Recent Results on Sizing Router Buffers

Guido Appenzeller Nick McKeown
Stanford University
appenz|nickm@stanford.edu

Joel Sommers Paul Barford
University of Wisconsin
jsommers|pb@cs.wisc.edu

Abstract

Today all Internet routers are built with buffers that hold packets in times of congestion. These buffers can typically store between 250ms to one second worth of data. According to a widely used "rule-of-thumb", a link needs a buffer of size $B = \overline{RTT} \times C$, where $\overline{RTT}$ is the average round-trip time of a flow passing across the link, and C is the data rate of the link. For example, a 10Gb/s router linecard needs approximately 250ms $\times$ 10Gb/s = 2.5Gbits of buffers; and the amount of buffering grows linearly with the line-rate. Such large buffers are challenging for router manufacturers, who must use large, slow, off-chip DRAMs. They also lead to high latencies in case of congestion. Recent research suggest that the rule-of-thumb ($B = \overline{RTT} \times C$) is now outdated and incorrect for routers serving highly aggregated traffic. According to these new results, a link with n flows requires no more than $B = (\overline{RTT} \times C)/\sqrt{n}$, for long-lived or short-lived TCP flows. These lower buffer requirements have been verified in simulation and laboratory experiments as well as on real networks with live traffic. The consequences on router design are enormous: a 2.5Gb/s link carrying 10,000 flows could reduce its buffers by 99% with negligible difference in throughput, and a 10Gb/s link carrying 50,000 flows requires only 10Mbits of buffering, which can easily be implemented using fast, on-chip SRAM.

1 Introduction and Motivation

1.1 Background

A substantial amount of the space, power and cost of a high-end router line card today is used by the buffer memory that stores packets during times of congestion. If these buffers fill up, they cause queueing delay and delay-variance; when they overflow they cause packet loss, and when they underflow they can degrade throughput. Given the significance of their role, we might reasonably expect the dynamics and sizing of router buffers to be well understood, based on a well-grounded theory, and supported by extensive simulation and experimentation. This is not so.

Router buffers are sized today based on a rule-of-thumb commonly attributed to a 1994 paper by Villamizar and Song [1] [1]. Using experimental measurements of at most eight TCP flows on a 40 Mb/s link, they concluded that a router needs an amount of buffering equal to the average round-trip time of a flow that passes through the router, multiplied by the capacity of the router's network interfaces. This is the well-known $B = \overline{RTT} \times C$ rule. We will show later that the rule-of-thumb does indeed make sense for one (or a small number of) long-lived TCP flows.

Network operators follow the rule-of-thumb and require that router manufacturers provide 250ms (or more) of buffering [3]. The rule is found in architectural guidelines [4], too. Requiring such large buffers complicates router design, and is an impediment to building routers with larger capacity. For example, a 10Gb/s router linecard needs approximately $250ms \times$ 10Gb/s= 2.5Gbits of buffers, and the amount of buffering grows linearly with the line-rate.

[1] While commonly attributed to this paper, the rule-of-thumb is actually much older [2].

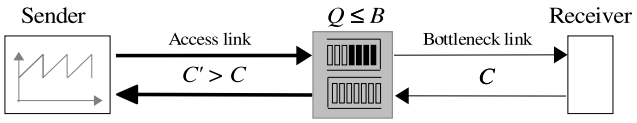

Figure 1: Single flow topology consisting of an access link of latency l_{Acc} and link capacity C_{Acc} and a bottleneck link of capacity C and latency l.

As the assumptions that lead to the rule of thumb have changed, it seems justified to ask if it still holds. While it is still true that most traffic uses TCP, the number of flows has increased significantly. Today, backbone links commonly operate at 2.5Gb/s or 10Gb/s, and carry over 10,000 flows.

Recent results [5] indeed suggest that this rule-of-thumb is no longer correct. We believe that significantly smaller buffers could be used in backbone routers (e.g., by removing 99% of the buffers) without a loss in network utilization. We demonstrate this theoretically, in simulations and experiments in Section 5.1 and 5.2 respectively. At the very least, we believe that the possibility of using much smaller buffers warrants comprehensive testing on real backbone networks. This paper isn't the last word, and our goal is to persuade one or more network operators to try reduced router buffers in their backbone network.

