Performance of TCP over a link using Fair Queueing

Extended abstract

Jordan Augé, James Roberts
{jordan.auge, james.roberts}@francetelecom.com

1 Introduction

Two recent fields of research will certainly have a major impact on the future performance of the Internet. These are firstly, the evaluation of the buffer size required to ensure fair, stable and efficient sharing of link bandwith and secondly, the proposition of new congestion control algorithms, more suitable for increasingly high link speeds than current version of TCP. In this paper, we re-examine the arguments developed in the light of the complementary proposition to use per-flow fair scheduling in router queues.

2 Buffer sizing

Despite its apparent simplicity, buffer dimensioning is not a well understood topic. A simple rule based on the operation of TCP Reno recommends a buffer size equal to the Bandwith Delay Product. Since link capacity is currently around 10GB and the delay used for the computation close to 250ms, buffer sizes are attaining the limit of feasibility. Several propositions have recently appeared in the litterature. Appenzeller et al. [1] have made a big step in favour of a drastic reduction of the buffer size in the order of the square root of the number of flows in presence. Raina and Wischik [2] argue for a fixed buffer size of 20 packets for stability reasons. Finally Enachescu [3] propose a dimensioning proportional to the maximum TCP window size.

3 High speed TCP versions

Until recently, the TCP congestion control has succeeded in making the network stable and efficient. However, as networks with high speed and long delays are more and more present, this protocol has revealed its limits and doesn't allow an efficient use of the available bandwith. Such proposals as HSTCP, FastTCP and Scalable TCP are among the most popular ; [4] provides interesting experimental results. In this paper we will only consider HSTCP [5] which has been extensively studied. The idea behind this protocol is to modify the initial AIMD algorithm with dictates the evolution of the congestion window to make it more tolerant to the losses at high rates.

4 Fair queueing

Fair queueing allows the network to give a fair share of the available bandwith to each user without relying on their cooperation. Associated with the longest queue drop policy, it provides very good performance. Its fairness and protection properties may also act in favour of the introduction of new and more efficient transport protocols, since there is no more the need to be TCP-friendly. Finally it can be useful to realize implicit service differenciation as shown in [6].

As long as link load is not higher than 90%, traffic models predict that the number of bottlenecked flows in progress is less than 100 with high probability. Thus, fair queueing is scalable since only a small number of flows need to be scheduled, independently of the link rate. It is also feasible since the maximum number of flows is around 500 at loads below 90% [7] [8].

5 Traffic on a backbone link

An essential characteristic of a data flow is its peak rate. Some flows to be scheduled are bottlenecked at a link in that they could attain a higher rate if the link in question had a greater capacity. Most flows in progress however are not bottlenecked. Their rate is limited by other constraints on their path (access links, notably) to a peak value less than the fair rate offered by the link. In order to get a proper evaluation of congestion control, we must account for a realistic traffic mix.

6 Performance results

We studied the performance of TCP flows over a fair queueing link by simulation. The topology used is a 50Mbps bottlenecked link, with a 100ms RTT. Initially, non-bottlenecked flows are represented with Poisson arrivals of TCP flows with 1Mbps peak rate. Then they are replaced with a Poisson stream of packets which has been proved to have the same characterictics. One, two or four permanent high rate flows (TCP Reno or HSTCP) are sharing the available capacity, in order to account for the most probable situations. We used several buffer sizes between 20 packets (the value recommended in [2]) and *Bandwith Delay Product*. The scheduling is either FIFO with DropTail or Fair Queueing with Longest Queue Drop. We present here the key results arising from our simulations.

By dropping the first packet in the flow which has the longest backlog, fair queueing ensures the protection of the non-bottlenecked flows which cannot reach their fair rate. Eliminating the losses and reducing the RTT if the flows is a good guarantee of quality of service, particularly when this background traffic includes real time flows, and when the buffer is large.

We should notice that the presence of background traffic highly modifies the closed-loop congestion control algorithm of TCP. A TCP connection on a bufferless link can attain 75% utilization. The larger the buffer, the higher the utilization. However, the throughput of the flow falls drastically below 50% of the available capacity as soon as we inject some background traffic : there is a non null probability that the buffer is saturated when one packet from the TCP connection arrives. Thus a 20 packets buffer seems not sufficient to ensure the performance of high speed TCP flows, as they are not able to sustain high rates and they exit their slow start phase prematurely. A larger buffer around a hundred packets provides good performance.

We notice an improvement when the number of multiplexed bottlenecked flows increases, all the more so as they become unsynchronized. Yet our model predicts that this number is reduced to a few units with high probability. Note that fair queueing has the property to desynchronize the losses and thus the behaviour of the flows.

Even if HSTCP behaves very poorly with FIFO and very small buffers, it brings gain in utilization even on such low capacities as 50Mbps. It is however responsible for higher loss rates for background flows, and appears quite unfair.

When the buffer is sufficiently sized, fair queueing is effective : each flow obtain its fair share, even with different protocols. Of course, less efficient protocols as standard TCP will have a slightly inferior throughput than HSTCP because of the drastic decrease of its congestion windows when a loss occurs. Fair queueing is also necessary to protect established connections from the effects of slow starts of other flows.

Finally, fair queueing regulates the arrivals of the packets inside a flow. It results in smoother evolutions of the queue, which minimizes its probability of overflow. Thus the flow can keep a higher window and have a better rate when this doesn't mean a larger RTT.

7 Conclusion

Those preliminary results are very encouraging in order to introduce fair queueing into the network. Further research will provide a theoretical basis for these statements, and investigate the possibility to use techniques such as *packet pair* [9] to probe the fair rate.

Références

[1] G. Appenzeller, I. Keslassy, N. McKeown, Sizing Router Buffers, Proceeding of ACM SIGCOMM '04, Portland, Oregon, September 2004.

[2] G. Raina, D. Wischik, Buffer sizes for large multiplexers : TCP queueing theory and instability analysis NGI 2005

[3] M. Enachescu, Y. Ganjali, A. Goel, N. McKeown, T. Roughgarden, Part III : Routers with Very Small Buffers, ACM/SIGCOMM CCR 2005.

[4] Y-T. Li, D. Leith, R.N. Shorten, Experimental Evaluation of TCP Protocols for High-Speed Networks,

[5] S. Floyd, HighSpeed TCP for Large Congestion Windows, IETF Internet Draft

[6] A. Kortebi, S. Oueslati and J. Roberts, Cross-protect : implicit service differentiation and admission control, IEEE HPSR 2004, Phoenix, USA, April 2004.

[7] A. Kortebi, L. Muscariello, S. Oueslati and J. Roberts, On the scalability of fair queueing, ACM HotNets-III, San Diego, USA, November 2004

[8] A. Kortebi, L. Muscariello, S. Oueslati and J. Roberts, Evaluating the Number of Active Flows in a Schedular Realizing Fair Statistical Bandwidth Sharing, Sigmetrics'05, Banff, Canada, June 2005.

[9] S. Keshav, Congestion Control in Computer Networks, PhD Thesis, published as UC Berkeley TR-654, September 1991