $r(n)$ decreases as n increases. Figure 4(a) and 4(b) show the utilization of the 155Mb/s bottleneck link and the ratio between the short flow's allocated rate and its max-min fair share respectively. As we can see, both the link utilization and the short flow's bandwidth allocation decrease as the number of long flows n increases.

Intuitively, this problem occurs because XCP-enabled routers independently allocate bandwidth to each flow based on their local information. Although a long flow is bottlenecked at the downstream 100Mb/s link, the upstream router at the 155Mb/s link still attempts to increase its bandwidth share to ensure local fairness. As a result, the short flow that is only bottlenecked at the upstream link cannot fully utilize the link and obtain its max-min fair share.

We explain it in more depth to shed light on the solution. The problem is primarily caused by XCP's fairness controller. The "bandwidth shuffling" operation essentially uses the *Additive-Increase Multiplicative-Decrease (AIMD)* algorithm to adjust rates among flows. Thus, a flow with a larger bandwidth share will be tariffed more and get back less, and vice versa. In a single bottleneck environment, the *AIMD* policy will equalize all flow's bandwidth shares, thereby achieving fairness.

The problem arises in a multi-bottleneck environment. In such an environment, the de-allocated bandwidth from a shuffling operation may not be fully utilized, when some flows are bottlenecked at other links (no matter downstream or upstream links) and cannot further increase their sending rates. This deallocated but not re-used bandwidth leads to

link under-utilization. The flows that could have used this bandwidth have a net loss in a shuffling operation. Recall that the fairness controller is also in charge of allocating the spare bandwidth computed by the efficiency controller. When the net loss of a flow from the shuffling operation balances out its net gain in the spare bandwidth allocation, the system has reached an equilibrium, in which a flow's sending rate cannot further increase although there remains un-used bandwidth.

We illustrate this problem with a numerical example. In the example shown in Figure 1, when there are four long flows, with XCP, the equilibrium rate allocation for the short flow is 43Mb/s, and the rate for each long flow is 25Mb/s. This leads to a 12Mb/s under-utilization at the 155Mb/s link. We show that this is an equilibrium using XCP's default parameters [5]. At the 155Mb/s upstream link, the used bandwidth is 143Mb/s and the unused bandwidth is 12Mb/s. The efficiency controller will compute the spare bandwidth as 0.4 * 12 = 4.8Mb/s. The fairness controller will compute the shuffled bandwidth as 0.1 * 143 - 4.8 = 9.5Mb/s. The additive increase for each flow is the sum of the spare and shuffled bandwidth evenly distributed among all flows: (4.8 + 9.5)/5 = 2.86Mb/s; the multiplicative decrease for the short flow is proportional to its rate: 9.5/143 * 43 = 2.86 Mb/s, and similarly, the decrease for a long flow is 9.5/143 * 25 = 1.66 Mb/s. The decrease for the short flow is the same as the increase for the short flow, and the long flows cannot increase beyond 25Mb/s due to the 100Mb/s bottleneck link. Therefore, no flow's rate can change over time. The system has reached an equilibrium, in which 12Mb/s bandwidth is not used.

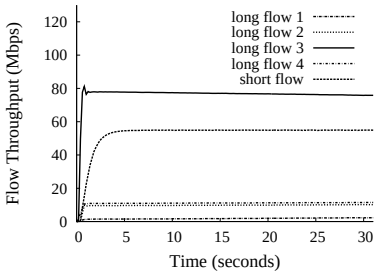


Fig. 2. **When there is no shuffling operation, XCP can achieve full link utilization but no fairness ensured.**

One might think that if the problem is caused by the bandwidth shuffling operation, we can simply disable bandwidth shuffling to fix the problem. However, this simple fix does not work. Bandwidth shuffling is essential to prevent starvation for new flows and to achieve some degree of fairness. The efficiency controller of XCP will not make any rate adjustment when the bandwidth is fully utilized. Without bandwidth shuffling, XCP can achieve a full link utilization, but the rate allocations to different flows can be arbitrarily unfair. When existing flows have used up a link's bandwidth, a new flow may be starved. As an example, if we disable bandwidth shuffling in Figure 1, four long flows and a short flow may obtain a bandwidth allocation as shown in Figure 2. Three long flows that start late are almost starved.

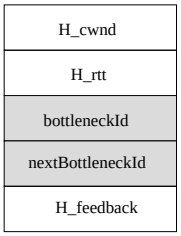


Fig. 3. **Congestion header for iXCP. We add two additional fields: the *bottleneckId* field and the *nextBottleneckId* field.**

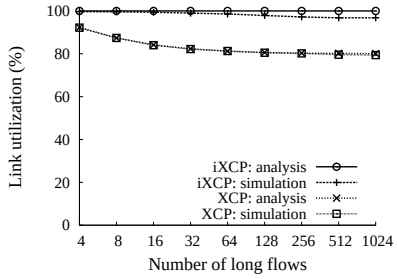
III. Improved XCP (iXCP)

In this section, we describe our improvement to XCP. As we discussed in the previous section, the under-utilization problem is caused by a router's attempt to shuffle bandwidth from flows that can further increase their rates to flows that cannot, i.e., flows that are bottlenecked at other routers. Therefore, we modify XCP's control algorithm to shuffle bandwidth only among flows that are bottlenecked at current routers. This modification ensures that the deallocated bandwidth by the shuffling operation will be re-used. Therefore, a link will be fully utilized.

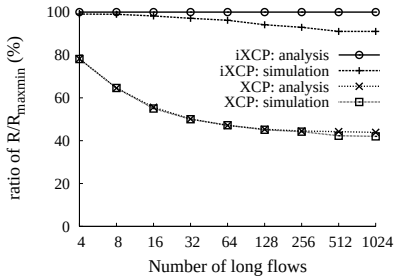
To implement this improvement, a router must be aware whether a flow is bottlenecked at the router or not. For a router to obtain this information, we include two additional fields in the XCP's congestion header: the *bottleneckId* and the *nextBottleneckId* field, as shown in Figure 3. The control algorithm uses the *bottleneckId* field to compute per-packet feedback. A router sets the *nextBottleneckId* field on the fly based on the feedback value. If the feedback from this router is smaller than the one in the packet header, then the outgoing link of this router becomes the new bottleneck for the flow, and the router writes its outgoing link identifier to this field. To ensure uniqueness, a router may use the IP address of an interface card or a random 32-bit value as a link identifier. Initially, both the *bottleneckId* and the *nextBottleneckId* fields are set to a sender's access link identifier. An iXCP receiver acknowledges back the *feedback* and the *nextBottleneckId* field to the sender. The sender copies the *nextBottleneckId* field from the acknowledgement to the *bottleneckId* field of its outgoing packets.

The control algorithm of XCP is modified as follows. On a packet arrival, a router estimates the spare bandwidth of an outgoing link as in the original XCP, and the router estimates the shuffled bandwidth only from those packets whose *bottleneckId*s match the link identifier of the router. On a packet departure, if the packet's *bottleneckId* matches the current outgoing link identifier of the router, the router follows the original XCP algorithm and computes the feedback from both the spare bandwidth and the shuffled bandwidth. Otherwise, the feedback is computed only from the spare bandwidth, but using the same algorithm as in XCP. Pseudo code of the algorithm is presented in Appendix.

The drawback of our modification is that iXCP increases the congestion header of XCP by eight bytes and the acknowledgement header by four bytes. If we assume the average packet size is 400 bytes [10], and each packet has both a con-

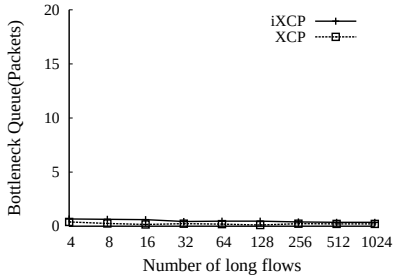


(a) Utilization for the 155Mb/s link in Figure 1.

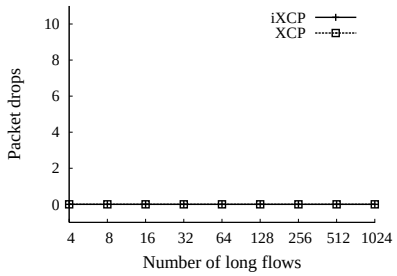


(b) Ratio of the short flow's rate (R) over its max-min fair rate (R_{maxmin}).

Fig. 4. **iXCP achieves nearly optimal link utilization and max-min fair rate for the short flow. The figures compare iXCP with XCP using both simulation and theoretical results.**



(a) Average queue size.



(b) Packet drops.

Fig. 5. **Small average queue size and zero packet drops are achieved for the 155Mb/s link in Figure 1.**

gestion header and an acknowledgement header, we increase the packet header overhead by 3%. However, as we will show in the following sections, iXCP can increase link utilization by almost 20% in some multi-bottleneck environments. Besides, packet sizes may increase in the future due to advanced technologies, e.g., gigabit Ethernet with a jumbo frame size of 9000 bytes. Therefore, we think it is a worthy tradeoff to design iXCP to achieve efficient and fair bandwidth allocation in both single- and multi-bottleneck topologies at the cost of slightly increased header overhead.

