Robust Transport Protocol for Dynamic High-Speed Networks: enhancing the XCP approach

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*— Transport protocols have the difficult task of providing reliability and fair sharing of the bandwidth to end-users. In this paper, we focus on very dynamic high-speed networks 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. Therefore end-to-end solutions show their limitations for exploiting the current very high-speed 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 ACKs to adapt its congestion window size, XCP performances can be affected if many losses occur on the reverse path. This paper proposes to calculate the congestion window size at the receiver side and to overcome the problem of ACk losses. The result is a more robust XCP transport protocol, capable of achieving a high level of performance in very dynamic high-speed networks, thus able to function in a broader range of network conditions.**

I. Introduction

Transport protocols have usually the difficult task of providing reliability and fair sharing of the bandwidth to end-users. TCP (Transmission Control Protocol) originally defined in RFC 793 has been the transport protocol of the Internet for more than 2 decades. Since the congestion collapse observed by V. Jacobson in 1986 and the well-known slow-start and congestion avoidance algorithms proposed in 1988 [7], 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 [2], [6]: wireless links [11], [5], asymmetric links and very high-speed and long delay links [4], [8], [9], [12]. 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.

On high bandwidth-delay product networks, the main optimizations consist in adding more efficient mechanisms for acquiring bandwidth faster. For example HSTCP [4] modifies the standard TCP response function to faster acquire the available bandwidth (more efficiency) and to quickly recover 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. FAST TCP [8] 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 [9] 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.

In this paper, we focus on very dynamic high-speed networks where the available best-effort bandwidth can vary over time. There are several reasons to this behavior. For example, it is not unrealistic to foreseen dynamic subscriptions to guaranteed bandwidth for resource-consuming applications (grid computing, distributed simulation, virtual reality experiences,...). 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 traffics 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 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 [13], 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. This paper proposes to calculate the congestion window size at the receiver side and

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

to overcome the problem of ACk losses. The result is a more robust XCP transport protocol, capable of achieving a high level of performance in very dynamic high-speed networks, thus able to function in a broader range of network conditions.

The paper is organized as follows. Section 2 describes the XCP protocol and presents some of its main problems on dynamic networks and especially those related to the high dependancy of XCP on ACK packets. Section 3 presents the performances of XCP on dynamic networks. Our contributions are described in Section 4. Section 5 concludes the paper.

II. The XCP protocol

A. General description

XCP [9] (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` field), the estimated RTT and a feedback field `H_feedback`. 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 Fig. 1). Basically, the `H_feedback` field which can take positive or negative values represents the amount by which the sender's congestion window size is increased or decreased. On reception of data packets, the receiver copies the congestion header (which has been modified accordingly by the routers) into ACK packets sent back to the source. It is not important that these ACK packets follow the same path than data packets since all the computations are done on the forward data path. On reception of ACK packets, the sender would update its congestion window size as follows: $cwnd = max(cwnd + H\_feedback, packetsize)$, with $cwnd$ expressed in bytes. The core mechanism resides in XCP routers that use an efficiency controller (EC) and a fairness controller (FC) to update the value of the 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, and to the persistent queue size Q (to avoid a feedback value of zero when input traffic is equal to output capacity).

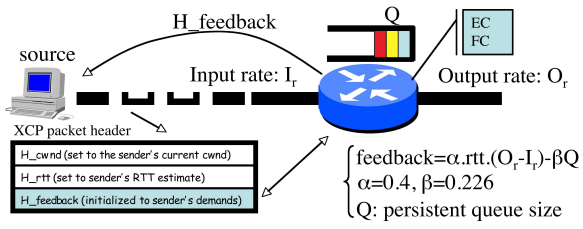

Fig. 1. The XCP protocol

The authors in [9] proposes the following EC equation: $feedback = \alpha.rtt.S - \beta.Q$, with $\alpha = 0.4$ and $\beta = 0.226$. 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, but decoupled from drops because only the difference between input traffic rate and output link capacity (S) is used instead in the EC. Note that no per-flow states are used by XCP routers to perform all these operations: as a data packet carries in its header the current sender $cwnd$ and the RTT, it is easy to compute how many data packets are sent per congestion window in order to assign the available bandwidth in a proportional manner.

The original XCP proposition did not mention any mechanism for handling severe congestion situations as it was believe that such situations should not occur with the XCP kind of control. However, some works have shown that severe congestions do happen and that it is desirable to keep the TCP mechanism which consists in resetting $cwnd$ to 1 in case of severe congestion[1] [13], [10]. Our simulations did confirm this assumption and therefore we assume that XCP does react as TCP does in case of severe congestion. The previous XCP studies have also shown that XCP has very good convergence time to full utilization and fairness to other flows. Some implementations are also available and deployment issues are being studied.

