Laboratoire de l'Informatique du Parallélisme

École Normale Supérieure de Lyon
Unité Mixte de Recherche CNRS-INRIA-ENS LYON-UCBL n° 5668

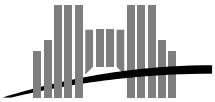
Fairness Issues When Transferring Large Volumes of Data on High Speed Networks With Router-Assisted Transport Protocols

Dino M. Lopez Pacheco ,
Laurent Lefèvre ,
Congduc Pham

December 2007

Research Report N° RR2007-46

École Normale Supérieure de Lyon
46 Allée d'Italie, 69364 Lyon Cedex 07, France
Téléphone : +33(0)4.72.72.80.37
Télécopieur : +33(0)4.72.72.80.80
Adresse électronique : lip@ens-lyon.fr



Fairness Issues When Transferring Large Volumes of Data on High Speed Networks With Router-Assisted Transport Protocols

Dino M. Lopez Pacheco , Laurent Lefèvre , Congduc Pham

December 2007

Abstract

This report presents improvements of the eXplicit Control Protocol (XCP) in order to support fairness between XCP and TCP flows. Our mechanisms are based on two main components: estimations of the resources needed by XCP, and limitation of the resources taken by TCP. The set of simulations shown in this report proves that we can guarantee a fairness level between XCP and TCP. We think that our TCP-XCP fairness mechanism can be perfectly applied for Long Distance Data Grids, where senders need to transfer large volumes of data.

Keywords: TCP, XCP, routers-assisted protocols, inter-protocol fairness, active flow number estimations

Résumé

Ce rapport présente nos solutions pour résoudre le problème d'équité entre les flux XCP (eXplicit Control Protocol) et TCP. Nos mécanismes sont basés sur deux principaux composants : une estimation des ressources requis par XCP, et une limitation de la quantité des ressources pris par TCP. L'ensemble des simulation montré dans ce rapport preuve que nous somme capables de garantir un certain niveau d'équité entre XCP et TCP. Nous croyons que notre solutions d'équité TCP-XCP peut être parfaitment appliqué dans des grilles des données long distance, où les émetteurs ont besoin d'envoyer grand quantités de données.

Mots-clés: TCP, XCP, protocole assistés par des routeurs, équité inter-protocolaire, estimation du nombre des flux actifs

1 Introduction

The eXplicit Control Protocol (XCP[3]) is a transport protocol that uses the assistance of specialized routers to 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 worse than TCP, in the presence of non-XCP routers thus limiting the benefit of having XCP running in some parts of the network.

In previous work, we have improved the robustness of XCP on high speed networks (*XCP-r* architecture [6]) and the interoperability of XCP with heterogeneous network equipments (*XCP-i* module [5]). The goal of this paper is to study and to improve the interoperability of XCP with existing protocols (TCP). The working context of the work described in this paper is focused on high performance networks dedicated to managed infrastructures (i.e. data grids). The considered flows are Long-Term High-Bandwidth streams adapted to large data transfers.

While fairness problem between XCP flows have been studied [7], the fairness issues between XCP and heterogeneous E2E protocols (TCP) have not been explored. This aspect must be covered if we want to benefit from XCP transport protocol for large volume of data transfers on high speed networks.

This paper presents the problem description (section 2) and the definition of XCP-TCP fairness (section 3). The proposed fairness mechanism is described in section 4 and some simulation results are presented in section 5. Improvements to the approach are described in section 6. Section 7 concludes this paper and presents our current and future works.

2 Problem description

When an XCP flow shares the resources with a flow using any end-to-end protocol, the XCP flow will yield as many networks resources as the end-to-end protocol will need. This phenomenon is caused by the XCP feedback equation (eq. 1) that we will describe briefly to better understand the problem.

$$feedback = \alpha.rtt.(O - I) - \beta.Q \tag{1}$$

In equation 1, α and β are constant, rtt is the average of the all RTT values given by the packets crossing the router, O is the Output Link Capacity, I the input traffic rate, and Q is the persistent queue size.

The XCP feedback equation lets the routers maximize the link utilization while avoiding packets dropping. This goal is possible because the feedback decreases as the input traffic rate increases (note that the sender rate increases proportionally to the feedback value). However, XCP does not take care about the cause of the traffic rate increasing, it will just try to decrease the transmission rate of the sender. As the XCP routers can change the transmission rate of the senders using XCP but not those using an E2E protocol like TCP, an XCP flow will only take the bandwidth left by the flows using E2E protocols.

