Enabling Large Data Transfers on Dynamic, Very High-Speed Network Infrastructures

D. M. Lopez-Pacheco
INRIA RESO/LIP, France
Email:dmlopezp@ens-lyon.fr

C. Pham, *Member, IEEE*
LIUPPA, University of Pau, France
Email: Congduc.Pham@univ-pau.fr

***Abstract*— High-speed networks are being built using the most up-to-date networking technologies such as optical fibers with dense wavelength multiplexing (DWDM) in order to meet the performance constraints of large scientific problems (grid infrastructures) or new multimedia applications. As technologies are moving forwards, many high-speed network infrastructures have amazing capabilities in the order of several gigabits/s. Our work focuses on very dynamic high-speed environments where the available best-effort bandwidth for regulated traffics can vary over time. Foreseen problems introduced by such highly dynamic environments are inefficiency due to convergence time and high amount of packet losses due to dramatic reductions of the available bandwidth. In a previous study, we proposed the XCP-r protocol which is a more robust XCP-based transport protocol. In this paper, we present simulation results comparing XCP-r, XCP, TCP (NewReno) and HSTCP for large data transfers on very dynamic high-speed networks. We show that XCP-r succeeds in providing a high level of performance thus able to function in a broader range of network conditions.**

I. Introduction

Reliable transfers of data is usually the task of transport protocols located at layer 4 in the OSI stack model. In the Internet world, and by extension in all IP-based networks, the TCP protocol originally defined in RFC 793 is the main protocol in charge of the difficult task of providing reliability and fair sharing of the bandwidth to end-users. Since the congestion collapse observed by V. Jacobson in 1986 and the well-known slow-start and congestion avoidance algorithms proposed in 1988 [8], the networking community has proposed many enhancements and optimizations to the original proposition in order to make TCP more efficient in a large variety of network conditions (to better react to congestions) and technologies [3], [7]: wireless links [14], [6], asymmetric links and very high-speed and long delay links [5], [9], [10], [15], [11]. Most of these new protocols are also made available in the newest Linux kernels facilitating the deployment of these new protocols on new network infrastructures.

High-speed networks are being built using the most up-to-date networking technologies such as optical fibers with dense wavelength multiplexing (DWDM) in order to meet the performance constraints of large scientific problems (grid infrastructures) or new multimedia applications. As technologies are moving forwards, many high-speed network infrastructures have amazing capabilities in the order of several gigabits/s. On these networks, usually referred to as high bandwidth-delay product networks, existing protocols such as TCP need to be tuned to the new networking conditions (socket buffer size, maximum congestion window size, timer setting,...). Tuning is however a very difficult task that usually needs deep knowledge of the internals of the protocols and of the operating systems. This is why the `web100` project (`http://www.web100.org/`) aims at distributing ready-to-use tuned versions of protocol stacks adapted to high-speed networks. The DataTAG project (see `http://datatag.web.cern.ch`) did also address this problem specifically for grid environments. Note that some works also address the problem of large file transfers by using multiple flows in parallel. For instance `GridFTP` [2] and `bbFTP` (`http://doc.in2p3.fr/bbftp/`) are two parallel flows tools that are currently being used in the grid community. These approaches, though efficient, show scalability problems when used on gigabits/s infrastructures.

In new transport protocol propositions, the main optimizations consist in adding more efficient mechanisms for acquiring bandwidth faster. For example HSTCP [5] modifies the standard TCP response function to both acquire faster the available bandwidth (more efficiency) and to recover faster from packet losses in the network. The drawback of such a behavior is that fairness between TCP and HSTCP, and even between HSTCP flows, is affected since HSTCP is much slower to give back bandwidth (see [15] for discussion). FAST TCP [9] is basically a modification of TCP Vegas which uses the delay variation (round-trip time variation) to predict congestion conditions in the network. FAST TCP shows very good performances but suffers from non-congestion based delay variations such as rerouting. While TCP, HSTCP and FAST TCP can be classified as end-to-end solutions, XCP [10] belongs to the router-assisted approaches that use the assistance of routers to more accurately signal congestion conditions in the network and to compute the optimal congestion window size to be applied at the source. Therefore, XCP shows very stable behavior but is also able to get bandwidth very fast while preserving fairness among flows which is an important criterion on a computational grid infrastructure shared by many users.

