# Buffer sizing for elastic traffic

Jordan Augé, James Roberts

# Buffer Sizing for Elastic Traffic

Jordan Augé and James Roberts
France Télécom R&D Division
{jordan.auge, james.roberts}@francetelecom.com

***Abstract*— Two areas of active research are likely to 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 bandwidth and secondly, the proposition of new congestion control algorithms, more suitable for increasingly high link speeds than the current version of TCP. In this paper, we re-examine propositions for a drastic reduction in buffer size in the light of our understanding of the nature of Internet traffic at flow level. We demonstrate through simulation that the trade-off between throughput and delay is not as favourable as previously predicted. We also illustrate the significant gains in this context that would accrue from the implementation of per-flow fair scheduling in router queues.**

NGI 2006 Topic: *Evolution of the IP network architecture*.

## I. Introduction

As the transmission speed of Internet links continues to increase, the sustainability of the current rule of thumb used for sizing router buffers has recently been called into question [1]. The rule is to provide a buffer capable of storing the data that can be transmitted at link rate in one round trip time (RTT). With an assumed maximum RTT of 250ms, the buffer requirement for an OC768 (40Gbps) link would be some 1.2GBytes. Realizing such high speed memories in electronic routers is a significant design challenge. In optical routers, it is very hard to build buffers with a capacity of more than a few tens of packets.

The origin of the rule of thumb is that this amount of buffering is necessary to ensure full utilization when a link is used by one TCP connection or by a group of TCP connections undergoing synchronized losses [2]. This requirement is not obviously relevant for very high speed links that typically handle hundreds of thousands of simultaneous flows. Indeed, studies of the performance of bandwidth sharing by a very large number of flows using TCP reveal performance degradations due to instability when the buffer size is large. It has been demonstrated that a buffer with a capacity as small as 20 packets has much better performance [3]. Clearly such small buffers would also have the advantage of ensuring very low latency for any streaming flows sharing the link in question.

In the present paper we question the assumptions about the nature of link traffic used in the studies referred to above. Specifically, we contest the notion that a large number of TCP connections are actively *competing* for a share of link bandwidth. The vast majority of the hundreds of thousands of connections using an OC192 link at any time are in fact limited in rate by other constraints on their path. These constraints include the users' access link which is the most common flow bottleneck. The number of connections that are actually bottlenecked on a considered high speed link is arguably very small.

Non-bottlenecked flows together consume a large part of the link bandwidth and require a relatively small buffer to avoid significant loss. However, if some traffic is generated by users whose flow rate is otherwise unconstrained, it is highly probable that the number of such flows in progress at any time is very small. When only one such flow is in progress, for instance, the rule of thumb applied to the residual link capacity is indeed necessary to ensure full link utilization.

Despite this observation, we do not conclude that large buffers are desirable or necessary. In the first place, full utilization is not essential. More significantly, the assumption that connections use legacy New Reno TCP is not appropriate when discussing buffer size for high throughput bottlenecked flows. It is now well-known that the particular additive increase multiplicative decrease algorithm of legacy TCP cannot sustain high throughput. New versions of TCP have been proposed that are more responsive to congestion and do not require such large buffers.

The performance of congestion control in the present context has been studied assuming FIFO queuing with a simple drop tail discard policy or with Gentle RED active queue management (in [4]). In previous work we have argued that it would be preferable to implement per-flow fair queuing instead [5]. Under the realistic assumption that link traffic is composed of a large number of non-bottlenecked flows and just a relatively small number of

bottlenecked flows, it has been demonstrated that such a scheduling mechanism is scalable and perfectly feasible [6].

In this paper we therefore seek to evaluate the impact of buffer size on congestion control and bandwidth sharing performance assuming a realistic traffic mix. We use simulation to illustrate the potential for throughput loss with small buffers and to investigate the impact of more responsive congestion control and the use of fair queuing.

## II. Buffer sizing and TCP

In this section we review recent work on buffer sizing and discuss the impact of new TCP versions and non-FIFO scheduling.

### A. Requirements for a large number of TCP connections

The recent paper by Appenzeler et al. [1] argues that the rule of thumb used for buffer sizing (capacity = bandwidth $\times$ RTT) is not appropriate for high speed links because they typically handle a very large number of simultaneous flows. In this case it is unlikely that TCP window sizes evolve synchronously and the combined buffer requirement is considerably less than the delay bandwidth product. They evaluate the mean and variance of the window size of each flow and approximate the distribution of the overall input rate by a Gaussian. The authors' conclusion was that a sufficient buffer requirement is the bandwidth delay product divided by the square root of the number of flows in progress. This typically reduces buffer requirements for a high speed link by two to three orders of magnitude.

Raina and Wischik [3] consider the performance of TCP congestion control with many connections under the assumption of small, medium and large buffer sizes. They conclude that the protocol brings instability (wide fluctuations in queue size and loss of utilization) for medium and large buffers, including buffers dimensioned following the recommendation in [1]. They point out that instabilities for medium size buffers occur for a number of connections somewhat larger than was used in the simulations reported in [1]. Their recommendation is for a very small buffer of 20 packets, for any link size.

