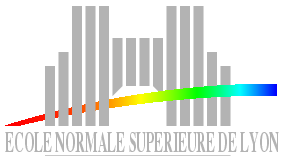
TITLE OF THE PRESENTATION



Dino M. López P. ***RESO – LIP, ENS de Lyon, France***
dmlopezp@ens-lyon.fr
Laurent Lefèvre ***RESO – LIP, ENS de Lyon, France***
Laurent.Lefevre@ens-lyon.fr
Congduc Pham ***LIUPPA – Université de Pau, France***
congduc.pham@ens-lyon.fr

Towards interoperability of XCP

Losses of ACK packets = Loss of network state information

XCP-r

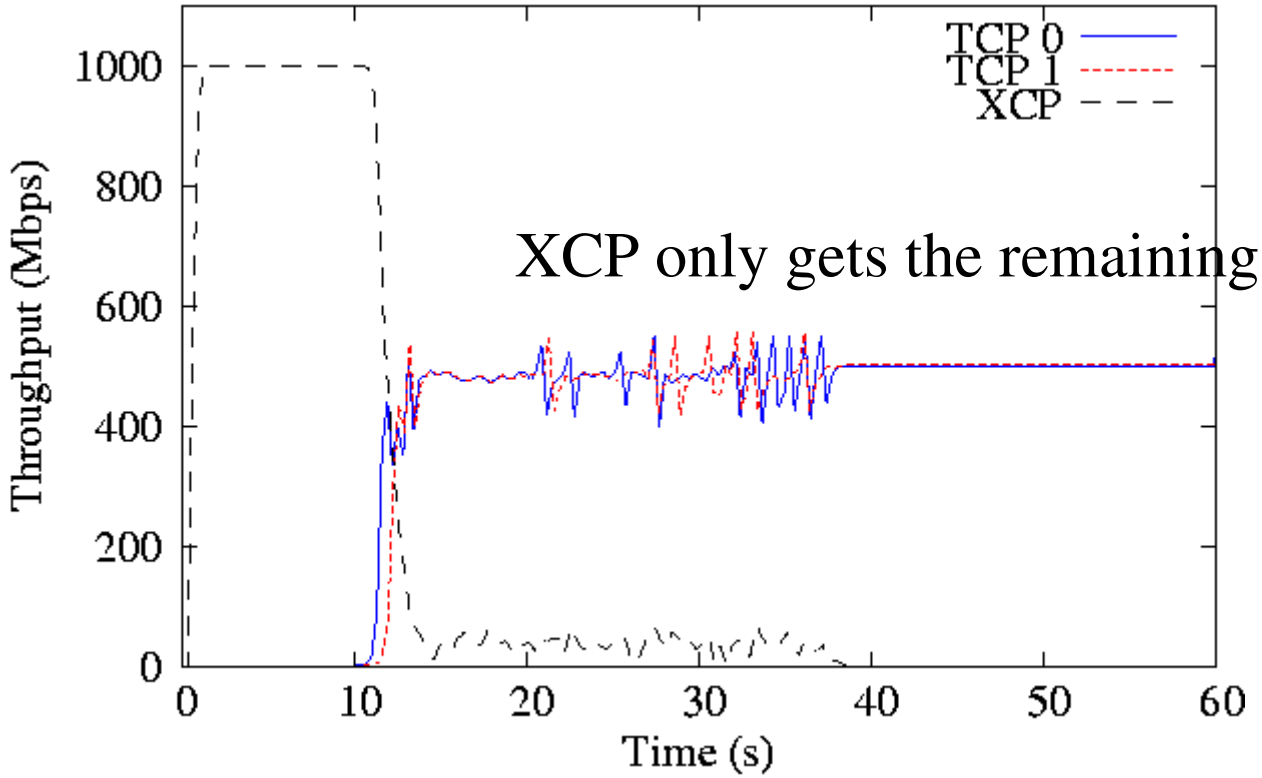
No interoperability between equipments = Bad performance when it is used in combination with IP routers.

XCP-i

No interoperability between protocols = Low fairness when the bottleneck is shared with End-to-End (E2E) protocols.