In this paper, we focus on very dynamic high-speed infrastructures where the available best-effort bandwidth can vary over time. There are several reasons to this behavior. For

This work was done while C. Pham was with INRIA RESO/LIP, France.

example, it is not unrealistic to foreseen dynamic subscriptions to guaranteed bandwidth for resource-consuming grid applications. In this case, the bandwidth available for the best-effort traffic have large variations over time. An other reason could be the presence of unregulated traffic such as UDP packets used in many multimedia applications which compete in an unfair manner with regulated flows. Foreseen problems that are amplified by such highly dynamic environments are inefficiency due to convergence time (because it is difficult to quickly and safely acquire the available bandwidth when it suddenly increases) and high amount of packet losses due to dramatic reductions of the available bandwidth. Therefore end-to-end solutions show their limitations for exploiting in an optimal manner the current very high-speed as data grid infrastructures.

XCP is a promising approach as the evolution of the sender congestion window size is dictated by the routers. However, as the XCP sender relies on the returned ACK packets to adapt its congestion window size, XCP can be affected if many losses occur on the reverse path. In [16], the authors observe that the lost of ACK packets have little impact on the ability of XCP to adjust the sender's congestion window size. However, the authors did not consider situations where there are bandwidth fluctuations over time (i.e. dynamic networks). We believe that in such environments, the problem of lost ACKs on the reverse path has much more impact on the XCP performances than what have been found in previous studies. Therefore, XCP-r is an enhancement of XCP [13] capable of achieving a high level of performance even in very dynamic high-speed environments, thus able to function in a broader range of network conditions. In this paper, we present simulation results comparing XCP-r, XCP, TCP (NewReno) and HSTCP for large data transfers on very dynamic high-speed networks.

The paper is organized as follows. Section 2 reviews the XCP protocol and its enhancement XCP-r that removes the high dependancy of XCP on ACK packets. Section 3 presents the performances of XCP and XCP-r on dynamic, very high-speed networks. Section 4 compares XCP and XCP-r to TCP and HSTCP. Section 5 concludes the paper.

II. The XCP and XCP-r protocols

A. XCP

XCP [10] (eXplicit Control Protocol) uses router-assistance to accurately inform the sender of the congestion conditions found in the network. In XCP, data packets carry a congestion header, filled in by the source, that contains the sender's current congestion window size (H_cwnd), the estimated RTT and a feedback field ($H\_feedback$ that could be positive or negative) representing the increase amount to be applied to the sender's congestion window. The $H\_feedback$ field is the only one which could be modified at every hop (XCP router) based on the value of the two previous fields (see figure 1).

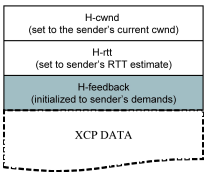

Fig. 1. XCP congestion header.

The core mechanism resides in XCP routers that use an efficiency controller (EC) and a fairness controller (FC) to update the value of the $H\_feedback$ field over the average RTT which is the control interval. The EC has the responsibility of maximizing link utilization while minimizing packet drop rate. The EC basically assigns a feedback value proportional to the spare bandwidth S deducted from monitoring the difference between the input traffic rate and the output link capacity (S could be positive or negative) and to the persistent queue size Q (to avoid a feedback value of zero when input traffic is equal to output capacity). Then the FC translates this feedback value, which could be assimilated to an aggregated increase/decrease value, into feedback for individual flows (to be put in the data packet's congestion header) following fairness rules similar to the TCP AIMD principles. Note that no per-flow states are used by XCP routers to perform all these operations. On reception of data packets, the XCP receiver copies the congestion header (which has been modified accordingly by the routers on the forward data path) into ACK packets sent back to the source which will then update its congestion window size as follows: $cwnd = max(cwnd + H\_feedback, packetsize)$, with $cwnd$ expressed in bytes. More information on XCP can be found in Katabi's paper [10] and in [16].

