Performance of TCP over a Fair Queueing Link

Jordan Augé and James Roberts France Télécom Division R&D
Issy-les-Moulineaux, France

***Abstract*— 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.**

I. Introduction

The internet has been designed to be a Best-Effort network. Hence the basic service it offers is the transport of packets, ensured by the IP protocol. The need to establish reliable connections between two end hosts is dedicated to the TCP protocol, built over IP. TCP has evolved to integrate congestion control as we know it today, and has become the major protocol in the network with regards to the volume of traffic.

Congestion control mechanism based on a window of unacknowledged packets is at the basis of buffer dimensioning in the routers. This has consequences on router buffers since they have to be designed to ensure a good quality of service for TCP flows. Explain Bandwith Delay Product.

- tcp dominated links
- Villamizar et al.
- B = C * RTT (avg rtt of the flows traversing the link)

This rule leads to very huge buffers. For instance, a buffer on an OC192 link (10Gb/s) would require 2.5Gb buffers to provision 250ms of traffic (the value commonly used, which represent a transatlantic Round Trip Time). On one hand, such big buffers will offer a poor QoS for interactive traffic (e.g. VoIP), because of the latency and jitter such buffers create. On the other hand, the more link capacity will increase, the harder the realization of such big memories will be.

Several criticisms are addressed to large buffers. The more obvious one is in the plan of the quality of service which is severely degraded by the latency and uncertainty they add. Real time and interactive applications such as voice over IP (VoIP) have strict requirements on delays, while elastic data transfers see their throughput reduced by the increase of the round trip time due to waiting time in buffers.

But technical realisation of such buffers is also getting more and more problematic with the increase of the link capacities [1]. The evolution of electronic memories hasn't grown thus quickly, so that currently we can distinguish two kinds of chips. SRAM provides fast access times (microsecond), but is very expensive, power consuming and offer a small amount of buffering. On the contrary, DRAM chips are convenient to implement cheap large buffers, but they cannot sustain high link rates. In practice we find hybrid designs based on a combination of both types, with intelligent dimensioning and synchronization between [2]. **??** gives an example of such scheme and how it can be used to store a FIFO queue

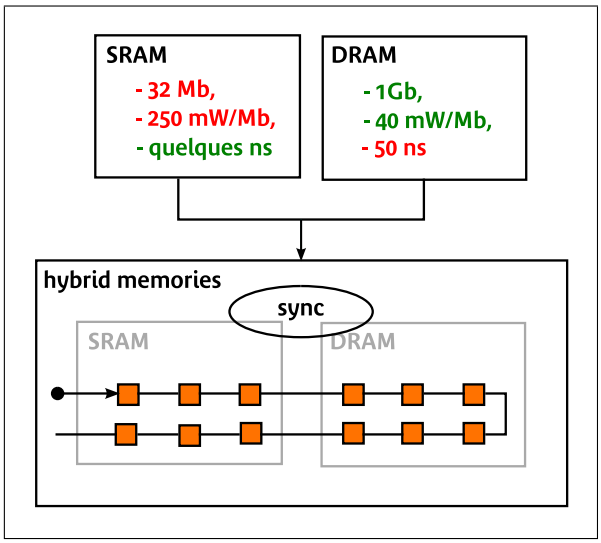

Fig. 1.

Another interesting remark is a misconception of the importance of the full utilization of a link. Though it is a good criterion to estimate the quality of service for data flows, it can be sometimes better to find a compromise between throughput and buffer size. All the more so that a reduction of buffer size could reduces costs, allow the integration of memories within the chips, giving the opportunity to rise up the capacities. This argument is

particularly interesting when it comes to consider all-optical networks, where only a few packet buffers can be emulated by the utilization of delay lines.

II. Recent proposals for buffer sizing

While the topic has remained relatively not much active, many recent proposals have argued for a drastic reduction of buffer size. The models they use generally assume FIFO scheduling with a DropTail policy.

Appenzeler *et al.*[1] state that on links with a high level of aggregation, such as in the backbone, some TCP flow desynchronization dynamics occur which help keeping buffers much smaller without sacrificing utilization. Synchronization effects are known to appear in usual networks since losses happen at instants of saturation of the buffer, and then affect most of the flows. Simulation has shown that statistical multiplexing of a high number of flows, say a few hundreds, leads to a high level of desynchronization, that is a reduction of the variance of the sum of the congestion windows of the connections. Basically, a gaussian approximation gives the result that buffer size can be cut to the Bandwith Delay Product divided by the square root of the number of active flows. When this number is equal to millions like in the backbone, it represents a improvement of several orders of magnitude.

