# XCP-i : eXplicit Control Protocol for heterogeneous inter-networking of high-speed networks

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

L. Lefèvre
INRIA RESO/LIP, France
Email: laurent.lefevre@inria.fr

***Abstract*— XCP is a transport protocol that uses the assistance of specialized routers to very accurately determine the available bandwidth along the path from the source to the destination. In this way, XCP efficiently controls the sender's congestion window size thus avoiding the traditional slow-start and congestion avoidance phase. However, XCP requires the collaboration of all the routers on the data path which is almost impossible to achieve in an incremental deployment scenario of XCP. It has been shown that XCP behaves badly, worse than TCP, in the presence of non-XCP routers thus limiting dramatically the benefit of having XCP running in some parts of the network. In this paper, we address this problem and propose XCP-i which is operable on an internetwork consisting of XCP routers and traditional IP routers without loosing the benefit of the XCP control laws. The simulation results on a number of topologies that reflect the various scenario of incremental deployment on the Internet show that although XCP-i performances depend on available bandwidth estimation accuracy, XCP-i still outperforms TCP on high-speed links.**

## I. Introduction

In the Internet world, 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 [6], the networking community has proposed many enhancements to the original proposition in order to make TCP more efficient in a large variety of network conditions ([1], [5]) and technologies such as wireless links ([14], [3]), satellite, etc. On high-speed networks where the link capabilities can be in the order of several gigabits/s (usually referred to as high bandwidth-delay product networks) TCP need to be tuned to the new networking conditions (socket buffer size, maximum congestion window size,...) but remains limited by the slow increase of the congestion window during the congestion avoidance phase. On these high bandwidth-delay product networks, a number of new propositions have been made [2], [8], [9], [15] which mainly consist in adding more efficient mechanisms for acquiring bandwidth faster. For example, HSTCP [2] modifies the standard TCP response function to both faster acquire the available bandwidth and to faster recover from packet losses in the network. The main drawback of such a behavior is that fairness between TCP and HSTCP flows, 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 round-trip time variation to predict congestion 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] is a router-assisted approach that use the assistance of routers to more accurately signal congestion 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 thus maximizing the utilization of high-speed links, while preserving fairness among XCP flows.

XCP is therefore a promising approach on very high-speed networks and several studies have analytically shown the performances of XCP [10], proposed enhancements to XCP for making it more robust to packet losses on the reverse path [11] and performed extensive experimental measures on a UNIX-based implementation [16]. In most of these studies, the problem of incremental deployment of XCP has been discussed as XCP requires the collaboration of all the routers on the data path. It has been shown that XCP behaves badly, worse than TCP, in the presence of non-XCP routers thus limiting dramatically the benefit of having XCP running in some parts of the network. In this paper, we address this problem and propose enhancements to XCP to make it operable on an internetwork consisting of XCP routers and traditional IP routers without loosing the benefit of the XCP control laws. The simulation results on a number of topologies that reflect the various scenario of incremental deployment on the Internet show that our modifications are efficient while keeping the core of the XCP control laws unchanged.

The paper is organized as follows. Section 2 reviews the XCP protocol and presents the problem of XCP's sensitivity to non-XCP routers. Section 3 presents the design objectives and the mechanisms we propose for detecting non-XCP clouds and take into account the non-XCP resources. Section 4 shows the simulation results. Section 5 discusses some limitations and the open issues while section 6 concludes our article.

## 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), 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. 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 $Q$.

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 believed that such situations should not occur with the XCP kind of control laws. 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] [16], [10]. Our simulations did confirm this assumption and therefore we assume that XCP does react as TCP does in case of severe congestion.

### B. Sensitivity to non-XCP routers

Since XCP relies on specialized routers to estimate the available bandwidth all along the path from the source to the destination, it can easily be foreseen that XCP will behave badly if there are non-XCP routers on the path with bottleneck link capacities (the term **non-XCP router** will refer to a traditional IP router, e.g. DropTail, RED, etc, with no XCP functionalities. An **non-XCP cloud** is a continuous set of $n$ non-XCP routers, $n \geq 1$.). Moreover, we can also predict that XCP will perform worse than TCP in this case because the feedback computation will only take into account the XCP elements on the path, ignoring the existence of the bottleneck link. This assumption has been first illustrated in [16] and we review below some simulation results exhibiting this problem for the purpose of making our paper clearer to the reader. Figure 1 presents 3 scenario: (a) shows a typical Internet network with non-XCP routers, (b) shows an all-XCP network with 100% XCP-routers and (c) shows a more realistic scenario of an incremental deployment of XCP around a non-XCP router. In all these scenario, the bottleneck capacity is 30 Mbps while the other links have a capacity of 80 Mbps.