B. XCP-r: a robust version of XCP

One problem of XCP comes for the fact that feedbacks are summed up by the source in order to incrementally compute the optimal sender's congestion window $cwnd$. One reason for doing so is that feedback values can be changed very quickly by the routers to take into account any changes in the networks. However, lost ACKs may cause a mismatch between the real network conditions in term of bandwidth availability and the conditions perceived by the source. XCP could then have unstable behavior which is amplified in dynamic high-speed networks where large amount of data could be sent per congestion window and where bandwidth reductions could occur.

XCP-r ("r" for receiver) [13] is a more robust version of XCP where the main idea is to avoid having the incremental feedbacks summed up by the source, while keeping the incremental feedback at the router level for flexibility and robustness. XCP-r reproduces at the receiver the computations that are performed at the source in the original proposition. To do so, the receiver maintains a $cwnd'$ variable per flow that evolves according to the `H_feedback` value received in data packets as follows: $cwnd' = cwnd' + H\_feedback$. As `H_feedback` evolves $cwnd'$ will evolve accordingly. A new $cwnd'$ variable set to 1 is initialized for each new

connection. Therefore, ACK packets sent by the receiver carry the new congestion window size instead of an increment or a decrement value. More details on XCP-r can be found in [13].

III. Performance of XCP and XCP-r

In XCP $cwnd$ can directly jump to the optimal value without any time wasted by a slow-start phase or by some incremental heuristics (like in HSTCP or FAST TCP). Therefore, achieving high utilization ratio on high-speed links is not a problem anymore for XCP. In this section, we show some ns-based [1] simulation results of XCP and XCP-r on dynamic, very high-speed networks. The original XCP ns code is the one provided by D. Katabi. Bandwidth variations for best effort traffic are due to ON/OFF UDP sources which compete with the XCP flows.

A. The dynamic, very high-speed network model

Figure 2 shows the classical dump-bell network model with 2 XCP routers used for our simulations. R1 is connected to the sources and R2 to the receivers (1 XCP flow and 17 UDP flows). All tail links have the same delay of 1-ms. R1 is connected to R2 by a 1Gbps link with 50ms/100ms of one-way delay which represents the bottleneck link.

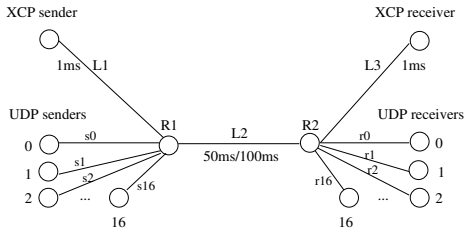

Fig. 2. Network topology

R1 and R2 have 2300 packet buffers per incoming link when the RTT is 100ms and 4300 when the RTT is 200ms. On a long, high-speed networks, the optimal buffer size usually follows the rule of thumb of $bandwidth.rtt$ product, e.g. for a 1Gbps link with an RTT of 100ms the buffer in number of 1Kbytes packets is about 12200. In this network model the amount of buffer is not optimal but we believe it better reflects what could be found in real networks. The queue management strategy is RED for both routers and uses the same configuration found in [10].

B. Simulation results of XCP and XCP-r

Figure 3 shows 10 XCP flows on a dynamic networks with small bandwidth fluctuations when compared to the link capacity (the maximum bandwidth fluctuation represents 18% of the link capacity).

We can see that although the bandwidth fluctuations are small, XCP shows a very unstable behavior with many time-outs and packet drops which considerably slows down the file transfer. When the bandwidth is reduced by half and causes a congestion at time 5s, we can see in the figure that the timeout could not be recovered quickly for some flows. The explanation is the following: on a timeout the sender $cwnd$ is set to 1 therefore the receiver could only send back 1 ACK that would carry a large feedback value if there is available bandwidth. If this ACK is dropped, the sender would need to wait for the retransmission timer (RTO) in order to send the packet again to get another ACK from the receiver.