Results from the above papers, [1] and [3], are summarized and consolidated in [7].

An alternative approach to buffer sizing is presented by Enachescu et al. in [4]. A significant observation in that paper is that most TCP connections do not emit bursts at rates close to the link rate since packets are naturally paced by the users' lower speed access links. The authors also account for the fact that the rate a connection can attain is limited by the maximum TCP window size. They recommend very small buffers of no more than 20 packets while recognizing that this can lead to some loss of utilization.

### B. TCP versions

The evaluations discussed above assume connections use TCP Reno. It may be argued that this protocol is not designed for high speed links and that buffer requirements should be re-examined in the light of newly proposed high speed versions of TCP.

RFC 3649 [8] defines so-called High Speed TCP (HSTCP). Noting that TCP Reno cannot sustain high throughput unless the rate of packet loss or error is unreasonably small, it is necessary to make the additive increase multiplicative decrease (AIMD) algorithm more aggressive. In order to additionally realize fair sharing with TCP Reno when the packet error rate is relatively high, HSTCP uses increase and decrease factors that vary with current window size.

FAST is another proposal for a more responsive high speed protocol [9]. Window size is adjusted based on estimated queue lengths as in TCP Vegas, this being considered a more comprehensive indication of congestion than the binary signal of packet loss (or marking).

Scalable TCP replaces AIMD by MIMD (for multiplicative increase multiplicative decrease) [10]. This yields a stable and efficient congestion control at all link speeds. These advantages come, however, at the cost of a loss in fairness when several flows share the link.

A number of disadvantages of the above protocols have been identified, notably by Li et al. [11]. The main drawbacks are slow convergence to fairness and incompatibility with legacy versions of TCP. The authors of [11] propose an new protocol, H-TCP, that is shown to outperform the alternatives.

In our evaluation below we consider only HSTCP using the publicly available ns2 implementation.

### C. Buffer management

The impact of Gentle RED on the stability of congestion control and consequent buffer requirements is evaluated by Raina et al. [12]. This AQM mechanism can ensure stability with larger buffers than possible with drop tail though not over the entire range of window sizes. The impact of alternative AQM mechanisms is an open question.

A more radical alternative to FIFO or AQM is to implement per-flow fair queuing. The well-known advantages of fair queuing include the fact that, fairness being ensured independently of user behaviour, it is possible to

introduce and experiment with new TCP versions with no detrimental impact on users of the legacy version. Fair queuing has the additional advantage of ensuring low packet latency for relatively low rate streaming flows.

Fair queuing might be considered infeasible if the number of flows to be controlled is large and increases with link speed, as assumed in the work referred to in Section II above. We argue in the next section that this assumption is in fact inappropriate. The number of flows to be scheduled is relatively small and independent of link speed, as demonstrated in [13], [6].

Widespread implementation of fair queuing would make it possible to use alternative congestion controls like the packet pair proposal of Keshav [14]. It remains to investigate the buffer requirements of such protocols.

## III. Traffic characteristics at flow level

In this section we discuss the nature of Internet traffic at flow level noting the significance for the present evaluation of a flow's exogenous peak rate. This determines whether the flow will be bottlecked or not by the considered link.

### A. Flow structure of traffic

IP traffic is composed of finite size flows. These flows arrive according to a certain stochastic process and each brings a certain amount of work equal to its size in bits. It is useful to distinguish elastic flows whose rate can vary to fill available capacity, usually under the control of TCP, and streaming flows that have an intrinsic rate determined by a codec. The number in progress varies as new flows are initiated, remain active for a certain duration and then leave.

The most significant traffic characteristic for performance is the average link load, equal to flow arrival rate $\times$ size / link rate. In many networks this load is very small (less than 50%, say) though our evaluation predicts performance would be satisfactory at higher levels (up to 80 or 90%, say).

At fixed load, the number of flows in progress at a given load depends on the distribution of exogenous flow rates. The exogenous rate is the rate the flow would realize if the considered link had infinite capacity. Though this rate varies quite widely throughout a flow's lifetime, the variations within a flow are much less significant than differences between distinct flows. Some flows have a high exogenous rate; most are limited by other constraints on their path (notably by access links) to a rate much smaller than the link rate.

For a given link capacity and load, it is significant to distinguish flows that are *bottlenecked* - their exogenous rate is greater than the rate to which they are limited by the link in question - and flows that are *non-bottlenecked* - their rate is not strictly limited by the considered link (although they may lose some packets due to confluent traffic from the other flows). Analysis of traces from a variety of network links reveals that the vast majority of flows (and often all flows) are non-bottlenecked [6], [15].

### B. Number of flows

The duration of non-bottlenecked flows is largely independent of the considered link (local packet loss will extend the duration of elastic flows but only marginally compared to losses on their actual bottleneck). The expected number, equal to the product of the arrival rate and flow duration, is thus roughly proportional to link capacity (assuming constant load). Trace statistics reveal a mean per flow rate of a few tens of Kbps so that an OC192 (10Gbps) at 50% load would have some hundreds of thousands of non-bottlenecked flows in progress.