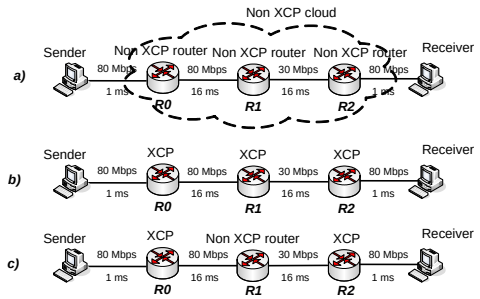


Fig. 1. (a) scenario for TCP, (b) and (c) scenario for XCP.

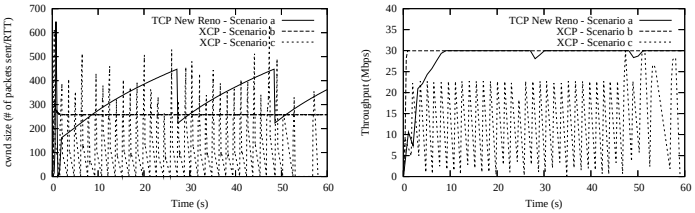


Fig. 2. Congestion window evolution and Throughput for scenario a,b,c.

Figure 2 shows the behavior of one TCP flow on scenario (a) and one XCP flow on scenario (b) and (c). The congestion window evolution (left figure) shows the typical saw-tooth curve of TCP and the typical XCP curve that directly jumps to the optimal congestion window size (no packet losses). For XCP on scenario (c), the congestion window size is very unstable and frequently goes well beyond the maximum value found by the linear search of TCP congestion avoidance mechanism, causing a high amount of packet losses. The explanation is as follows: since the non-XCP router is unable to update the feedback value carried in XCP packets to indicate the bottleneck, the XCP router that immediately follows the non-XCP router uses a feedback value that reflects the available bandwidth outside the non-XCP cloud, which is much greater than the 30 Mbps of the bottleneck in our scenario. In these $ns$-based simulations, TCP on scenario (a) successfully sent 215.004 MBytes, XCP on scenario (b) sent 223.808 MBytes and XCP on scenario (c) sent only 52.426 MBytes during one minute !

[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.

## III. Enhancing XCP for heterogeneous internetworking

We have seen in the previous section that XCP performs badly with non-XCP routers in the data path. This section describes the mechanisms we propose to make XCP operational in an incremental deployment scenario. We will call XCP-i this XCP version, the $i$ letter standing for *interoperable*. We will then use the **XCP-i router** term to refer to an XCP-capable router with *interoperable* functionalities. While extending XCP for internetworks it is desirable to keep the changes to a minimum and especially keep the core of the XCP's control laws unchanged. One main reason for doing this is because there are already some XCP implementations available (which have shown that the XCP computations are not trivial to implement [16]) and therefore major changes in the protocol require a lot of time in new software development. Also, XCP-i tries to maintain the XCP philosophy which is to avoid keeping state variables per flow.

The XCP-i algorithm introduces 2 main new functionalities: $(i)$ detects when an XCP packet has gone through a non-XCP cloud and $(ii)$ takes into account the available bandwidth in the non-XCP cloud in the feedback computation. We will in the following subsections present how these new functionalities have been incorporated into the XCP protocol while keeping the core of the XCP control laws unchanged.

### A. XCP-i : architecture and algorithm in routers

*1) Detecting non-XCP clouds:* XCP-i detects non-XCP clouds by using the TTL counter (defined in the RFC 791). We suppose that all routers in the network support the regular TTL operations, especially the one that decreases the TTL's value in the IP packet header before forwarding the packet. With this assumption, we add a new field in the XCP packet header (which is different to the IP header) named `xcp_ttl_` which is decremented only by XCP-i routers. TTL and `xcp_ttl_` have to be initialized by the sender with the same value. In this way, on an all-XCP network, the TTL and `xcp_ttl_` fields will always have the same value. When an XCP-i router receives a packet with the TTL field smaller than the `xcp_ttl_` field, it can conclude that the packet has gone through a non-XCP cloud. After processing packet, the XCP-i router will update $xcp\_ttl\_ = TTL$ in order to hide this non-XCP cloud to the others XCP-i routers and to detect new non-XCP clouds if they are present in the data path. This solution is simple, does not require any special message between the routers and the overhead for processing this additional field is small.