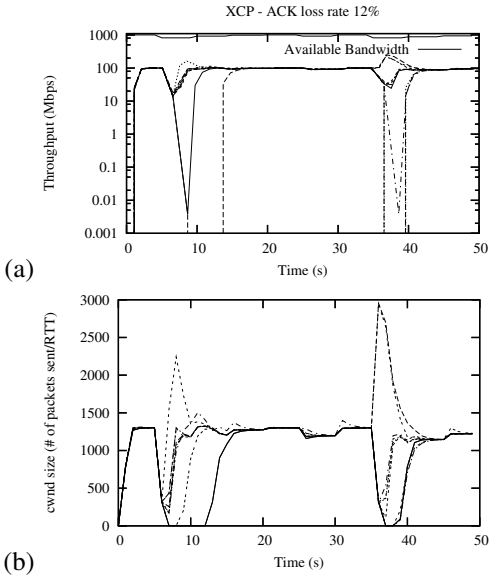

Fig. 3. 10 XCP flows, 1Gbps, 100ms RTT, 12% lost ACKs

The sender could also received ACK packets from the receiver for the in-transit data packets sent before the congestion. Unfortunately, these feedback values are small (because the congestion window was previously large) and do not allow $cwnd$ (previously set to 1) to increase fast enough to grab for the available bandwidth. In addition, if it happens that an important amount of dropped packets belongs to the same flow, then the probability to have a timeout increases while the others flows will try to get the newly available bandwidth. We can verify this assumption in the figure at time 5s and 30s where the losses of ACKs have more impact on some flows while other flows try to get all the available bandwidth. For instance, at time 5s one flow remains inactive for more than 6s.

With the same simulation settings, figure 4 shows the performance of XCP-r where we can see that the timeouts at time 5s and 30s are avoided which represents an important improvement over the original XCP protocol. XCP-r succeeds in maintaining fairness between flows which translates into a more stable evolution of the congestion window which was not the case with XCP.

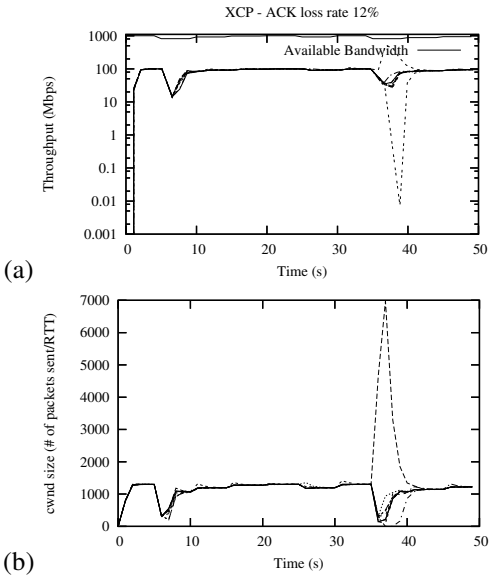

Fig. 4. 10 XCP-r flows, 1Gbps, 100ms RTT, 12% lost ACKs

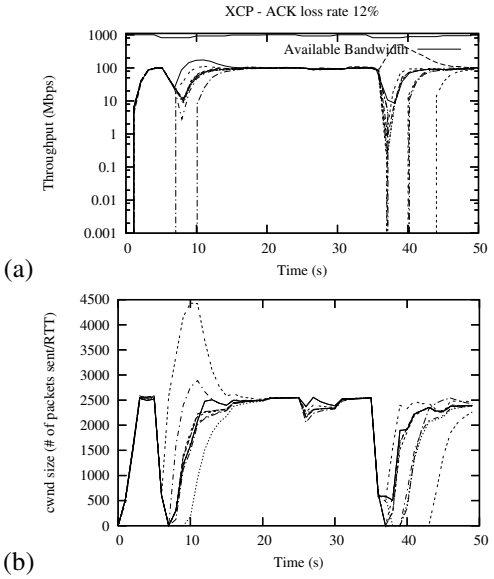

Fig. 5. 10 XCP flows, 1Gbps, 200ms RTT, 12% lost ACKs

On a 200ms RTT bottleneck link, XCP-r still performs better than XCP (see figure 5 for XCP and figure 6 for XCP-r) although the timeouts can not be totally avoided: in 50s, XCP-r sends with 10 flows 5157.458 MBytes of data whereas XCP only sends 5102.866 MBytes. However, the main improvement of XCP-r over XCP is the better fairness between flows. Figure 7 shows the amount of MBytes transfered per flow and the standard deviation of the set of results for XCP and XCP-r. The rounder the curve, the more fairness the protocol achieves.

