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.

1 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 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 2, we briefly summarize how XCP works and its weakness. Section 3 describes iXCP. We use extensive

[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.

simulations to evaluate the performance of iXCP in Section 4. We compare our work with related work in Section 5. Section 6 concludes the paper.

2 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 underutilize link bandwidth.

2.1 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 computed by XCP is as follows:

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

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.

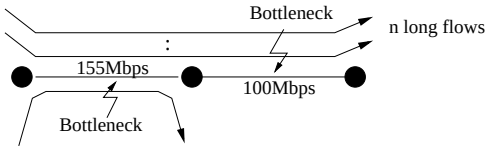


Fig. 1. In this topology, XCP under-utilizes the 155Mb/s link. The n long flows are bottlenecked at the 100Mb/s link. So the short flow should be able to send at 55Mb/s, but with XCP, it may get a bandwidth allocation anywhere between 24Mb/s to 55Mb/s, depending on the number of long flows

2.2 The Weakness Revealed

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 2(a) and 2(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.

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, bandwidth shuffling is essential to prevent starvation for new flows and to achieve some degree of fairness. Without bandwidth shuffling, XCP can achieve full link utilization, but the rate allocations to different flows can be arbitrarily unfair (See [10] for a concrete example).

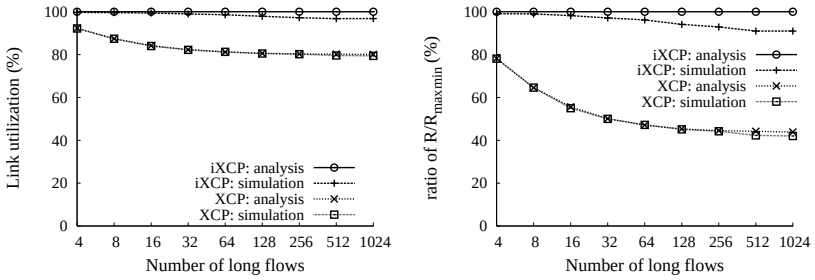
3 Improved XCP (iXCP)

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. 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 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 shown in Equation (1), and the router estimates the shuffled bandwidth only from those packets whose *bottleneckId*s match the link identifier of the router. In iXCP, the shuffled bandwidth can be represented as follows:

$$h_{ixcp} = \max\{0, \gamma(y - y_0) - |\phi|\} \tag{3}$$



(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. 2. 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

As y_0 denotes the input traffic rate for flows whose *bottleneckId*s do not match router's link identifier, $y - y_0$ denotes the input rate for flows bottlenecked at current router whose *bottleneckId*s match router's link identifier.

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 computed from Equation (1) and the shuffled bandwidth computed from Equation (3). Otherwise, the feedback is computed only from the spare bandwidth. Pseudo code of the algorithm is presented in [10].

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 [11], and each packet has both a congestion 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.

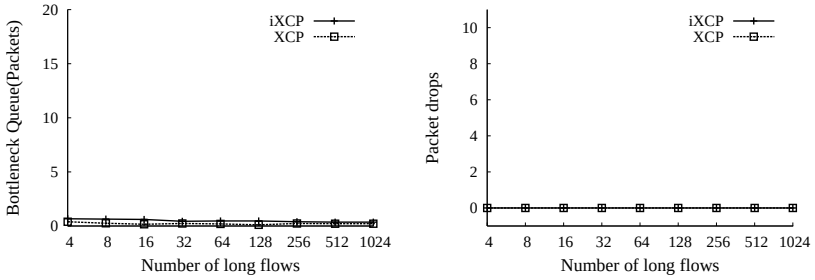
We build on Low's theoretical model [7] to prove the following theorem and we refer interested readers to [10] for detailed proof.

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.*

4 Simulation Results

In this section, we use extensive ns2 [12] simulations to evaluate the performance of iXCP and compare it with XCP. Due to space limitation, we only present a subset of our results. More results can be found in [10].