XCP vs TCP

$$H\_feedback = \alpha.rtt.(O - I) - \beta.Q$$



XCP only gets the remaining bandwidth

Improving the fairness

- Obtaining fairness on a link :
  - Need to estimate the resources needed by XCP and non-XCP flows
  - Need to estimate the number of XCP and non-XCP flows
  - Hard work!!
- Some approaches :
  - Record the ID of every flow crossing a router = impossible
  - Apply the NRED's Bloom filter [Li-Su05] = Hard in terms of processing and time. Accuracy (according to the authors).
  - SRED-like mechanisms = Lightweight in terms of processing, time and low memory usage. Good accuracy?

Estimation of the number of active XCP and non-XCP flows

We recycle the active flow estimation algorithm as described in SRED :

- After filling up a zombie list with the ID flow of the first 1000 incoming data packets
- Each arriving packet is compared with a packet randomly chosen from the zombie list
  - if (ID arriving packet != ID chosen packet), then with a probability *r* overwrite the ID of the arriving packet and set *hit = 1*. Either case set *hit = 0*
- Update the hit frequency estimator
  - $P(t) = (1-\alpha)P(t-1) + \alpha.hit$
  - $P(t)^{-1}$

Resources needed by XCP flows

After having an idea about the number of XCP and non-XCP flows, we can estimate the bandwidth needed by XCP flows

BW_{XCP} = # XCP flows * Link Capacity / (# XCP flows + # non-XCP flows)

And now?

- We have estimated the number of XCP and non-XCP flows. We don't know the exact number.
- When TCP exceeds the limit found by our estimations we will drop TCP packets with a probability *p*
- The utilization of a probabilistic method to drop packet can amortize the inaccuracy of our estimations.

Dropping TCP packets with a variable probability

The probability *p* to drop TCP packets must be adapted to the aggressiveness of TCP flows.

If (XCP input traffic rate > BW_{XCP})

Decrease the probability of dropping non-XCP packets

p = min(0, p * Ddrop);

else if (XCP input traffic rate < BW_{XCP})

Increase the probability of dropping non-XCP packets

p = p * Idrop;

