

Achieving Faster Access to Satellite Link Bandwidth

Aman Kapoor, Aaron Falk, Ted Faber, Yuri Pryadkin
USC/Information Sciences Institute
{kapoor, falk, faber, yuri} @isi.edu

Abstract

TCP with Van Jacobson congestion control (VJCC) is known to have poor performance over large bandwidth-delay product paths. Long delay paths, in particular, can display very poor behavior with VJCC slowly probing to acquire available capacity. TCP Performance Enhancing Proxies (PEPs) constitute one mechanism for ameliorating poor VJCC end-to-end performance by splitting TCP connections around a long-delay link or network and using alternate congestion control dynamics across the troublesome portion. Previous congestion control techniques for use over satellite links have typically constituted either tweaks to the Van Jacobson algorithms, Vegas-style congestion control, or disabling congestion control altogether in favor of a manual send-rate. This paper analyzes results from measurements of a new congestion control mechanism, the eXplicit Control protocol or XCP, between PEPs with a simulated geosynchronous satellite link in the path. We show that connections using an XCP PEP acquire their share of expensive satellite bandwidth up to 70 times faster than end-to-end TCP with VJCC.

1. Introduction

TCP [1, 2] carries a major portion of the Internet traffic. It has shown a remarkable resilience to the large magnitudes of Internet growth. However, with the large bandwidth-delay paths, wireless and satellite links, TCP's congestion control algorithm, known as Van Jacobson congestion control or VJCC [3, 4], clearly exhibits its limitations. Long delay paths in particular can have very poor behavior with VJCC being slow to acquire available capacity. TCP has other difficulties in this operating regime, but the slow acquisition of capacity is the one we address here.

TCP with VJCC makes inefficient use of long fat pipes. In the *congestion avoidance* phase, additive increase allow TCP to open its congestion window (*cwnd*) by 1 packet every round-trip time (RTT). In high delay networks it would take a long time to fill up the high bandwidth pipe resulting in underutilization of network resources. VJCC's throughput is inversely proportional to the RTT, thus hav-

ing an inherent bias towards the short RTT flows. In a mixture of short and long RTT flows, VJCC allocates bandwidth unfairly between them. VJCC treats packet loss as the indication of congestion in the network. Wireless links where losses may be due to corruption lead to poor performance[5, 6] using VJCC.

Several schemes have been proposed to improve TCP's performance over wireless and satellite links, such as having a reliable link layer [7], a snoop agent at the base station for faster retransmit [8], TCP Selective Acknowledgments [9], or a *Performance Enhancement Proxy (PEP)* [10] around the long delay link or network. PEPs commonly split the end-to-end TCP connection into three, the first between the sender and PEP end-point on the sender side, the second between the two PEP end-points (spanning the high delay portion of the network) and the last between the other PEP end-point and the receiver. An alternate congestion control algorithm may be applied between the two PEPs to achieve a higher end-to-end performance. Other congestion control techniques for use over satellite links have typically constituted either tweaks to the Van Jacobson algorithms [11, 12, 13], Vegas-style congestion control [14, 15], or disabling congestion control altogether in favor of a manual send-rate. However, these approaches have only been partially successful.

In this paper we show that fast access to satellite link bandwidth can be achieved using an *XCP PEP*.

The eXplicit Control Protocol (XCP) is a new congestion control proposed by Katabi, et al. [16, 17, 18]. XCP makes use of explicit feedback from the routers. Its scalability is achieved due to the router not having to maintain any per-flow state and calculating feedback on aggregate traffic. It also decouples efficiency from fairness and adjusts sender's rates using two algorithms: a *multiplicative-increase multiplicative-decrease (MIMD)* approach when adjusting efficiency and an *additive-increase multiplicative-decrease (AIMD)* approach when adjusting fairness. This results in higher network utilization and faster convergence to fair allocation among multiple flows. The initial studies have shown a marked increase in end-to-end link utilization in comparison to VJCC [16, 17].

Since XCP only changes the congestion control algo-

rithm in TCP, i.e., TCP reliability mechanisms are unaffected, it is not a new transport protocol. The XCP algorithm could also be used in other transport protocols.