B. The importance of ACKs in XCP

The problem we are tackling in this work is the high dependancy of XCP on ACK packets sent on the reverse path. As opposed to TCP where ACK packets indicate good reception of data packets, each XCP ACK packet also carries the feedback value filled in by the XCP routers that controls the evolution of $cwnd$ at the sender. This feedback represents an increment or a decrement to be applied to $cwnd$. The larger the congestion window, the higher the number of ACKs and, the smaller the feedback value per ACK is (see [9] for more details).

With TCP, if some ACK are lost the only consequence is to delay the release of data buffers at the sender. For XCP, lost ACKs will cause a mismatch between the real network conditions in term of bandwidth availability and the conditions perceived by the source. The direct consequence is either a too large $cwnd$ while there are bandwidth shortage somewhere in the network, or a too small $cwnd$ that decreases utilization while bandwidth is available. While the second case only affects the utilization of the link, the first case has more dramatic consequences: as the sender $cwnd$ increases well beyond the optimal point, and does not reflect the network conditions anymore, severe congestions are created inside the network. While this phenomenon has little impact on slow networks it is amplified in dynamic high-speed networks where large amount of data could be sent per congestion window and where bandwidth reductions could occur. In this case, if the returned feedbacks (those that have not been dropped) are not

[1] However, as the original ns model of XCP was implemented on top of the TCP model, the XCP simulation model did benefit from this TCP mechanism.

sufficient to reduce $cwnd$ at the sender, then large amount of data can be dropped causing costly retransmissions but also very penalizing timeouts.

III. Performance of XCP on dynamic networks

With XCP, $cwnd$ can directly jump to the optimal value without wasting time in a slow-start phase or in a slow probing phase with 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. On dynamic networks where the available bandwidth can vary over time, XCP is also expected to achieve good performances. In this section, we show some ns-based [1] simulation results of XCP on such dynamic networks. The original XCP ns code is the one provided by D. Katabi. In our simulations scenario, the bandwidth variations for best effort traffics are due to ON/OFF UDP sources which compete with the XCP flows.

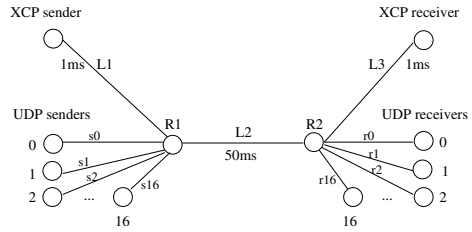

Fig. 2. Network topology

Fig. 2 shows the classical dump-bell network model with 2 XCP routers used for our ns-based 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 1ms. R1 is connected to R2 by either a 200Mbps or a 1Gbps link with 50ms of one-way delay which represents the bottleneck link.

R1 and R2 have 1000 buffers slots (expressed in number of packets) per incoming link (2300 with the 1Gbps link). On a long, high-speed networks, the optimal buffer size usually follows the rule of thumb of $bandwidth.rtt$ product, e.g. for a 200Mbps link with an RTT of 100ms the buffer in number of 1Kbytes packets is about 2440. For a 1Gbps link it 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 [9].

A. XCP with no lost ACKs

Fig. 3 shows the throughput for one XCP connection. As expected, XCP is able most of the time to follow the best-effort bandwidth variations. The only problem we could observe is at time 5s when the bandwidth decreases by half. At this time, there is a timeout introduced by the numerous losses of data packets but it is quickly recovered as $cwnd$ is increased almost instantaneously afterwards.

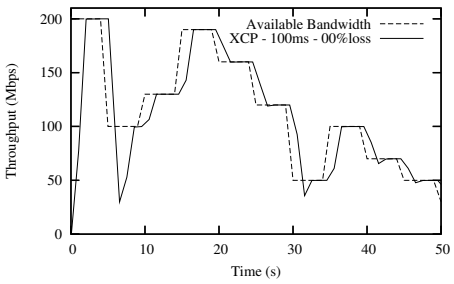

Fig. 3. 1 flow, 200Mbps, no lost ACKs

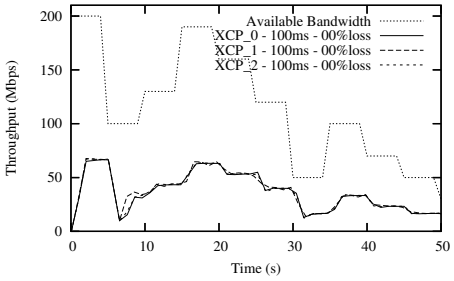

Fig. 4. 3 flows, 200Mbps, no lost ACKs

With several flows, XCP is able to follow the bandwidth variations as can be seen in Fig. 4.