The duration of bottlenecked flows depends on link capacity and load. The number of such flows in progress can be roughly estimated using a queuing model. If we assume all flows are elastic and bottlenecked and that congestion control realizes perfect fair sharing, the processor sharing model is a reasonable statistical bandwidth sharing model [16]. This model predicts a geometric distribution for the number of flows in progress:

$$\Pr[n \text{ flows}] = (1-\rho)\rho^n$$

where $\rho$ is the link load. Note that if $\rho = 0.5$, for example, the probability there are more than 4 flows in progress is only 0.03.

In practice there is a mixture of non-bottlenecked flows and bottlenecked flows, the latter dynamically sharing the residual bandwidth unused by the former. If the traffic due to the bottlenecked flows (arrival rate $\times$ size) is equal to half the average residual capacity, for instance, the above results from the processor sharing model suggest we will rarely have more than 4 bottlenecked flows sharing the link with a large number of non-bottlenecked flows.

The number of bottlenecked flows is large only when overall link load is very close to 1. If the offered load exceeds 1, the number of flows increases (arrivals occur more frequently than departures) and all flows eventually become bottlenecked. The generally considered scenario of a large number of bottlenecked flows is thus hardly typical. It is in fact exceptional and, in our opinion, not useful for evaluating the effectiveness of congestion control algorithms or assessing buffer requirements.

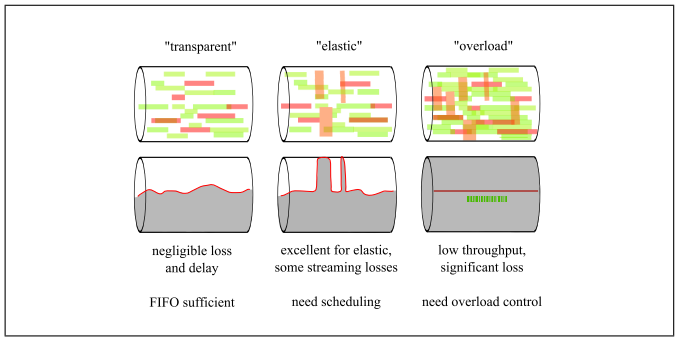


Fig. 1. Three link operating regimes: "transparent", "elastic" and "overload"

### C. Link utilization regimes

From the above discussion, we identify three basic link operating regimes. These are depicted in Figure 1. In the upper sketches flows are represented as boxes of height equal to their average exogenous rate and horizontal position within the link determined by the starting time and duration. The vertical position of the box within the link is chosen at random. The lower figure represents the evolution in time of the overall rate of flows in progress.

The left hand case depicts a “transparent” regime where all flows are non-bottlenecked and the sum of flow rates is always less than link capacity. Packet loss and delay are then extremely small. This is the regime of most links in the current Internet due to over provisioning and the fact that most users have a relatively low access rate.

The “elastic” regime in the middle occurs when some bottlenecked flows are added to a non-bottlenecked background while the overall offered load remains well within link capacity. These bottlenecked flows can only be generated by users with particularly high access rates. Whenever one or several bottlenecked flows are in progress, we can expect some packet level congestion leading to delays and some loss. The delay may be significant if large buffers are used, particularly for streaming flows. Some form of scheduling may be considered necessary to preserve quality of service in this regime.

The right hand case corresponds to a “overload” regime where the offered traffic exceeds the available link capacity. Congestion control algorithms like those of TCP are inadequate to deal with this kind of saturation and alternative means must be employed to avoid saturation, e.g., [17], [18].

## IV. Buffer requirements under realistic traffic assumptions

In this section we evaluate the impact of buffer size on the performance of a realistic mix of bottlenecked and non-bottlenecked flows, as discussed in the previous section. We assume congestion control is performed using TCP New Reno and buffer management is FIFO drop tail.

### A. Buffer sizing in the transparent regime

The transparent regime is characterized by the fact that the sum of rates of a large set of non-bottlenecked flows is less than available link capacity (with high probability). The link buffer is necessary to absorb momentary bursts of packet arrivals from a fraction of the large number of independent connections. From well-known theoretical limit results, and as confirmed by measurements [19], the packet arrival process is very nearly Poisson.

If we additionally (conservatively) assume exponential packet sizes, the relation between buffer size and packet loss probability is approximately geometric:

$$Pr[\text{loss} \mid \text{buffer size} = n] \approx \rho^n$$

where $\rho$ is the link load. If, for example, link load is less than 90%, a buffer of 100 packets maintains a loss rate better than 1 in 30000. A 20 packet buffer can keep the loss rate better than 0.01 if load is less than 71%. Delay is extremely small on high speed links: at 2.5Gbps, assuming average packet size 500 bytes, a 100 packet buffer empties in 160 microseconds.

