

Performance of TCP over a link using Fair Queueing

Extended abstract

Jordan AUGÉ, James ROBERTS

{jordan.auge, james.roberts}@francetelecom.com

1 Main trends in IP networking

1.1 Reducing the size of the buffers

Despite its apparent simplicity, buffer dimensioning is not a well understood topic. Until quite recently, a simple rule based on the functioning of TCP Reno, advised a buffer size equal to the Bandwidth Delay Product : the buffer must have enough place to never become idle when a single TCP flow is emitting. Since today's available link capacity is around 10Gb, and the mean delay used for the computation close to 250ms, buffer sizes should be very high (2,5Gb) which cause several problems of feasibility or quality of service.

Since then, several proposition have appeared in the litterature. [7] shows us the problems encountered by TCP connections which cannot place a few packets in the buffer, and thus propose a dimensioning proportional to the number of flows. The biggest step in favor of a drastic reduction of the buffer size has been done by [1]. The study shows that in presence of a huge number of multiplexed flows, it is possible to ensure a full utilization of the link with a buffer \sqrt{n} times smaller than one sized with the rule-of-thumb. [9] exposes the stability problems of big buffers and recommend a fixed buffer size of a few dozens of packets. Finally, [8] puts a fixed buffer size into question, and proposes a dimensioning proportional to the logarithm of the maximum TCP size as soon as packets arrivals are sufficiently spaced, which is the case in practice given the low access rates.

1.2 Introducing highspeed 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 bandwidth. Due to the wide spread of standard TCP, any new proposal has to be TCP friendly, that is behaving similarly and grab a fair share of capacity. Among the proposals, the more popular are HSTCP, FastTCP and Scalable TCP. In this paper we will only consider HSTCP which has been extensively studied. Note that [6] tells us that HSTCP seems to be the least unfair among those highspeed protocols.

An efficient use of a high capacity link would imply that its loss rate is far below what technology offers nowadays. Moreover, the AIMD algorithm that controls the evolution of the congestion windows is not efficient. Because of its drastical decrease and its slight increase, too much time is needed to recover throughput from a

loss. Thus, HSTCP proposes to modify this AIMD algorithm to make it more aggressive in its increases, and more tolerant to losses when the congestion window is high. It can be seen as a modification of the response graph linking the window size to the loss rate [?].

2 Traffic model on a backbone link

We distinguish two classes of traffic on the link. Some flows to be scheduled are bottlenecked in that they could attain a higher rate if the link in question had unlimited capacity. Most flows in progress at any instant 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 ordered by the link.

Assuming the sessions arrive according to a Poisson process, the link sharing realized by TCP makes the link behave like a M/M/1 queue. Supposing all flows are bottlenecked, $\bar{n} = E[\text{flows in progress}] = \frac{\rho}{1-\rho}$. Thus, for a load $\rho < 0.9$, we should have less than 9 flows (even if $\lim_{\rho \rightarrow 1} \bar{n} \rightarrow \infty$). In practice, $\rho < 0.5$ and $\bar{n} = O(10^4)$, which shows that most flows are non-bottlenecked. In fact, each flow emits packets rarely. At low loads, there is little queueing and FIFO is sufficient, but at higher loads the mix of bottlenecked and non-bottlenecked flows needs a better scheduling.

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, and that we have $O(100)$ packets from non-bottlenecked flows. Fair queueing deals only with flows having packets in queue. Since this number does not increase with link rate, but just depends on the proportion of each class of flows, fair queueing is scalable. It is also feasible since the maximum number of flows is around 500 at loads below 90% [3] [4].

3 Performance of TCP over a FQ link

We studied the performance of TCP flows over a fair queueing link by simulation. The topology used for the simulation is a 50Mbps bottlenecked link, with a 100ms RTT. We establish on this link a 25Mbps background traffic with Poisson arrivals of TCP flows with 1Mbps peak rate (or Poisson arrivals of packets for the sake of simplicity), and 1, 2 or 4 permanent high rate flows (TCP Reno or HSTCP) are sharing the available capacity left. We used several buffer sizes between 20 packets

and a rules-of-thumb dimensioning. The scheduling is either FIFO with DropTail or Fair Queueing with drop from front of the longest queue. We present here the key results illustrated by our simulations.