For TCP New Reno

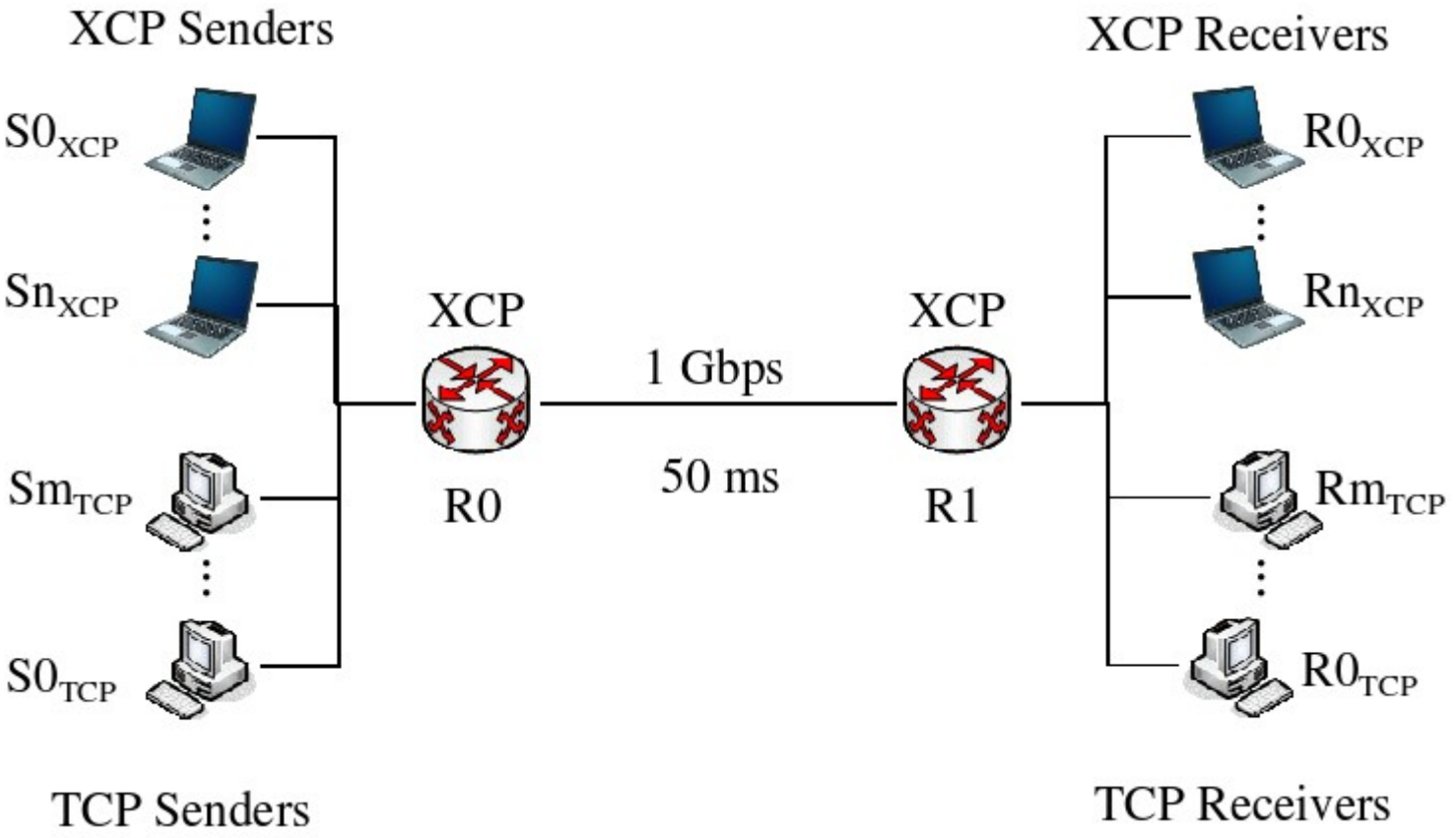
0.99 < Ddrop < 1

1.01 > Idrop > 1

Discussion/Limits

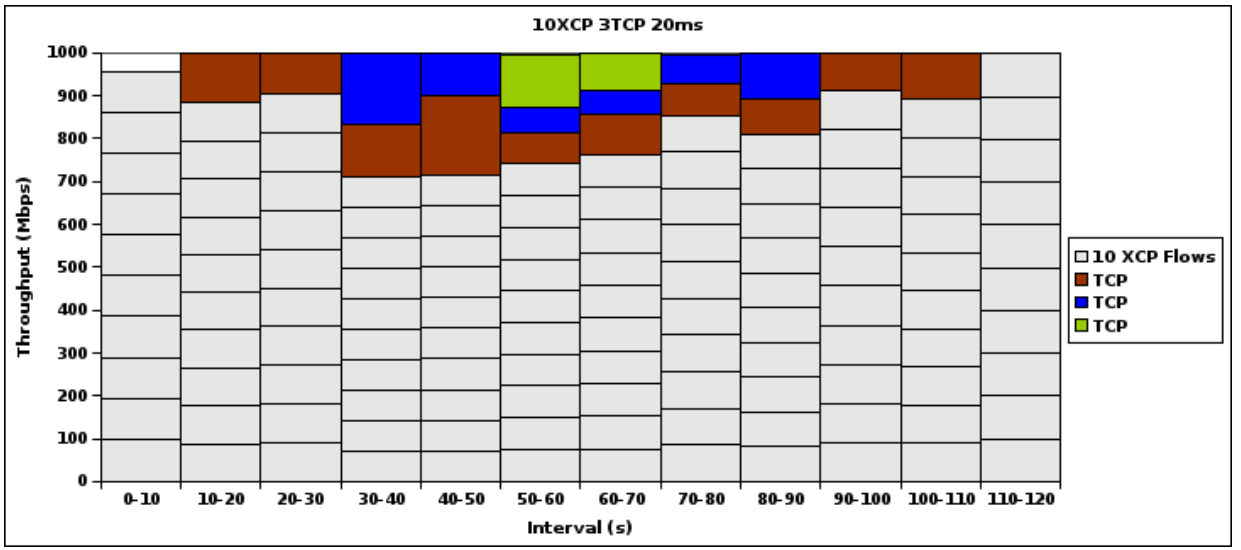
- We don't monitor the queue occupancy of the routers. We monitor the XCP input traffic rate and the BW_{XCP}
- This mechanism is executed only in a XCP router when XCP and non-XCP flows share the link.
- This mechanism is executed only when the total input traffic rate is bigger than 97% of the output link capacity.
- We never drop XCP packets
- The *p* probability is updated at every control interval of XCP