It is worth asking why we care to accurately size router buffers; with declining memory prices, why not just overbuffer routers? Overbuffering is a bad idea for two reasons. First, it complicates the design of high-speed routers, leading to higher power consumption, more board space, and lower density. Second, overbuffering increases end-to-end delay in the presence of congestion. Large buffers conflict with the low-latency needs of real time applications (e.g., video games, and device control).

1.2 Where does the rule-of-thumb come from?

The rule-of-thumb comes from a desire to keep a congested link as busy as possible, so as to maximize the throughput of the network. We are used to thinking of sizing queues 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.

The rule-of-thumb comes from the dynamics of TCP's congestion control algorithm. In particular, a single TCP flow passing through a bottleneck link requires a buffer size equal to the bandwidth-delay product in order to prevent the link from going idle and thereby losing throughput. Here, we will give a quick intuitive explanation of where the rule-of-thumb comes from; in particular, why this is just the right amount of buffering if the router carried just one long-lived TCP flow. In Section 2 we will give a more precise explanation, which will set the stage for a theory for buffer sizing with one flow, or with multiple long- and short-lived flows. Later, we will confirm that our theory is true using simulation and experiments in Sections 5.1 and 5.2, respectively.

Consider the simple topology in Figure 1 in which a single TCP source sends an infinite amount of data with packets of constant size. The flow passes through a single router, and the sender's access link is much faster than the receiver's bottleneck link of capacity C, causing packets to be queued at the router. The propagation time between sender and receiver (and vice versa) is denoted by T_p. Assume that the TCP flow has settled into the additive-increase and multiplicative-decrease (AIMD) congestion avoidance mode.

The sender transmits a packet each time it receives an ACK, and gradually increases the number of outstanding packets (the window size), which causes the buffer to gradually fill up. Eventually a packet is dropped, and the sender doesn't receive an ACK. It halves the window size and pauses. The sender now has too many packets outstanding in the network: it sent an amount equal to the

old window, but now the window size has halved. It must therefore pause while it waits for ACKs to arrive before it can resume transmitting. The key to sizing the buffer is to make sure that while the sender pauses, the router buffer doesn't go empty and force the bottleneck link to go idle. By determining the rate at which the buffer drains, we can determine the size of the reservoir needed to prevent it from going empty. It turns out that this is equal to the distance (in bytes) between the peak and trough of the "sawtooth" representing the TCP window size. We will show later that this corresponds to the rule-of-thumb: $B = \overline{RTT} \times C$.

It is worth asking if the TCP sawtooth is the *only* factor that determines the buffer size. For example, shouldn't statistical multiplexing, and the sudden arrival of short bursts have an effect? In particular, we might expect the (very bursty) TCP slow-start phase to increase queue occupancy and frequently fill the buffer. We will find in Section 4 that bursts from short flows do have an effect. However it is very small, and that the buffer size is, in fact, dictated by the number of long flows.

1.3 How buffer size influences router design

Having seen where the rule-of-thumb comes from, let's see why it matters; in particular, how it complicates the design of routers. At the time of writing, a state of the art router linecard runs at an aggregate rate of 40Gb/s (with one or more physical interfaces), has about 250ms of buffering, and so has 10Gbits (1.25Gbytes) of buffer memory.

Buffers in backbone routers are built from commercial memory devices such as dynamic RAM (DRAM) or static RAM (SRAM).[2] The largest commercial SRAM chip today is 36Mbits, which means a 40Gb/s linecard would require over 300 chips, making the board too large, too expensive and too hot. If instead we try to build the linecard using DRAM, we would just need 10 devices. This is because DRAM devices are available up to 1Gbit. But the problem is that DRAM has a random access time of about 50ns, which is hard to use when a minimum length (40byte) packet can arrive and depart every 8ns. Worse still, DRAM access times fall by only 7% per year, and so the problem is going to get worse as line-rates increase in the future.

In practice router linecards use multiple DRAM chips in parallel to obtain the aggregate data-rate (or memory-bandwidth) they need. Packets are either scattered across memories in an ad-hoc statistical manner, or use an SRAM cache with a refresh algorithm [6]. Either way, such a large packet buffer has a number of disadvantages: it uses a very wide DRAM bus (hundreds or thousands of signals), with a huge number of fast data pins (network processors and packet processor ASICs frequently have more than 2,000 pins making the chips large and expensive). Such wide buses consume large amounts of board space, and the fast data pins on modern DRAMs consume a lot of power.

