Part I: Buffer Sizes for Core Routers

Damon Wischik*
Computer Science, UCL
D.Wischik@cs.ucl.ac.uk

Nick McKeown†
Computer Science, Stanford
nickm@stanford.edu

ABSTRACT

In this article we describe recent work on buffer sizing for core Internet routers. This work suggests that the widely-used rule of thumb leads to buffers which are much larger than they need to be. For example, the buffer in a backbone router could be reduced from 1,000,000 packets to 10,000 without loss in performance. It could be reduced even further, perhaps to 10–20 packets, at the cost of a small amount of bandwidth utilization. This tradeoff is worth considering, for example for a possible future all-optical router.

Categories and Subject Descriptors

C.2 [**Internetworking**]: Routers; C.4 [**Performance of systems**]: Modeling techniques

General Terms

Design, Performance, Theory

Keywords

Internet router, buffer size, bandwidth delay product, TCP, synchronization, multiplexing

1. INTRODUCTION

The buffer in an Internet router has several roles. It accomodates transient bursts in traffic, without having to drop packets. It keeps a reserve of packets, so that the link doesn't go idle. It also introduces queueing delay and jitter. Arguably, router buffers are the single biggest contributor to uncertainty in the Internet. Given their significance, we might reasonably expect buffer sizing to be well understood, based on well-grounded theory and supported by extensive experiments. This is not yet so.

At the moment, router manufacturers typically use a rule of thumb: routers should provide at least one round trip time's worth of buffering, often taken to be around 250ms. In this article and the two that follow [5, 11], theory and simulations illustrate why buffers could be *much much smaller*, perhaps as small as a few μs, without making changes to TCP. We hope this encourages network operators to experiment with small buffers. For more detail on many of the issues raised here see [2].

Hardware considerations.

For a 10Gb/s router linecard to provide 250ms of buffering, it needs a buffer of 250ms $\times$ 10Gb/s = 2.5Gb, which is roughly 310MBytes or 210 thousand packets (assuming a packet size of 1pkt = 1500Bytes). This is modest in comparison to the amount of memory on a desktop computer. Why do we care about buffer size? Why not just overprovision?

There are several hardware reasons for wanting smaller buffers. First, the memory in a router linecard has to be as fast as the line rate—much faster than the speed of memory in desktop computers. This requirement complicates the design of high-speed routers, leading to higher power consumption, more board space, and lower density. Second, while switching speeds double every 18 months according to Moore's Law, memory access speeds double only every 10 years. Therefore memory requirements will increasingly become a limiting aspect of router design. Third, all-optical routers will perhaps be able to buffer 100 packets or so (using e.g. fiber delay lines). If this amount of buffering is sufficient to get good performance, an all-optical packet switched core network is feasible.

Traffic considerations.

The most obvious way to measure the impact of buffer size is by seeing how it affects bandwidth utilization. The naive view is that the larger the buffer, the higher the utilization. There are two problems with this:

First, utilization is not necessarily the right metric. It's a useful metric when capacity is scarce/expensive and the network operator wants to be sure that all the capacity is able to be used. But core networks today are run significantly below 100% utilization, and the need for high utilization is not nearly as strong as it once was. Other quality-of-service metrics like latency and jitter must be considered, and these will definitely be improved with smaller buffers. Furthermore, small buffers may enable cheaper/faster routers, so even if utilization is lower throughput may still be higher.

Second, it isn't always the case that larger buffers give higher utilization. This paradox arises because of synchronization. We will argue here that if TCP flows are synchronized then we need a large buffer to get high utilization, and that if they are desynchronized then we can get away with much smaller buffers thanks to statistical multiplexing. However, synchronization itself depends on buffer size,

*Research supported by a Royal Society university research fellowship, and DARPA Buffer Sizing Grant no. W911NF-05-1-0254.
†Research supported under DARPA/MTO DOD-N award no. W911NF-04-0001/KK4118 (LASOR PROJECT) and the Buffer Sizing Grant no. W911NF-05-1-0224.

because of the interplay between TCP's control loop and queueing dynamics at the router. Reference [11] argues that large buffers tend to *induce* synchronization, which means that large buffers can actually hurt utilization.