In Section V, we build on Low's theoretical model [7] to prove the following theorem:

THEOREM 1: iXCP achieves max-min fair bandwidth allocation in steady state. That is, with iXCP, all bottleneck links will be fully utilized; and a flow cannot increase its rate without decreasing the rate of a flow that has a smaller or equal share.

IV. Simulation Results

In this section, we use ns2 [11] simulations to evaluate the performance of iXCP and compare it with XCP.

The simulation scenarios include multiple bottleneck topologies, heterogeneous RTTs and web-like traffic. There are two key questions we aim to answer: 1) does iXCP fix the problem of XCP and achieve max-min fair allocation? 2) does iXCP make other properties of XCP worse? To answer the first question, we compute two metrics: 1) link utilization; 2) a flow's rate normalized by its theoretical max-min rate. If the link utilization is 100%, and a flow's normalized rate is 1, it shows that iXCP achieves max-min fair bandwidth allocation. To answer the second question, we examine three metrics: 1) the persistent queue sizes; 2) the packet drops; 3) the convergence speed when flows join and leave the network.

In our simulations, we use the same parameter settings as those chosen by XCP [5] for the purpose of comparison. The coefficients for the efficiency controller α and β are set to be 0.4 and 0.226 respectively, and the coefficient for the fairness controller γ is set to be 0.1. The packet size is 1000 bytes. The propagation delay of each link in all topologies is 20ms unless otherwise noted, and the bandwidth of each link is specified in the figures.

A. A simple two-bottleneck environment

We simulated the scenario as shown in Figure 1. The short flow is bottlenecked at the first 155Mb/s link, and all the other long flows are bottlenecked at the second 100Mb/s link. We vary the number of long flows from 4 to 1024. We only show the results for the short flow on the first link, because XCP cannot fully utilize the bandwidth of the first link and does not allocate the max-min rate to the short flow, as explained in Section II-B. The second link is fully utilized in both XCP and iXCP, and each long flow obtains its max-min rate of $100/n$ Mb/s.

Figure 4(a) and 4(b) show the link utilization for the first bottleneck link and the ratio between the short flow's rate and its theoretical max-min share. We show both the simulation and the theoretical results for XCP and iXCP. Theoretical results for XCP are obtained using Low's model [7]; theoretical results for iXCP are obtained using our analysis described in Section V. As can be seen, as the number of long flows increases, the link utilization of XCP decreases and approaches its theoretical lower bound 80%. The short flow only obtains 40% of its max-min rate. In contrast, iXCP achieves more than

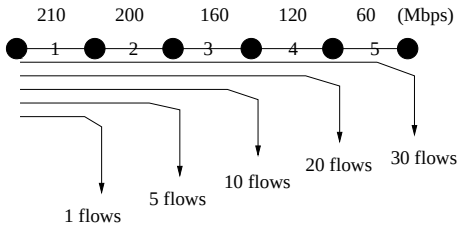


Fig. 6. **A multi-bottleneck topology.**

97% link utilization; the short flow's rate is more than 90% of its max-min rate.

Figure 5(a) and 5(b) show the average queue size and the packet drops at the 155Mb/s link. As can be seen, both XCP and iXCP have very small queue size, and negligible packet drops. (In the simulations, the packet drops are zero.)

The simulation results for iXCP match well with the theoretical analysis until the number of long flows becomes large. We examined packet traces and concluded that the discrepancy is caused by rounding errors. XCP's control algorithm computes a window adjustment value in bytes, but a sender advances its sending window in packets. When the window increment is less than one packet, the window size in the congestion header of a packet is actually larger than a sender's true sending window. This rounding error affects the calculation of per-packet feedback, stopping a flow from further increasing its window. When the number of flows is large, the aggregated rounding errors are not negligible and therefore leads to slight under-utilization.

B. A multi-bottleneck environment

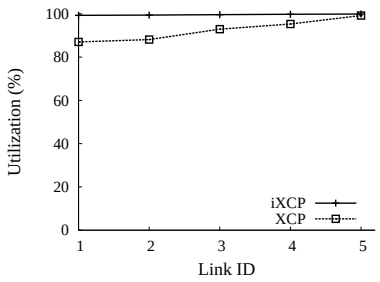
We studied how iXCP performs in a multi-bottleneck environment as shown in Fig 6. In this topology, there are a total of five hops and all flows are bottlenecked at the last hop they cross. A link is labeled with an identifier ranging from 1 to 5. For example, the thirty longest flows are bottlenecked at the fifth link; the twenty second longest flows are bottlenecked at the fourth link; and so on. The max-min rates for flows bottlenecked at link 1 to 5 are 10Mbps, 8Mbps, 4Mbps, 3Mbps, 2Mbps, respectively.

Figure 7(a) and 7(b) show the utilization and the normalized flow rate at each link achieved by both iXCP and XCP. We only show the bottlenecked flows for each link, and the normalized rates are averaged over all bottlenecked flows. The standard deviations among flows are too small to be visible. Thus, they are not shown in the figure. As can be seen, iXCP achieves full link utilization and max-min rate allocations for all flows at all bottleneck links, while XCP cannot fully utilize the bandwidth of the first four links.

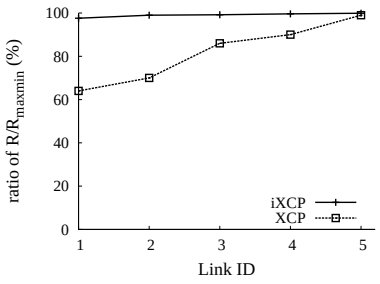
Figure 8(a) and 8(b) show the average queue size and packet drops for the multi-bottleneck topology. Both of them are small in XCP and iXCP.

C. Dynamic flows

We simulated the case in which the bottlenecks of flows change over time as flows dynamically join and leave the network. This simulation is to show how well iXCP converges when there are sudden traffic demand changes.

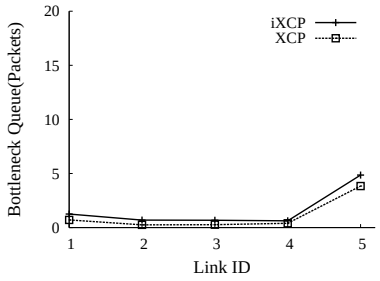


(a) Link utilization.

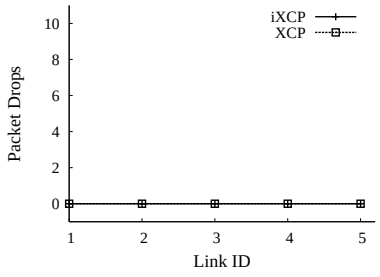


(b) Ratios of flow rates (R) over their max-min fair rates (R_{maxmin}).

Fig. 7. **iXCP achieves nearly 100% link utilization and max-min fair flow rates for each link in the multi-bottleneck topology shown in Figure 6.**



(a) Average queue size.



(b) Packet drops.

Fig. 8. **iXCP achieves small average queue size and zero packet drops for each link in the multi-bottleneck topology shown in Figure 6.**

The simulation topology is shown in Figure 10. Each link has round-trip propagation delay 20ms. We start the long flow, and four short flows from s_1 to s_4 at time $t = 0$. Each short flow only crosses one link. We then start three short flows s_5, s_6, and s_7 at $t = 20$. At time $t = 40$, the short flows from s_4 to s_7 stop.

Figure 11(b) shows the utilization of each link. As can be

seen, iXCP is robust to the sudden changes in traffic demand. After four flows exist the system, there is only a temporary dip, and the link is quickly fully utilized again.

Figure 12(a), 12(b), and 12(c) show how each flow's rate changes over time as its bottleneck shifts with traffic demand in iXCP. All flows converge to their max-min rates shortly after a change in the network. In the first 20 seconds, the long flow is bottlenecked at the 120Mb/s link. The max-min rates for the long flow is 40Mb/s; and the rates for the short flows are: s_1 = 110Mb/s, s_2 = 60Mb/s, and $s_3 = s_4$ = 40Mb/s. After the simulation starts, all flows converge to their max-min rates within ten seconds. When flows s_5 to s_7 start at $t = 20$, the bottleneck link for the long flow shifts to the 150Mb/s link. The flow's rate quickly converges to its new max-min rate 30Mb/s. Correspondingly, the short flows s_1, s_5, s_6 and s_7 converge to their max-min rates 30Mb/s, and s_2 increases to its new max-min rate 70Mb/s, and s_3 and s_4 increase to 45Mb/s. At $t = 40$, the short flows from s_4 to s_7 stop. The new bottleneck for the long flow is the 100Mb/s link. It quickly converges to its max-min rate 50Mb/s. Similarly, s_1 converges to its new max-min rate 100Mb/s, s_2 converges to 50Mb/s, and s_3 converges to 70Mb/s.