Fig. 1 demonstrates the difficulty for an XCP flow to deal with 2 concurrent TCP flows. The TCP streams compete for half of the bandwidth and the XCP can not obtain any network

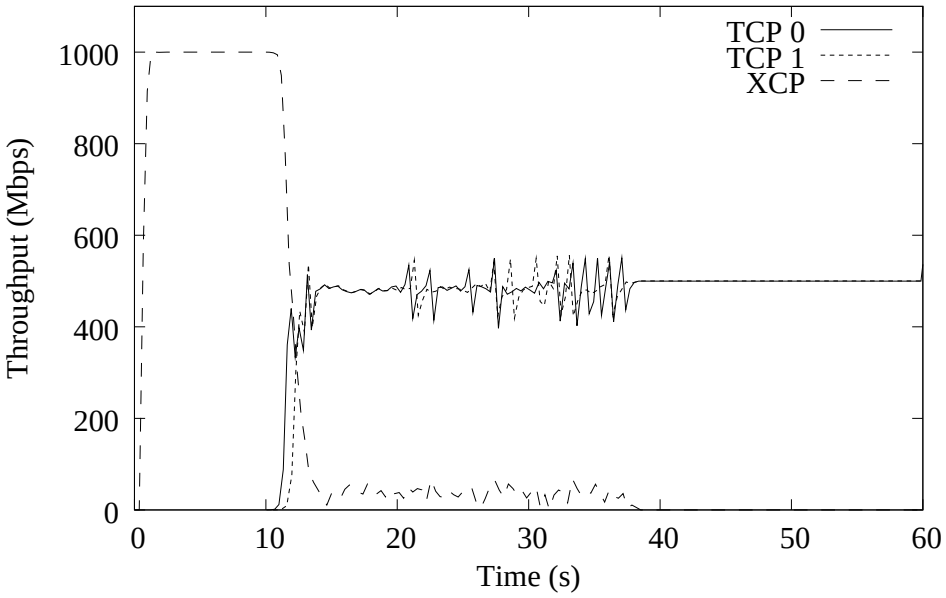


Figure 1: XCP flow dealing with TCP concurrent flows

resources.

3 Defining the XCP-TCP fairness

The objective of this paper is to present our mechanism to ensure the fair share between XCP and TCP flows: more precisely, ensure an inter-protocol XCP-TCP fairness. We want to remark that in any case we are not looking at intra-protocol fairness of TCP or XCP.

Thus, before presenting our fairness mechanism, it is necessary to clarify a definition of the XCP-TCP fairness. We will define the fairness between TCP and XCP by the fact that TCP lets XCP get a total bandwidth equivalent to the ratio between the link capacity, and the sum of the number of XCP and TCP flows, multiplied by the number of XCP flows. Equation 2 shows this relationship.

$$BW_{XCP} = \frac{LinkCapacity}{N + M} * N \tag{2}$$

In equation 2, BW_{XCP} represents the optimal amount of resources obtained by XCP for considering that TCP is fair with XCP. N is the number of active XCP flows and M the number of active TCP flows.

4 Proposing a XCP-TCP fairness mechanism

In order to solve the problem described in section 2, we have proposed a mechanism based on two main steps: (i) to estimate the resources needed by XCP, and (ii) to limit the resources taken by TCP.

4.1 Estimating the resources needed by XCP flows

Estimating the resources needed by the XCP flows (BW_{XCP}) is a complex work since it implies the knowledge of the number of active XCP flows N and the number of active TCP flows M, as expressed in equation 2.

The best known strategy to compute the number of active flows consists in looking at the ID flow of every incoming packet in a router, to search for this ID in a table and, if it does not exist into the table then to record this new ID. This way, after an interval control, we need only to count the number of IDs recorded in our table to know the number of active flows, and finally, to refresh the table (erasing all the IDs stored). This strategy can compute exactly the number of active flows, but it is very hard in terms of CPU and time-consuming since it is necessary to search the ID of every incoming packet in a table that could expand to thousands of IDs.

Another available approach to estimate the number of active flows (less accurate) is based on the algorithm of Bloom filters [1], used in the NRED [4] mechanism. When a packet arrives, NRED hashes the source-destination pair of the packet into a bin. A bin is a small cache and needs only 1 bit of memory to mark 0 or 1. The estimator maintains $P.L$ bins. The bins are organized in L levels, each of which contains P bins. The scheme employs L independent hash functions, each of which is associated with one level of bins. Each hash function maps a flow into one of the P bins in that level. At the beginning of measurement interval, all bins and a counter N_{act} are set to zero. When a packet arrives at a router, the L hashed bins for the source-destination IP address pair of the packet are set to 1 and N_{act} increases by 1 if at least one of the L hashed bins was zero before hashing. However, it could happen that for packets belonging to two different flows are hashed into the same level L bins causing a "misclassification". The probability of having a "misclassification" decreases as the number of L hash functions increases.