*2) Detecting the XCP-i edge routers:* When an non-XCP cloud has been detected by an XCP-i router, XCP-i requires the identity of the first XCP-i router before the non-XCP cloud to be known. The reason is because XCP-i will then try to determine the available bandwidth between the 2 XCP-i routers located at the edge of the non-XCP cloud. In order to discover the upstream XCP-i edge router, we add a new field in the XCP packet header named `last_xcp_router_` which contains the IP address of the last XCP-i router that has processed the XCP packet. An XCP-i router would simply put its own IP address in this field prior to send the packet on the wire. In this way, when a non-XCP cloud is detected by an XCP-i router, this router will automatically know which XCP-i router is located at the other side of the non-XCP cloud. Once again, this solution is simple, does not require any special message between the XCP-i routers and the CPU usage to process this additional field is kept to a minimum.

*3) Determining the bottleneck bandwidth:* Let's note by XCP-i$_{k-1}$ and XCP-i$_k$ the 2 XCP-i edge routers of the non-XCP cloud. The idea in the XCP-i algorithm is to initiate a bandwidth estimation procedure at the XCP-i$_{k-1}$ router. To do so, XCP-i$_k$ sends a request to XCP-i$_{k-1}$ and waits for an acknowledgment of its request during a $xcp\_req\_timeout$ time period. If this acknowledgment does not arrive the process is restarted. After 3 unsuccessful requests, XCP-i$_k$ concludes that the path between XCP-i$_{k-1}$ and XCP-i$_k$ is broken. The bandwidth estimation procedure will only be restarted on reception of a new packet from XCP-i$_{k-1}$. Now, upon reception of a request, XCP-i$_{k-1}$ will acknowledge the request and will try to find the available bandwidth, $BW_{k-1,k}$, between XCP-i$_{k-1}$ and XCP-i$_k$. Many algorithms has been proposed in the literature to do this (e.g. packet pair, packet train, etc...), and we will only suppose that the router will implement one of these to find the most accurate value (for instance in [13] the authors reported that pathchirp [12] or pathload [7] present very accurate bandwidth estimations, without producing a lot of load in the network [2] ). After having obtained $BW_{k-1,k}$, XCP-i$_{k-1}$ will send it to XCP-i$_k$ which will add an entry in a hash table based on XCP-i$_{k-1}$'s IP address to record the available bandwidth between XCP-i$_{k-1}$ and XCP-i$_k$. Then the bandwidth estimation procedure should be performed periodically at a given frequency. This procedure should be stopped after an inactivity period of XCP-i$_{k-1}$ and the corresponding entry in the hash table should be removed in order to keep the hash table as small as possible.

Note that it is important that XCP-i$_k$ stores the available bandwidth (and therefore performs the feedback computation as this will be explained in the next section) and not XCP-i$_{k-1}$, because XCP-i$_{k-1}$ is unable to distinguish between flows that go through the non-XCP cloud to XCP-i$_k$ from those that go to another XCP-i router through the same non-XCP cloud (see figure 3 for an example). This is why XCP-i$_{k-1}$ communicates the available bandwidth to XCP-i$_k$ even though this is XCP-i$_{k-1}$ which computes it.

This solution does not keep any state per flow: only 1 entry in the hash table needs to be kept per upstream XCP-i router.

*4) The XCP-i virtual router:* When XCP-i$_k$ receives a packet that has gone through a non-XCP cloud, and if an available entry $BW_{k-1,k}$ exists in the hash table for `last_xcp_router_` XCP-i$_k$ will use a virtual router, XCP-iv$_k$, to compute a feedback that will reflect the network condition in the non-XCP cloud. The purpose of the virtual router is to emulate an XCP-i router located upstream from XCP-i$_k$ with a virtual output link connected to XCP-i$_k$ which

[2] bursts for only a few milliseconds.