Figure 13(a), 13(b) and 13(c) show how XCP's flow rate changes as bottleneck shifts. Figure 14(a), 14(b) and 14(c) show TCP's flow rates on each link. TCP cannot achieve max-min fairness and its AIMD control algorithm leads to flow rates fluctuation. Additionally, fairness is an issue for TCP as flow's throughput is inversely proportional to its RTT, in the above topology, flows have different RTTs and considerable unfairness happens. For example, the long flow has the largest RTT and its flow is almost starved.

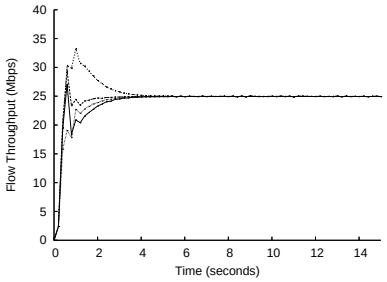
The convergence property of iXCP to equilibrium is similar to that of XCP. Both converge to equilibrium within seconds, whereas TCP persistently oscillates and never converges [12]. However, in our simulations, it takes iXCP slightly longer (about 1.6 seconds) to ramp up a flow's rate from zero to the equilibrium rate. This is because iXCP is fairer than XCP: it requires additional round trip times to shuffle bandwidth to fully achieve max-min fairness.

Another experiment we conducted is to test the convergence speed when iXCP and XCP converge to the same flow rates. We use the topology in Figure 1 and let four long flows passing both the 155Mb/s link and the 100Mb/s link without any short flows. The four long flows are all bottlenecked at the 100Mb/s link with max-min fair flow rate 25Mb/s each. Figure 9(a) and Figure 9(b) show the flow rates and convergence speed for iXCP and XCP respectively. There is no noticable convergence speed difference between those two schemes.

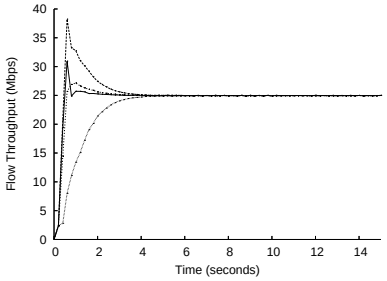
D. Varying RTTs

We study how RTT values may impact the performance of iXCP using the topology shown in Figure 1. There are one short flow bottlenecked at the 155Mb/s link and sixteen long flows bottlenecked at the 100Mb/s link.

In the first set of experiments, all flows have the same RTTs, and we vary the RTTs from 40ms to 1.4s. Figure 15(a), 15(b)



(a) iXCP flows.



(b) XCP flows.

Fig. 9. **iXCP and XCP exhibit similar convergence speed when their flows converge to the same max-min fair rates.**

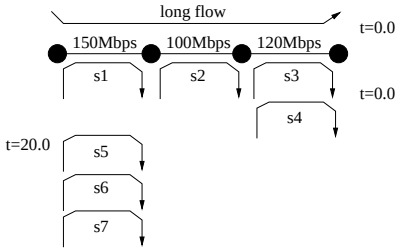
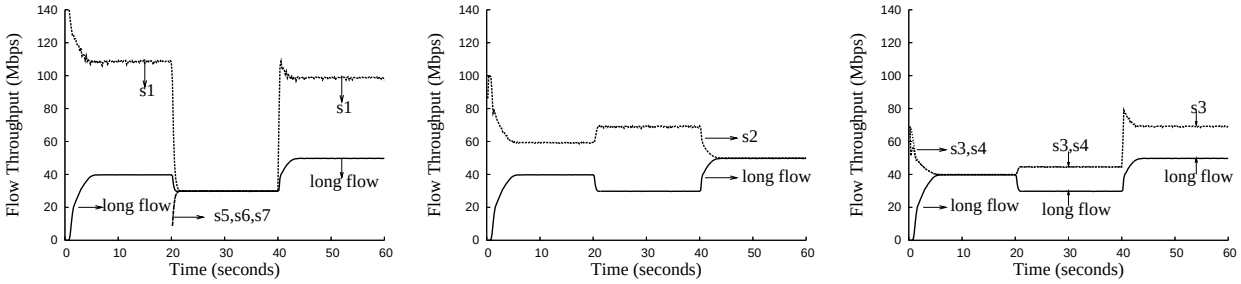


Fig. 10. **The long flow and four short flows s_1 to s_4 start at time $t = 0$. Three short flows s_5, s_6, and s_7 start at $t = 20$. At time $t = 40$, the short flows from s_4 to s_7 stop.**

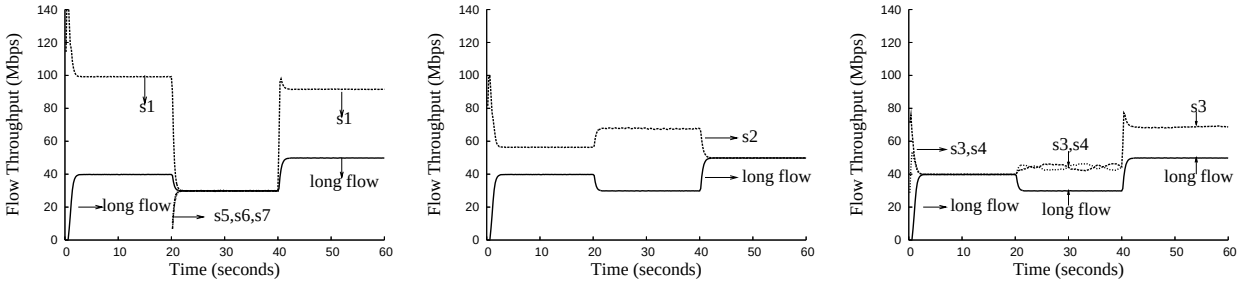
and 15(c) show that iXCP preserves the good properties of XCP and is robust to large RTT values. Both iXCP and XCP maintain high link utilization, low persistent queue size, and near-zero packet drops. This result is expected, because unlike TCP, both iXCP and XCP take into consideration the RTT value of a flow when adjusting a flow's congestion window. Each flow also achieves its max-min rate for all RTT values we simulated.

In the second set of experiments, flows have heterogeneous RTT values. The purpose is to study whether iXCP ensures fairness among flows with heterogeneous RTTs. In our experiments, the short flow has a 40ms RTT value. The other sixteen long flows have RTTs ranging from 60ms to 1.4s with an 85ms increment. Figure 16 shows the ratio between a flow's rate and its theoretical max-min rate. The flow with ID 0 is the short flow and the flows with IDs from 1 to 16 are the sixteen long flows. With iXCP, all flows achieve their max-min rates regardless of their RTT values. With XCP, the sixteen long flows achieve max-min fair rates; but the short flow does not obtain its max-min rate due to the under-utilization problem stated previously (Section II-B).



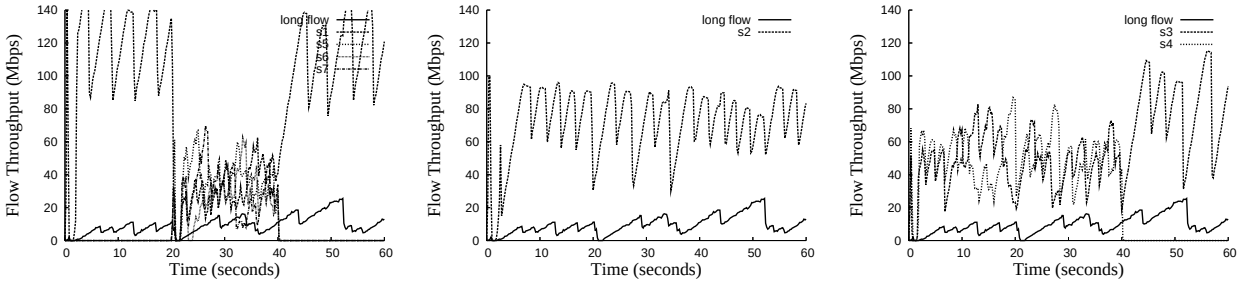
(a) Flow rates at the 150Mb/s link (b) Flow rates at the 100Mb/s link (c) Flow rates at the 120Mb/s link

Fig. 12. **These figures show how iXCP adapts to the flow dynamics of the network. The simulation topology is shown in Figure 10. Each flow can quickly converge to its max-min rate when other flows join or leave the network.**