The problem of the Bloom filter algorithm is based on the fact that to avoid "misclassification", we need an important number of hash functions. Since every hash function is executed one time for every incoming packet, this algorithm could be expensive in CPU terms if we want to avoid "misclassifications". Finally, even if the hash functions could be executed in parallel, they need a point of synchronization in order to modify the N_{act} variable, which could be very time-consuming.

Finally, another considered approach for estimating the active flows in a router is the probabilistic method used in the SRED algorithm [8]. SRED keeps a table called the *zombie table* able to keep up to 1000 ID flows (the ID flow could be src_addr:src_port::dest_addr:dest_port). The *zombie* table is filled at the beginning with the ID flows belonging to the first 1000 incoming packets. When the *zombie* table has been filled in, every arriving packet to the router is examined. First, the ID flow of the packet is taken, and compared with a ID flow taken randomly from the *zombie* table. If the ID are the same, then an event *hit* is declared. In other case, an event *mis* is declared. When a *mis* is detected, with a probability of 25%, the old stored ID in the *zombie* table will be replaced by the packet's flow ID. After a *hit* or *mis*, $P(t)$ which is an equation that computes the probability to make a *hit* is updated. Now, if the total of active flows in a router is N, then the probability to get a hit is $1/N$. Since $P(t)$ reflects this probability $1/P(t)$ represents the estimation of the number of active flow seen by a router at time t.

The operations executed by SRED, as opposed to others methods, do not require a high utilization of CPU since SRED only needs one comparison, two random numbers, and some-

times one memory write. However, one of the weaknesses of SRED is based on the fact that it does not compute the exact number of active flows, but only make an estimation of this number.

We believe that in high performance infrastructures it is very difficult to know the exact number of active flows at a given time (as we have seen earlier, we need faster CPU and/or bigger memory). We believe also that having an estimation of the number of active flows is enough to ensure a max-min fairness between XCP and TCP, if we are able to mitigate in a certain way the error produced during the estimation procedure. With this hypothesis, we have chosen the mechanism proposed in the SRED algorithm in order to estimate the number of active TCP and XCP flows. A mechanism will divide the resources between XCP and TCP in a flexible way in order to correct the error produced during the estimation.

The implemention of the active flow number estimation method in the XCP routers is very similar to the one proposed in SRED. However, in our case it is necessary to keep two *zombie* tables: the first one will be filled in with the XCP flow IDs, while the second one will be filled in with the TCP flow IDs. Thus, we will have one variable $P(t)_{XCP}$ that will keep the probability to get an *XCP-hit*, and another one $P(t)_{TCP}$ that will keep the probability to get a *TCP-hit*. The way to update both variables is the same described in the SRED mechanism. Therefore, we can get an estimation about the number of XCP flows $N = P(t)^{-1}_{XCP}$, and TCP flows $M = P(t)^{-1}_{TCP}$ at a given time t. Based on these estimations, the XCP routers calculate allocation for each type of flows, taking into account the output link capacity (already known by the router) and then trying to limit the resources taken by these two types of flows.

4.2 Limiting the resources taken by E2E protocols

Because the XCP flows only take the available bandwidth not used by the TCP flows, our main goal is to limit the performance of the latter. It is well known that most of end-to-end protocols only decrease their transmission rate when a loss is detected. Thus we propose to execute an intelligent packets drop over end-to-end flows when the routers detect that they are using more resources than they should.

When the output link capacity uses more than a given threshold (95% for instance in our tests), the proposed algorithm performs the following steps :

1. If the resources taken by the end-to-end flows exceed our estimation: drop packets with a small initial probability (the fixed minimum value is 0.01%);

2. If end-to-end flows do not decrease their transmission rate: increase the drop probability;

3. If end-to-end flows have less than the determinate resources utilization: decrease the drop probability;

4. If the resources taken by the end-to-end flows match with our estimated utilization: do not modify the drop probability.

For our tests with TCP NewReno, the drop probability is increased by a 1.01 factor and decrease by a 0.99 factor. It is possible to have different values according to the type of end-to-end protocols: HSTCP, BIC,... Also, it is possible to have dynamic parameters depending on the network conditions, as will be explained later on.

5 Simulation results

We evaluate the proposed fairness solution by simulation (based on the *ns-2* XCP modules provided by Katabi [3]). We focus on 2 different scenario : national small distance Grid (such as the French Grid5000 [2] infrastructure) with an average 20 ms RTT and larger scale Grids with 100 ms RTT.

5.1 1 XCP flow & 2 TCP competing flows

For our first set of experiments, we evaluate the same scenario as Fig. 1 when an XCP flow competes with two new TCP streams.