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



(a) Average queue size (b) Packet drops

Fig. 3. Small average queue size and zero packet drops are achieved for the 155Mb/s link in Figure 1

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.

4.1 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 2.2. 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 2(a) and 2(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 in [10]. 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 97% link utilization; the short flow's rate is more than 90% of its max-min rate.

Figure 3(a) and 3(b) show the average queue size and the packet drops at the 155Mb/s link. 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

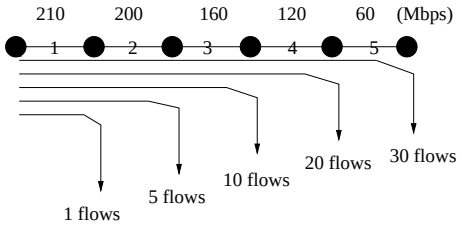


Fig. 4. A multi-bottleneck topology

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.

4.2 A multi-bottleneck environment

We studied how iXCP performs in a multi-bottleneck environment as shown in Figure 4. 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 5(a) and 5(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 figures. 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.

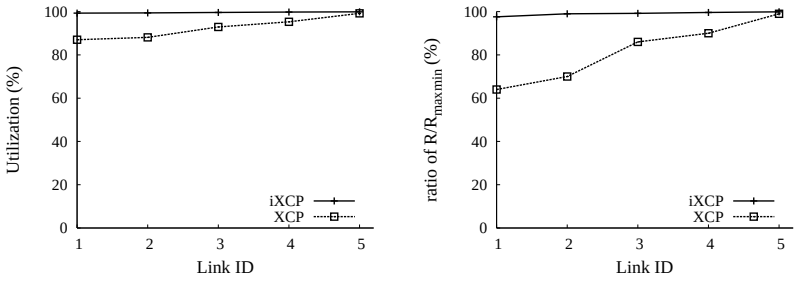
Small average queue lengths and zero packet drops are achieved for both iXCP and XCP. The results are presented in [10].

4.3 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.

The simulation topology is shown in Figure 6. Each link has a 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 7(a), 7(b), and 7(c) show how each flow's rate changes over time as its bottleneck shifts with traffic demand. 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



(a) Link utilization

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

Fig. 5. iXCP achieves nearly 100% link utilization and max-min fair flow rates for each link in the multi-bottleneck topology shown in Figure 4

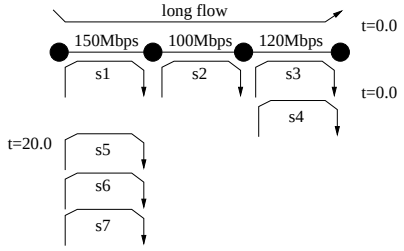
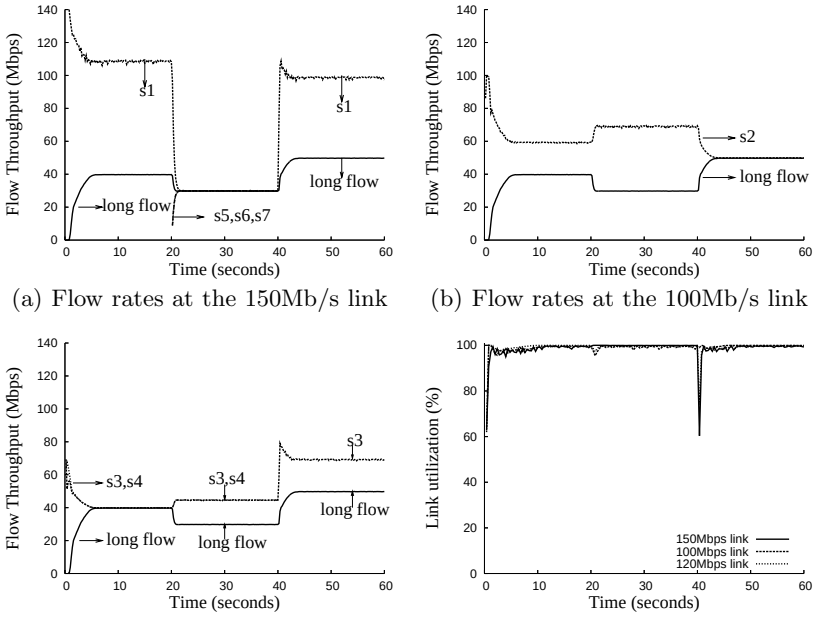