We will show that a PEP using XCP congestion control acquires capacity in a high bandwidth-delay product network up to 70 times faster than an end-to-end VJCC. We also present preliminary results indicating that the XCP PEP also converges to fair allocations with multiple flows quickly.

Section 2 gives a brief description on XCP congestion control. In Section 3 we describe the XCP PEP in more detail. Section 4 lays out the testbed topology and experimental analysis. Section 5 describes some future work and concludes the paper.

2. XCP Protocol

XCP is a feedback-based congestion control system. Routers in the network provide explicit feedback to the end points advising a change in sending rate. The end-points thus change their sending rate based on the explicit feedback from the network rather than inferring the state of congestion in the network. Scalability is achieved due to the routers not having to maintain any per-flow state. All the calculations done in the routers are based on the aggregate traffic that it sees. The algorithm introduces minimal overhead (a few multiplications and additions) on router’s processing. The feedback is a function of the aggregate throughput, capacity of the bottleneck link and the bottleneck queue [16, 17, 18].

XCP decouples efficiency from fairness. It applies a *multiplicative-increase multiplicative-decrease* (MIMD) policy for efficiency. The efficiency controller calculates the aggregate feedback (positive or negative) that is to be handed out in the next control interval. Using MIMD results in faster ramp up to the bottleneck bandwidth and is advantageous for satellite links (long RTTs) where VJCC slowly ramps up. The fairness controller applies *additive-increase multiplicative-decrease* (AIMD) to fairly allocate the capacity to the flows traversing that link.

Explicit feedback to the end-points results in a quicker ramp up of the flow’s sending rate to the optimal and fair allocation of bandwidth amongst the flows. XCP’s congestion control is insensitive to loss. Routers hand out negative feedback on seeing any persistent bottleneck queue. Even if a packet is lost, end-points can quickly recover based using network feedback rather than the slow *congestion avoidance phase* as in VJCC [3]. XCP also eliminates VJCC’s inherent bias towards short RTT flows thus resulting in a greater fairness when RTTs are diverse.

Initial studies on XCP have shown significant performance improvements over VJCC [16, 17]. In this paper we show that XCP proxies can be used to achieve a higher

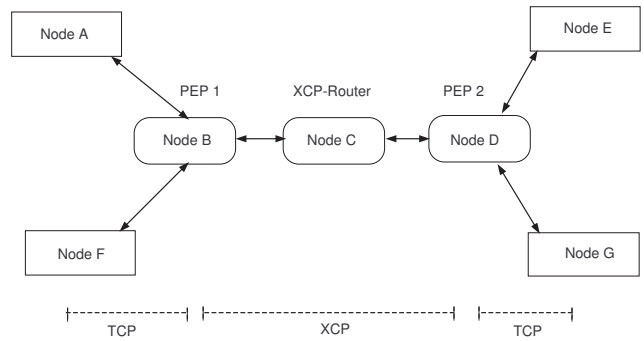


Figure 1. Testbed Setup

end-to-end performance over satellite links than VJCC end-to-end. The next section provides a details of our XCP PEP.

3. XCP-PEP design and implementation

Performance Enhancement Proxies are often used to improve poor TCP performance in certain environments, like satellite communications, having long RTTs (at least 480ms for geosynchronous satellites) and high loss rates. For example, in satellite networks PEPs are deployed around the long delay link and use alternate congestion control mechanisms such as tweaks to the Van Jacobson algorithms [11, 12, 13], Vegas-style congestion control [14, 15], or no congestion control (using UDP).

XCP congestion control has been designed to accommodate long bandwidth-delay product networks. In this section we describe the XCP-PEP in detail.

3.1. XCP-PEP Design Philosophy

Figure 1 shows a basic PEP configuration. Nodes A, E, F, and G are the endpoints. Nodes B and D are the XCP-PEPs and Node C is an XCP capable router.

