Review of "Congestion Control for High Bandwidth-Delay Product Networks" by Katabi et al.

Sreenivasa Pavan Kuppili

February 23, 2007

TCP becomes inefficienct and oscialltory (irrespective of the queueing scheme) in the presence of high bandwidth-delay links. So this paper comes up with a protocol which is efficient, fair, scalable, and stable as the bandwidth-delay product increases. The routers give explicit and precise feedback to the senders, telling the sender how to change the cwnd value (feedback of the bottleneck link prevails). The paper puts forth the idea of decoupling utilization control from fairness control. Utilization control tries to optimize the aggregate throughput. To calculate the desired amount of throughput change in the next d units of time, it uses the equation $\phi = \alpha.d.S - \beta.Q$, where α and β are constant parameters, d is the mean RTT of the flows through the link, S is the spare bandwidth, and Q is the persistent queue size. So the efficiency controller using an MIMD approach and tries to quickly attain the spare bandwidth-delay product. This MIMD approach increases the efficiency in links with a high-bandwidth delay product. The efficiency controller also tries to drain the persistent queues. This proactive reduction of queue size results in small queues and very small packet losses. The fairness controller distributes this aggregate throughput among the various flows in an AIMD way to attain fairness. If $\phi > 0$, it is allocated so that the throughputs of all the flows increase by the same amount in the same time (flows with larger RTT increase their cwnd quickly, and TCP's bias against flows with large RTT's is avoided). If $\phi < 0$, it is allocated so that the decrease in throughput is proportional to the throughput. The fairness controller can naturally be extended to perform service differentiation. Further, the protocol puts control state in packet headers and does not require per-flow state in routers, which is a good thing as having the per-flow state in a router might not be feasible for the high bandwidth links.

One disadvantage of XCP is all the routers along the path need to be XCP-enabled. Even if the source checks that all the routers along the path are XCP-enabled at the beginning of the connection, the routes can change in the middle, and this might require some state in the routers to detect this case. One serious issue which is not discussed in the paper is a router getting compromised. A malicious router has significant control over the sending rates of all the XCP flows passing through it, and it can do bad things like stifling the sending rate, or erasing the feedback of an upstream bottleneck link. Another small problem pointed out by the authors is that the shuffling of bandwidth can cause the aggregate throughtput to fall below the optimal level.

XCP is a clean-slate idea which starts off discussing how the congestion control should be, without bothering about backward compatibility or deployment. Only later does it bother about these issues. As a clean-slate idea, it is very good in theory. It uses explicit congestion feedback to optimize throughput and attain fairness. Several other advantages like small queue sizes and low packet losses were shown. One good thing is it does not require state in the routers, which makes the protocol scalable. But, this protocol violates the end-to-end principle in a sense, and gives too much power to the routers. Unless careful policing of the routers can be done, it is not good to deploy XCP in a future Internet.