Fig. 6. The long flow and four short flows s_1 to s_4 start at time $t = 0$. Three short flows s_5, s_6, and s_7 start at $t = 20$. At time $t = 40$, the short flows from s_4 to s_7 stop

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, other shorts flows quickly converge to their max-min rates (the short flows s_1, s_5, s_6 and s_7 converge to 30Mb/s, and s_2 increases to 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 7(d) 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.

The convergence property of iXCP to equilibrium is similar to that of XCP as shown in the longer version of this paper [10]. Both converge to equilibrium within seconds, whereas TCP persistently oscillates and never converges [10]. 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.

Due to the lack of space, simulation results for heterogeneous RTTs and web-like traffic are not included here. A complete set of simulation results are presented in [10].



(a) Flow rates at the 150Mb/s link

(b) Flow rates at the 100Mb/s link

(c) Flow rates at the 120Mb/s link

(d) Link Utilization for all the three links shown in Figure 6

Fig. 7. These figures show how iXCP adapts to the flow dynamics and the link utilization of all three links during adaptation. The simulation topology is shown in Figure 6. Each flow can quickly converge to its max-min fair rate when there are flows joining or leaving the network. Each link can achieve nearly 100% link utilization

5 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 flows bottlenecked at itself. The router allocates more bandwidth to those bottlenecked flows 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 [10].

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 [13] 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. [14] 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 [15] and improvements of XCP in other areas. Zhang et al. [16] extensively studied the implementation and deployment challenges of XCP. Hsiao et al. [17] extended XCP to support streaming layered video. Kapoor et al. [18] proposes to combine XCP with a Performance Enhancement Proxy (PEP) to provide fast satellite access. Yang et al. [19] proposed an improvement to shorten XCP's response time for new flows to acquire their bandwidth. Zhang et al. [20] presented a control theoretical model that considers capacity estimation errors. XCP-r [21] proposes to calculate the congestion window size at the receiver side to cope with ACK losses.

6 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 in [10], 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.

7 Acknowledgement

The authors would like to thank Yong Xia for useful discussions, Dina Katabi for answering questions and Xin Liu for help with simulations.

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. 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`.
11. "Caida report," `http://www.caida.org/analysis/AIX/plen_hist/`, February 2000.
12. "The network simulation: ns2," `http://www.isi.edu/nsnam/ns`.
13. Y. Zhang, D. Leonard, and D. Loguinov, "JetMax: Scalable Maxmin Congestion Control for High-Speed Heterogeneous Networks," in *Proc. of INFOCOM*, 2006.
14. A. Charny, D. Clark, and R. Jain, "Congestion Control With Explicit Rate Indication," in *Proc. of ICC*, 1995.
15. A. Falk and D. Katabi, "Specification for the Explicit Control Protocol (XCP)," 2005.
16. Y. Zhang and T. Henderson, "An Implementation and Experimental Study of the eXplicit Control Protocol (XCP)," in *Proc. of INFOCOM*, 2005, pp. 1037–1048.
17. H. Hsiao and J. Hwang, "A Max-Min Fairness Congestion Control for Streaming Layered Video," in *Proc. of ICASSP*, 2004, pp. V981– V984.
18. A. Kapoor, A. Falk, T. Faber, and Y. Pryadkin, "Achieving faster access to satellite link bandwidth," in *Proc. of INFOCOM*, 2005, pp. 2870–2875.
19. 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.
20. Y. Zhang and M. Ahmed, "A Control Theoretic Analysis of XCP," in *Proc. of INFOCOM*, 2005, pp. 2831–2835.
21. 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.