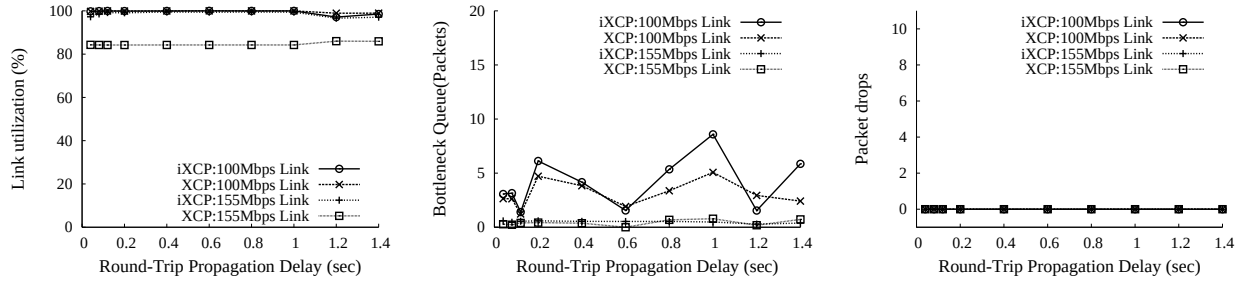
(a) Flow rates at the 150Mb/s link (b) Flow rates at the 100Mb/s link (c) Flow rates at the 120Mb/s link

Fig. 13. **These figures show how XCP adapts to the flow dynamics of the network. The simulation topology is shown in Figure 10. Each flow can quickly converge to its equilibrium rate when other flows join or leave the network. However, some flows' equilibrium rates in XCP are smaller than that in iXCP.**



(a) Flow rates at the 150Mb/s link (b) Flow rates at the 100Mb/s link (c) Flow rates at the 120Mb/s link

Fig. 14. **These figures show how TCP performs in dynamic case. The simulation topology is shown in Figure 10. TCP flows oscillates and never converges. TCP cannot achieve fairness among its flows.**

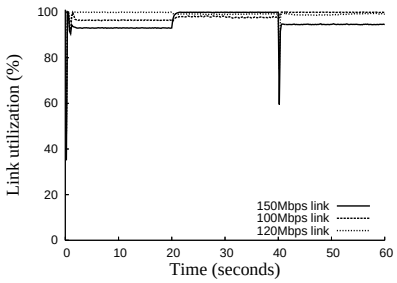


(a) Link utilization (b) Average queue size (c) Packet drops

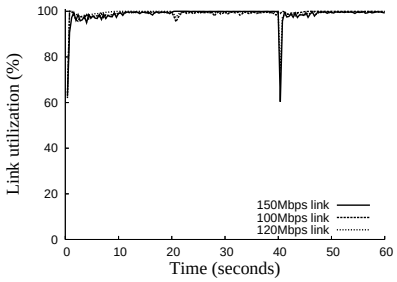
Fig. 15. **iXCP preserves the good property of XCP and is robust to large RTT values. It maintains high link utilization, low average queue size, and zero packet drops.**

E. Web-like Traffic

We study the impact of flow dynamics on the performance of iXCP using web-like traffic. We use the two-bottleneck topology as shown in Figure 1. The long-lived background traffic includes the short flow that crosses the 155Mb/s link and sixteen long flows that cross both the 155Mb/s link and



(a) Link utilization of XCP



(b) Link utilization of iXCP

Fig. 11. **XCP under-utilizes some links, while iXCP achieves nearly 100% link utilization for all the three links, as shown in Figure 10.**

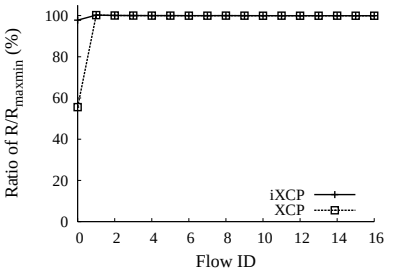


Fig. 16. **iXCP is max-min fair to all flows even in the presence of highly heterogeneous RTTs.**

the 100Mb/s link. The short-lived web-like traffic crosses both the 155Mb/s link and the 100Mb/s link. These flows arrive according to a Poisson process with arrival rates in the range of [10/sec, 1000/sec]. The transfer size of these flows is derived from a Pareto distribution with an average of 30 packets, which is consistent with real web traffic study [13].

Figure 17(a), 17(b) and 17(c) show the link utilization, average queue size and packet drops on the 155Mb/s link and the 100Mb/s link for iXCP and XCP.

As can be seen, both XCP and iXCP's performance are less than ideal in highly dynamic situations: link utilization decreases, queue size and packet drops increase. iXCP outperforms XCP, on the 155Mb/s link especially when the flow arrival rate is less than 400/sec. The performance degrades because web-like traffic exits the network before the flow allocation in the network has reached equilibrium. When a new flow joins the network, iXCP and XCP deallocate bandwidth from the existing flows to the new flow. Before the existing flows converge to their new rate allocations, bandwidth may be temporarily over-allocated, leading to congestion. When a flow departures from the network, the bandwidth allocated to the flow may be temporarily wasted before the network can assign it to other flows, causing link under-utilization. As the control algorithm of both iXCP and XCP assigns positive window increment inversely proportional to a flow's rate, a new flow with a small starting rate has a larger window increment than a flow that departures the network. Therefore, the over-allocation and the under-utilization effects can not exactly cancel out, even though the departure rates and the arrival rates of the web-like flows are the same. As shown in Figure 17(a), 17(b) and 17(c)), for the 100Mb/s link, when the arrival rates of the web-like flows are high ($>$ 400/sec), the over-allocation effect dominates. The link is fully utilized, but the queue size increases and packet drops occur. When the arrival rate is low ($\leq$ 400/sec), the departure effect dominates, and the link is under-utilized. For the 155Mb/s link, the link remains under-utilized when the arrival rates are high, because the over-allocation to the new web-like flows prevents the short flow from reaching its equilibrium rate.

The link utilization for iXCP will increase when the number of short flows increases, as this will reduce the over-allocation to the web-like flows. Figure 18(a), 18(b) and 18(c) show the simulation results when there are ten short flows passing the first 155Mb/s link. More performance improvements of iXCP over XCP are demonstrated when the arrival rates of web traffic are high.

We have conducted another experiment where each link has web traffic across it and only across it, i.e., at the 155Mb/s link, the short flow is mixed with a set of web traffic; at the 100Mb/s link, the sixteen long flows are mixed with another set of web traffic. In this scenario, both the short flow and long flows are greatly impacted by web traffic. Figure 18(a), 18(b) and 18(c) show the simulation results for this case. When arrival rate of web traffic is high, both links are dominated by the short-lived web traffic and over-allocation effect leads to high utilization, larger queue size and more packet drops.

It is our future work to further investigate how flow dynamics interact with the control algorithm of XCP and iXCP. Is there a lower bound on the link utilization in dynamic situations? Can the performance of iXCP be further improved?

F. Single-link topology

We use the single-link topology and the parking-lot topology as in [5] to perform sanity check for iXCP scheme. The simultion scenarios cover those with different link capacity, different round-trip delay, dynamic case with new traffic joining and the case with web traffic introduced. The performance metrics we are interested are link utilization, average queue size, packet drop and flow rate.

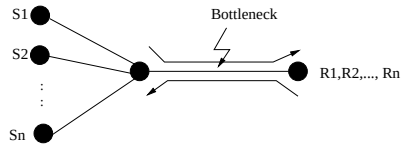
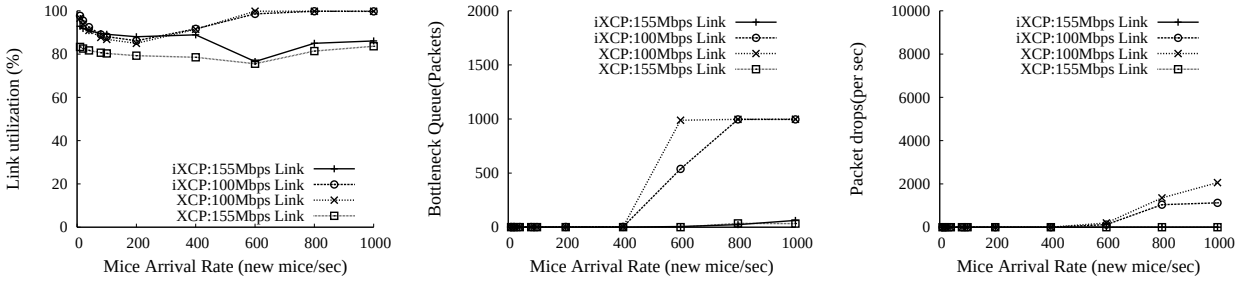
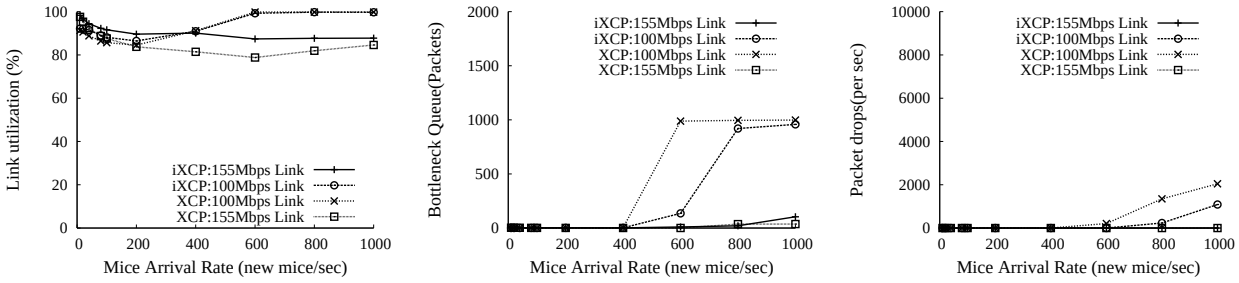