- End-point A initiates a regular TCP connection with Node E. Node B, the sender-side XCP PEP, splits the connection (by intercepting the TCP SYN) and initiates a TCP connection *using XCP* with Node D, the receiver-side XCP PEP. Node D, upon receiving the XCP PEP connection request, initiates a TCP connection with Node E, the receiver. Figure 2 shows the connection splitting. The PEP connection triggers a TCP 3-way handshake (between nodes B and D) at the application level. The *Hello* packets sent from the sender-side PEP (node B) to the receiver-side PEP (node D) contain details about node A’s TCP connection such as end-point addresses and TCP ports. The sender-side PEP waits until the B-D and D-E TCP connections are established before responding to nodes A’s SYN with a SYN/ACK to achieve rudimentary *fate sharing*. Once the connection is established data can

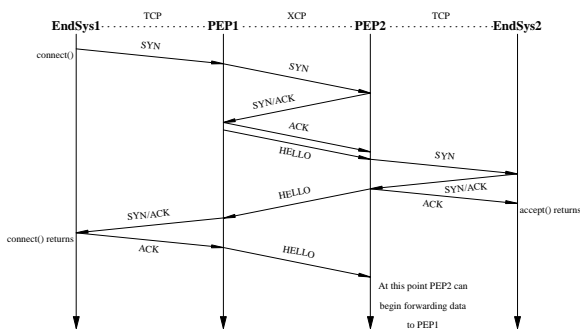


Figure 2. TCP Connection over PEP

flow between the end-points.

- Our XCP PEP is transparent to the end-points. The end-point sees a regular TCP connection to its peer. No application or transport layer modifications were required at the end-points.¹
- While a kernel implementation of the PEP functionality is possible, we chose to implement the PEP functions in user space. A user space implementation allows us to use TCP/IP stack all the way up to socket buffers to buffer and acknowledge data from end-points.
- There is a one-to-one mapping between TCP and PEP connections. We did not implement multiplexing of TCP connections, mainly for simplicity. It remains a promising direction for future work.

3.2. PEP Implementation Details

Here we provide some of the implementation level details of the XCP PEP. The PEP code has been written for FreeBSD to be compatible to XCP implementation.

3.2.1 Packet Interception

For the XCP PEP to work we configure it as a gateway that forwards packets. It intercepts the packets being forwarded by removing them from the IP-stack, changes the packet headers, fills in the XCP information and re-injects them back into the network. We make use of FreeBSD firewall *ipfw* mechanism in conjunction with *divert sockets*. *ipfw* redirects the packets to a *divert socket* where they are read, modified and re-injected back into the stack.

The PEP configuration file contains the triples in the form of *src, dst, peer*. *Src* and *dst* are the source and destination networks, e.g., 10.10.1.0/24, and *peer* is the peer

¹We note that the use of IPsec between nodes A and E will defeat the PEP's ability to intercept the connection. This is a known shortcoming of split-connection PEPs and is discussed in [10].

PEP's IP address. This file is used at the startup to install firewall rules to divert the packets.

3.2.2 Network Address Translation

For each intercepted packet coming from the end-system, PEP needs to translate its destination address/port to a local address/port. For the reverse packet direction, it must translate packet's source address/port (local) to the other end-system's address and port. In addition, IP and extended TCP checksums have to be adjusted.

This functionality is provided on FreeBSD systems by the *libalias* library. It allows for packet aliasing and de-aliasing rules to be installed: *PacketAliasRedirectPort()* and later applied by calling *PacketAliasIn()* and *PacketAliasOut()*. Thus, all address caching, translations, and checksums adjustments are done in user space.

We have a multi-threaded model where the master thread is responsible for the network address translation and creating new threads when it detects a new connection attempt. The child threads complete the end-to-end connection and data transfer.

Next we describe the testbed topology and experiments.

4. Experiments and Analysis

This section describes experiments we conducted to determine if the XCP PEP was effective in providing sources with higher bandwidth across a long latency path. We show that using the XCP PEP to traverse the high latency section allows sources to rapidly make full use of the link. We also show traces from typical runs to assist in the intuition and report on early fairness results.