Topology for testing our XCP-TCP fairness mechanism



10 XCP vs 3 TCP flows

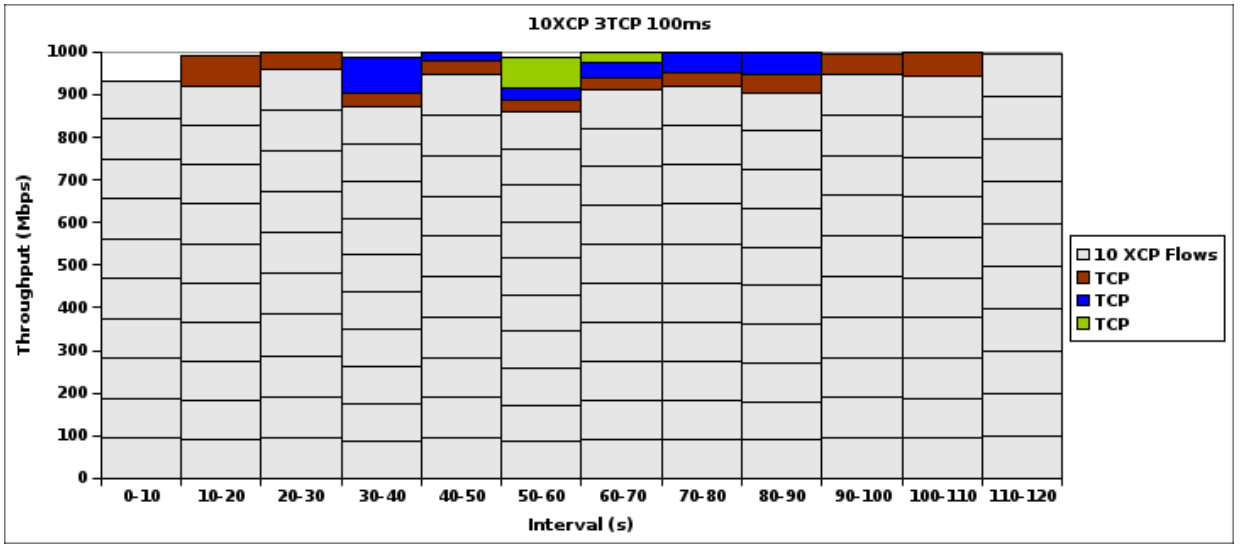
Link Propation delay = 10ms



- TCP flows arriving at seconds 10, 30 and 50
- Easy to identify Slow-Start effects
- After the Slow-Start phase, flows get stability