B. XCP with lost ACKs on the reverse path

When losses occur on the reverse path for some reasons (congestions, bandwidth variations…), XCP ACK packets may be dropped. As explained previously this situation may lead to an unstable behavior of XCP.

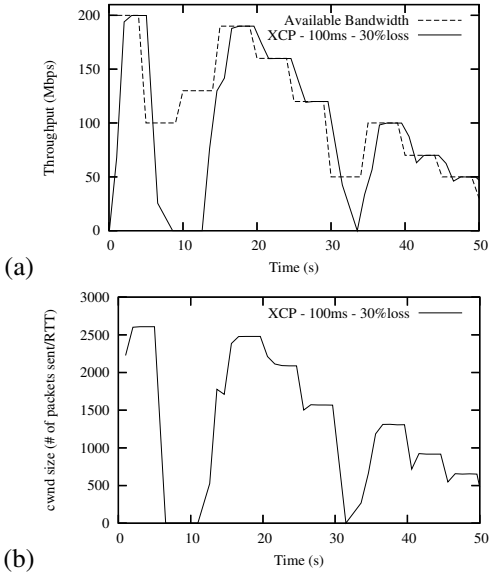

Fig. 5. 1 flow, 200Mbps, 30% lost ACKs

Fig. 5a and 5b show the sender's throughput and $cwnd$ respectively when there is an ACK loss rate of 30%[2]. As

[2]30% of lost ACK is not an unusual situation since it is possible to have some kind of black-out situation on the reverse path due to severe congestions. In this case, 100% of ACK could be lost during the blackout period.

opposed to the case shown in Fig. 3, when the bandwidth is reduced by half and causes a congestion at time 5s we can see in Fig. 5a that the timeout could not be recovered quickly. The explanation is the following: on a timeout the sender $cwnd$ is set to 1 therefore the receiver could only send back 1 ACK carrying a large feedback value if there is available bandwidth. If this ACK is dropped[3], 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.

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. At time 30s, there is another timeout which is quickly recovered this time because the RTO has a much more optimal value than at the beginning of the connection.

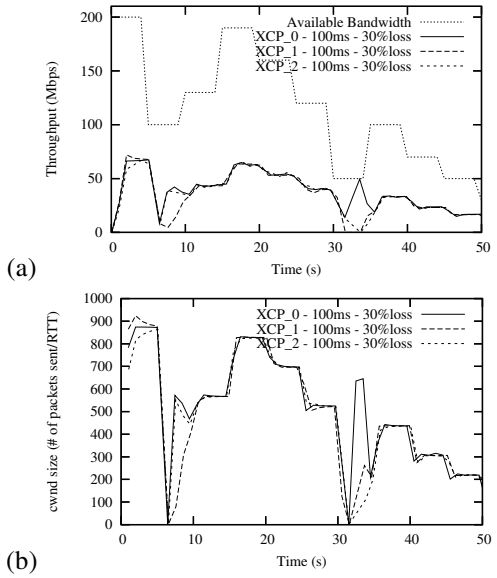

Fig. 6. 3 flows, 200Mbps, 30% lost ACKs

With several flows, each sender has a smaller $cwnd$ but the lost ACKs can produce an important number of dropped packets when the available bandwidth decreases. This is because the total number of data packets stored in the routers is similar to the 1-connection case. 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 for this flow while the others flows will try to get the newly available bandwidth. We can verify this assumption in Fig. 6a and 6b which show the throughput and $cwnd$ for 3 XCP flows. At time 5s the timeout is quickly solved but at time 30s the losses of ACKs have more impact on the second and third flow while the first flow tries to get all the available bandwidth. The second flow stays inactive for about 4s.

[3]An important ACK can be dropped even with a much lower loss rate.

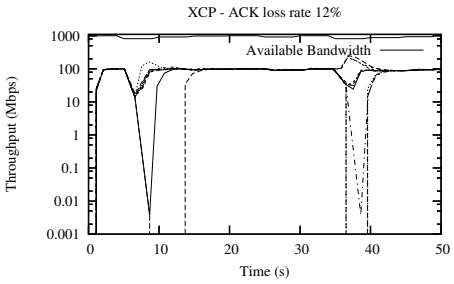

Fig. 7. 10 flows, 1Gbps, 12% lost ACKs

We also simulated XCP on a 1Gbps link with a smaller ACK loss rate (12%). Fig. 7 shows 10 XCP flows on a dynamic network 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 timeouts and packet drops.