Dovrolis :

- when 80% of the traffic is constituted by bottlenecked flows
- part I dit que ce resultat vaut quand les flots sont partiellement synchronisés
- adding some queueing delay reduces drop and load, bad tradeoff (part I) since losses can be recovered but not the delay (rt applications).

However it is not recent, Morris observes on simulations that TCP connections behaves quite badly (several timeouts) when they cannot place more than a few packets in the queue. Thus he recommends a dimensioning proportional to the number of flows which will be supported by the link.

[3] introduce an analytical model of synchronization for AIMD-based protocols. They claim that large buffers create instability for the connections, which is at the same time the main reason to introduce themi. They identify in which case they obtain the best overall results, and thus advice the use of small buffers, around 20 packets and no more than 50. Such as drastic reduction should annihilate the issue of flow synchronization. They also showed that the square root formula presented before was not a good plan when number of flows grows for stability concerns, and that again small buffers should be employed.

Another vision of routers with very small buffers is presented in [2]. Since buffers are present to account for bursts of traffic, it is natural to consider paced TCP flows. In typical network architectures, backbone link rates are far beyond access link rates, in such a way that the link introduces a natural spacing of packets. The paper gives bounds for which this assumption apply, which enable to accept small buffers (again around 20 packets) as far as we are willing to sacrifice a small part of throughput (at most 25%).

Why there is no self-pacing of TCP...

Pacing packets by client at the edge of the network has also be envisaged, but despite some good results, it is known to have several flaws. [2] tells us about some synchronization problems that can arise from the error due to a delayed feedback of the congestion. We can also imagine that requiring a host to compute and schedule the departure time of each packet is not tractable, because of both the CPU load it will create, but also because of certain timer granularities that cannot be bypassed.

The main issue with those models is that they generally assume multiplexing a large number of permanent flows, all of which are implementing TCP. We are to see that it is in fact not the case, and that a more realistic vision of the traffic mix can give very different results regarding the performance.

III. Traffic characteristics at flow level

A. Flow structure of traffic

a flow = ... micro flow local to link considered most flows use TCP and are elastic ; some streaming flows use UDP or TCP, generally require low packet delay and loss arrival process of flows, size distribution...

B. Numbers of flows

numbers of flows (statistics from traces) proportional to link rate and load => rate independent of link rate (because they are not bottlenecked) (exogenous peak) rate of flows (statistics from traces - eg ADSL users, vs Abilene) - notion of bottlenecked/non-bottlenecked flows number of bottlenecked flows geometric distribution cf PS model ; 0, 1 or 2 flows with high probability

C. Link utilization regimes

three regimes ("transparent" - all non-bottlenecked, "elastic" - some bottlenecked, "congested" - too many arrivals) realistic traffic mix = many non-bottlenecked flows (modulated Poisson arrival process (cite Cao, Cleveland) low amplitude modulation) + 0, 1 or small

number of bottlenecked flows ; NB many bottlenecked flows only occurs in congested regime.

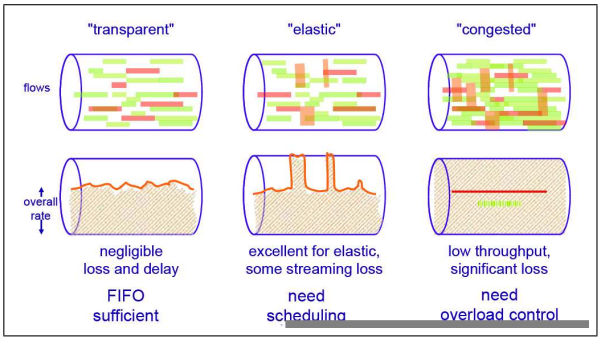

Fig. 2. Link operating regimes

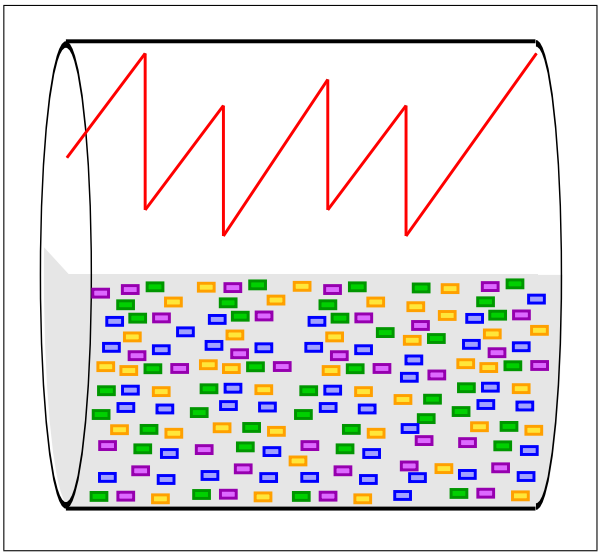

Fig. 3.