10 XCP vs 3 TCP flows

Link propagation delay = 50ms



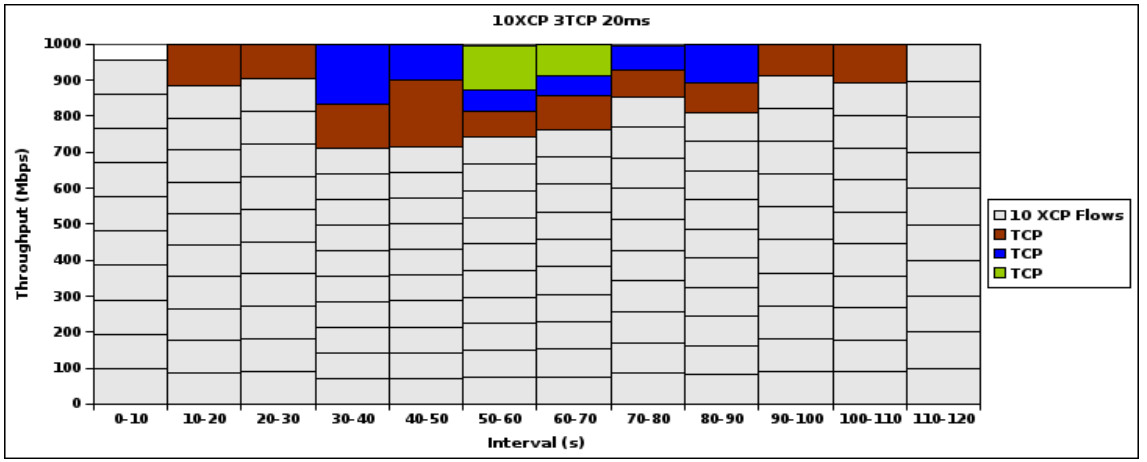
- Easy to identify Slow-Start effects even with a larger RTT
- After dropping packets to stop the aggressiveness of the Slow-Start phase, TCP is not enough reactive due to a larger RTT
- Unfairness due to the link propagation delay.

Towards a lightweight fairness solution

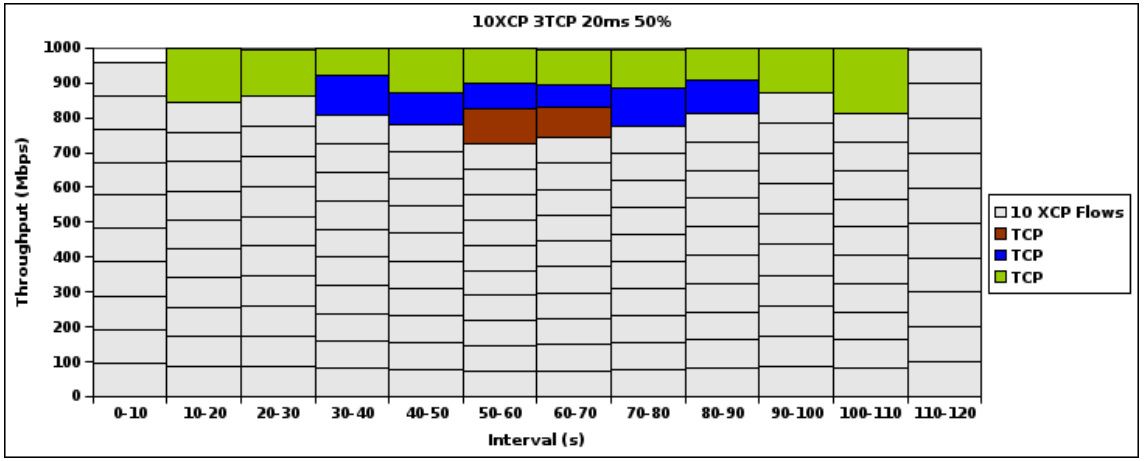
- In past experiments, we always analyzed every incoming data packet to estimate the resources needed by XCP. These operations joined to the XCP operations can be too expensive in a router with high performance links.
- We reduced the number of analyzed packets:
  - $P(t) = (1-\alpha)P(t-1) + \alpha.hit$ ---> Probability to make a *hit* analyzing 100% of incoming packets.
  - $P(t) = (1-\alpha)P(t-1)Pa + \alpha.hit.Pa$ ---> Probability to make a *hit* when packets are analyzed with a probability *Pa*.