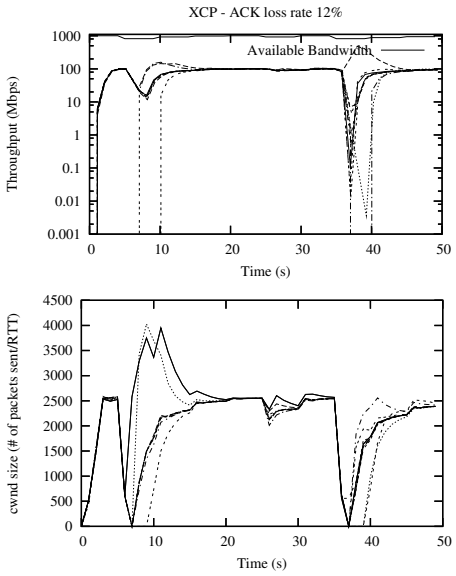

Fig. 6. 10 XCP-r flows, 1Gbps, 200ms RTT, 12% lost ACKs

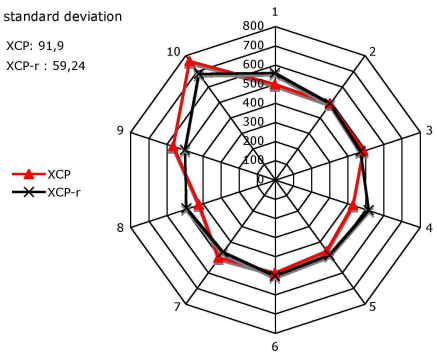

Fig. 7. Mbytes transfered per flow and standard deviation for XCP and XCP-r.

IV. Comparing XCP and XCP-r with TCP and HSTCP

In this section, comparisons with TCP and HSTCP are provided with the RTT set to 200ms. We tuned the TCP and HSTCP ns simulation model to the networking conditions: (i) the maximum window size is set to 25000 packets to avoid the congestion window to be limited in its growth, (ii) the initial slow-start threshold is set to 1000 packets for HSTCP (`max_ssthres_` parameter) to allow better startups for flows with very high congestion windows[1].

Figure 8 shows that XCP-r performs better than the other protocol: the amount of MBytes transfered in 50s is well above TCP and HSTCP. In figure 9, we plot the amount of MBytes transfered for each flow. We can see that both TCP and HSTCP suffer from fairness problem among flows[2].

[1] See http://www.icir.org/floyd/hstcp/slowstart/Slowstart.notes

[2] These are well-known problems of TCP and HSTCP in high-speed networks.

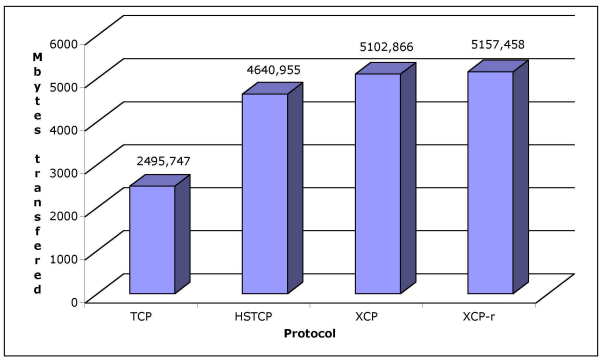

Fig. 8. Mbytes transfered: TCP, HSTCP, XCP and XCP-r.

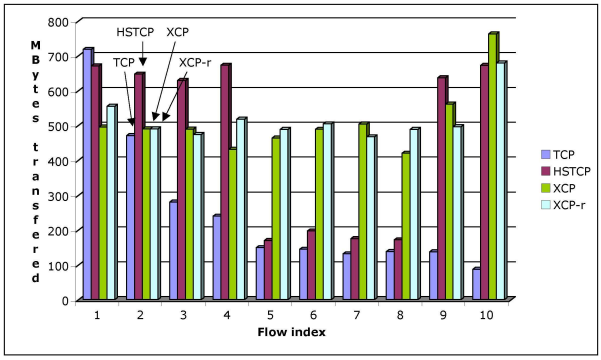

Fig. 9. Mbytes transfered per flow.