capacity is the available bandwidth found in the non-XCP cloud. Figure 3 shows the logical architecture of the XCP-i$_k$ router with one virtual router per non-XCP cloud. We can view the virtual router as a logical entity that replaces the non-XCP cloud. The equation to compute the feedback in XCP-iv is similar to the one of XCP (and therefore the same code could be reused):

$$feedback_{XCP-iv_k} = \alpha.rtt.BW_{k-1,k} - \beta.Q \qquad (1)$$

Rules for setting $\alpha$ and $\beta$ are the same than for XCP. $rtt$ and $Q$ are respectively the average RTT on all the incoming packets and the persistent queue size in the XCP-i router which contains the XCP-iv virtual routers. In equation (1) $BW_{k-1,k}$ replaces $S$ in the XCP's original equation therefore the virtual router does not need to know the amount of input traffic (see section II-A). Once the feedback is updated by the virtual router, XCP-i$_k$ will start its normal feedback computation as usual.

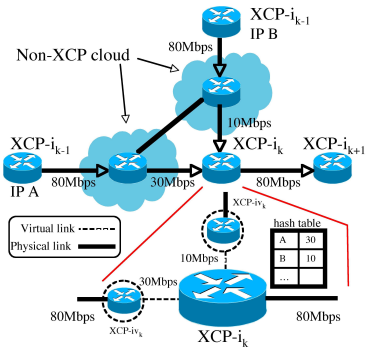


Fig. 3. An XCP-i router with 1 virtual router per non-XCP cloud.

### B. XCP-i : architecture in end-hosts

It is possible that during an incremental deployment of XCP, either the source or the receiver, or both, are not directly connected to an XCP router. For example, figure 4 shows an asymmetric deployment scenario where XCP-i routers are deployed near the receiver side with a non-XCP cloud at the sender side.

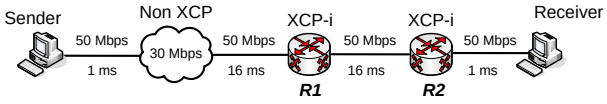


Fig. 4. Asymmetric deployment: optimized receiver side

In these cases, some parts of the XCP-i algorithm must also be supported by the end-hosts. If the XCP-i router is located at the receiver side (figure 4), the sender must be able to initiate a bandwidth estimation procedure upon reception of a request from the first XCP-i on the path. When the XCP-i router is located at the sender side, the receiver can either act as an XCP-i router by implementing both non-XCP cloud detection and feedback computation, or, if this solution is not desirable, it could ask the last XCP-i router to compute a feedback value corresponding to the non-XCP cloud's bottleneck value. We believe that this last solution is more complex than the first one, which has the benefit of simply duplicating the XCP-i code in the receiver's XCP protocol stack since the input traffic rate does not need to be known when an estimation of the available bandwidth is provided (see section III-A.4).

## IV. Simulation results

XCP-i has been simulated with $ns$ by extending Katabi's XCP simulation model in order to incorporate the enhancements of XCP-i. Unless specified, the bandwidth estimation procedure always gives the correct value at the end of each XCP control interval (in $ns$, the available bandwidth is found by subtracting the incoming traffic load to the bottleneck link capacity, which is known in the simulation).

### A. Incremental deployment around non-XCP clouds

The first scenario on which XCP-i is tested consists in a symmetric incremental deployment depicted in figure 5 which could be viewed as an optimized peering point scenario where 2 non-XCP clouds are connected by XCP routers. Figure 6 shows the sender's $cwnd$ and the receiver's throughput. As we can see, both $cwnd$ and throughput are stable with identical results when compared to the all-XCP scenario. Although not shown there were no timeouts nor packet losses. The XCP-i virtual router in R1 and R2 knows the available bandwidth in the non-XCP cloud and therefore computes an optimal feedback value accordingly. These results show that XCP-i is able to efficiently run in an heterogeneous network even though it is deployed only at some strategic locations.

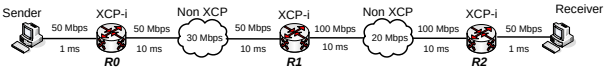


Fig. 5. Incremental deployment at peering point

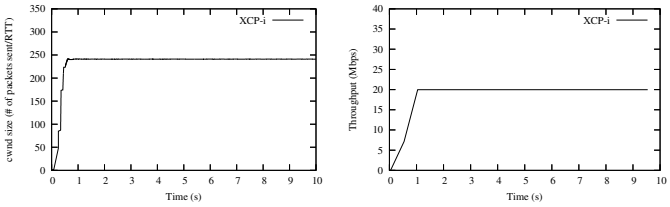


Fig. 6. $cwnd$ and throughput in incremental deployment

### B. Merge scenario: $n$ non-XCP clouds share 1 XCP path

The third scenario is a merge scenario where 2 non-XCP clouds share 1 XCP path as depicted in figure 7.