Fig. 20. A single bottleneck topology



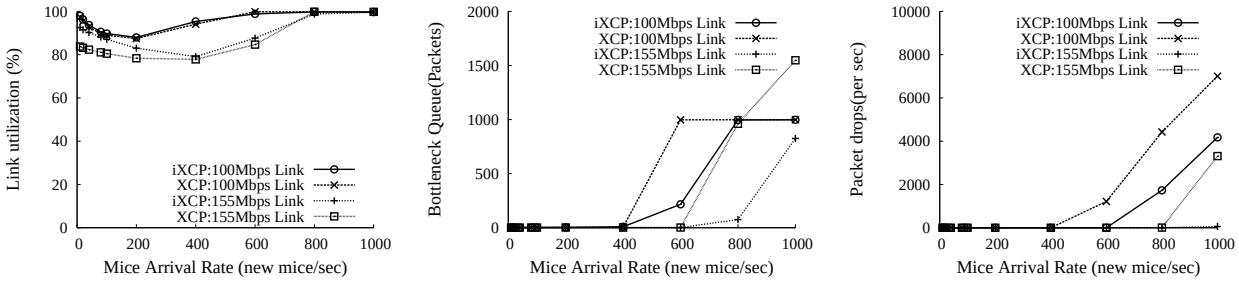
(a) Link utilization (b) Average queue size (c) Packet drops

Fig. 17. **iXCP outperforms XCP in a highly dynamic environment with the arrivals and departures of web-like traffic.**



(a) Link utilization (b) Average queue size (c) Packet drops

Fig. 18. **iXCP achieves more improvment over XCP if we increase the number of short flows to 10 at the 155Mb/s link, in a highly dynamic environment with the arrivals and departures of web traffic.**



(a) Link utilization (b) Average queue size (c) Packet drops

Fig. 19. **In this simulation, each link has web traffic traverse it. Thus, the single short flow is mixed with web traffic at the 155Mb/s link and the sixteen long flows are mixed with another set of web traffic at the 100Mb/s link. As web traffic dominates both links when arrival rate of web traffic is high, the 155Mb/s link utilization approaches to that of the 100Mb/s link.**

In this topology, 50 flows traverse the single link in the forward direction and 50 flows traverse in the reverse direction to stress iXCP and XCP.

1) Impact of bandwidth: single link bandwidth is increased from 50Mbps to 2Gbps. Link propagation delay is 80ms. Figure 21(a), Figure 21(b) and Figure 21(c) show that both iXCP and XCP are efficient with link capacity increases and can achieve high utilization, low queue size and zero packet drop.

2) Impact of RTT: We fix the link bandwidth to be 150Mbps and vary the link propagation delay to make flows have different RTTs. Figure 22(a), Figure 22(b) and Figure 22(c) show that both iXCP and XCP can achieve high utilization, low queue size and zero packet drop, robust to round-trip time increases.

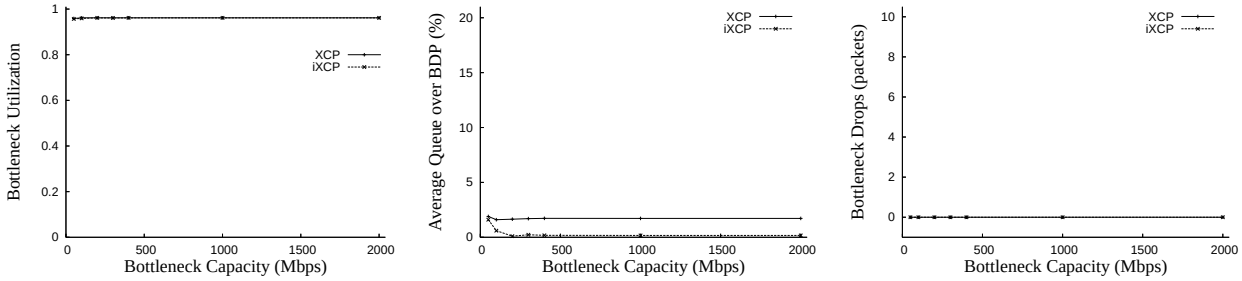
3) Convergence Dynamics: In this experiment, link capacity is set to be 45Mbps and RTT is 40ms. Five long-lived flows share the single link and flows start at different times, 0, 2, 4,6 8 seconds.

Figure 23(a) shows that whenever there is a new flow joining, both XCP and iXCP can reallocate a new max-min fair rate to each flow. Figure 23(b) and Figure 23(c) show that during convergence, utilization is not disturbed; queue can accomodate traffic burst and drain afterwards.

4) Impact of web traffic: We introduce web-like traffic into our simulations to study the impact of flow dynamics on the performance of iXCP and XCP in single-link topology. The web traffic arrives according to a Poisson process with arrival rates in the range of [10/sec, 1000/sec]. The transfer size of these flows is derived from Pareto distribution with an average of 30 packets, which is consistent with real web traffic study [13].

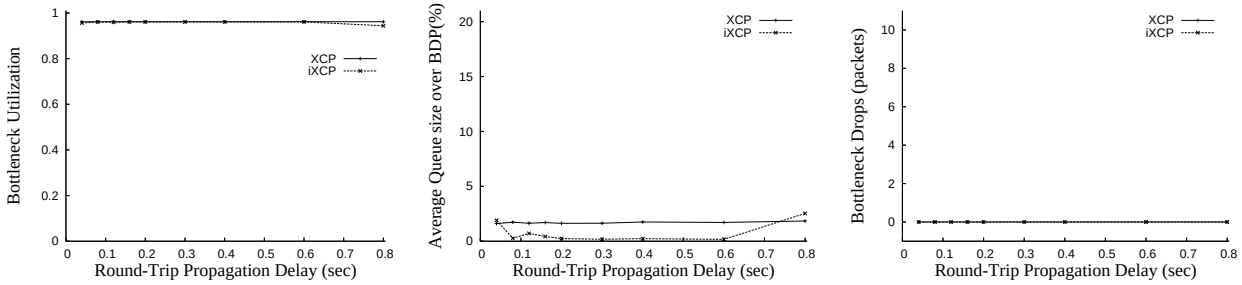
Figure 24(a), 24(b) and 24(c) show the link utilization, average queue size and packet drops on the single link for iXCP and XCP.

The utilization for the single link is similar in both iXCP



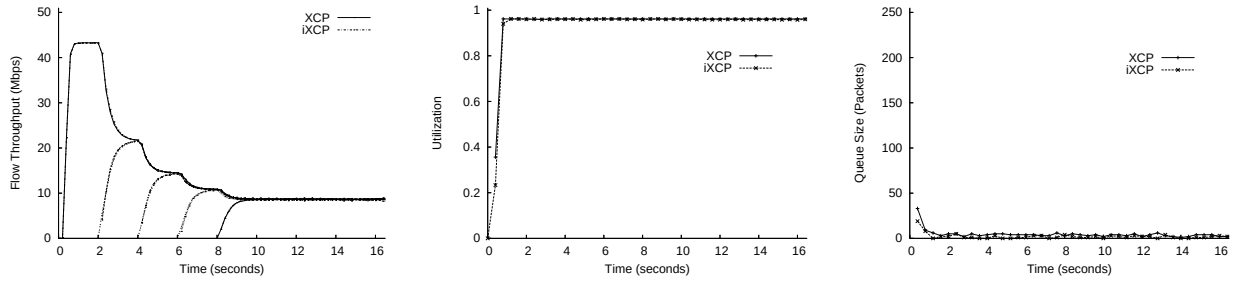
(a) Link utilization (b) Average queue size (c) Packet drops

Fig. 21. **Both iXCP and XCP have high link utilization, low persistent queue size and zero drop at single-link with different link bandwidth.**



(a) Link utilization (b) Average queue size (c) Packet drops

Fig. 22. **Both iXCP and XCP have high link utilization, low persistent queue size and zero drop at single-link with different RTTs.**



(a) Flow rate (b) Link utilization (c) Average queue size

Fig. 23. **Both iXCP and XCP smoothly converge to high fairness, good utilizationa and small queue size. Five flows start at times 0,2,4,6,8 respectively.**

and XCP. This is because our modification to XCP only affects the rate allocation when link under-utilization problem exists. In a single-link topology, we do not have this problem.