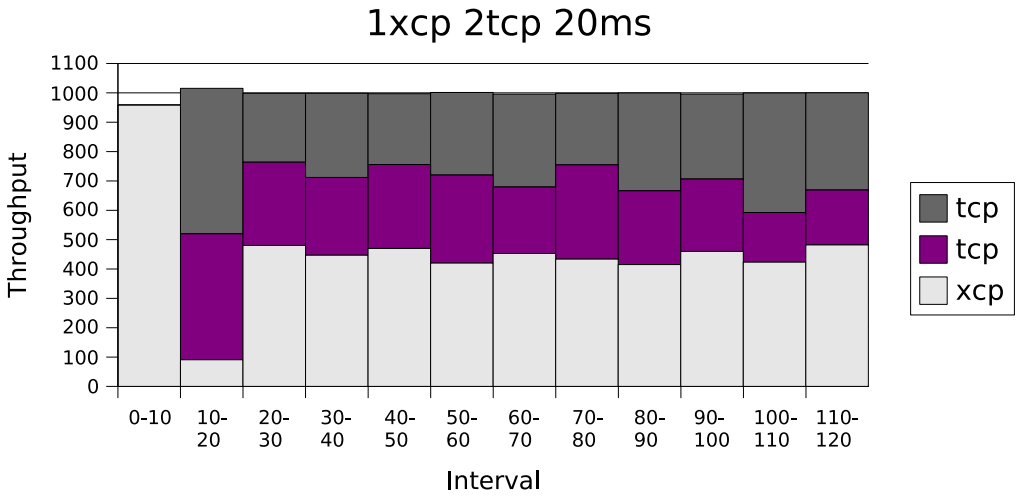
In Fig. 2, we show the average throughput for XCP and TCP computed every 10 seconds (the next results will be shown in a similar way) when our XCP-TCP fairness mechanism runs in every XCP router. We can clearly see the benefit of our algorithm: after a period impacted by the slow start effects of the two TCP flows (seconds 10-20), XCP flow is able to obtain some bandwidth.

An important remark: in Fig.2(a), where the throughput evolution is showed with a granularity of 10 seconds, it may appear that TCP never gets more than 60% of the total link capacity as it should be, which is not true. In some occasions TCP gets around 60% of the total link capacity, however, when our mechanism drops TCP packets, TCP flows decrease drastically theirs sending rate causing a smaller average throughput that does not reflect its bigger throughput level. On the other hand, every time TCP decreases its sending rate, XCP grabs the remaining available bandwidth quickly, therefore the computed average throughput does not reflect its lower throughput level. Due to the unstable nature of TCP, some fairness metrics, like the Jain index, are not able to reflect the fairness behavior of long-life connections.

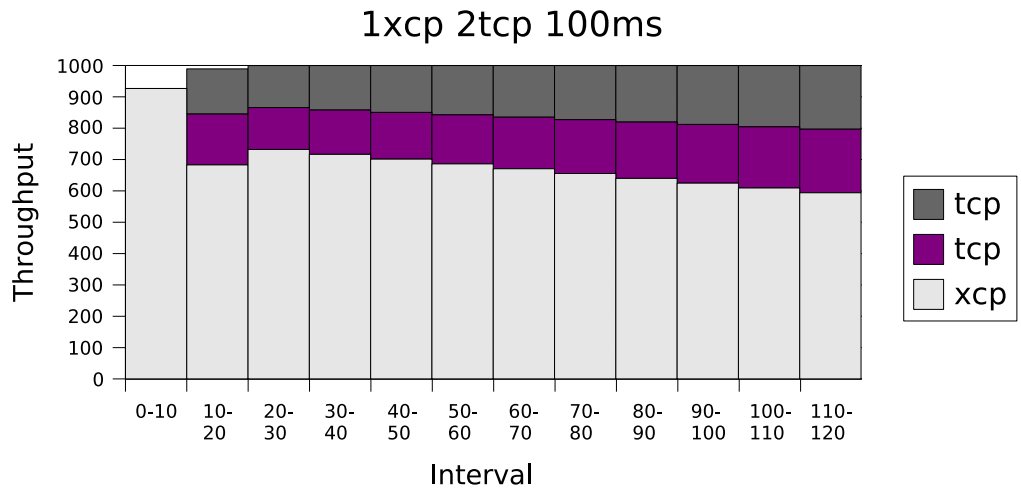
Fig.2(b) shows the same experiment with an RTT of 100ms. In this case, we can observe a very aggressive behavior of TCP during the startup phase that strongly penalizes the performance of XCP (seconds 10-20), triggering the execution of our fairness algorithm. When the RTT is large, the losses produced by the aggressiveness of TCP, added to the losses produced by our fairness mechanism, added to a very slow increase of the TCP throughput, result in a very low TCP throughput. It is very important to remark that after second 20, the drop probability used by our mechanism at its minimum and TCP flows are no longer penalized with dropped packets. However, the unfairness that can been seen after second 20 is caused mainly by the incapacity of TCP to regain the available bandwidth when RTT is large. Still, fairness metrics could not show properly the fairness evolution in this case.