4.1 Single Flow Throughput

Figure 1 shows the testbed topology. The nodes are Intel Xeon, CPU 2.80GHz, 1GB RAM and run FreeBSD 4.8 modified to support XCP. The same nodes are used for TCP and the stacks have been tuned for high bandwidth-delay product networks, e.g., with large transmit and receive buffers. (Throughout the rest of this paper TCP refers to TCP with VJCC and XCP refers to TCP with XCP congestion control.) The high latency is induced by delaying acknowledgments, instead of data packets, from E or G to A or F at C. Delaying acknowledgments reduces the buffering requirements at C because acknowledgments are smaller than data packets. Without this delay inducement, round trip times would be on the order of 1 ms. With the delay, sources see round trip times of 200ms.

The outgoing link from the router on the forward path (link between Node C and D in Figure 1) is made the bottleneck by using 100Mbit Ethernet, while all the other links use Gigabit Ethernet. In a real wide area network it is un-

Scheme	Average	Std. Dev
TCP (scenario a)	123.38	1.28
XCP (scenario b)	1.47	0.32
XCP-PEP (scenario c)	1.61	0.26

Table 1. Time(sec) to reach 95% utilization

likely that hosts would connect at gigabit rates, this was used as simple technique to confine the bottleneck to the long-delay portion of the network. The difference in round-trip times between the long-delay and short-delay portions of the network ensure that the TCP dynamics in the short network will not affect the end-to-end performance. Analysis of links where losses are not primarily congestive remains future work.²

We compare the XCP PEP to end-to-end TCP and end-to-end XCP to demonstrate that the PEP delivers nearly the same performance as end-to-end XCP, which is considerably better than TCP performance. In the single flow case, this performance is reflected in how quickly a source can use all of the channel. Under end-to-end XCP this ramp-up happens very quickly, under end-to-end TCP the congestion control is more conservative and happens more slowly.

We run our tests with a single end-to-end flow to compare utilization and then with two end-to-end flows to compare fairness. In all cases the round trip time is 200 ms and the endpoints have a bandwidth-delay product worth of transmit and receive buffers. In these experiments only one source/destination pair is sending.

Scenario a: *TCP Flows end-to-end*: Nodes B, C and D act as the routers. We set the TCP and socket buffers to bandwidth-delay product.

Scenario b: *XCP Flows end-to-end*: Here node C is an XCP-capable router. Nodes B and D are non-XCP routers, though, because they are not bottlenecks and have negligible queues, their behavior would not change if they were XCP-capable.

Scenario c: *TCP flows with XCP PEP*: Here node A opens up a TCP connection to Node E. Nodes B and D act as XCP PEP end-points. The TCP connection from the sender (node A) to the receiver (node E) is intercepted by Node B, which initiates an XCP connection with Node D who, in turn, opens a TCP connection with the receiving end. Node C is an XCP-capable router. No modifications to the TCP stack or application are required at the end-points.

Table 1 shows the average time for each scenario to reach 95 percent utilization of the link. Each experiment was repeated 10 times and the mean and standard deviation reported.

²Our model is of a network between the PEPs which consists only of XCP flows. Analysis of fair sharing between XCP and VJCC flows is outside the scope of this paper.

The XCP PEP ramps up nearly as quickly as end-to-end XCP and more than 70 times faster than TCP. The advantages for long-delay connections should be clear.

In our scenario (without losses due to corruption) TCP's throughput stays constant once it reaches the maximum. TCP uses $\min(\text{cwnd}, \text{receiver window})$ as its sending window, hence if properly tuned the receiver window would make TCP stable. However, tuning the receiver window to get the optimal utilization is a hard problem due to less knowledge of the bandwidth-delay product of the end-to-end path. Web100 [19] tries to solve this problem by auto-tuning the receiver window.