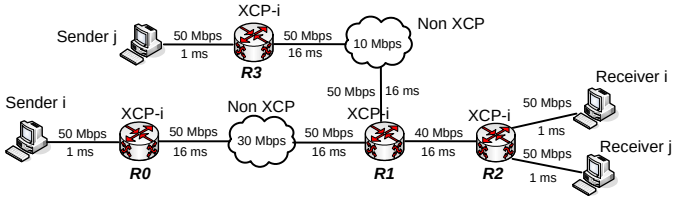


Fig. 7. 2 upstream non-XCP queues, $\Sigma input\_capacity = output\_capacity$

In this case, the XCP-i router at the merging point (R1 in the figure) has to create one virtual XCP-i router for each incoming non-XCP cloud. In addition, the sum of the bottleneck bandwidth of each non-XCP clouds is equal to the output link capacity of the XCP-i merging point. In this way, we also test the ability of XCP-i to correctly use the

legacy XCP feedback computation procedure to insure fairness between the 2 merging flows. Figure 8 shows that XCP-i succeeds in maintaining an XCP-like fairness since sender $j$ can get an optimal throughput of 10Mbps and sender $i$ can get approximately 28Mbps. The reason why sender $i$ only gets 28Mbps instead of 30Mbps is due to XCP control laws and is explained in more details in [10].

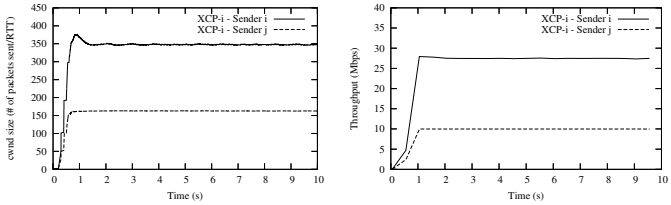


Fig. 8. $cwnd$ and throughput in the merge scenario

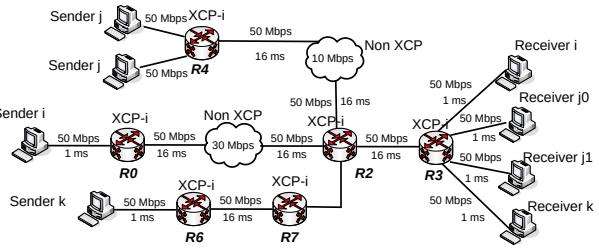


Fig. 9. 2 upstream non-XCP queues competing with an XCP path, $\Sigma input\_capacity > output\_capacity$

Figure 9 shows a more complex scenario where we have 2 non-XCP clouds and 1 XCP-i router connected to a single XCP-i router. In addition, the non-XCP cloud on the top carries 2 flows, $j_0$ and $j_1$, which should share the 10Mbps link. Also, if we consider the sum of all incoming link at the XCP-i merging point, it is much higher than the output link capacity.

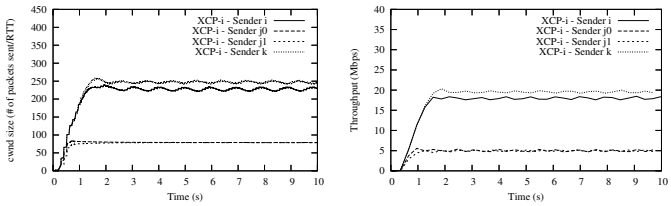


Fig. 10. $cwnd$ & throughput: non-XCP clouds compete with an XCP path

As we can see in figure 10, the sum of all the throughputs does not exceed the output link capacity at the merging point which is set to 50Mbps. In this complex scenario, the real XCP-i router executes the XCP Fairness Controller to insure that its output link is fairly used by all the flows. It is also important to see in this scenario that the XCP-i virtual router does execute the Fairness Controller to insure that the available bandwidth in the non-XCP cloud is shared in a fair manner. In our example, $j_0$ and $j_1$ get 5Mbps each.

### C. Fork scenario: 1 non-XCP cloud serves $n$ XCP paths

In this scenario, figure 11 shows a topology with a non-XCP cloud connected to 2 XCP paths. Figure 12 shows that XCP-i once again is able to fairly share the 50Mbps link in order to get 25Mbps for each flow.