5.2 10 TCP flows & 3 XCP competing flows

The impact in the fairness of the TCP flows' RTT could be reduced when the number of streams increases. For instance, in Fig.3(b) where some XCP flows are incorporated among TCP flows (at sec. 10, 30 and 50), we can see that each time a new XCP flow comes in, our fairness mechanism give some bandwidth to the flow. The amount of bandwidth recovered by our mechanism is bigger than needed, however, in this case, TCP is able to grab faster the lost bandwidth. The explanation is as follows: lets imagine a scenario with M TCP flows. Since in steady state every flow increases by 1 packet per RTT, then during a period of time $intv$, the total TCP rate increase will be $M * intv/RTT$ packets. Therefore, if we increase



(a) 20 ms RTT

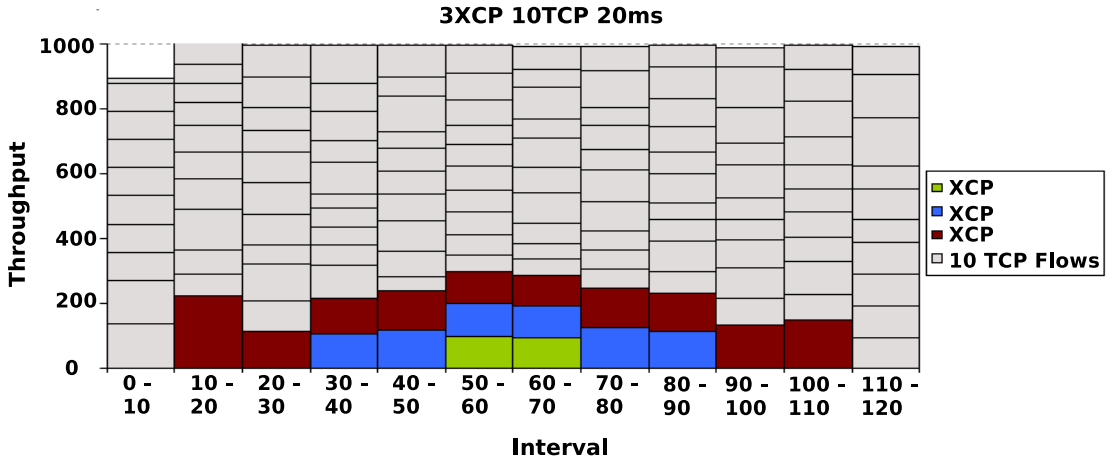


(b) 100 ms RTT

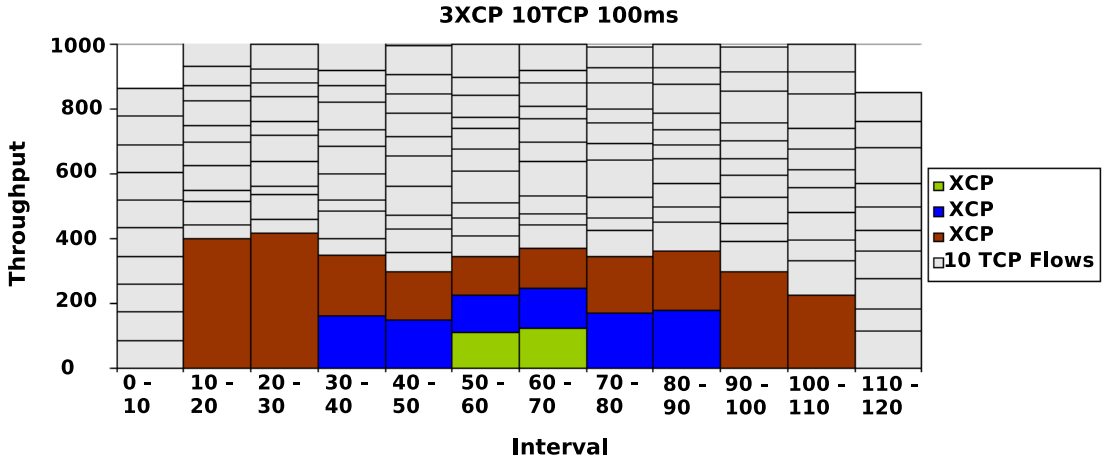
Figure 2: One XCP flow dealing with two concurrent TCP flows

the number of TCP active flow by a factor α, then, with the same RTT value, the total TCP rate increase will be $\alpha * M * intv/RTT$.