IV. Buffer sizing considering a realistic traffix mix (FIFO + Reno)

A. Buffer sizing in transparent regime

While the load of the link remains not too high, and we have only non-bottlenecked flows, it suffices to use well-known formulas of bufferless multiplexing to ensure a sufficient provisioning of the buffer size. The M/M/1 queue provides a good approximation of the behaviour of the queue, which only depends on the load. It is thus possible to ensure a small loss rate with high probability. For instance, at load 0.9, Pr[100 packets] $< 3.10^{-5}$. And the delay caused by the transmission of 100 MTU packets (1500B) on a 10GB link is less than one tenth of a millisecond). In practice, measurements on routers at low loads (there is a very low probability to have bottlenecked flows) show that buffers are hardly used, and that the delays are satisfactory.

INSERTING A FIGURE HERE ?

B. Buffer sizing in the elastic regime

In case there are some bottlenecked flows on the link, a proper dimensioning of the buffer size is not so easy. A set of simulation helps understanding which issues arise from this traffic mix, and are a starting point to a further investigation on how things work.

1) The simulation scenario: The scenario is very close to the traffic model presented previously. The topology is a typical bottlenecked link whose characteristics are presented on 4 and 5. Non-bottlenecked flows are represented by Poisson arrivals of finite TCP flows (peak rate of 1Mb) 4, which we approximate by Poisson packet arrivals 5 for the sake of simplicity since in both cases we have rather the same performance for the indicators we consider. They represent a load $\rho = 0.5$ in most cases, but which will evolve between 0.1 and 0.8. Bottlenecked flows are permanent long TCP flows, using either TCP Reno of HighSpeed TCP (it will be presented in the following). Scheduling is either FIFO or Fair Queueing and the buffer size will be made to vary between 20 packets and 625 packets (the Bandwith Delay Product).

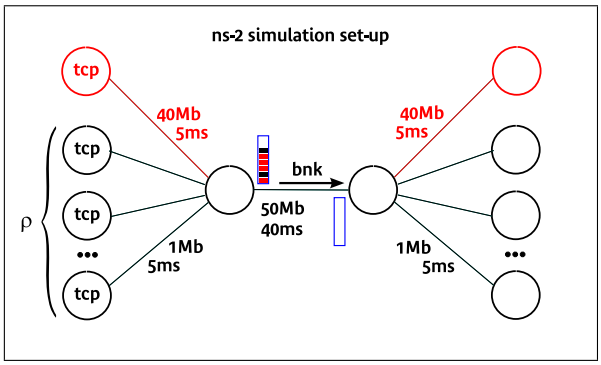

Fig. 4. Simulation scenario with arrivals of finite TCP flows

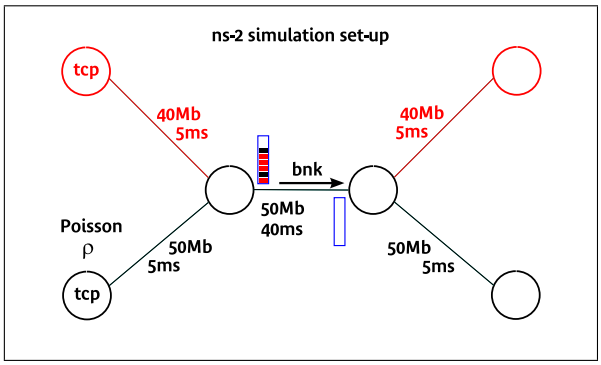

Fig. 5. Simulation scenario with Poisson packet arrivals

2) Simulation results with 1 TCP flow: When a TCP flow is alone on a link with no buffer, it is easy to see on a fluid model that it will be able to use 75% of the available capacity.