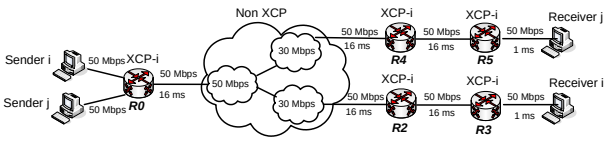


Fig. 11. 1 non-XCP queue shared by XCP-capable downstream nodes

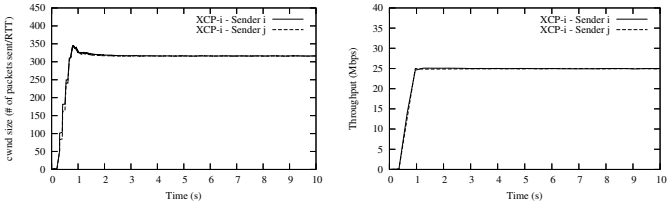


Fig. 12. $cwnd$ and throughput in the fork scenario

### D. Varying the bandwidth estimation accuracy

We supposed so far that the bandwidth estimation found by the routers are always accurate. This is not always true [13] and under certain conditions, the tools that are used to estimate the available bandwidth could overestimate or underestimate it. In this subsection, we took the topology of figure 7, multiplied all link capacities by 10 in order to compare XCP-i with TCP on high-speed links and supposed that the available bandwidth estimation is inaccurate: we randomly overestimate or underestimate the available bandwidth by a maximum of 10% and 20%.

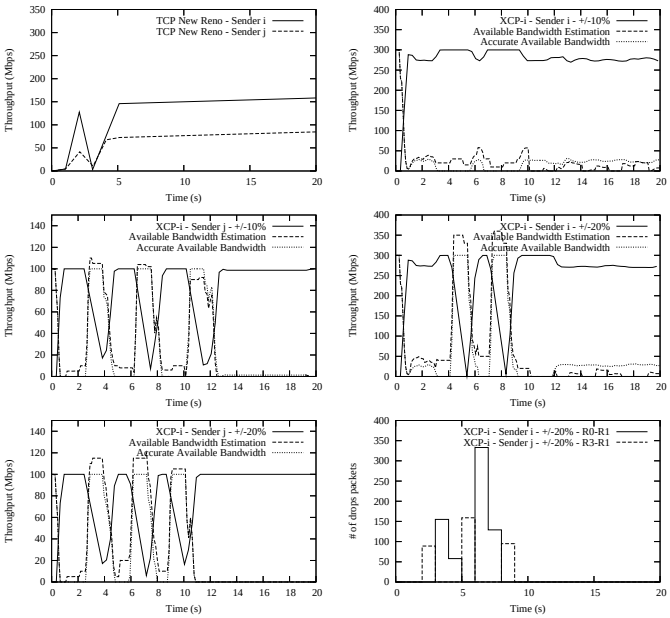


Fig. 13. Top: throughput for TCP (left), XCP-i - Sender i - 10% (right). Middle: throughput for XCP-i - Sender j - 10% (left), XCP-i - Sender i - 20% (right). Bottom: throughput for XCP-i-Sender j-20%, packet losses XCP-i-20% (right).

Figure 13 shows the throughput for sender $i$ and $j$, the accurate (real) and the estimated available bandwidth. As we can see in figure 13(top-left) TCP is not able to get all the available bandwidth (bottleneck link capacities are 300Mbps and 100Mbps) and sender $i$ and $j$ sent respectively 329Mbytes and 172MBytes in 20s. XCP-i with 10% and 20% estimation error still performs well: sender $i$ and $j$ sent respectively

690MBytes and 182MBytes with 10% error and 590MBytes and 187MBytes with 20% error. As a comparison, with XCP-i with 0% error (accurate estimation) sender $i$ and $j$ sent respectively 670MBytes and 244MBytes. As can be expected, the main consequences of overestimating the available bandwdith are packet drops and timeouts. This can be seen more easily for sender $j$: figure 13(bottom-left) shows that, in this case, the estimated bandwidth is always above the real available bandwidth resulting in packet drops at 3 moments (see figure 13(bottom-right)) which correspond to when the estimated bandwidth goes well beyonds the link capacity. For sender $i$ 10% of error does not produce timeouts as the router's buffers can compensate (1700-packet buffer) which is not the case for sender $j$ (700-packet buffer). However, although XCP-i performances depend on the estimation accuracy, XCP-i still outperforms TCP on high-speed links because it recovers quickly from packet losses.