In Fig.3(b), even though TCP grabs bandwidth faster than in Fig.2(b), and therefore the fairness between XCP and TCP is improved, XCP has still more bandwidth than needed.



(a) 20 ms RTT



(b) 100 ms RTT

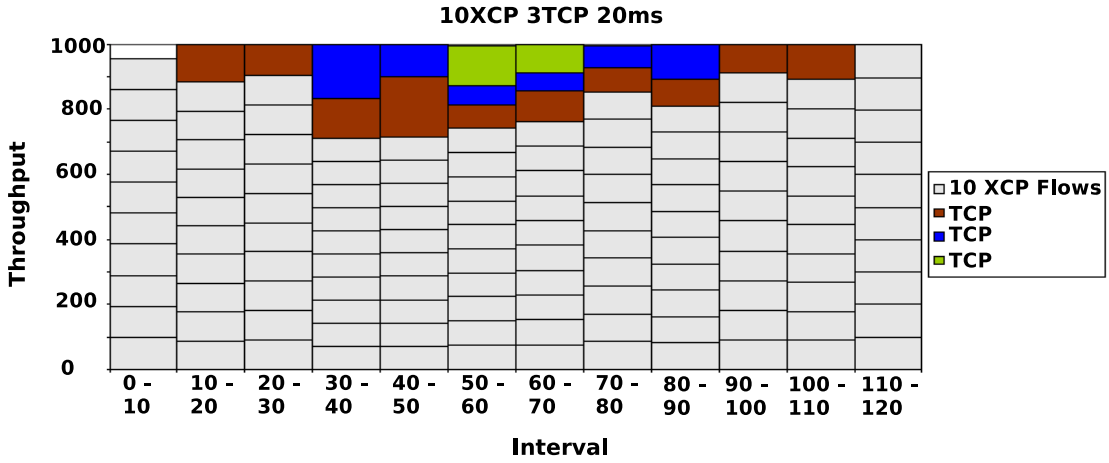
Figure 3: 3 XCP flows appear among 10 TCP flows

Thus, by increasing the number of TCP flows, and by decreasing the RTT, TCP should become more aggressive, and this aggressiveness should be limited by our XCP-TCP mechanism. In order to prove the robustness of our algorithm, we decided to execute the same experiment (3 XCP appearing gradually among 10 TCP flows), with a RTT value of 20ms (Fig.3(a)), that represents an increase of the TCP aggressiveness by a factor $\alpha = 5$, in comparison to the experiment shown in Fig.2(b) or the one shown in Fig.3(b). As we can observe in Fig.3(a), the increase in the aggressiveness is well managed by our fairness mechanism.

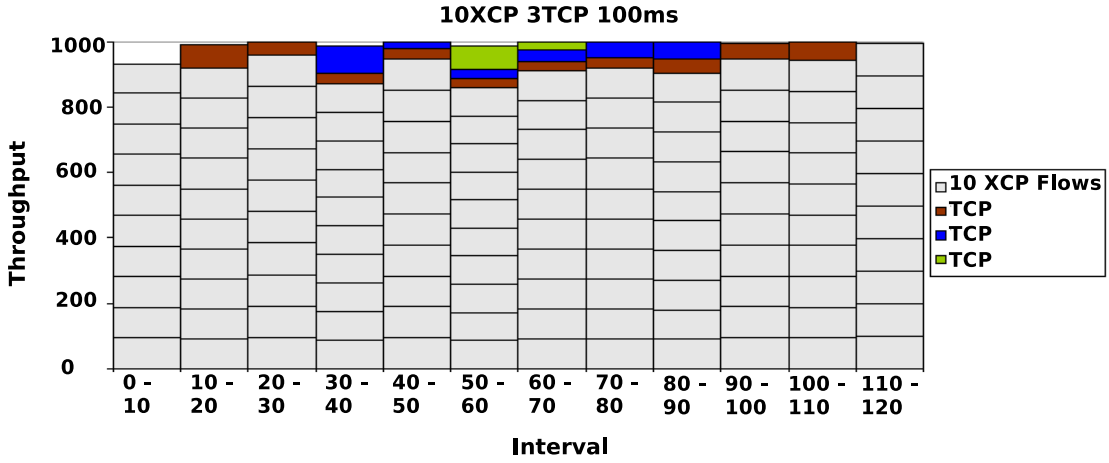
In fact, during most of the simulation time, XCP and TCP get almost the optimum needed bandwidth.