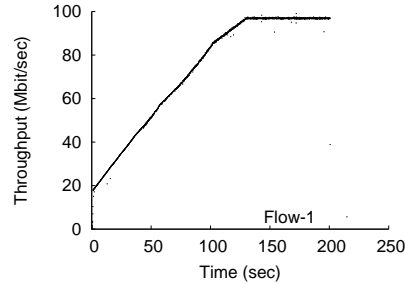


Figure 3. TCP: Single Flow, RTT=200ms

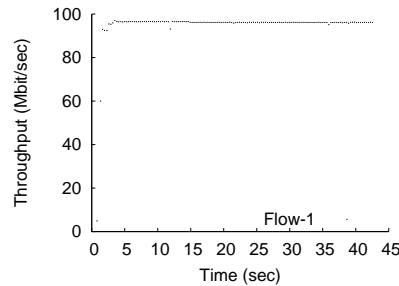


Figure 4. XCP: Single Flow, RTT=200ms

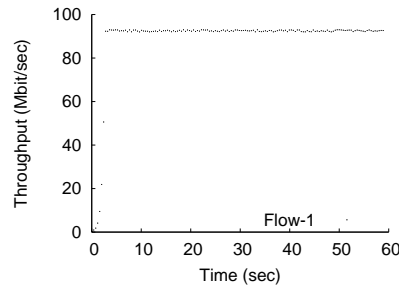


Figure 5. PEP: Single Flow, RTT=200ms

Figures 3-5 show the throughput of each source through the router calculated each round trip time. Each figure is a trace from a representative run of the experiments tabulated in Table 1, and they demonstrate how each system delivers the link bandwidth to the source.

Each RTT an XCP source gets information from the routers about excess capacity there, so they can converge to the link bandwidth quickly (Figures 4-5). Without this extra information a TCP source can only increase its rate linearly

each RTT once the *ssthresh* is reached causing it to leave the slow-start phase and enter congestion avoidance. This linear increase, following the initial exponential increase, is displayed in the slope of the line in Figure 3 which is inversely proportional to the RTT. Confining Reno to low-latency access networks keeps the ramp-up to the PEP site quick, because of the short RTT.

4.2 Fairness

In this section we briefly address the fairness properties of the XCP PEP. Space does not permit a full discussion.

Fairness is an important property for a congestion control system, in all but the simplest situations. Otherwise multiple sources must share the network resources without prior arrangement. Some proposed PEP systems neglect this issue and consequently require extensive manual configuration.

Although far from a complete investigation, Figures 6-8 show that both the end-to-end congestion systems and the PEP system deliver fairness that passes the eyeball test in the simple configuration we have. These figures are generated the same way as Figures 3-5, with each point representing the throughput at the bottleneck router over one RTT.

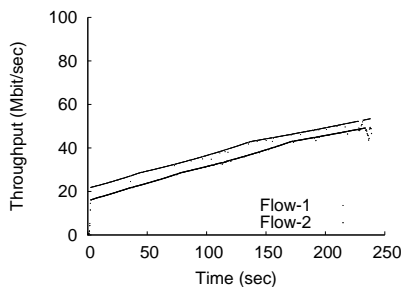


Figure 6. TCP: Two Flows, RTT=200ms

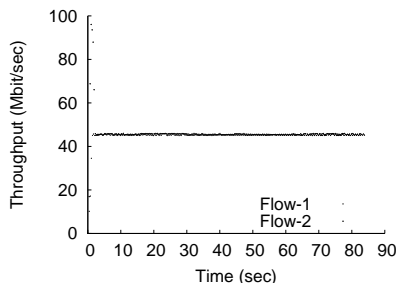


Figure 7. XCP: Two Flows, RTT=200ms

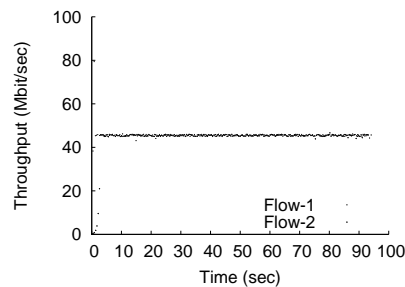


Figure 8. PEP: Two Flows, RTT=200ms

The primary result here is that using the PEP does not seriously disrupt the XCP fairness. TCP can be somewhat unfair, as shown in Figure 6, but without input from the routers convergence may take several peak and drop cycles. XCP, both end-to-end and with the PEP, converges quickly even with the large delay.

5. Conclusion and Future Work

This paper has described using an XCP PEP in high latency networks and shown that the benefits of the XCP protocol described elsewhere can be provided by a PEP in such a network. Specifically, we have shown that XCP provides access to full link capacity for a single source more than 70 times faster than a source using only Reno TCP. We have also presented some evidence, albeit preliminary and rough, that the XCP PEP will also provide fairness in such an environment.

