Buffer Requirements for High-Speed Routers

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

Abstract *A core Internet router can typically buffer 250ms worth of data (1.25GByte at 40Gb/s). This would be challenging for an all-optical router. Happily, recent theory suggests that the optimal buffer size is around 30kByte.*

1. Introduction

Today, the buffer size in core Internet routers is typically chosen according to a rule of thumb which says: provide at least one round trip time's worth of buffering. The round trip time is often taken to be around 250ms (it takes 134ms to send a packet half way round the world and back, but queueing delay may plausibly add 100ms). A 40Gb/s linecard therefore needs 1.25GByte of memory.

Such large memories are hard to build in electronics. (The problem is that data can arrive at line rate, so the memory needs to be writeable at line rate. Such a high memory bandwidth is hard to engineer—and DRAM access speeds increases at only 7.5% a year [6].) Such large memories are also wildly impractical for any all-optical router that we can conceive of today.

Recent theoretical work has challenged the rule of thumb: it seems that a buffer of just 20 packets should be sufficient. The modeling behind this deals with TCP's algorithm for congestion control, with very short-timescale traffic statistics, and with queueing theory for Poisson traffic. (Long-range dependence has no bearing on buffer size.) This article outlines the theory.

Practical work has also challenged the rule of thumb. A measurement study of a Sprint backbone router [1] found that the queue size hardly ever exceeded 10 packets! (perhaps not surprising given that Sprint aims to keep utilization below 20%). Preliminary experiments on working routers [15] suggest that buffers can be much smaller than they are now, though the experimentation does not yet make clear quite how much.

Small buffers have obvious practical benefits. In an electronic router, the buffer could be on-chip, giving much higher memory bandwidth. In an all-optical router, 20-packet buffers might become feasible in the coming few years.

There are various design goals on which to base a decision about buffer size, beyond the practical question of `how hard is it to build and how much does it cost?' Section 2 outlines some of these goals, and the mathematical models that have accompanied them. Section 3 outlines a model for the behaviour of a network with small buffers, and suggests a refinement of the design goals.

The work described here is covered further in [2] [3] [5]. There are other approaches to small buffer theory. See especially [2] [6].

2. Design goals for buffer sizing

The buffer in an Internet router has several roles. It accommodates 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.

Delay and jitter. For real-time traffic, queueing delay and jitter are bad. It's hard to find a universal model for these two quantities; but at least with small buffers it's easy to get useful bounds.

Example. Consider a 20-packet buffer, serving a 40Gb/s link, and assume a packet size of 1500Byte. The maximum possible queueing delay is 6μs, which is negligible compared to a propagation delay of say 10ms. Jitter is therefore also negligible.

Accommodate bursts. A classical queueing-theoretic approach to buffer sizing might go like this. The purpose of a buffer is to absorb transient bursts in traffic. Given the service rate, the buffer size, and a traffic model, we can estimate the probability of packet loss. Choose a sufficiently large buffer size so that the loss probability is less than some specified threshold.

Example. Suppose we have a 40Gb/s queue, and we aim to run it at 95% utilization. A widely accepted Internet traffic model is fractional Brownian motion: if A(t) is the amount of work arriving in an interval of length t then A(t) has a Gaussian distribution with mean μt and standard deviation σt^H. We're aiming for 95% utilization, so let μ=38Gb/s. There are numerous measurement studies of long-range dependence in Internet traffic; a typical estimate of H is H≈0.85. Traffic measurements on the link of interest can be used to find σ; perhaps σ≈1Gb/s. Queueing theory (for reference see [7]) says that the packet loss probability p with a buffer of size B satisfies

$$\log(p) \approx - \inf_{t \geq 0} (B+(C-\mu)t)^2 / 2\sigma^2 t^{2H}$$

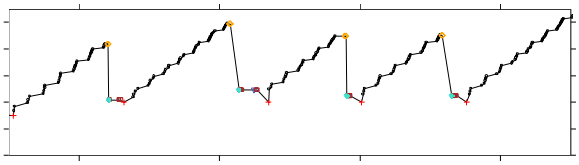
In order to guarantee a packet loss probability of less than say 10^{-4} we'd need a buffer of B=19.4Gb.

This approach is that it is rather sensitive to the traffic model. (Try changing σ and see how much B changes.) It also assumes that traffic statistics do not change over the time period of interest, which is hard to judge.

TCP's congestion avoidance. One trouble with naively using queueing theory to work out buffer size is that it doesn't take account of TCP's feedback control loop. How does this control loop work?