The throughput of the flow is expected to increase linearly with the buffer size. In order to make some comparisons, we simulated such a scheme on a 25Mb link (so that the available bandwith is the same than on a 50Mb link loaded at 50%). 6 plots the proportion of available capacity the TCP flow is able to use as a function of the buffer size. For very small buffer sizes we get the behaviour predicted by Morris [2] and the flow cannot behave correctly. Then, it seems possible to consider sacrificing a small part of the throughput (less than 25%) in order to reduce the buffer size significantly.

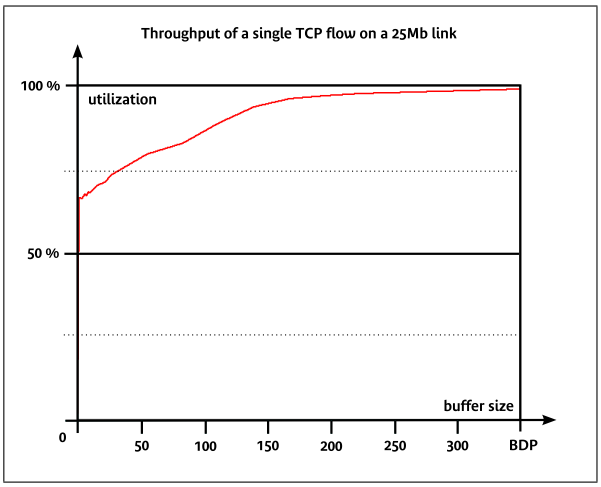

Fig. 6. Utilization of the available capacity by a single TCP flow with no background traffic

Though, this statement is no more true if the TCP flow is mixed with some background traffic . 7 represents the evolution of the utilization of the link as a function of time on the 50Mb link with 20 packet buffers. Considering that background flows represent 50% of the load of the link, it appears quite clearly that the TCP connection cannot manage to use available capacity. The throughput of the flow falls drastically below 40% utilization. When using a large buffer (Bandwith-Delay Product) 8, link utilization is again close to 100%.

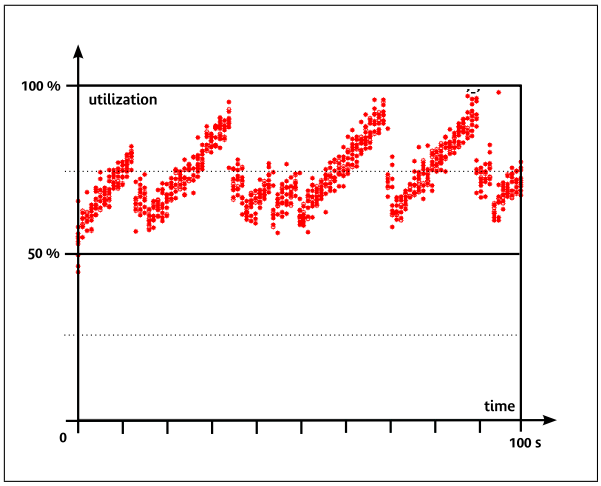

Fig. 7. Utilization of the link with a load of 50% of non-bottlenecked flows and 1 bottlenecked flow - buffer = 2O packets

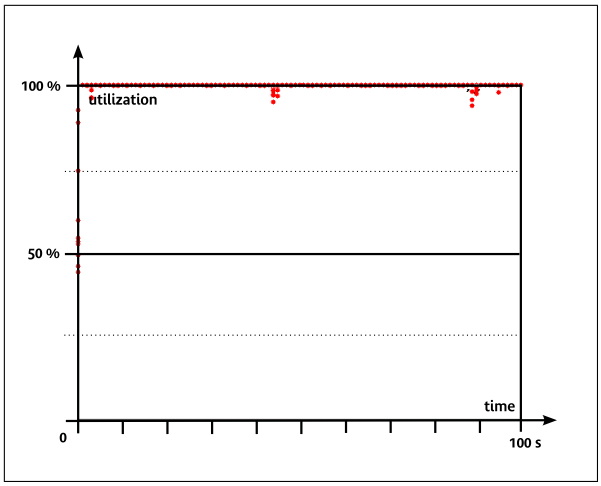

Fig. 8. Utilization of the link with a load of 50% of non-bottlenecked flows and 1 bottlenecked flow - buffer = 2O packets

3) Simulation results when multiplexing many flows: 9 plots the evolution of the utilization of the link as a function of time when 4 TCP Reno flows are competing on the link. The buffer size is 20 packets and this size implies some kind of desynchronization of the flows [3]. Thus we see that the multiplexing of the flows can help reducing the imapct of the buffer on the throughput : less abrupt changes in the sum of the window sizes have to be supported by the buffer. Yet, we have to keep in mind that the model predicts a small number of bottlenecked flows. We cannot then rely on the multiplexing of the flows. The case with 1 bottlenecked is thus a highly probable worst case. We notice that it is also when changing between 1 and 2 bottlenecked flows that we will suffer from the more abrupt changes regarding the available bandwith on the link.