Fair Queueing protects non bottlenecked flows

By dropping the first packet in the flows which has the longest backlog, fair queueing ensures the protection of the flows which cannot reach their fair rate. Those flows emit packets rarely, thus they only need to be scheduled from time to time, while bottlenecked flows have several packets backlogged in the buffer. It has been shown that this number is around a hundreds with a high probability. Fair queueing thus eliminates losses for the background flows, and reduces their RTT, which guarantees their quality of service, particularly when the buffer is large.

Small buffers may not be sufficient First 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 (in fact a one packet buffer) can achieve 75% utilization. The more the buffer, the more the utilization. But if we feed the link with some background traffic, the throughput of the flow falls drastically below 50% of the available capacity : there is a non null probability that the buffer is saturated by packets from the background traffic when a packet from the TCP connection arrives. The performance gets better when the number of bottlenecked flows increase, all the more so as they become unsynchronized. Yet, our model predicts that this number is reduced to a few units with a high probability.

A 20 packets buffer seems to small to ensure the performance of high speed TCP flows, as they will not be able to sustain high rates even with fair queueing. We noticed that HSTCP behaves very poorly with such small buffers, and exits its slow start phase prematurely. This is a major issue with standard TCP which also suffers from an unefficient AIMD algorithm for high rates. Though, a buffer sized to the *Bandwidth Delay Product* may not be necessary, is we consider better protocols. As soon as the buffer is around 100 packets, we obtain very good results for the fair queueing - HSTCP association.

Compared to TCP Reno, HSTCP brings gain in utilization even on a 50Mbps link, but is responsible for higher loss rates for background flows. With a small buffer, fair queueing avoids background losses but has little impact on bottlenecked flows. While TCP Reno offers with FIFO DropTail an approximate fairness, HSTCP remains unfair.

Fair queueing ensures fairness between bottlenecked flows

With a large buffer, fair queueing is effective : two flows sharing the same link (both TCP Reno or HSTCP, or a mix of them) will have a higher throughput, and each one will approximately get its fair share. This makes the coexistence of several TCP protocols possible and their TCP friendliness is no more required. More efficient propositions can be tested since fair queueing will enforce fairness. With a mix of a HSTCP and a standard TCP flows, we can notice that the former one can't gather all its available bandwidth because of its AIMD

algorithm : after a window halving, the HSTCP flow will grab more bandwidth until the standard TCP flow recovers. Fair queueing is also necessary to protect established connections from the effects of slow starts of other flows.

We can also note that fair queueing regulates the arrivals of the flows. 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 higher rate when this doesn't mean a higher RTT. The longest queue drop tackles the issue of losses synchronization.

4 Conclusion

Those results are very encouraging to introduce fair queueing in the network, associated with admission control to ensure scalability. More than the benefits in protection, fairness and efficiency, such mechanisms allow the introduction of more efficient TCP protocols, not necessarily TCP friendly. We may consider, for example, techniques such as packet pair [5] in order to probe the fair rate.

Références

- [1] Sizing Router Buffers, G. Appenzeller, I. Keslassy, N. McKeown, Proceeding of ACM SIGCOMM '04, Portland, Oregon, September 2004.
- [2] HighSpeed TCP for Large Congestion Windows, S. Floyd, IETF Internet Draft
- [3] Evaluating the Number of Active Flows in a Scheduler Realizing Fair Statistical Bandwidth Sharing, A. Kortebi, L. Muscariello, S. Oueslati and J. Roberts, To Appear in Sigmetrics'05, Banff, Canada, June 2005.
- [4] Evaluating the number of Active Flows in a Scheduler Realizing Fair Statistical Bandwidth Sharing, A. Kortebi, L. Muscariello, S. Oueslati and J. Roberts, SIGMETRICS'05, Banff, Canada, June 2005
- [5] Congestion Control in Computer Networks, S. Keshav, PhD Thesis, published as UC Berkeley TR-654, September 1991
- [6] Experimental Evaluation of TCP Protocols for High-Speed Networks, Y-T. Li, D. Leith, R.N. Shorten,
- [7] Scalable TCP Congestion Control, R. Morris, In Proceedings IEEE INFOCOM 2000, March 2000.
- [8] Part II : Control theory for buffer sizing G. Raina, D. Towsley, D. Wischik, ACM/SIGCOMM CCR 2005.
- [9] Buffer sizes for large multiplexers : TCP queueing theory and instability analysis G. Raina, D. Wischik, NGI 2005