10 XCP vs 3 TCP flows
Link propagation delay = 10ms

100%

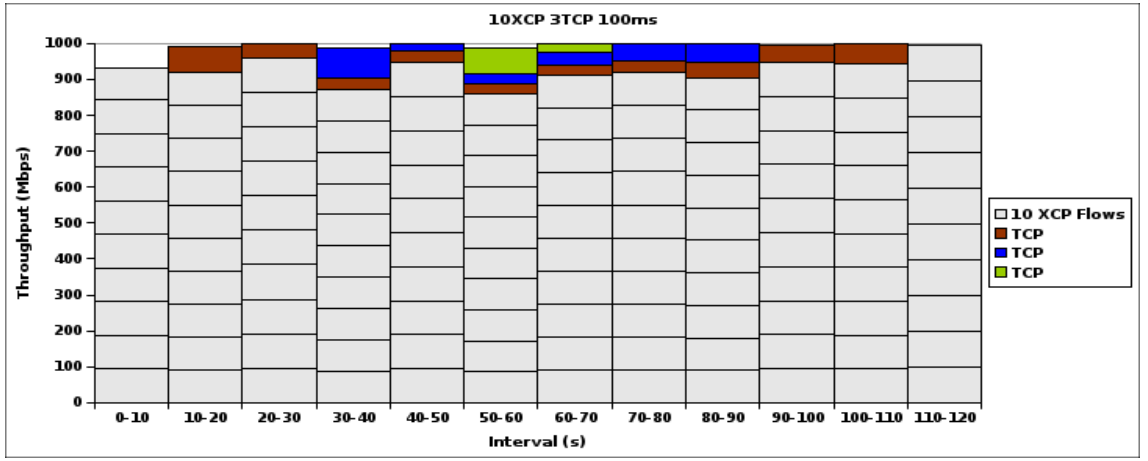


50%

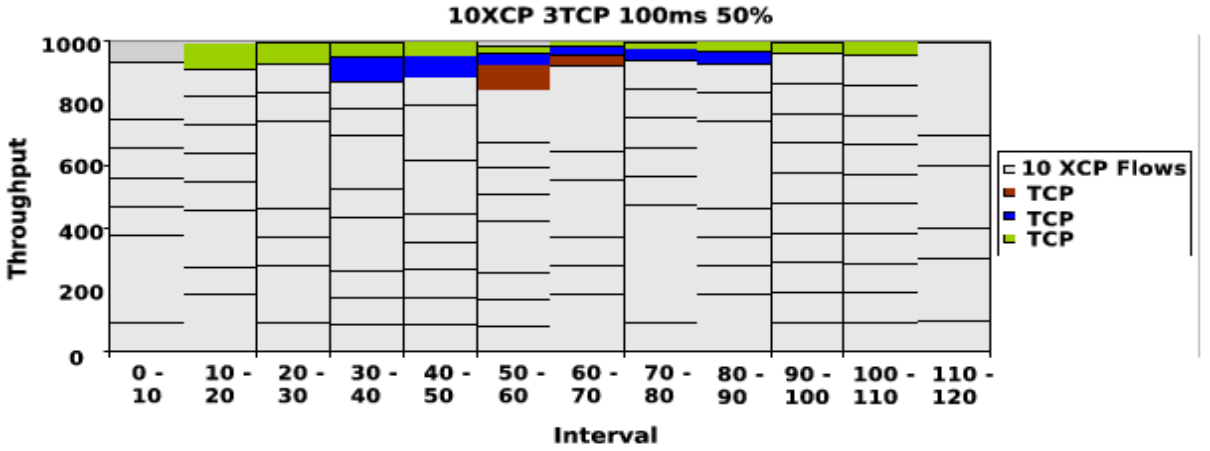


10 XCP vs 3 TCP flows
Link propagation delay = 50ms

100%



50%



Cost of our XCP-TCP fairness mechanism (2)

When analyzing 100% of incoming packets, by every one we must execute:

- 1 access to the UP header.
- Generation of 3 random numbers.
- 2 comparisons.
- 2 multiplications.
- 2 sums

Cost of our XCP-TCP fairness mechanism

When analyzing 50% of incoming packets, by every one we must :

- Generate 1 random number and 1 comparison.

And approximately by ½ incoming packets :

- 1 access to the UP header.
- Generation of 3 random numbers.
- 2 comparisons.
- 2 multiplications.
- 2 sums

Conclusions

•Last step for interoperability of XCP with external world (losses, heterogeneous equipments, heterogeneous protocols)

• Scalable and lightweight in terms of routers CPU / memory usage

• Appropriate for high bandwidth * delay product networks running long live flows.

• Limit of simulation tools (ns-2). We want to validate on real emulated XCP routers : Developing an XCP implementation for large scale validation (Grid5000 platform)

•More information on :

http://www.ens-lyon.fr/LIP/RESO/Projects/XCP