### B. Buffer sizing in the elastic regime

In the elastic regime a large set of non-bottlenecked flows shares the link with a small number of bottlenecked flows. We illustrate the behaviour of this regime by means of ns2 simulations. The simulation scenario is depicted in Figure 2. We initially simulated a background traffic made up of a process of finite size TCP flows each with a maximum rate of 1Mbps. However, to facilitate the evaluation of a range of configurations, we replaced the flow level process (resulting in a variable rate background traffic) by a Poisson packet process of the same intensity. We first verified that the principal comparative results discussed below were not changed by this simplification.

The link capacity is 50Mbps and the background traffic accounts for half of this. We simulated 1, 2 and 4 permanent bottlenecked New Reno TCP flows and observed the resulting performance for varying buffer size. We only describe a small sample of the results obtained. Further experiments are reported in the next section.

Figure 3 illustrates the impact of buffer size on the evolution of the TCP window size *cwnd* (bottom graphs)

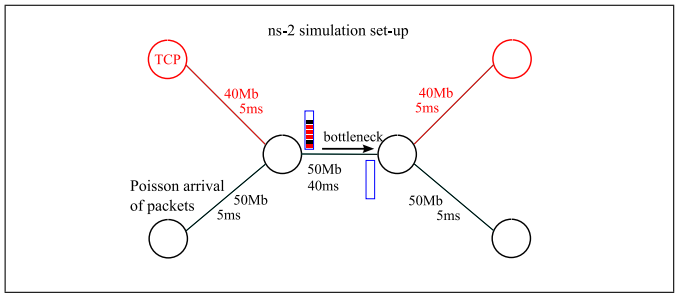


Fig. 2. Simulation scenario

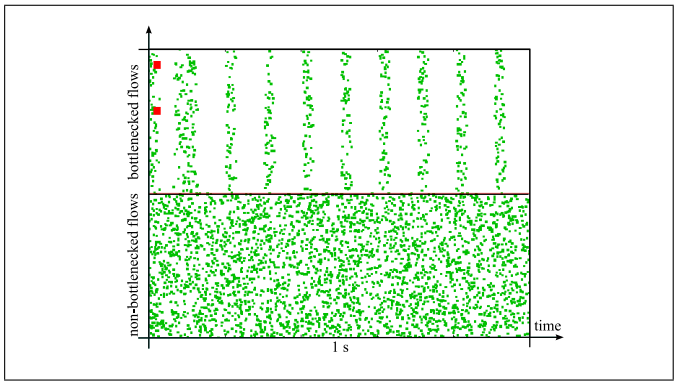


Fig. 4. Representation of TCP New Reno packets arrivals on a link with 20 packet buffers. Each dot is a packet; its vertical position is random; squares on the left are lost packets.

and of link utilization averaged over an RTT (top graphs) for a single bottlenecked flow. We show results for three buffer sizes: 20 packets as recommended in [3], 625 packets corresponding to the bandwidth delay product, and an intermediate value of 100 packets. Results for the 20 packet buffer show that this buffer size is clearly inadequate for the considered (realistic) traffic mix. The bottlenecked flow only manages to acquire around 40% of the residual bandwidth. The large buffer enables 100% utilization, as expected. The intermediate size of 100 packets appears as a reasonable compromise, realizing reasonable throughput while ensuring low latency for any streaming traffic included in the set of background non-bottlenecked flows.

Loss of utilization with a small buffer is not unexpected but the magnitude of the loss is at first surprising. A single bottlenecked TCP flow on an empty link with a buffer of only 1 packet should acquire a throughput of 75% of the link capacity and we would expect this to grow to around 80% with a 20 packet buffer. The presence of background traffic reduces the realized throughput to only 40% of the residual capacity.

The reason is that the competing background traffic combines with the bursts of the bottlenecked TCP flow to momentarily saturate the link. The emission of packets is depicted in the scatter plot of Figure 4. This figure shows the background packets as dots in the lower half of the graph and bottlenecked flow packets in the upper half. The stripes reflect the TCP window mechanism which tends to emit packets in bursts.

During each burst packets arrive at link rate but with some randomness and this tends to fill the buffer. If the buffer is not big enough the randomness of the arrival process will lead to loss. For the 20 packet buffer this event happens quite often, even for relatively small window sizes. In the figure, packet loss occurs at the end of the first depicted burst. This is recognized one RTT later leading to a halving of the current window size. Had there been no background traffic, loss would only have occurred when the window exceeds the bandwidth delay product.

Figure 5 illustrates the impact of increasing the number of bottlenecked flows. We observe that even with only two flows, there is little synchronized loss and utilization improves as the number of flows increases. The evolution of *cwnd* for each flow shows that bandwidth sharing is approximately fair.

On the strength of these limited results it seems clear that a recommendation to use buffers of only 20 packets is not justified when taking account of a realistic mix of bottlenecked and non-bottlenecked flows. While some loss of utilization is acceptable, it would be preferable to increase the buffer size in the considered case to around 100 packets. The requirement is to have enough buffer space for the random (Poisson-like) arrivals of packets from non-bottlenecked flows together with a further allowance to enable the full development of the window size of the bottlenecked flows. The case of a single bottlenecked flow is a worst case in this respect and, according to the processor sharing model alluded to above, is a quite frequent occurrence.