G. Parking-lot Topology

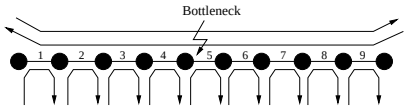


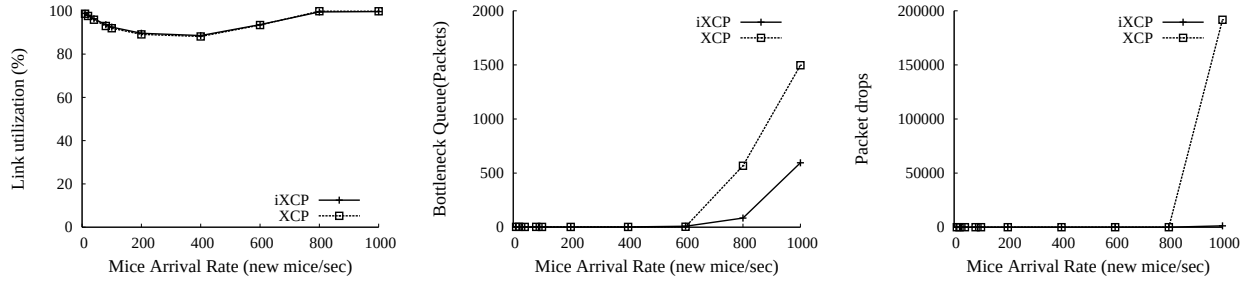
Fig. 25. A parking lot topology

In the parking lot topology, there are total nine links. Each link has capacity 100Mb/s, except link 5 has 50Mb/s. There are 50 long flows traversing all the links in forward direction and 50 long flows in the reverse direction. At each link, there are 50 flows across only that link. Thus, long flows are all bottlenecked at link 5. As we explained before, in this multi-bottleneck topology, link under-utilization problem exists for each link. For the first four links, long flows are bottlenecked at downstream links, however, upstream links still attempt to increase long flows' rates but they cannot increase any further. Thus, iXCP can increase long flows' rates and iXCP outperforms XCP in terms of link utilization (reverse traffic stresses the schemes and makes it not achieve 100% utilization). For the last four hops, long flows are bottlenecked at upstream link, in XCP's implementation, residual terms help to save some bandwidth for the bottleneck flows, but it does not solve the link under-utilization problem and cannot achieve 100% utilization.

H. Comparison of XCP, iXCP and P-XCP

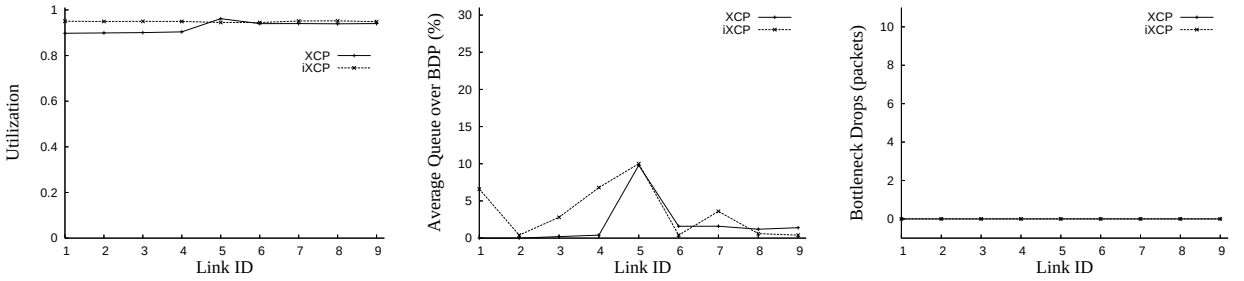
Zhou et al. also forsaw XCP's problem and proposed P-XCP [9] to improve XCP's under-utilization problem.

With P-XCP, a router estimates the number of bottlenecked flows by comparing the feedback in a packet header with the one it computes. If the computed feedback is smaller, then the router considers the flow as bottlenecked at itself. The router allocates more bandwidth to those flows that are bottlenecked



(a) Link utilization (b) Average queue size (c) Packet drops

Fig. 24. **Both iXCP and XCP are robust and efficient in highly dynamic environments with arrivals and departures of web traffic.**



(a) Link utilization (b) Average queue size (c) Packet drops

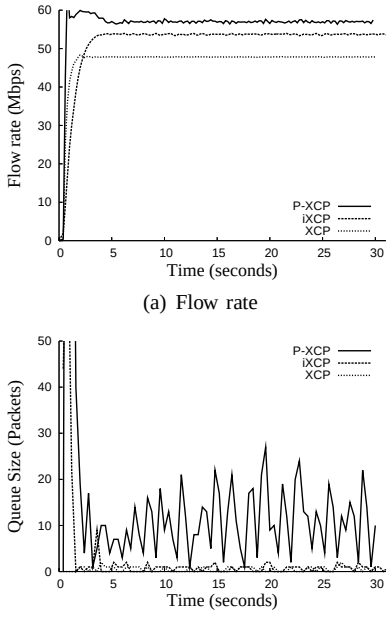
Fig. 26. **Simultion of iXCP and XCP in multi-bottleneck parking-lot topology.**

at itself by scaling the spare bandwidth with a factor N/N_b, in which N is the total number of flows, and N_b is the flows that are bottlenecked at the router. This scheme only increases link utilization when a bottleneck link is upstream to an under-utilized link. For instance, it does not increase link utilization in the example shown in Figure 1. Moreover, as P-XCP over-allocates spare bandwidth, it causes rate fluctuations and increases the persistent queue size.

As P-XCP only works for the cases that a bottleneck link is upstream to an under-utilized link, we reverse the topology in Figure 1 and let four long flows pass the 100Mb/s link first and then pass the 155Mb/s link. Thus, the long flows are bottlenecked at the 100Mb/s link, which is upstream to the under-utilized 155Mb/s link. The max-min fair rate for the short flow is 55Mb/s. Figure 27(a) and Figure **??** show the short flow rate and instantaneous queue length by P-XCP, iXCP and XCP. From the simulation results, we can see that P-XCP allocates the highest rate to the short flow which is about 57.4Mb/s and higher than its max-min fair rate. As a cost, P-XCP has the largest persistent queue length. In contrast, iXCP can make short-flow approach to its max-min fair rate with very small queue length, while XCP makes short flow not achieve its max-min fair rate and makes the 155Mb/s link under-utilized.

I. Summary

In all our simulations except the simulations for web traffic, iXCP achieved a near optimal bandwidth allocation: 100% link utilization with max-min fairness, as predicated by our theoretical analysis (Section V). iXCP also preserves all the good properties of the original XCP, including small persistent queue sizes, near-zero packet drops, and fast convergence. In



(a) Flow rate

(b) Queue size

Fig. 27. **Comparing to iXCP and XCP, P-XCP can overshoot link bandwidth and lead to rate fluctuation and queue size increase.**

a highly dynamic environment where a large number of short-lived flows depart and arrive in the network, flows cannot reach their equilibrium rates and thus iXCP's improvement degrades, but iXCP still outperforms XCP in terms of link utilization.

V. Theoretical Model of iXCP

In this section, we build on Low's theoretical model [7] to prove that iXCP achieves full link utilization and max-min fairness (Theorem 1 in Section III). As iXCP differs from

XCP in its shuffling operation, we only show the derivation for feedback computed from the shuffling operation in detail, and use Low's result for the spare bandwidth analysis.

The high-level idea of our analysis is as follows. We first convert the per-packet feedback computed by iXCP's control algorithm to a per-flow feedback, and then derive a flow's rate at an equilibrium from the feedback, using the condition that in an equilibrium, the feedback for window adjustment is zero. We then show that the equilibrium rate is the max-min rate. Our analysis assumes that a flow has a fixed packet size.

We first define the following notations:

- L: the set of links in the network.
- N: the number of flows in the network.
- i: the index of a flow.
- l: the index of a link.
- k: the index of a packet.
- R: the $L \times N$ routing matrix. $R_{li} = 1$ if flow i uses link l, and 0 otherwise.
- $L(i)$: the set of links in the path of each flow i. $L(i) = \{l|R_{li} = 1\}$.
- $I(l)$: the set of flows that cross the link l: $I(l) = \{i|R_{li} = 1\}$.
- $I_1(l)$: the set of flows that cross the link l and are bottlenecked at link l.

For each flow i, we define the following variables:

- $w_i(t)$: the congestion window size at time t (in packets).
- $T_i(t)$: the round-trip time (RTT) at time t.
- $x_i(t)$: the flow i's sending rate in unit of packets per second, and $x_i(t) = w_i(t)/T_i(t)$.

As each packet also carries a window size and a RTT in its congestion header, we use k as the index for a packet to indicate the value carried in a packet. For instance, $w_{i,k}(t)$ is the window size carried by packet k that belongs to flow i.