In most of this article we will be concerned with TCP flows. Long-lived TCP flows account for the bulk of packets in the Internet. Therefore TCP dynamics govern the behaviour of the queue at an Internet router, and hence the quality of service experienced by non-TCP flows. If buffer sizes are chosen so that TCP is well-behaved and queueing delays are small, then other traffic (UDP flows, short-lived TCP flows) will be happy. If in the future TCP ceases to dominate, we hope that its replacement is engineered taking account of the sort of theory we describe below.

2. WHAT ARE BUFFERS FOR?

Queueing theorists are used to thinking of sizing buffers so as to prevent them from overflowing and losing packets. But TCP's "sawtooth" congestion control algorithm is designed to fill any buffer, and deliberately causes occasional loss to provide feedback to the sender. No matter how big we make the buffers at a bottleneck link, TCP will cause the buffer to overflow.

Rule of thumb for synchronized flows.

The rule of thumb comes from a desire to keep a congested link as busy as possible, so as to maximize the throughput. The basic idea is that when a router has packets in its buffer, its outgoing link can be kept busy. Consider a single long-lived TCP flow which passes though a single router, and suppose it has settled into congestion avoidance. When TCP detects a packet drop and responds by halving its window, the buffer will drain, and if there aren't enough packets in the buffer then the link will idle. It turns out that the buffer size we need to prevent this is $C \times RTT$, where C is the link speed and RTT is the round trip time (RTT) of the flow (see [2] for references).

When there are several TCP flows with identical RTTs and they are all synchronized (i.e. they all halve their windows at the same time) the buffer requirement is exactly the same. This idea has recently been extended [4] to the case of synchronized flows with non-identical RTTs. They find that the buffer size needed is $C\widetilde{RTT}$, where $\widetilde{RTT}$ is the harmonic mean of the RTTs, i.e. if there are N flows with RTTs $RTT_1, \ldots, RTT_N$ then

$$1/\widetilde{RTT} = \frac{1}{N}\sum_{i=1}^{N} 1/RTT_i.$$

By convexity, this is always less than or equal to the arithmetic mean of the RTTs; it gives more weight to small RTTs.

Desynchronized flows.

When there are many desychronized flows, the buffer requirement is much smaller. Here is a simple model. Suppose that there are N flows with identical RTTs, with mean window size wnd. Treat each flow as following a periodic TCP sawtooth with average window size wnd, which means that its window varies between $2wnd/3$ and $4wnd/3$. Suppose that the flows are unsynchronized, i.e. that the sawtooths have uniformly distributed phase. Then, by the central limit theorem, the sum of window sizes can be approximated by a normal distribution:

$$W \sim \text{Normal}(Nwnd, N\sigma_{wnd}^2)$$

where σ_{wnd} is the standard deviation of the window size of a single flow, namely

$$\sigma_{wnd} = \sqrt{\frac{(4wnd/3 - 2wnd/3)^2}{12}} = \alpha wnd,$$

where $\alpha \approx 0.192$. Suppose that TCP achieves the objective of matching the average total transmission rate to link capacity. The total sending rate is $Nwnd/RTT$, so $Nwnd = CRTT$. This lets us express the standard deviation of W in terms of N and $CRTT$: it is

$$\text{sd}(W) = \sqrt{N\sigma_{wnd}^2} = \alpha CRTT/\sqrt{N}.$$

Therefore a 99.9% confidence interval for W is roughly

$$CRTT \pm 3.3\alpha\frac{CRTT}{\sqrt{N}}.$$

Now, $CRTT$ is the number of packets that can be accomodated in the pipe; the outstanding $3.3\alpha CRTT/\sqrt{N}$ packets need to be buffered. Therefore a buffer of size $3.3\alpha CRTT/\sqrt{N}$ is sufficient to cope with the variability of the TCP sources. ($3.3\alpha \approx 0.63$.) This argument was proposed by [2]. Recent experiments on operational networks appear to validate it; they will be described in a future paper.