A TCP source attempts to limit congestion by using windowed flow control. The standard behaviour is that TCP sends a packet, the packet reaches the destination, the destination sends an acknowledgement back to the source. TCP limits the number of not-yet-acknowledged packets it allows into the network, and thereby limits its transmission rate. Let W be the permitted number of not-yet-acknowledged packets (the window size), and let RTT be the round trip time. In one round trip time, a TCP flow would send W packets and receive W acknowledgements, so its transmission rate is W/RTT.

By changing W, TCP can control its transmission rate. TCP steadily increases W when it receives acknowledgements (giving an average increase rate of roughly 1 packet per RTT), and it cuts W by $W/2$ when it detects a loss, inferred from missing acknowledgements. Therefore TCP will always experience loss, even in the absence of transmission errors, because it itself causes loss.



A plot of TCP window size [0-11 pkt] as a function of time [0-8s] for a flow with RTT 220ms. This graph is called the `TCP Sawtooth'.

Keep utilization high. The rule of thumb for buffer sizing derives from the following goal: If capacity is limited, it's desirable to make sure the link never goes idle. Therefore there should always be some packets in the buffer. By reasoning about how TCP responds to loss, we can work out how big the buffer needs to be.

Illustration. Consider a single bottleneck link with a single TCP flow. TCP will gradually increase the window size W, say to W_{max}, until it detects a loss, when W is cut to $W_{max}/2$. But this still leaves W_{max} unacknowledged packets in the network, so TCP is not allowed to send anything more until it has received $W_{max}/2$ acknowledgements. When it has, it starts up again sending new packets. We want the buffer never to empty, so packets reach the destination at rate C, so this pause lasts for $W_{max}/2C$. The buffer needs to be at least $W_{max}/2$ to avoid going empty during the pause.

What is W_{max}? Suppose the buffer is just big enough to avoid emptying. When TCP starts sending packets again, we want it to send at rate at least C, so the link doesn't idle. But its window size after the drop is $W_{max}/2$, so we need $W_{max}/2RTT=C$.

Putting these arguments together, the buffer needs to have capacity at least C×RTT packets to avoid going empty.

It is much harder to calculate the required buffer size in this way when there are multiple flows and multiple links. The trouble is that the flows may have different *RTT*s, the TCP flows may halve their windows at different times, etc. For some further work in this direction see [6] [8] [9].

Discussion of design goals. The objectives and the models discussed above should not be taken separately. The queueing-theory model suggests that loss can be reduced by making buffers large enough—but TCP's design means it fills up any buffer, no matter how large. It is good to have reasonably high utilization of the output link—but higher utilization generally means more loss and also larger fluctuations in buffer size, i.e. jitter. The small-buffer model in the next section marks a compromise between the objectives we have described here.

3. Theory of small buffers

In this section, we present a model for networks with small buffers and its justification. We also explain the design goals that such a network meets.

Summary of model. The network model comprises two parts: one part describing the dynamics of TCP, another describing the dynamics of the queue. For the first, consider a single long-lived TCP flow which experiences packet drop probability p and has round trip time RTT. According to the classic throughput formula [10], it achieves average transmission rate

$$x \approx 0.87 / RTT \sqrt{p}. \qquad (1)$$

If the drop probabilities at the queues along the route are $q_1,...,q_n$ then $p=1-\Pi(1-q_i.)$ The drop probability q at a given queue depends on the total incoming traffic rate y: q is approximately the drop probability when that queue is fed by Poisson traffic of rate y. All these equations (for flow rates, and for drop probabilities) should then be solved simultaneously.

This procedure, of setting up equations for link rates and for drop probabilities and then solving them simultaneously, is called the fixed point approach. References are collected in [11].

Example. Consider a single link serving 100 flows with common RTT of 200ms. The total load is

$$y \approx 100 \times 0.87 / 0.2 \sqrt{p} \quad \text{pkt/s.}$$

Suppose the queue has service rate C=2000 pkt/s (24Mb/s for packet size 1500Byte) and a buffer for B=20 packets. We could set up a Markov chain model to calculate the drop probability, but to save a tedious calculation we will make the crude approximation

$$p \approx (y/C)^B \qquad (2)$$

(i.e. pretend the queue has exponential service and an infinite buffer, and calculate the probability that the queue size exceeds B). Solving these two equations simultaneously yields

$$p \approx 0.062, \quad y \approx 1741 \text{ pkt/s.}$$

Non-TCP traffic and short-lived TCP flows can be incorporated into the model as an extra component to the total traffic rate y.