## V. Open issues

### A. Fairness and over-estimatation in a non-XCP cloud

The topology depicted by figure 14 is currently not fully supported. In this case there is a bottleneck link in an non-XCP cloud that is shared by 2 XCP paths. The problem is as follows: let assume in a first step that all links are unload. If router $b$ detects the non-XCP cloud, it will request a bandwidth estimation procedure from router $a$. The result will be BW=30Mbps if the link is not loaded. Now, suppose that almost at the same time router $c$ also detects the non-XCP cloud and requests a bandwidth estimation procedure from router $d$. Again, BW=30Mbps. Then senders $i$ and $j$ will both try to transmit at 30Mbps resulting in a 60Mbps load for the bottleneck link. When another estimation will be triggered, BW will certainly be less than 30Mbps (typically near zero). Depending on how large are the router's buffer, some packets could be dropped because XCP-i can conclude that the available bandwidth is $n$ times the real available bandwidth if there are $n$ XCP independent paths. However, this problem could be diminished if the frequency of bandwidth estimation is increased so that senders $i$ and $j$ get a more accurate view of the available bandwidth, but it will increase the load in the network with control messages. The impact of controls messages in the network's load will be studied in futures works.

A second problem is when there is already 1 XCP flow that takes all the bottleneck link capacity. When the second sender starts, XCP-i is not able to correctly allocate bandwidth in a fair manner because of the XCP control laws that prevent any aggressive behavior (see next subsection). The second flow will only get the bandwidth given by the bandwidth shuffling procedure.

We plan to investigate these 2 issues in future works.

### B. Fairness with TCP

The fairness with TCP is not an XCP-i problem but an XCP problem in general. XCP is only able to get the remaining bandwidth with the objective of not causing packet drops. XCP-i being based on the XCP control laws has the same problem. In this paper, we did not consider fairness between XCP-i and TCP. Non-XCP clouds can carry non-XCP flows but XCP-i will only consider the available bandwidth left by these non-XCP flows. The problem of XCP and TCP cohabitation will be studied in future works.

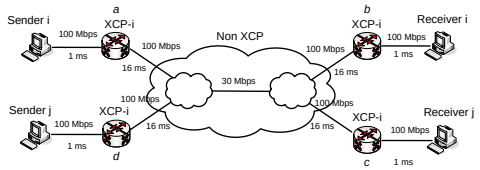


Fig. 14. 1 bottleneck link shared by $n$ XCP paths

## VI. Conclusion

This paper presented XCP-i which is an enhancement to the XCP protocol that enables XCP to deal with heterogeneous networking. The main design goal of XCP-i is to keep the control laws of XCP unchanged while adding new features for detecting and handling non-XCP clouds. The simulation results show that XCP-i can succeed in a large variety of scenario to provide an XCP-like level of performances. Although XCP-i performances depend on the available bandwidth estimation accuracy, XCP-i still outperforms TCP on high-speed links because it recovers quickly from packet losses. Current works concern the implementation of XCP-i in XCP capable routers, a large scale validation on the Grid5000[4] platform, XCP-i performance study in heterogeneous networks and some extensions on XCP fairness.

## References


[1] C. Barakat, E. Altman, W. Dabbous. On TCP performance in a heterogeneous network: a survey. IEEE Comm. Mag., January 2000.

[2] S. Floyd. HighSpeed TCP for Large Congestion Windows. RFC 3649.

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

[4] F. Cappello et al. Grid'5000: A Large Scale, Reconfigurable, Controlable and Monitorable Grid Platform. 6th IEEE/ACM International Workshop on Grid Computing, Nov. 2005

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

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

[7] M. Jain, C. Dovrolis. Pathload: an available bandwidth estimation tool. PAM 2002.

[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] D. M. Lopez Pacheco, C. Pham. Robust Transport Protocol for Dynamic High-Speed Networks: enhancing the XCP approach. Proceeding of the IEEE MICC-ICON 2005.

[12] V. Ribeiro. pathChirp: Efficient Available Bandwidth Estimation for Network Path. PAM 2003.

[13] Alok Shriram et al. Comparison of Public End-to-End Bandwidth Estimation Tools on High-Speed Links. PAM 2005.

[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.