For a 10Gb/s linecard with $RTT = 250$ms, carrying 10,000 flows, the rule of thumb recommends $CRTT = 2.5$Gb of buffering, whereas this calculation recommends 16Mb.

Partially synchronized flows.

When flows are partially synchronized, the buffer requirements are intermediate [4].

What large buffers really do.

When flows are desynchronized, what is the effect of large buffers? A single TCP flow, which experiences packet drop probability p, attains a throughput of roughly

$$x = \frac{h}{RTT\sqrt{p}} \text{ pkt/s} \quad (1)$$

where $h = 0.87$ and RTT is the average round trip time, comprising propagation delay and queueing delay [9].

Now consider a link shared by several flows, where flow r sends packets at rate x_r. Let $y = \sum x_r$ be the total sending rate. Suppose the link has service rate C and that the buffer is large enough to keep the link fully utilized. Clearly $y > C$ since otherwise the link could not remain fully utilized, and the queue drops only what it cannot serve, so the drop probability p satisfies

$$(1-p)y = C. \quad (2)$$

The queue has more arrivals than it can serve, and so the buffer will fill up and stay full or nearly full, adding queueing delay to every flow.

Let flow r have propagation time PT_r and let the common queueing delay be QT. Since the buffer stays full or nearly full, this is B/C where B is the buffer size. From (1) and (2), the drop probability p is the solution to

$$\sum_r \frac{h(1-p)}{(PT_r + QT)\sqrt{p}} = C.$$

If B is very big then every flow will have a larger RTT, so its throughput x will be smaller, so the drop probability p will be smaller.

In other words, the buffer's function is not only to keep the link busy. It also adds queueing delay, which keeps the transmission rates of the flows low, which in turn keeps the loss probability low. *It acts as an artificial dampener on demand.* Is this a useful function for the buffer in a router? We believe it is very unhelpful[1]. There are two ways that a router can signal that it is overloaded: by increasing the queueing delay, or by dropping packets; either signal will make TCP back off. Both signals are effective, in that they have the effect of keeping the link fully utilized. But the former signal causes irreparable damage to e.g. real-time flows, whereas flows can recover from the latter signal by retransmitting the dropped packet or by using forward error correction. In order that the Internet should be able to evolve to carry new services, we emphatically believe that packet loss is a better signal than delay. Furthermore, the hardware resources needed for large delays/buffers would be better spent in building faster routers. (Others however argue [4] that it is better to keep loss probability small by increasing delay.)

3. THE CASE FOR SMALL BUFFERS

Let us now give up the objective of keeping the link fully utilized, and see what effect this has on buffer size. Recall the normal approximation to the sum of window sizes, in the case of N desynchronized flows each with average window size wnd:

$$W \sim \text{Normal}(N\mathit{wnd}, N\alpha^2\mathit{wnd}^2).$$

Suppose we could arrange things so that $N\mathit{wnd} = \gamma C\mathit{RTT}$ for say $\gamma = 95\%$. Then a 99.9% confidence interval for W is roughly

$$\gamma C\mathit{RTT} \pm 3.3\alpha\gamma\frac{C\mathit{RTT}}{\sqrt{N}}.$$

If the upper bound for this confidence interval is less than $C\mathit{RTT} + B$ then the queue size (that is, the total number of packets in flight W, minus the number in the pipe $C\mathit{RTT}$), should rarely exceed B. A sufficient condition is

$$N > 3.3\alpha\frac{\gamma}{1-\gamma} \approx 12.0.$$

This indicates that by sacrificing 5% utilization, and with just a small amount of statistical multiplexing, we can nearly eliminate the need for buffers altogether! (But see the analysis in the next section.)

Our argument rests on the supposition that the TCP flows are desynchronized. The analysis of synchronization in [11] relies on more sophisticated theory which we now describe.