Figure 10 highlights this fairness problem and shows that XCP-r is a real improvement over TCP and HSTCP since the standard deviation computed on the set of flows for XCP-r is very small. XCP-r succeeds in providing an efficient and reliable data transport service for dynamic, high-speed network infrastructures such as those found in grid testbeds.

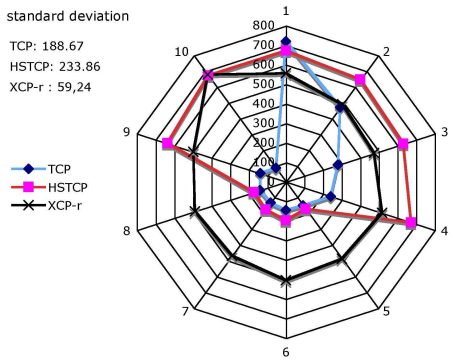

Fig. 10. Mbytes transfered per flow and standard deviation for each protocol.

V. Conclusion

This article addresses the problem of transferring large amount of data on dynamic, very high-speed networks. In such environments, end-to-end solutions show their limits in terms of fairness and efficiency. In this paper, we show simulation results of XCP and XCP-r that show their superiority on TCP and HSTCP both in term of fairness and efficiency. XCP-r, which is an improvement of XCP, succeeded in providing a high level of performance thus able to function in a broader range of network conditions. We believed such router-assisted approaches have high potentials for being deployed in dynamic, very high-speed networking infrastructures such as data or computational grids where efficient transfers of large amount of data is required.

References

[1] The network simulator ns-2. http://www.isi.edu/nsnam/ns.
[2] W. Allcock, editor. GridFTP Protocol Specification (Global Grid Forum Recommendation GFD.20). March 2003.
[3] C. Barakat, E. Altman, W. Dabbous. On TCP performance in a heterogeneous network: a survey. IEEE Communication Magazine, January 2000.
[4] Cerf, V., and R. Kahn. A Protocol for Packet Network Intercommunication. IEEE Transactions on Communications, 22(5), May 1974.
[5] S. Floyd. HighSpeed TCP for Large Congestion Windows. RFC 3649, Experimental, December 2003.
[6] L. A. Grieco, S. Mascolo. Performance evaluation and comparison of Westwood+, New Reno and Vegas TCP congestion control. ACM CCR, 34(2), April 2004.
[7] G. Hasegawa, M. Murata. Survey on fairness issues in TCP congestion control mechanisms. IEICE Transactions on Communications, January 2001.
[8] V. Jacobson. Congestion avoidance and control. ACM SIGCOMM 1988.
[9] C. Jin, D. X. Wei, S. H. Low. FAST TCP: Motivation, Architecture, Algorithms, Performance. IEEE Infocom 2004.
[10] D. Katabi, M. Handley, C. Rohrs. Congestion Control for High Bandwidth-Delay Product Networks. ACM SIGCOMM 2002.
[11] R. King, R. Baraniuk, R. Riedi. TCP-Africa: An Adaptive and Fair Rapid Increase Rule for Scalable TCP. IEEE Infocom 2005.
[12] S. H. Low, L. Andrew, B. Wydrowsk. Understanding XCP: Equilibrium and Fairness. IEEE Infocom 2005.
[13] D.M. Lopez-Pacheco. Robust Transport Protocol for Dynamic High-Speed Networks: enhancing the XCP approach. Technical report.
[14] R. Wang et al. Adaptive Bandwidth Share Estimation in TCP Westwood. Globecom 2002.
[15] L. Xu, K. Harfoush and I. Rhee Binary Increase Congestion Control (BIC) for Fast Long-Distance Networks. IEEE Infocom 2004.
[16] Y. Zhang and T. Henderson An Implementation and Experimental Study of the eXplicit Control Protocol (XCP). IEEE Infocom 2005.