## V. Enhanced Bandwidth Sharing

In considering buffer requirements in the future high speed network, it is appropriate to consider other evolutions that can affect the performance of congestion control. In this section we consider the impact of new TCP versions and the possible use of per-flow fair scheduling.

### A. New high speed protocols

We confine the present evaluation to the use of HSTCP as defined in RFC 3649 [8]. The performance of other protocols will be considered in future work.

Figure 6 shows results for three configurations where bottlenecked connections use HSTCP instead of New Reno. The left hand plots demonstrate that a single HSTCP connection uses bandwidth more efficiently than Reno when the buffer size is considerably less than the

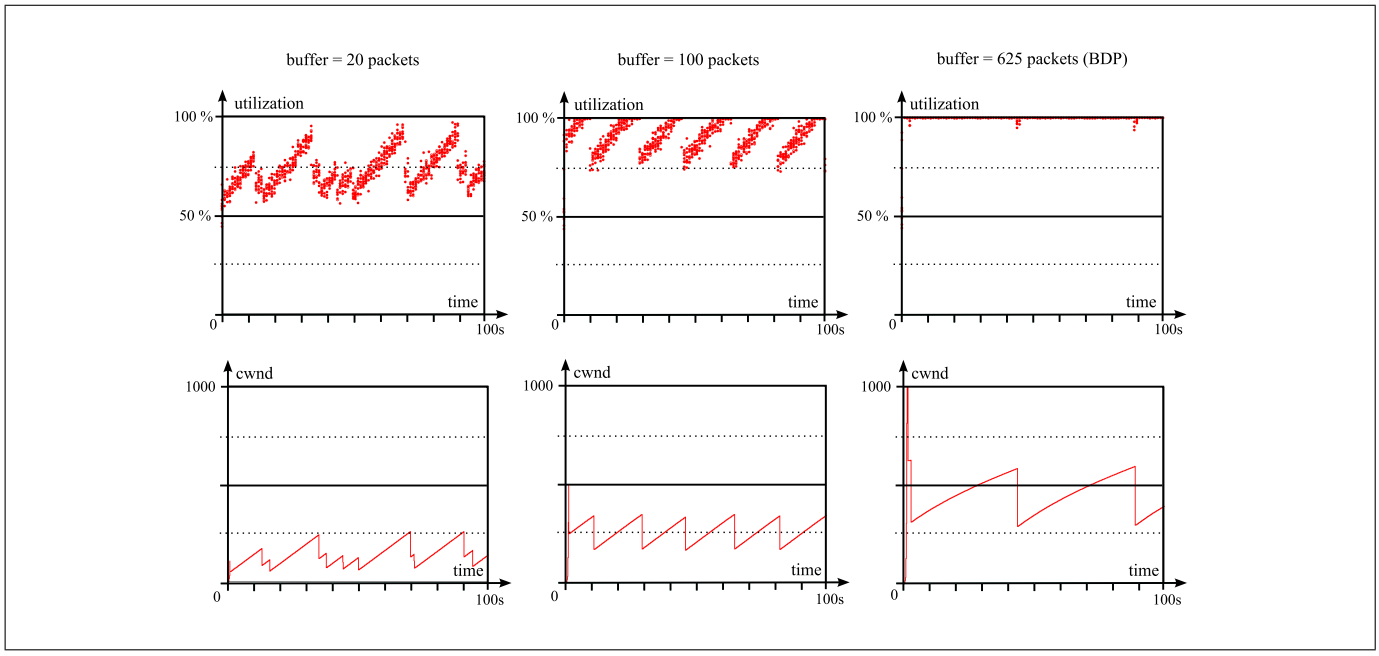


Fig. 3. From left to right, utilization and cwnd size as a function of time for a range of buffer sizes : 20, 100 and 625 packets (bandwith delay product). All flows use TCP New Reno and the scheduling discipline is FIFO.