[1]Current TCP does benefit from large queueing delays in one case: very small propagation delays. If the round trip time is very small then the TCP window is likely to be very small, which can lead to problems with timeouts. Large buffers add queueing delay, making the round trip time larger, which alleviates the problems. Our concern in this article is with core routers, which by assumption carry flows with large propagation delays, so this is not an issue. For small buffers to work in campus routers, TCP would need to be modified to work better with small windows, e.g. as in [8].

4. ANALYSIS OF NEARLY-BUFFERLESS MULTIPLEXERS

Queueing delay.

Our first remark is trivial. In a queue with service rate C and buffer size B, the maximum queueing delay is B/C. If buffer size is kept fixed as line rates increase, then the maximum queueing delay tends to zero. Therefore jitter tends to zero. Further consequences of small delays are discussed in [7].

Poisson modeling.

In the last section we said that the total number of packets in flight W should have a normal distribution, and suggested that the queue size distribution should be $(W - C\mathit{RTT})^+$. This doesn't take into account the fact that some packets are likely to be spaced close together and others spaced far apart, just because of inherent randomness. When there is a cluster of closely-spaced packets, the queue will build up transiently. *Pace* [10], we will use a Poisson model for this.

The evidence in favour of Poisson traffic models is threefold [3]. First, theory says that as the number of independent traffic flows increases, the aggregate arrival process converges to Poisson over short timescales. Second, there is empirical evidence that this holds for Internet traffic. Third, theory says that for large multiplexers with small buffers, overflow is most likely to happen over very short timescales. (This is basically because the length of a typical busy period is inversely proportional to the speed of a queue.) The case for Poisson traffic models is argued in more detail in [12], taking explicit account of the closed-loop feedback of TCP's congestion control. In summary, the loss probability in a queue with total arrival rate y should be approximated by the loss probability for Poisson arrivals of rate y.

To be concrete: in a queue running at 95% utilization, with a buffer of 50 packets, the loss probability is approximately 3.3×10^{-4}.

The key insight is that, if the buffer is small and the line rate is large, then the queue size fluctuations are very fast—so fast that it is impossible to control the queue *length*. Instead, TCP acts to control the *distribution* of queue length.

Fixed point approximation.

We have given several equations to describe the system. For a flow which experiences drop probability p the throughput is roughly

$$x = \frac{h}{\mathit{RTT}\sqrt{p}}.$$

In a network with small buffers, RTT is essentially just the propagation delay. Let $L_B(y, C)$ be the drop probability in a queue with Poisson arrivals of rate y, service rate C and buffer size B. Then the drop probability at a link is

$$p = L_B(y, C)$$

where y is the total load at that link, i.e. the sum of the throughputs of flows that arrive at that link, thinned by the drop probability they experience at upstream links. This gives us a set of simultaneous equations to solve. The solution gives us the throughputs x and drop probabilities p.

This is called the fixed-point approach. For references see [6]. It is a refinement of our simple assumption in Section 2, namely that TCP achieves the objective of matching total transmission rate to link capacity.

Modeling burstiness.

Simulations suggest that packet spacing makes a tremendous difference [5]. If packets are spaced, either by an upstream link with limited speed or by an implementation of TCP which implements rate-pacing, then the Poisson arrival model is good. If not, then TCP tends to send its entire window in a single burst, and so a more appropriate model is batch-Poisson arrivals, where the batch size distribution reflects the TCP window size distribution. A rough heuristic is that, if the mean batch size is wnd, the batch arrival rate is y, the buffer is B, and the service rate is C, then the probability that at least one packet in a burst is dropped is roughly $p = L_{B/wnd}(y, C/wnd)$.

Synchronization.

Our arguments for small buffers suppose that there is no synchronization. The theory which enables us to predict synchronization is left for [11].

5. SIMULATION TIPS

When we scale up the number of flows N, it's a good idea to scale up the service rate in proportion. If C is the total service rate, then the notional 'ideal' window size for each flow is roughly $wnd = CRTT/N$. If C were kept fixed then wnd would change, which would change the dynamics of TCP. For example, if wnd is too small, then timeouts are more likely. We found it most convenient to study the effects of multiplexing (N) and of window size (wnd) separately.