IV. Enhancing the robustness 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 $cwnd$. The reasons for doing so are (i) lost of ACKs have little impact and (ii) feedback values can be changed very quickly by the routers to take into account any changes in the networks. However, in the previous sections, we have shown that on dynamic networks XCP could have unstable behavior due to lost of ACKs. In this section, we are describing the modifications we did to the XCP sender and receiver while keeping the complex router's tasks untouched. We will call this version XCP-r ("r" for receiver).

The main idea behind XCP-r is to avoid having the incremental feedbacks summed up by the source, while keeping the incremental feedbacks at the router level for flexibility and robustness. This paper proposes to calculate the congestion window size at the receiver and to send this value to the source in a redundant manner. As each ACK packet now carries the target $cwnd$, the only consequence of some ACKs losses is to delay the update of the sender $cwnd$ since any subsequent single ACK packet can be used.

In a first version of XCP-r, the receiver simply uses the `H_cwnd` field of the data packet (which stores the current sender's congestion window size) to recompute the target $cwnd$ from the `H_feedback` field (see Fig. 1). The formula was $cwnd = (H\_feedback * H\_cwnd) + H\_cwnd$ which simply relies on the fact that the `H_feedback` field was computed by the routers so as to distribute the total increase amount in congestion window into `H_cwnd` increments of `H_feedback` value. Unfortunately, the simulations we made show that this formula can over-estimate the congestion window size because it assumes that the `H_feedback` value remains constant within a congestion window.

Our proposition in this paper is to reproduce 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 created on reception of an XCP SYN packet that has the same meanings than a TCP SYN packets. Therefore, ACK packets sent by the receiver carry the new congestion window size instead of an increment or a decrement value.

Note that whenever the source decides to modify $cwnd$, it must synchronize with the receiver beforehand. In XCP, such a modification happens only when there is a congestion that either set $cwnd$ to 1 or to half of its value. In both cases, the sender indicates in an additional special *congestion flag* field of the XCP header that the receiver should read the value of `H_cwnd` again. Fig. 8 illustrates the XCP-r receiver's set of operations.

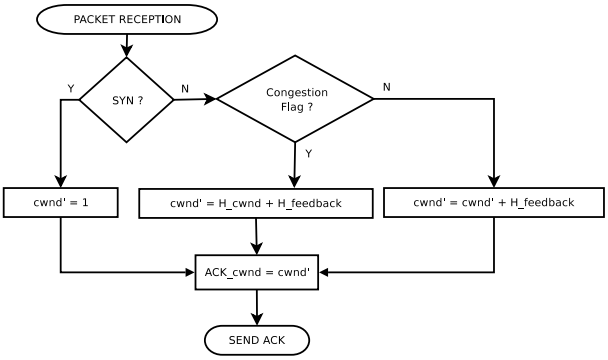

Fig. 8. A receiver in XCP-r.

Regarding the XCP-r source, the only modifications are (i) instead of doing $cwnd = cwnd + H\_feedback$ simply do $cwnd = H\_feedback$ and, (ii) set the *congestion flag* whenever the $cwnd$ is reset. The next subsections present the performances of XCP-r.

A. XCP-r with no lost ACKs

When there is no lost ACKs XCP-r is similar to XCP without any side effects.

B. XCP-r with lost ACKs on the reverse path

With the same simulation settings, Fig. 9 shows that the impact of the timeout at time 5s is reduced and that the period of inactivity is almost negligible which represents an important improvement over the original XCP protocol. After the timeout, the computed feedback is very important because the value needs to be large in order to quickly grab the available bandwidth. With XCP, if the ACK which carries the largest increment value is lost, the sender will not be able to increase its congestion window and the timeout will last longer (see the case depicted in Fig. 5). XCP-r reduces this problem because the next ACK will carry the correct value for the congestion window since $cwnd'$ remains unchanged at the receiver side.