For each link l, we define the following variables:

- c_l: the capacity (in packets per second).
- $b_l(t)$: the queue size at time t (in packets).
- $y_l(t)$: the aggregate input traffic rate at time t, and $y_l(t) = \sum_i R_{li}x_i(t)$.
- $y_{l0}(t)$: the aggregate rate for flows crossing link l but not bottlenecked at link l at time t.
- N_l: the number of flows crossing link l. $N_l = |I(l)|$.
- N_{l0}: the number of flows crossing link l but not bottlenecked at link l. $N_{l0} = |I(l)| - |I_1(l)|$
- $\phi_l(t)$: the spare bandwidth computed by iXCP's control algorithm.
- $h_l(t)$: the shuffled bandwidth computed by iXCP's control algorithm.
- $K_l(t)$: the number of packets from flows that are bottlenecked at link l during the control interval that proceeds time t.
- $P_{l,i}(t)$: the total number of packets from flow i during the control interval that proceeds time t.
- r_l: the equilibrium rate allocated to a flow at the bottleneck link l.

Next, we define the control parameters of iXCP and the feedback computed by iXCP. The control parameters are the same as defined in XCP [5].

- α, β: the damping coefficients used to compute the spare bandwidth.
- γ: the coefficient used to compute the shuffled bandwidth.
- d: the control interval in unit of seconds.
- p: the per-packet positive feedback. The variable p may have four subscripts: 1) l: the index of the link this positive feedback is computed from; 2) i: the index of the flow a packet belongs to; 3) k: the index of the packet for which the feedback is computed; 4) ϕ or h: indicating whether the feedback is computed from the spare bandwidth or the shuffled bandwidth.
- n: the per-packet negative feedback. The subscript definitions for n are the same as those for p.
- H: the overall per-packet feedback, and $H = p - n$. The subscripts for H are the same as those for p and n.

The spare bandwidth and shuffled bandwidth at link l are defined by the following equations:

$$\begin{aligned} \phi_l(t) &= \alpha d(c_l - y_l(t)) - \beta b_l(t) \\ h_l(t) &= \max\{0, \gamma(y_l(t) - y_{l0}(t)) - |\phi_l(t)|\} \end{aligned}$$

From iXCP's control algorithm, we can represent the per-packet positive feedback and negative feedback computed from the shuffled bandwidth on packet k as follows:

$$p_{l,i,k,h}(t) = h_l(t)\frac{T_{i,k}(t)}{d}\frac{T_{i,k}(t)/w_{i,k}(t)}{\sum_{u\in I_1(l)}\sum_{v\in P_{l,u}(t)} T_{u,v}(t)/w_{u,v}(t)} \quad (3)$$

$$n_{l,i,k,h}(t) = h_l(t)\frac{T_{i,k}(t)}{d}\frac{1}{K_l(t)} \quad (4)$$

We note that the feedbacks computed for packets from the same flow are the same. We then convert the per-packet feedback into a per-flow feedback. Let the flow rate computed from a packet be $x_{i,k}(t) = w_{i,k}(t)/T_{i,k}(t)$ and replace it in (3), we obtain:

$$p_{l,i,k,h}(t) = h_l(t)\frac{T_{i,k}(t)}{d}\frac{1/x_{i,k}(t)}{\sum_{u\in I_1(l)}\sum_{v\in P_{l,u}(t)} 1/x_{u,v}(t)} \quad (5)$$

The number of packets from a flow received within a control interval equals to the rate of the flow times the length of the control interval: $P_{l,i}(t) = R_{li}x_i(t)d$, and $x_{i,k} = x_i$. We have:

$$\begin{aligned} \sum_{u\in I_1(l)}\sum_{v\in P_{l,u}(t)} 1/x_{u,v}(t) &= \sum_{u\in I_1(l)} R_{lu}x_u(t)d\frac{1}{x_u(t)} \\ &= (N_l - N_{l0})d \end{aligned} \quad (6)$$

By substituting (6) in (5), we obtain:

$$p_{l,i,k,h}(t) = \frac{T_{i,k}(t)}{d^2}\frac{h_l(t)}{(N_l - N_{l0})x_{i,k}}$$

The number of packets from flows that are bottlenecked at the link l equals to the aggregate rates from those flows times the control interval:

$$K_l(t) = (y_l(t) - y_{l0}(t))d \quad (7)$$

By plugging (7) into (4), we obtain the negative feedback:

$$n_{l,i,k,h}(t) = \frac{T_{i,k}(t)}{d^2}\frac{h_l(t)}{y_l(t) - y_{l0}(t)} \quad (8)$$

Therefore, the net feedback from the shuffled bandwidth is:

$$H_{l,i,k,h}(t) \quad = \quad \frac{T_{i,k}(t)}{d^2}(\frac{h_l}{(N_l - N_{l0})x_{i,k}(t)} - \frac{h_l}{y_l(t) - y_{l0}(t)})$$

From Low's model [7], the feedback from the spare bandwidth is computed as:

$$H_{l,i,k,\phi}(t) = \frac{T_{i,k}(t)}{d^2}(\frac{\phi_l^+}{N_l x_{i,k}(t)} - \frac{\phi_l^-}{y_l(t)})$$

where $\phi_l^+(t) = max(\phi_l(t), 0)$ and $\phi_l^-(t) = max(-\phi_l(t), 0)$.

Putting it all together, the feedback for packet k from both the spare bandwidth and the shuffled bandwidth is:

$$H_{l,i,k}(t) = H_{l,i,k,\phi}(t) + H_{l,i,k,h}(t)$$

Since flow i receives $x_i(t)$ feedback packets per unit time, its window evolves according to the minimum feedback from its packets:

$$\Delta w_i(t) \quad = \quad x_i(t) \min_{l \in L(i)} H_{l,i,k}(t)$$

In an equilibrium, a flow's window will stop changing [7], $\Delta w_i(t) = 0$, which leads to $\min_{l \in L(i)} H_{l,i,k}(t) = 0$. Note that the flow's rate and feedbacks will not change over time in an equilibrium. So we can drop the time variable t, and the packet index k. Replacing the terms in $H_{l,i,k}$, we have:

$$\min_{l \in L(i)} (\frac{\phi_l^+}{N_l x_i} - \frac{\phi_l^-}{y_l} + \frac{h_l}{(N_l - N_{l0})x_i} - \frac{h_l}{y_l - y_{l0}}) = 0$$

We obtain this condition for all flows i at all links the flow crosses, i.e., $l \in L(i)$. Therefore, we have:

$$x_i \leq \frac{y_l(y_l - y_{l0})(\phi_l^+(N_l - N_{l0}) + h_l N_l)}{N_l(N_l - N_{l0})(\phi_l^-(y_l - y_{l0}) + h_l y_l)} \tag{9}$$

The equality holds for flows that are bottlenecked at link l. Note that the right side of (9) does not depend on the flow index i. So all flows that are bottlenecked at a link l must have the same rate r_l. That is:

$$x_i = r_l, \forall i \in I_1(l)$$

On the other hand, from the definitions of y_l, y_{l0}, N_l, and N_{l0}, we can compute the common rate r_l as

$$r_l = \frac{y_l - y_{l0}}{N_l - N_{l0}}$$

Moreover, only one of ϕ_l^+ and ϕ_l^- can be non-zero. Here we ssume ϕ_l^+ is non-zero (similar proof can be shown when ϕ_l^- is non-zero). Thus, $\phi_l^+ = \phi_l$. We have:

$$\frac{y_l - y_{l0}}{N_l - N_{l0}} \frac{\phi_l(N_l - N_{l0}) + h_l N_l}{h_l N_l} = \frac{y_l - y_{l0}}{N_l - N_{l0}} \tag{10}$$

The above equality leads to $\phi_l = 0$, which shows that the link utilization is 100%, and each bottlenecked flow obtains a rate $\frac{c_l - y_{l0}}{N_l - N_{l0}}$. These rates can be shown to be max-min fair following the proof in [8]. □

VI. Related work

Low et al. simulated and analyzed XCP's under-utilization problem [7]. This work is inspired by their discovery. Our analysis is built on Low's model. Zhou et al. also foresaw this problem and proposed P-XCP [9]. However, P-XCP does not address the root cause of the under-utilization problem, and only alleviates the problem in some topologies. With P-XCP, a router estimates the number of bottlenecked flows by comparing the feedback in a packet header with the one it computes. If the computed feedback is smaller, then the router considers the flow as bottlenecked at itself. The router allocates more bandwidth to those flows that are bottlenecked at itself by scaling the spare bandwidth with a factor N/N_b, in which N is the total number of flows, and N_b is the flows that are bottlenecked at the router. This scheme only increases link utilization when a bottleneck link is upstream to an under-utilized link. For instance, it does not increase link utilization in the example shown in Figure 1. Moreover, as P-XCP over-allocates spare bandwidth, it causes rate fluctuations and increases the persistent queue size. Comparisons between P-XCP and iXCP are shown in [12].