It's important to pay close attention to traffic burstiness. Look close-up at plots of queue size and of packet arrivals from a single flow, and see how much the queue varies in between packets arrivals. This makes a big difference to the drop probability, to synchronization, and to buffer requirements.

6. CONCLUSION

We conclude by answering some of the arguments in favour of large buffers.

Large buffers are needed to get 100% utilization. Not only is this not true, it is not necessarily a worthwhile goal. By sacrificing a little utilization, we reduce latency and jitter. Maybe routers with small buffers could run faster, so that even at 95% utilization they have higher throughput.

With small buffers, bursty flows and flows in slow-start will be unfairly penalized because their packets arrive back-to-back and are more likely to be dropped. TCP sources can reduce their burstiness by using rate-pacing. So it is a philosophical issue, whether to regard this discrimination as unfairness or as just desserts. [1] reports synchronization problems with rate-pacing, because "feedback is delayed until the network is saturated, making it difficult for senders to 'avoid' overwhelming the network"; we recommend that buffers should be small because then feedback is not delayed and synchronization is avoided.

If the flows become synchronized, buffers need to be sized according to the rule of thumb. [11] argues that large buffers *induce* synchronization. Small buffers are the solution, not the problem.

Large buffers increase round trip time at a congested link, which increases queueing delay, which reduces load and drop probability. We have argued that this is a bad tradeoff. It is better to have lower latency and higher drop probability, since applications can protect themselves against packet drops but lost time can never be recaptured.

Acknowledgements.

We are grateful for helpful discussions with M. Enachesu, Y. Ganjali, P. Glynn, A. Goel, C.V. Hollot, F.P. Kelly, Y. Liu, G. Raina, T. Roughgarden, D. Towsley.

7. REFERENCES

[1] A. Aggarwal, S. Savage, and T. Anderson. Understanding the performance of TCP pacing. In *IEEE Infocom*, 2000.

[2] G. Appenzeller, I. Keslassy, and N. McKeown. Sizing router buffers. In *SIGCOMM*, 2004. Extended version available as Stanford HPNG Technical Report TR04-HPNG-060800.

[3] J. Cao and K. Ramanan. A Poisson limit for buffer overflow probabilities. In *IEEE Infocom*, 2002.

[4] A. Dhamdhere, H. Jiang, and C. Dovrolis. Buffer sizing for congested Internet links. In *IEEE Infocom*, 2005.

[5] M. Enachescu, Y. Ganjali, A. Goel, N. McKeown, and T. Roughgarden. Part III: Routers with very small buffers. *ACM/SIGCOMM CCR*, 2005.

[6] V. Firoiu, J.-Y. L. Boudec, D. Towsley, and Z.-L. Zhang. Theories and models for Internet quality of service. *Proceedings of the IEEE*, 2002.

[7] F. P. Kelly. Models for a self-managed Internet. *Phil.Trans. of the Royal Society*, A358, 2000.

[8] T. Kelly. On engineering a stable and scalable TCP variant. Technical Report CUED/F-INFENG/TR.435, Cambridge University Engineering Department, 2002.

[9] M. Mathis, J. Semke, J. Mahdavi, and T. Ott. The macroscopic behavior of the TCP congestion avoidance algorithm. *ACM/SIGCOMM CCR*, 1997.

[10] V. Paxson and S. Floyd. Wide-area traffic: the failure of Poisson modeling. *ACM/SIGCOMM CCR*, 24:257–268, 1994.

[11] G. Raina, D. Towsley, and D. Wischik. Part II: Control theory for buffer sizing. *ACM/SIGCOMM CCR*, 2005.

[12] G. Raina and D. Wischik. Buffer sizes for large multiplexers: TCP queueing theory and instability analysis. In *EuroNGI*, 2005. Extended version to appear in Queueing Systems.