5.3 10 XCP flows & 3 TCP competing flows

We already explored the case where the number of TCP flows is greater than the number of XCP flows. The experiments shown in the last subsection tested the robustness of our mechanism when TCP increase its aggressiveness. Now, in this subsection we explore the case where most of the flows are XCP and only some are TCP. This new set of experiments investigate whether our mechanism lets TCP compete adequately against XCP flows or not.



(a) 20 ms RTT



(b) 100 ms RTT

Figure 4: 3 TCP flows appear among 10 XCP flows

In Fig.4(a), with a $RTT = 20ms$, we can see that every time a new TCP flow starts, it

gets a large amount of bandwidth (seconds 10, 30, 50). However, after Slow Start finishes, our XCP-TCP mechanism succeeds in ensuring a fairness level between XCP and TCP. For instance, at seconds 60 and 70, we can observe that TCP gets slightly less than 25% of the total link capacity, if we take into account the proportion of TCP flows in the total number of active flows.

However, in the case where the RTT increases to $100ms$, since TCP flows loss packets at startup, due to the aggressiveness of the Slow Start phase and our fairness mechanism, the amount of bandwidth taken by TCP, after Slow Start, is small. Similar to the case shown by Fig.2(b), even if our mechanism does not penalize TCP flows since the drop probability decreases, the time needed by TCP to get enough resources is very large (see seconds 60 to 110 in Fig.4(b)). These results confirm that our XCP-TCP mechanism is mainly adapted for long-life flows, since they are able to recover the lost available bandwidth when TCP is penalized.

6 Limitations and optimizations

In the presented XCP-TCP fairness mechanism, we have used the active flow number estimation as proposed in [8]. This mechanism has the benefits to be lightweight in CPU utilization since the number of operations is minimum and most of these operations can be executed in parallel. However, these operations need to be executed for every incoming packet. Thus, the estimation operations, added to the XCP routers operations, could delay significantly the packets.

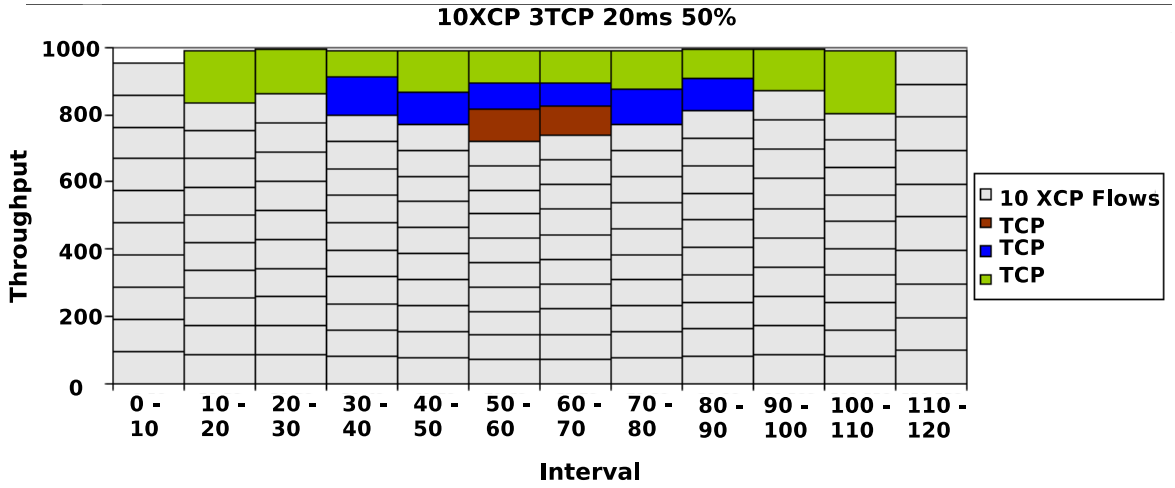
For this reason, we propose to modify the *hit* probability equation, in order to avoid inspecting every incoming packet in the router. The modification lies basically in inspecting the ID incoming packet with a given probability. We present in the next figure, two experiments where we have only inspected 50% of the total incoming packets.

After analyzing the results shown in Fig.5, and comparing them with the results shown in Fig.3, we can see that inspecting only 50% of the total incoming packets does not alter significantly the fairness between XCP and TCP.