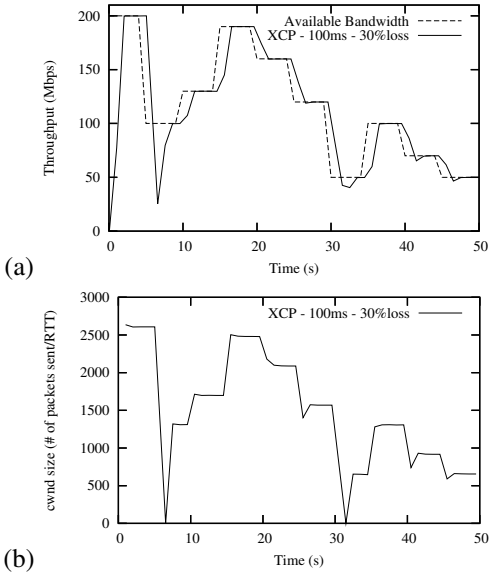

Fig. 9. 1 XCP-r flow, 30% lost ACKs

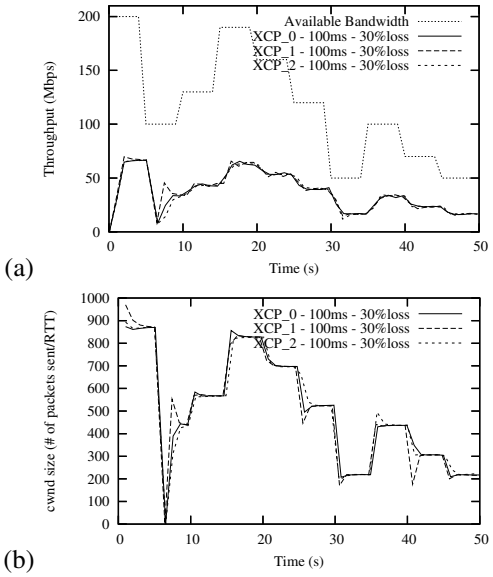

Fig. 10. 3 XCP-r flows, 30% lost ACKs

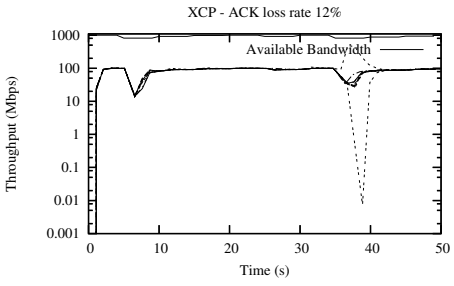

Fig. 11. 10 XCP-r flows, 1Gbps, 12% lost ACKs

With several XCP flows (Fig. 10a and 10b), we can see that XCP-r succeeds in maintaining fairness between flows

which directly translates into a more stable evolution of the congestion window than with XCP. On a 1Gbps link, Fig. 11 shows that XCP-r maintains for the 10 flows a high level of performance.

V. Conclusion

This article focuses on the XCP approach which is a promising router-assisted approach for high-speed networks. The problem we addressed in this paper is the high dependance of XCP with regards to the returned ACK packets for maintaining a coherent view of the network conditions. We showed that this dependance has high impact on the XCP performances in case of ACK losses on the reverse path, making XCP very unstable if used as it is on dynamic, very high-speed networks.

In this paper we propose to calculate the congestion window size at the receiver side and to overcome the problem of ACk loss. We showed that the resulting protocol succeeds in providing a high level of performance in very dynamic high-speed networks, 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] C. Barakat, E. Altman, W. Dabbous. On TCP performance in a heterogeneous network: a survey. IEEE Communication Magazine, January 2000.

[3] Cerf, V., and R. Kahn. A Protocol for Packet Network Intercommunication IEEE Transactions on Communications, 22(5), May 1974.

[4] S. Floyd. HighSpeed TCP for Large Congestion Windows. RFC 3649, Experimental, December 2003.

[5] L. A. Grieco, S. Mascolo. Performance evaluation and comparison of Westwood+, New Reno and Vegas TCP congestion control. ACM CCR, 34(2), April 2004.

[6] G. Hasegawa, M. Murata. Survey on fairness issues in TCP congestion control mechanisms. IEICE Transactions on Communications, January 2001.

[7] V. Jacobson. Congestion avoidance and control. ACM SIGCOMM 1988.

[8] C. Jin, D. X. Wei, S. H. Low. FAST TCP: Motivation, Architecture, Algorithms, Performance. IEEE Infocom 2004.

[9] D. Katabi, M. Handley, C. Rohrs. Congestion Control for High Bandwidth-Delay Product Networks. ACM SIGCOMM 2002.

[10] S. H. Low, L. Andrew, B. Wydrowsk. Understanding XCP: Equilibrium and Fairness. IEEE Infocom 2005.

[11] R. Wang et al. Adaptive Bandwidth Share Estimation in TCP Westwood Globecom 2002.

[12] L. Xu, K. Harfoush and I. Rhee Binary Increase Congestion Control (BIC) for Fast Long-Distance Networks. IEEE Infocom 2004.

[13] Y. Zhang and T. Henderson An Implementation and Experimental Study of the eXplicit Control Protocol (XCP). IEEE Infocom 2005.