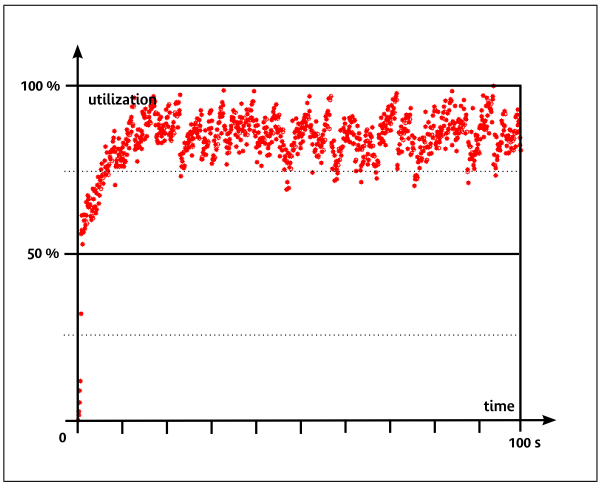

Fig. 9. Utilization of the link with a load of 50% of non-bottlenecked flows and 4 bottlenecked flows - buffer = 2O packets

INSERT HERE THE UTILIZATION OF THE LINK WRT THE NUMBER OF FLOWS WITH THE PROBABILITY TO HAVE SUCH A SCHEME

TALK ABOUT LOSSES FOR NON-BOTTLENECKED FLOWS

4) The trade-off: On one hand, small buffers offer a poor throughput for bottlenecked flows and cannot guarantee a low loss rate for non-bottlenecked ones. On the other hand, large buffers aren't scalable and lead to packet latency which penalizes both types of flows. It should be possible to establish a relation between buffer size and performance results which could help designing network buffers by chossing an optimal trade-off. In 10, we plot the relation between the buffer size and the throughput of a single TCP flow.

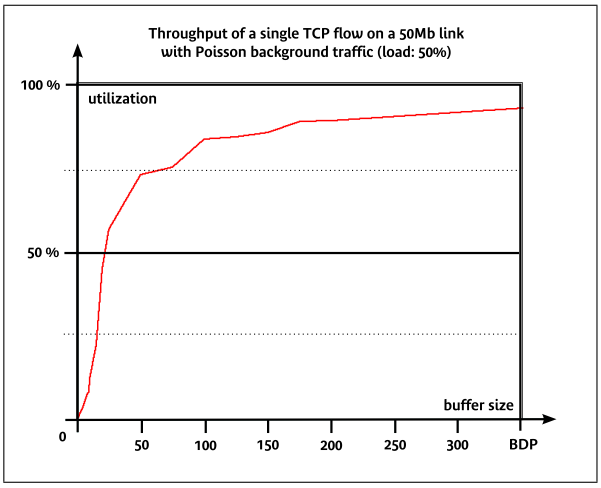

Fig. 10. Utilization of the available capacity by a single TCP flow with 50% poisson background traffic

INSERT HERE SOME COMMENTS AND PLOTS ABOUT THE INFLUENCE OF CAPACITY AND LOAD

Though, understanding the mechanisms behind the poor performance of small buffers is necessary since the results are directly appliable on all optic networks.

C. A simple model to understand the performance with 20 packet buffers

The simulations can help understanding why TCP cannot use residual capacity :

- why 1 TCP cannot use residual capacity
- local load =approx 1 when window emitted
- figure with scatter plot

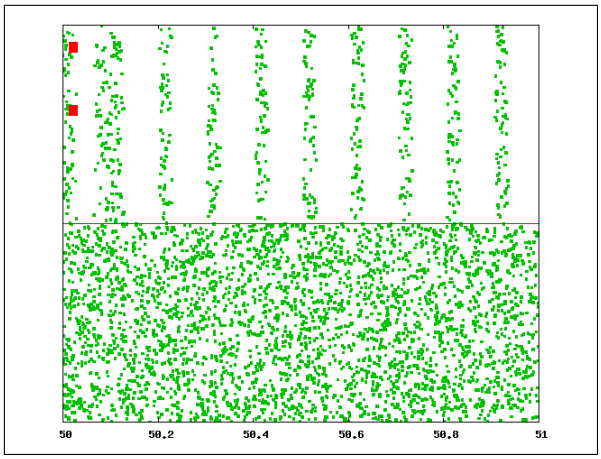

Fig. 11.