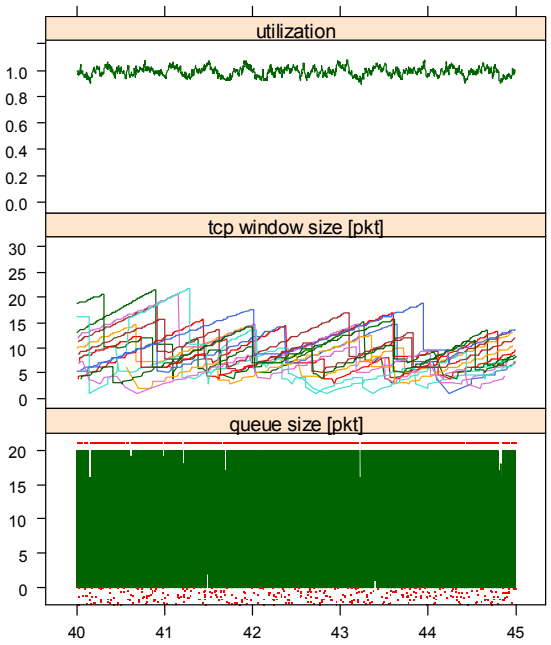
Justification of model. The first part of the model, giving the average transmission rate of a long-lived TCP flow, is well-known. A simplistic model shows us why x depends on RTT and p in the way it does. Suppose that drops are periodic, once every $1/p$ packets. Then the window size follows a precise sawtooth, increasing from W_{min} packets to W_{max} packets and then falling back to W_{min}. Since the window halves when there is a loss, $W_{min}=W_{max}/2$. Since the window size increases at rate 1 packet per RTT, and it increases by $W_{max}/2$ packets in every sawtooth, the period is $RTT \times W_{max}/2$. The number of packets sent in any single sawtooth can then be found by integrating the transmission rate, which increases from $W_{max}/2RTT$ to W_{max}/RTT pkt/s over $RTT \times W_{max}/2$ seconds: it is $W_{max}^2/2$. But we have assumed that there is one drop every $1/p$ packets. Therefore $W_{max}=\sqrt{(2/p)}$. Averaging over a period, the average transmission rate is $(W_{max}^2/2)/(RTT \times W_{max}/2)$, i.e. $\sqrt{2} / RTT\sqrt{p}$. A more careful calculation, accounting for random packet drops, gives the formula (1).

More interesting is the justification for the use of a Poisson queueing model (especially given the empirical evidence for long range dependence in Internet traffic). There are several supporting pieces of evidence. First, the aggregate of many independent point processes converges to a Poisson process over very short timescales. Second, measurements of Internet traffic confirm this short-timescale Poisson phenomenon. Third, classic queueing models show that a queue with a fast service rate and a small buffer size will most likely overflow over short timescales. (These claims are justified in [12].) Fourth, the aggregate arrival rate only changes slowly, over several RTTs (in fact, it satisfies a differential equation [20]). Over a short timescale we therefore expect the aggregate traffic to vary hardly at all, we expect the queue size to fluctuate rapidly, and we expect the queue size distribution to match that of a queue fed by a Poisson process. This argument is made in more detail in [5] [13].

This is not inconsistent with published results on long-range dependence [14]. Over short timescales Internet traffic looks Poisson; over long timescales it looks long-range dependent. If buffers are small then small bursts in traffic are the likely cause of buffer overflow. Since small bursts occur over short timescales, long-range statistical characteristics are irrelevant.

(There is one proviso about Poisson traffic modeling. Simulations show that, if access link speeds are very high, the packets from a single TCP flow tend to arrive in bursts—the burst size being equal to the TCP window size [8]. If this is the case, a batch-Poisson traffic model should be used instead, and equation (2) needs to be updated. The practical implication is that the buffer needs to be able to accommodate 20 *bursts*, not 20 *packets*.)

The following plot (from a simulator by Mark Handley, UCL) clearly shows the separation of timescales. The queue size fluctuates so rapidly that, in a plot spanning 5 seconds of simulated time, all we see is a blur.



Five-second simulation trace for 2000 TCP flows with average round trip time 120ms and packet size 1500Byte, sharing a single bottleneck link with capacity 1.6Mb/s. Even though the TCP flows are all following the sawtooth, and varying their rates by a factor of two, total utilization hardly changes. Queue size fluctuates very rapidly, emptying and filling again repeatedly every 1ms.

It is natural to ask whether the utilization graph is always flat, or if it is possible that the TCP sawtooths can become synchronized, which would lead to large

fluctuations in utilization. This is indeed possible. The theory for predicting synchronization is described in further detail in [3] [5]. We will come back to synchronization in the following sections.