To the best of our knowledge, our work is the first that systematically addresses the under-utilization problem of XCP and to prove that the improved XCP is max-min fair in all types of topologies in steady state.

JetMax [14] is a rate-based congestion control algorithm that aims to provide max-min fairness. Similar to our scheme, a Jetmax packet header includes a bottleneck identifier field, but its control algorithm is rate-based, completely different from XCP, which is a window-based protocol. Charny et al. [15] also proposes a rate-based algorithm to realize max-min flow rate allocation, using a different control algorithm and feedback scheme. However, their approach requires per-flow state at routers. In contrast, both XCP and iXCP are stateless.

Other work has focused on the implementation of XCP [16] and improvements of XCP in other areas. Zhang et al. [17] extensively studied the implementation and deployment challenges of XCP. Hsiao et al. [18] extended XCP to support streaming layered video. Kapoor et al. [19] proposes to combine XCP with a Performance Enhancement Proxy (PEP) to provide fast satellite access. Yang et al. [20] proposed an improvement to shorten XCP's response time for new flows to acquire their bandwidth. Zhang et al. [21] presented a control theoretical model that considers capacity estimation errors. XCP-r [22] proposes to calculate the congestion window size at the receiver side to cope with ACK losses.

VII. Conclusions and Future Work

XCP [5] is a new and promising protocol that outperforms TCP in terms efficiency, stability, queue size, and convergence speed. However, Low et al. [7] discovered a weakness of XCP. In some multi-bottleneck environments, XCP may only utilize as low as 80% of bottleneck bandwidth.

This paper proposes an improved XCP (iXCP) that solves the link under-utilization problem of XCP. We use extensive

simulations as well as a theoretical analysis to show that iXCP is able to efficiently utilize bottleneck bandwidth and is max-min fair in steady state. iXCP also preserves other features of XCP, including small queue size, near-zero packet drops, and fast convergence.

The performance of both iXCP and XCP degrades in highly dynamic situations. We have analyzed the cause of this performance degradation, but it is our future work to further investigate the interactions between flow dynamics and iXCP's control algorithm and to propose improvement that makes the control algorithm more robust to flow dynamics.

REFERENCES

[1] C. Jin, D. Wei, and S. Low, "Fast TCP: Motivation, Architecture, Algorithms, Performance," in *Proc. of Infocom*, March 2004.

[2] S. Floyd, "HighSpeed TCP for Large Congestion Windows," in *IETF RFC 3649*, Dec. 2003.

[3] T. Kelly, "Scalable TCP: Improving Performance in HighSpeed Wide Area Networks," in *1st International Workshop on PFLDN*, Feb. 2003.

[4] L. Xu, K. Harfoush, and I. Rhee, "Binary increase congestion control for fast long-distance networks," in *Proc. of Infocom*, Mar. 2004.

[5] D. Katabi, M. Handley, and C. Rohrs, "Congestion Control for High Bandwidth-Delay Product Networks," in *Proc. of Sigcomm*, 2002.

[6] D. Clark, K. Sollins, J. Wroclawski, D. Katabi, J. Kulik, X. Yang, R. Braden, T. Faber, A. Falk, V. Pingali, M. Handley, and N. Chiappa, "New Arch: Future Generation Internet Architecture," Tech. Rep., USC ISI, 2003.

[7] S. Low, L. Andrew, and B. Wydrowski, "Understanding XCP: Equilibrium and Fairness," in *Proc. of Infocom*, 2005, pp. 1025–1036.

[8] D. Bertsekas and R. Gallager, "Data Networks," in *Prentics-Hall Inc., 2nd ed.*, 1992.

[9] K. Zhou, K. Yeung, and V. Li, "P-XCP: A Transport Layer Protocol for Satellite IP Networks," in *Proc. of GLOBECOM*, 2004, pp. 2707–2711.

[10] "Caida report," `http://www.caida.org/analysis/AIX/plen_hist/`, February 2000.

[11] "The network simulation: ns2," `http://www.isi.edu/nsnam/ns`.

[12] Lei Zan and Xiaowei Yang, "Improving XCP to Achieve Max-Min Fair Bandwidth Allocation," Tech. Rep., UCI ICS, 2006, `http://www.ics.uci.edu/~lzan/ixcp_techreport.pdf`.

[13] M. Crovella and A. Bestavros, "Self-Similarity in World Wide Web Traffic: Evidence and Possible Causes," in *Proc. of IEEE/ACM Trans. Networking*, 1997.

[14] Y. Zhang, D. Leonard, and D. Loguinov, "JetMax: Scalable Maxmin Congestion Control for High-Speed Heterogeneous Networks," in *Proc. of Infocom*, 2006.

[15] A. Charny, D. Clark, and R. Jain, "Congestion Control With Explicit Rate Indication," in *Proc. of ICC*, 1995.

[16] A. Falk and D. Katabi, "Specification for the Explicit Control Protocol (XCP)," 2005.

[17] Y. Zhang and T. Henderson, "An Implementation and Experimental Study of the eXplicit Control Protocol (XCP)," in *Proc. of Infocom*, 2005, pp. 1037–1048.

[18] H. Hsiao and J. Hwang, "A Max-Min Fairness Congestion Control for Streaming Layered Video," in *Proc. of ICASSP*, 2004, pp. V981– V984.

[19] A. Kapoor, A. Falk, T. Faber, and Y. Pryadkin, "Achieving faster access to satellite link bandwidth," in *Proc. of Infocom*, 2005, pp. 2870–2875.

[20] Y. Yang, C. Chan, P. Wang, and Y. Chen, "Performance Enhancement of XCP for High Bandwidth-Delay Product Networks," in *Proc. of ICACT*, 2005, pp. 456–461.

[21] Y. Zhang and M. Ahmed, "A Control Theoretic Analysis of XCP," in *Proc. of Infocom*, 2005, pp. 2831–2835.

[22] D.M. Lopez-Pacheco and C. Pham, "Robust Transport Protocol for Dynamic High-Speed Networks: Enhancing the XCP Approach," in *Proc. of IEEE MICC-ICON*, Nov. 2005.

APPENDIX
PSEUDOCODE OF iXCP

There are three blocks of code in the implementation of an iXCP router. The first block is executed upon packet arrival and the following estimates are updated by the router. In iXCP, packets from bottleneck flows are involved in shuffle operation and are recorded separately in order to compute estimate coefficients for shuffle bandwidth.

On each packet arrival:

```
input_traffic += pkt_size
sum_rtt_by_cwnd += H_rtt×pkt_size/H_cwnd
sum_rtt_square_by_cwnd += H_rtt×H_rtt×pkt_size/H_cwnd
If(bottleneckId = current RouterID) then
   input_traffic_shuffle += pkt_size
   sum_rtt_by_cwnd_shuffle += H_rtt × pkt_size/H_cwnd
```

The second block of code is run upon expiration of estimation-control timer. It involves computing control variables, reinitializing estimation parameters. Control variables are computed and updated from spare bandwidth and shuffle bandwidth separately.

On estimation-control timeout:

```
avg_rtt=sum_rtt_square_by_cwnd/sum_rtt_by_cwnd
φ=α×avg_rtt×(capacity-input_traffic)-β×Queue
shuffle_traffic = 0.1×input_traffic_shuffle-|φ|
ξ_p_spare = max(φ,0)/(avg_rtt×sum_rtt_by_cwnd)
ξ_n_spare = max(−φ,0)/(avg_rtt×input_traffic)
ξ_p_shuffle = shuffle_traffic/(avg_rtt×sum_rtt_by_cwnd_shuffle)
ξ_n_shuffle = shuffle_traffic/(avg_rtt×input_traffic_shuffle)
input_traffic = input_traffic_shuffle = 0
sum_rtt_by_cwnd = sum_rtt_by_cwnd_shuffle = 0
```

The third block of code is to compute feedback for each packet and update its congestion header fields whenever necessary. If current router has smaller feedback, it overwrites both the feedback and nextBottleneckId fields in packet congestion header. On each packet departure:

```
If(bottleneckId = current RouterID) then
   pos_fbk = (ξ_p_spare+ξ_p_shuffle)×H_rtt×H_rtt×pkt_size/H_cwnd;
   neg_fbk = (ξ_n_spare+ξ_n_shuffle)×H_rtt×pkt_size;
else
   pos_fbk = ξ_p_spare×H_rtt×H_rtt×pkt_size/H_cwnd;
   neg_fbk = ξ_n_spare×H_rtt×pkt_size;
feedback = pos_fbk - neg_fbk
if(H_feedback≤feedback) then
   H_feedback = feedback
   H_nextBottleneckId = current RouterID
ξ_p_spare = ξ_n_spare = 0
ξ_p_shuffle = ξ_n_shuffle = 0
```