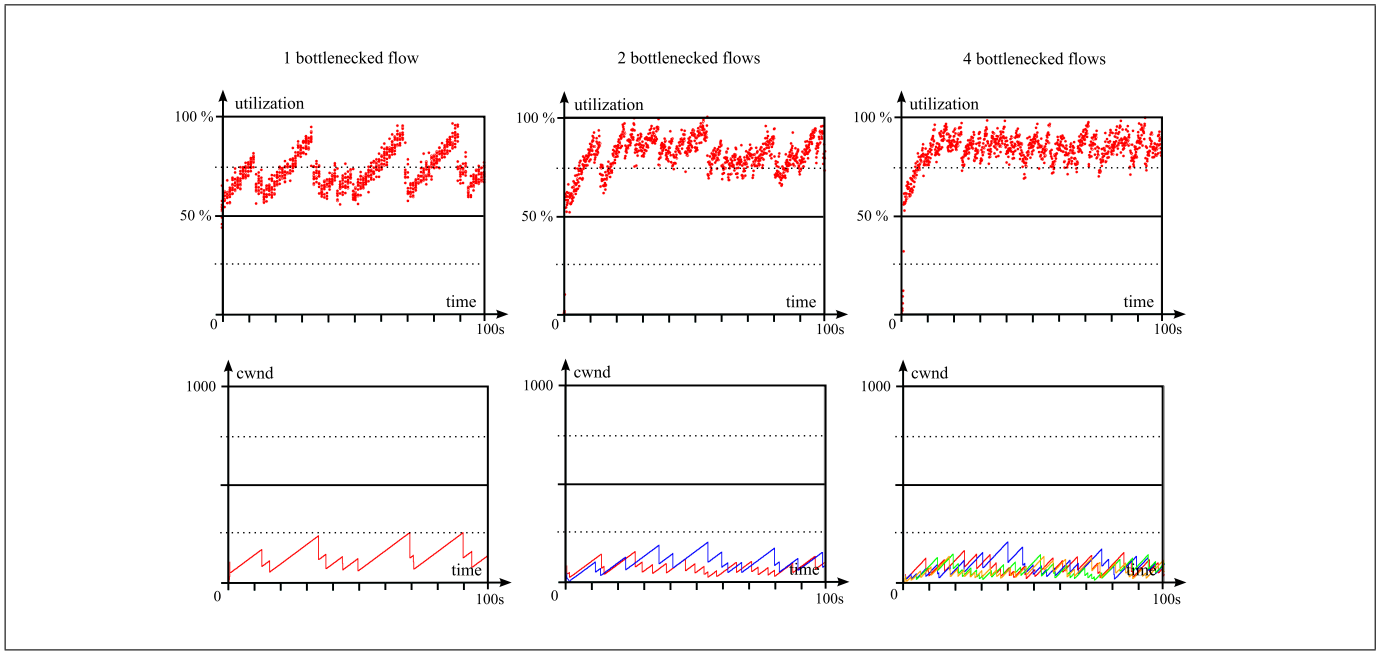


Fig. 5. From left to right, utilization and cwnd size as a function of time for 1, 2 and 4 bottlenecked flows with a 20 packet buffer. All flows use TCP New Reno and the scheduling discipline is FIFO.

bandwidth delay product. This is an encouraging result illustrating that the evolution of TCP can improve the efficiency of small buffers.

In the middle, an HSTCP flow shares the 25Mbps residual bandwidth with a Reno flow using a large buffer. The depicted behaviour of the respective flow windows *cwnd* is typical: the Reno connection is unable to compete fairly with the more aggressive HSTCP connection.

The last plots on the right relate to 4 bottlenecked HSTCP flows. We observe quite chaotic behaviour of the flow *cwnd*s as one flow after the other gains a temporary ascendancy. As soon as one flow has a large value of *cwnd*, it tends to act more aggressively and preserves its relative advantage until it is eventually superseded by one of the other flows. This kind of behaviour has previously been observed by Li et al. [11].