Design goals. From this model, we can see that small buffers do not satisfy exactly the design goals that we suggested in Section 2. There is no obvious way to set a certain target drop probability—TCP will adapt, and achieve the drop probability it wants. This reflects an early design goal of the Internet: that *robustness* is the most important design goal. TCP adapts to network conditions, and it will continue to function even if the network is hopelessly underprovisioned. If that is the case, we have to expect the drop probability to be high.

The network operator can control drop probability by providing enough capacity. For a given buffer size B, and for given traffic conditions (number of flows and their *RTT*s), it is easy to calculate the service rate C needed to achieve a certain target drop probability. (The answer depends on traffic conditions, so in fact C should be chosen based on the expected range of traffic conditions.)

What is the effect of buffer size B, and why have we suggested B=20 packets? Equation (2) gives the tradeoff between buffer size, drop probability, and utilization. For $B \geq 20$ we get packet loss probability of less than 1% at 75% utilization. If we can build a small-buffer router which runs more than 33% faster than a standard large-buffer router, then it's worth sacrificing the 25% utilization in order to obtain higher throughput.

There is in fact a deeper reason why B=20 packets is a good compromise. The synchronization analysis alluded to above shows that the system tends to become synchronized for B much larger than 20 packets and much smaller than $C \times RTT$ packets. The consequences of synchronization are jitter, loss of utilization, and general unpredictability. Therefore buffers must either be very small or very large; they can't be in between.

Note: one design goal which is automatically met with small buffers is the goal of keeping queueing delay and jitter small.

4. Evolution of TCP

TCP determines a relationship between drop probability and throughput and round trip time, specified by equation (1). For some users (e.g. particle physicists who want very high throughput for transatlantic file transfers) that equation is too restrictive. This has prompted work on modifying TCP's congestion control algorithm [16] [17]. For any new algorithm, one can write down a modified version of (1), solve the fixed point equations, and calculate the operating point. One can also analyse synchronization. The modified TCP described in [17] has in fact been proven to avoid synchronization, assuming small buffers, in an arbitrary network with heterogeneous round trip times [18]. Such general results are not known for networks with large buffers. For further discussion see [3] [19].

5. Conclusion

Buffers in core routers can be very small, as small as 25kByte. Such small buffers reduce queueing delay and jitter. They also lead to slightly reduced utilization—though the sacrifice is worth considering if it permits all-optical packet-switched routers.

References

1 N.Hohn et al., ACM Sigmetrics 2004, "Bridging router performance and queuing theory"

2 D.Wischik et al., ACM CCR 2005, "Part I: Buffer sizes for core routers"

3 G.Raina et al., ACM CCR 2005, "Part II: Control theory for buffer sizing"

4 M.Enachescu et al., ACM CCR 2005, "Part III: Routers with very small buffers"

5 G.Raina et al., EuroNGI 2005, "Buffer sizes for large multiplexers: TCP queueing theory and instability analysis"

6 G.Appenzeller et al., ACM SIGCOMM 2004, "Sizing router buffers"

7 A.Ganesh et al., "Big Queues", Springer 2004

8 S.Shenker et al., ACM CCR 1990, "Some observations on the dynamics of a congestion control algorithm"

9 A.Dhamdhere et al., IEEE Infocom 2005, "Buffer sizing for congested Internet links"

10 M.Mathis et al., ACM CCR 1997, "The macroscopic behaviour of TCP congestion control"

11 V.Firoiu et al., Proc. IEEE 2002, "Theories and models for Internet quality of service"

12 J.Cao et al., IEEE Infocom 2002, "A Poisson limit for buffer overflow probabilities"

13 S.Dep et al., ACM Sigmetrics 2004, "Rate-based versus queue-based models of congestion control"

14 V.Paxson et al., ACM CCR 1994, "Wide-area traffic: the failure of Poisson modeling"

15 G.Appenzeller, PhD thesis, Stanford 2004

16 S.Floyd, RFC 3649, 2003.

17 T.Kelly, technical report CUED/ F-INFENG/ TR.435, Cambridge Univ. Eng. Dept., 2002

18 G.Vinnicombe, Proc. IFAC World Congress on Automatic Control 2002, "On the stability of networks operating TCP-like congestion control"

19 F.Kelly, Euro. J. Control 2003, "Fairness and stability of end-to-end congestion control"

20 V.Misra et al., ACM CCR 2000, "Fluid-based analysis of a network of AQM routers supporting TCP flows with an application to RED"