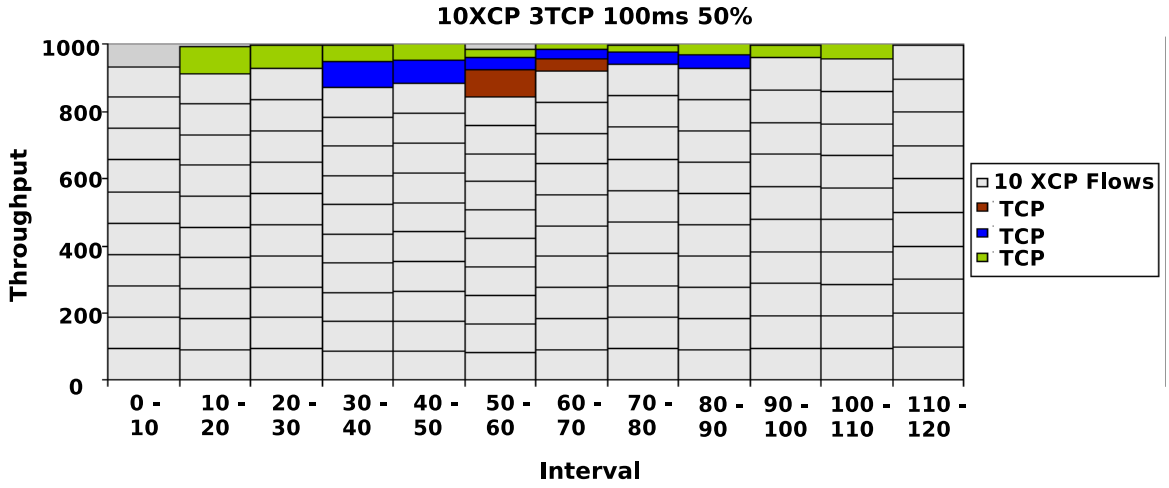
7 Conclusion and future works

This paper is a new step to stimulate and improve the deployment of a router-assisted transport protocol like XCP in heterogeneous high-speed networks. However, the proposed solutions can not be transposable as it is to the Internet, since we rely on the assumption that we observe long term high bandwidth flows. We think that this context can be perfectly applied for Long Distance Data Grids. In this case, large volume of data transfers could greatly benefit of XCP dynamic usage of the network without being too much impacted by existing TCP flows.

In the experiments we have seen that our fairness mechanism is able to limit the performance of TCP, to let XCP take some resources. When the RTTs are small, the effect of our fairness mechanism is translated into a high fairness level between XCP and TCP. However, when the RTT is larger, then TCP is unable to recover efficiently the lost bandwidth. We believe that this phenomenon could be avoided if we decrease the drop probability used in our mechanism faster. However, this could not be always a viable solution since TCP could



(a) 20 ms RTT



(b) 100 ms RTT

Figure 5: 3 TCP flows appear among 10 XCP flows - inspecting 50% of packets

take much more resources than XCP when the RTT is smaller. Future works consist in estimating the aggressiveness when the throughput of TCP increases, in order to update the drop probability in a more intelligent way.

Finally, simulation tools (like *ns-2*) limit our possibility to evaluate the proposed solution at higher link capacity. We are currently designing a Linux implementation of an interoperable XCP router in order to evaluate it with 10 Gbps links on the Grid5000 platform [2].

Acknowledgments

These works is partly supported by the "Consejo Nacional de Ciencia y Tecnología CONACyT" (www.conacyt.mx).

References

[1] Burton H. Bloom. Space/time trade-offs in hash coding with allowable errors. *Commun. ACM*, 13(7):422–426, 1970.

[2] F. Cappello, F. Desprez, M. Dayde, E. Jeannot, Y. Jegou, S. Lanteri, N. Melab, R. Namyst, P. Primet, O. Richard, E. Caron, J. Leduc, and G. Mornet. Grid'5000: A large scale, reconfigurable, controlable and monitorable grid platform. In *6th IEEE/ACM International Workshop on Grid Computing, Grid'2005*, Seattle, Washington, USA, November 2005.

[3] D. Katabi, M. Handley, and C. Rohrs. Congestion control for high bandwidth-delay product networks. In *ACM SIGCOMM*, 2002.

[4] Jung-Shian Li and Yong-Shun Su. Random early detection with flow number estimation and queue length feedback control. *J. Syst. Archit.*, 52(6):359–372, 2006.

[5] Dino M. Lopez Pacheco, Cong-Duc Pham, and Laurent Lefevre. XCP-i : eXplicit Control Protocol for heterogeneous inter-networking of high-speed networks. In *Globecom 2006*, San Francisco, USA, Nov. 2006.

[6] Dino M Lopez-Pacheco and Congduc Pham. Enabling Large Data Transfers on Dynamic, Very High-Speed Network Infrastructures. In *Proceedings of ICN/ICONS/MCL 2006*, Mauritius, April 2006.

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

[8] Teunis J. Ott, T. V. Lakshman, and Larry H. Wong. SRED: Stabilized RED. In *INFOCOM*, pages 1346–1355, 1999.