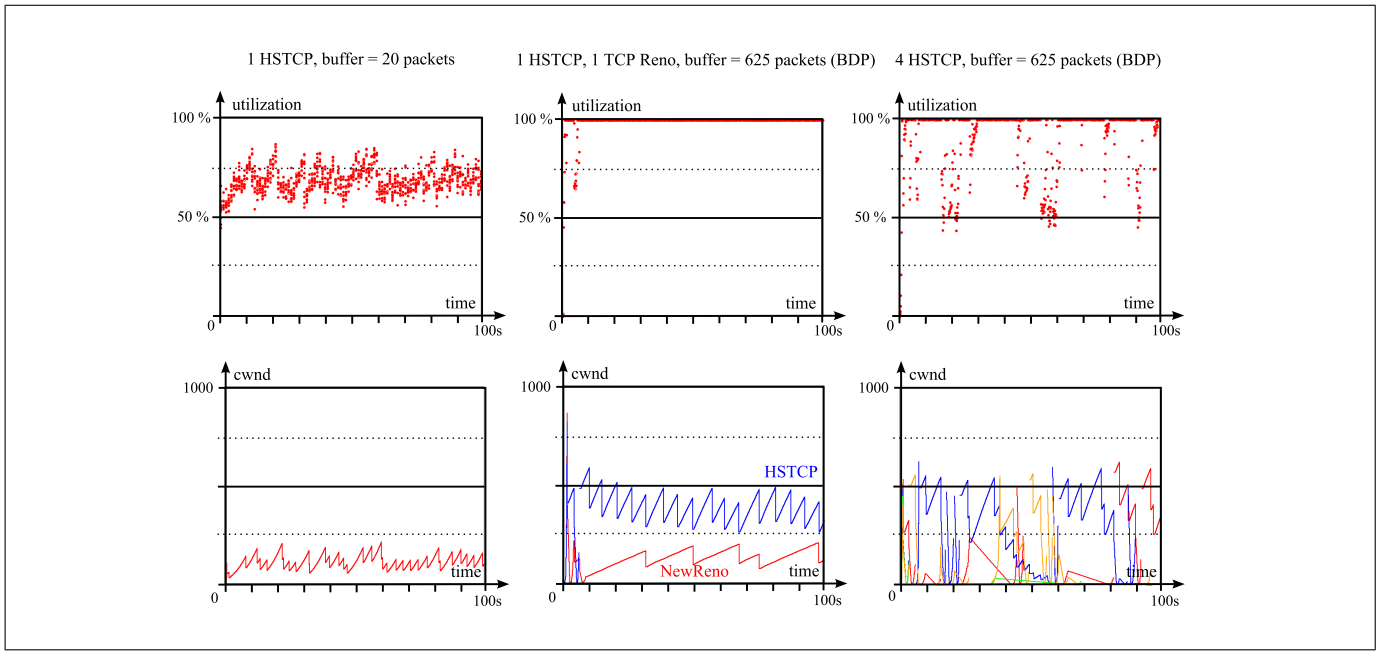


Fig. 6. From left to right, utilization and cwnd size as a function of time for: - 1 HSTCP flow with a 20 buffer packet, - 1 HSTCP flow and 1 TCP New Reno flow with a 625 packet buffer (bandwith delay product), - 4 HSTCP flows with a 625 packet buffer. The scheduling discipline is FIFO.

### *B. Using fair queuing*

We have implemented per-flow fair queuing in ns2 with a drop from longest queue discard policy. The latter has been shown to largely outperform RED as an active queue management mechanism [20]. In the present case the drop policy is such that non-bottlenecked flows remain loss free. Their packets also have very low latency[1], a highly desirable feature for any streaming flows included in the background traffic.

Figure 7 illustrates the evolution of utilization and *cwnd* for the three cases previously illustrated under FIFO in Figure 6. The plot on the left corresponds to a single TCP Reno connection sharing the link with background traffic using a 20 packet buffer. Comparison with the left hand plot of Figure 6 shows that fair queuing barely alleviates the loss in performance due to a very small buffer (it does, however, avoid loss for the non-bottlenecked flows).

The middle plot should be compared to the middle plot of Figure 6. It is clear that fair queuing considerably reduces the bias in favour of HSTCP. The HSTCP flow does, however, gain slightly more throughput than the Reno flow due to its more aggressive behaviour.

Finally the rightmost plot confirms that fair queuing effectively counters the chaotic behaviour illustrated in the corresponding plot of Figure 6. Fair queuing also considerably attenuates the impact on existing flows of newly arriving flows whose aggressive slow-start behaviour can otherwise lead to quite severe loss events.

Two supplementary benefits of fair queuing are desynchronized loss events due to flow isolation and the fact that packet emissions are paced at the current fair rate. The latter effect means acknowledgements are paced leading to less bursty input in the following window cycle. Both phenomena appear to improve bandwidth sharing performance somewhat, notably when the buffer size is small.

[1]This is can be reduced still further using the priority mechanisms described in [5], [21].

## VI. Conclusions

Several authors have recently pointed to the need to reduce Internet buffer sizes by several orders of magnitude. This is necessary for technological reasons but is also claimed to have a positive impact on performance. In the present paper we have contested the relevance of the traffic model assumed in the published evaluations where the link is shared by a very large number of permanent bottlenecked TCP flows.

In fact, it is necessary to account for the random nature of traffic at flow level. At normal link loads, the large number of flows that are observed to share link bandwidth are necessarily limited in rate by constraints that are exogenous to the considered link. The number of bottlenecked flows, whose rate is determined by the considered link through the action of the congestion control algorithms, is typically very small (i.e., 0, 1 or 2, . . . flows) with high probability.

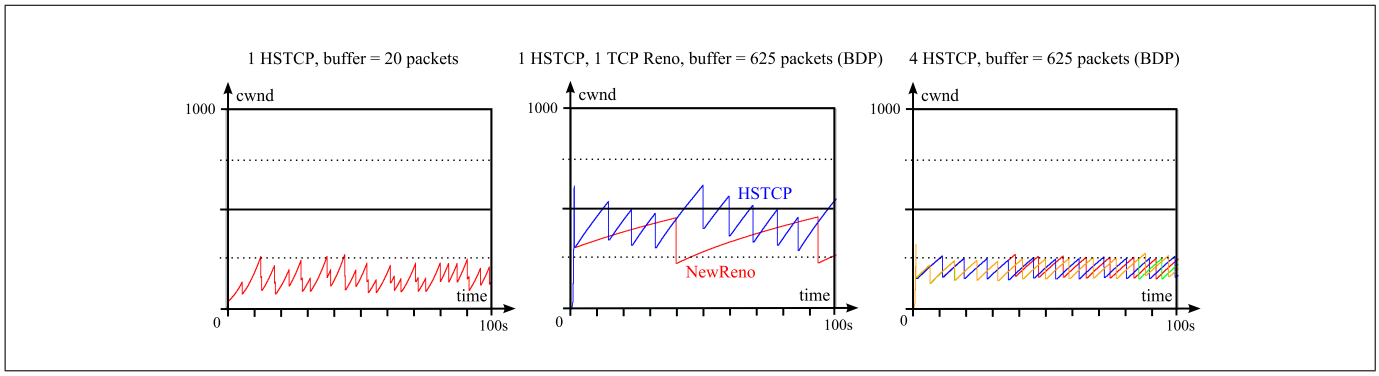


Fig. 7. From left to right, utilization and cwnd size as a function of time for: - 1 TCP New Reno flow with a 20 buffer packet, - 1 HSTCP flow and 1 TCP New Reno flow with a 625 packet buffer (bandwith delay product), - 4 HSTCP flows with a 625 packet buffer. The scheduling discipline is Fair Queueing.