We believe that these results are promising, but there are other topics that need to be more fully addressed before the case for an XCP PEP is compelling. We would like to compare XCP PEP with more sophisticated solutions proposed for satellite networks like SCPS-TP [15]. SCPS-TP uses TCP-Vegas-based [14] congestion control and tries to differentiate between losses due to congestion from those due to corruption. TCP-Vegas is expected to perform poorly over systems using bandwidth-on-demand since TCP-Vegas uses delay variation as an indication of congestion and bandwidth-on-demand systems have high delay variation. We believe that in such cases XCP's fast reaction time and fairness would result in a higher performance.

References

- [1] Jon Postel. Transmission control protocol. RFC 793, Internet Request For Comments, September 1981.
- [2] W. Richard Stevens. *TCP/IP Illustrated*, volume 1. Addison-Wesley, 1994.
- [3] Van Jacobson. Congestion avoidance and control. In *Proceedings of the SIGCOMM '88*, pages 314–329, Stanford, California, August 1988. ACM.

- [4] M. Allman, V. Paxson, and W. Stevens. TCP congestion control. RFC 2581, Internet Request For Comments, April 1999.
- [5] Mark Allman, Chris Hayes, Hans Kruse, and Shawn Ostermann. TCP performance over satellite links. In *Proceedings of the Fifth International Conference on Telecommunications Systems*, Nashville, TN, March 1997.
- [6] Ramon Caceres and Liviu Iftode. Improving the performance of reliable transport protocols in mobile computing environments. *IEEE Journal of Selected Areas in Communications*, 13(5):850–857, 1995.
- [7] E. Ayanoglu, S. Paul, T. F. LaPorta, K. K. Sabnani, and R. D. Gitlin. AIRMAIL: A link-layer protocol for wireless networks. *ACM Wireless Networks*, 1(1):47–60, 1995.
- [8] Hari Balakrishnan, Srinivasan Seshan, Elan Amir, and Randy H. Katz. Improving TCP/IP performance over wireless networks. In *Proceedings of the First ACM Conference on Mobile Computing and Networking*, pages 2–11, Berkeley, CA, USA, November 1995. ACM.
- [9] M. Mathis, J. Mahdavi, S. Floyd, and A. Romanow. Tcp selective acknowledgment options. RFC 2018, Internet Request For Comments, October 1996.
- [10] J. Border, M. Kojo, J. Grinder, G. Montenegro, and Z. Shelby. Performance enhancement proxies intended to mitigate link related degradations. RFC 3135, internet-rfc, June 2001.
- [11] Robert C. Durst, Gregory J. Miller, and Eric J. Travis. Tcp extensions for space communications. In *Proceedings of the 2nd ACM Conference on Mobile Computing and Networking*. ACM, November 1996.
- [12] V. Jacobson and R.T. Braden. Tcp extensions for long delay paths. RFC 1072, internet-rfc, October 1988.
- [13] Mark Allman, Jim Griner, Keith Scott, Joe Touch, and Diepchi Tran. Ongoing tcp research related to satellites. RFC 2760, February 2000.
- [14] L. S. Brakmo, S. W. O’Malley, and L. L. Peterson. TCP Vegas: New techniques for congestion detection and avoidance. In *Proceedings of the ACM SIGCOMM Conference*, pages 24–35. ACM, September 1994.
- [15] R. Durst, G. Miller, and E. Travis. TCP extensions for space communications. *Proceedings of the second annual international conference on Mobile computing and networking*, White Plains, NY USA, page 15, 1996.
- [16] D. Katabi, M. Handley, and C. Rohrs. Internet congestion control for future high bandwidth-delay product environments. In *Proceedings of the ACM SIGCOMM Conference*, 2002.
- [17] D. Katabi. *Decoupling Congestion Control from the Bandwidth Allocation Policy and its Application to High Bandwidth-Delay Product Networks*. PhD thesis, Massachusetts Institute of Technology, March 2003.
- [18] A. Falk and D. Katabi. Specification for the explicit control protocol (XCP). October 2004.
- [19] J. Heffner. High bandwidth tcp queueing, july 2002. http://www.psc.edu/jheffner/papers/senior_thesis.ps.