Future work, better than poisson ?

V. Possible enhancements to bandwith sharing

A. Introduction of new high speed protocols

Until now, we have been investigating the performance of TCP flows on high bandwith links. Though, TCP Reno is known to be inefficient in such conditions. The problems lies on the Additive Increase Multiplicative Decrease (AIMD) congestion control algorithm. Indeed, when a loss is detected (which can have other causes than congestion), the halving of the congestion window is too drastic, and the time it takes to reach again the fair rate is too consequent (sometimes a few hours). Another aspect pointed in [2] is related to the response function of the congestion window to the loss rate. Indeed, the losses due to physical errors on the medium is far too large so that TCP can attain large windows. This observation is the main reason for the introduction of new protocols such as HSTCP and can be a starting point to require that protocols adapt themselves to the network, and not the opposite.

1) Modifications to the AIMD algorithm: Scalable TCP and HighSpeed TCP : cf my presentations

2) Estimating queueing delay: Fast TCP :

– cf Vegas
– a binary signal is not sufficient
– need a correct dimensioning of the buffers
– FINALLY, what if different protocols, maybe instabilities cf [3] ?

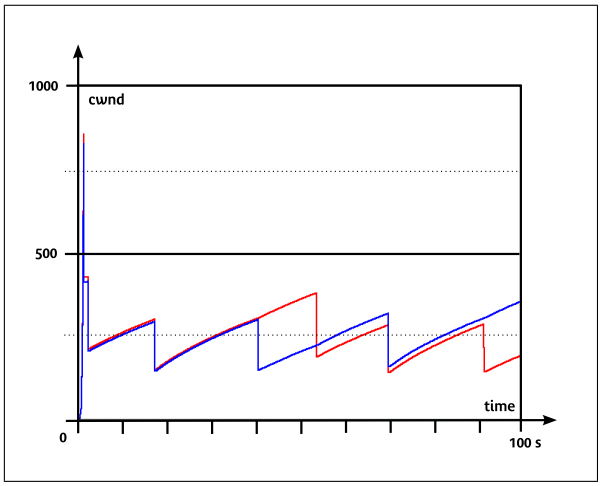

Fig. 13.

B. Impact of the use of HighSpeed TCP in our simulations

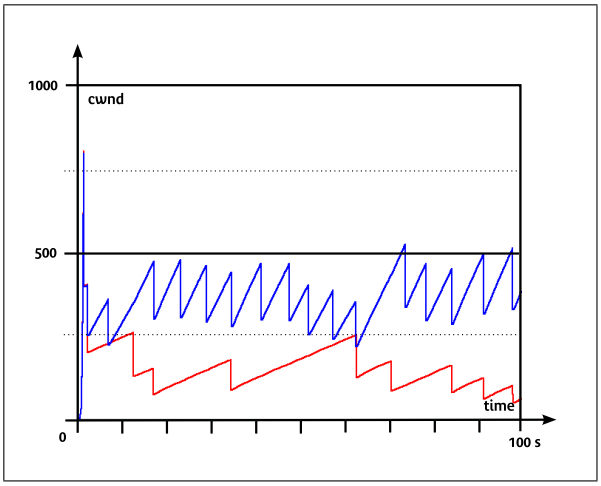

Fig. 12.

HSTCP improves utilization, but :

– more loss for non-bnk flows
– severe bias in favour of HSTCP

– HSTCP + Reno, ok
– ok for non tcp friendly protocols, unresponsive flows, cohabitation of flows
– though, HSTCP will have a little advantage because of the AIMD halving which is too severe
– protection of non-bottlenecked flows (losses, delay)
– little gain in throughput
– tendency to desync flows
– tendency to pace packets since it interleaves many flows

C. Fair Queueing

– Feasability and scalability of fq has been proved provided that the load stays under control [2].
– Problems of being TCP friendly, Problems of unresponsive flows.
– TODO : FQ and desync.