We have conducted a preliminary performance evaluation by means of simulation. The results show that the previously recommended reduction in buffer capacity to 20 packets is rather too drastic. For the considered configuration, a buffer of 100 packets seems a more appropriate choice allowing reasonable throughput while maintaining low packet latency. The buffer must be sized to accommodate near coincident packet arrivals from independent non-bottlenecked flows (this requirement can be evaluated using a simple queuing model) while preserving sufficient free space to allow a single flow to efficiently utilize the residual link capacity. For a link of arbitrary capacity, a buffer size of one or two hundred packets appears as a reasonable choice. We are currently developing a mathematical model in order to more precisely determine required buffer size as a function of bottlenecked and non-bottlenecked flow loads.

Use of new high speed TCP versions can significantly reduce the required buffer size. A smaller buffer is sufficient to maintain high throughput when the window additive increase rate is higher and the multiplicative decrease rate is smaller than legacy TCP, as in the proposed HSTCP congestion avoidance algorithm. It is clear, however, that the fairness and convergence properties of these new protocols are less than perfect.

Bandwidth sharing would be further enhanced by the implementation of per-flow fair queuing in router queues. Fair queuing enables the coexistence of legacy and high speed TCP versions by avoiding the above mentioned fairness and convergence problems. Fair queuing also ensures low packet latency for non-bottlenecked flows, including any streaming flows within the traffic mix. Finally, we recall that under the realistic traffic conditions discussed here, fair queuing is both scalable and feasible since the number of flows with packets to be scheduled at any instant is only measured in hundreds for any link capacity.

## REFERENCES


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

[2] C. Villamizar and C. Song, "High performance TCP in ANSNET," *Computer Communications Review, V. 24 N. 5, October 1994, pp. 45-60*.

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

[4] M. Enachescu, Y. Ganjali, A. Goel, N. McKeown, and T. Roughgarden, "Part III: routers with very small buffers," *ACM SIGCOMM Computer Communication Review, v.35 n.3*, July 2005.

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

[6] 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.

[7] D. Wischik and N. McKeown, "Part I: buffer sizes for core router," *ACM SIGCOMM Computer Communication Review, v.35 n.3*, July 2005.

[8] S. Floyd, "Highspeed TCP for large congestion windows," *RFC 3649, Experimental, December 2003*.

[9] C. Jin, D.X. Wei, and S.H. Low, "Fast TCP: Motivation, architecture, algorithms, performance," *Proceedings of IEEE Infocom, Hong Kong*, March 2004.

[10] T. Kelly, "Scalable tcp : Improving performance in highspeed wide area networks," .

[11] Y-T. Li, D. Leith, and R.N. Shorten, "Experimental evaluation of TCP protocols for high-speed networks," *Hamilton Institute, NUI Maynooth*, 2005.

[12] G. Raina, D. Towsley, and D. Wischik, "Part II: control theory for buffer sizing," *ACM SIGCOMM Computer Communication Review, v.35 n.3*, July 2005.

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

[14] S. Keshav, "Congestion control in computer networks," *PhD Thesis, published as UC Berkeley TR-654*, September 1991.

[15] Y. Zhang, L. Breslau, V. Paxson, and S. Shenker, "The characteristics and origins of internet flow rates," *Proceedings of ACM SIGCOMM, Aug. 2002, Pittsburgh, PA.*

[16] S. Ben Fredj, T. Bonald, A. Proutière, G. Régnié, and J.W. Roberts, "Statistical bandwidth sharing: a study of congestion at flow level," *SIGCOMM 2001, San Diego, CA, USA*, August 2001.

[17] S. Oueslati and J. Roberts, "A new direction for quality of service: Flow aware networking," *NGI 2005, Rome*, April 18-20, 2005.

[18] S. Kandula, D. Katabi, B. Davie, and A. Charny, "Walking the tightrope: Responsive yet stable traffic engineering," *ACM SIGCOMM 2005*.

[19] J. Cao, W. Cleveland, D. Lin, and D. Sun, "Internet traffic tends toward Poisson and independent as the load increases," 2002.

[20] B. Suter, T. V. Lakshman, D. Stiliadis, and A. Choudhury, "Design considerations for supporting TCP with per-flow queuing," *Proc. of INFOCOMM '98*, Apr. 1998.

[21] J. Roberts A. Kortebi, S. Oueslati, "Implicit service differentiation using deficit round robin," *Proceedings of ITC 19, Beijing*, August 2005.