Until now, we have only been interested about the global utilization of the link, without any concern about individual performance. With both small and large buffers, fairness is approximative with TCP Reno (see fig. **??** for the case of large buffers). If the introduction of fair queueing can solve the issue of fairness in case of not too small buffers, it has almost no effect when applied on a link with 20 packet buffers.

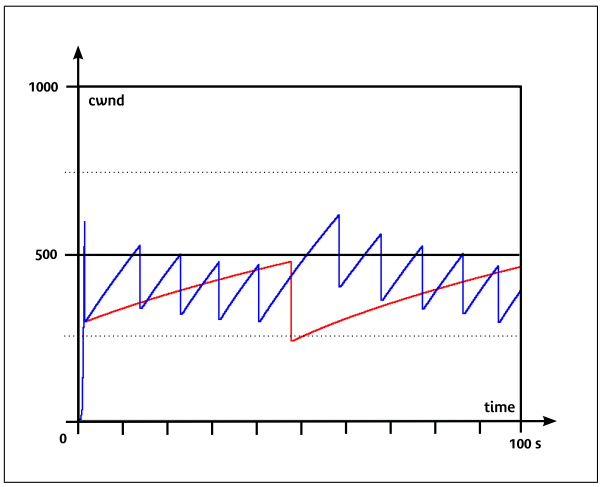

Fig. 14.

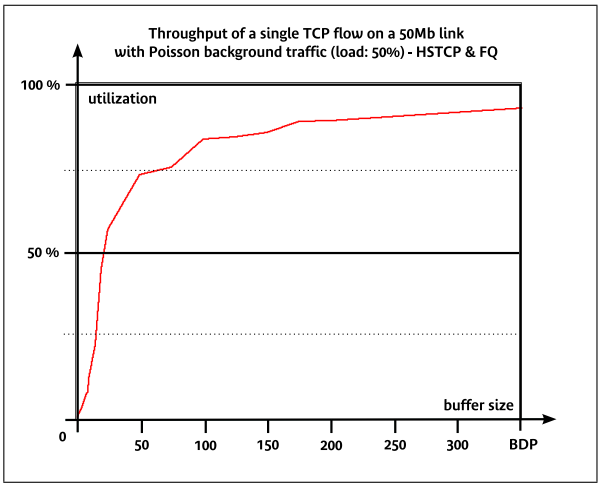

Fig. 15.

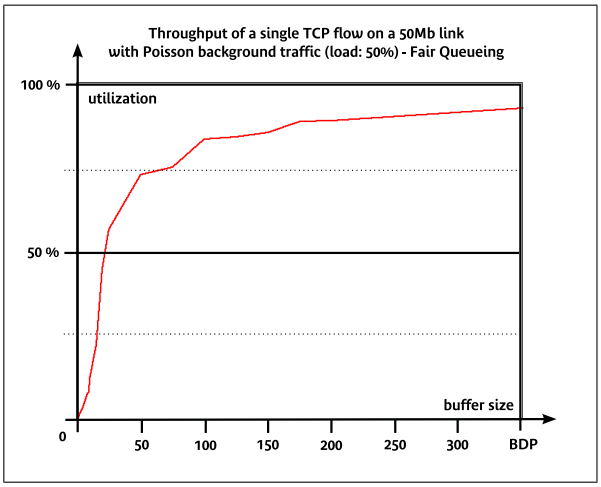

Fig. 16.

D. Perspectives

Maybe packet pair is similar to pacing packets if we want to send packets at the fair rate, but we could envisage the use of PP only to fix the size of the congestion window. Avoid SS which is a cause for losses.

VI. Conclusion

TODO : parler du slow start TODO : parler des temps de convergence TODO : parler de l'influence du RTT TODO : attacks related to buffer size (small buffers, sensitive to low rate attacks which inject several packets, no more this pb with Fair Queueing) TODO : need for experimental simulation

TODO : no model accounting for loss rate according to the window size TODO : Poisson approximation is too tight need for understanding traffic performance, relation between protocols, scheduling, buffer size and demand.

NOTE : scalable TCP a ete concu pour eviter la synchronisation des flots (part II)

References

[1] G. Appenzeller, I. Keslassy, and N. McKeown, "Sizing router buffers," *Proceeding of ACM SIGCOMM '04, Portland, Oregon*, September 2004.

[2] TODO, "Todo," .

[3] G. Raina and D. Wischik, "Buffer sizes for large multiplexers : Tcp queueing theory and instability analysis," *NGI'05*.