In summary, it is extremely difficult to build large packet buffers at 40Gb/s and beyond. Given how slowly memory speeds improve, this problem is going to get worse over time.

Substantial benefits could be gained by placing the buffer memory directly on the chip that processes the packets (a network processor or an ASIC). In this case, very wide and fast access to a single memory is possible. Commercial packet processor ASICs have been built with 256Mbits of "embedded" DRAM. If memories of 2% the delay-bandwidth product were acceptable (i.e., 98% smaller than they are today), then a single-chip packet processor would need no external memories. We will present evidence later that buffers this small might make little or no difference to the utilization of backbone links.

2 Buffer Size for a Single Long-lived TCP Flow

In the next two sections we will determine how large the router buffers need to be if all the TCP flows are long-lived. We will start by examining a single long-lived flow, and then consider what

[2]DRAM includes devices with specialized I/O, such as DDR-SDRAM, RDRAM, RLDRAM and FCRAM.

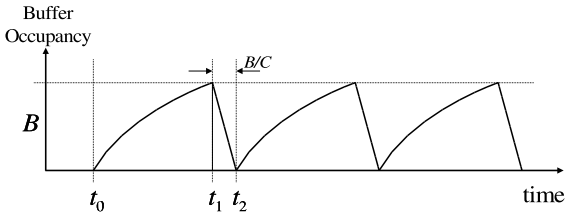

Figure 2: Schematic evolution of a router buffer for a single TCP flow.

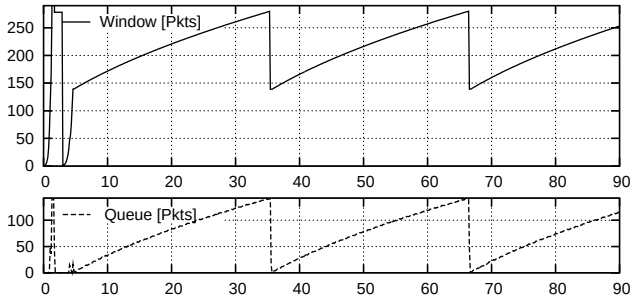

Figure 3: A TCP flow through a single router with buffers equal to the delay-bandwidth product. The upper graph shows the time evolution of the congestion window $W(t)$. The lower graph shows the time evolution of the queue length $Q(t)$.

happens when many flows are multiplexed together.

Starting with a single flow, consider again the topology in Figure 1 with a single sender and one bottleneck link. The schematic evolution of the router's queue (when the source is in congestion avoidance) is shown in Figure 2. From time t_0, the sender steadily increases its window-size and fills the buffer, until the buffer has to drop the first packet. Just under one round-trip time later, the sender stops sending at time t_1, because it detects that a packet has been dropped. It halves its window size from W_{max} to $W_{max}/2$ packets[3]. Now, the window size limits the number of unacknowledged (i.e., outstanding) packets in the network. Before the loss, the sender is allowed to have W_{max} outstanding packets; but after the timeout, it is only allowed to have $W_{max}/2$ outstanding packets. Thus, the sender has too many outstanding packets, and it must pause while it waits for the ACKs for $W_{max}/2$ packets. Our goal is to make sure the router buffer never goes empty in order to keep the router fully utilized. Therefore, the buffer must not go empty while the sender is pausing.

If the buffer never goes empty, the router must be sending packets onto the bottleneck link at constant rate C. This in turn means that ACKs arrive to the sender at rate C. The sender therefore pauses for exactly $(W_{max}/2)/C$ seconds for the $W_{max}/2$ packets to be acknowledged. It then resumes sending, and starts increasing its window size again.

The key to sizing the buffer is to make sure that the buffer is large enough, so that while the sender pauses, the buffer doesn't go empty. When the sender first pauses at t_1, the buffer is full, and so it drains over a period B/C until t_2 (shown in Figure 2). The buffer will just avoid going empty if the first packet from the sender shows up at the buffer just as it hits empty, i.e., $(W_{max}/2)/C \leq B/C$, or

$$B \geq W_{max}/2.$$

To determine W_{max}, we consider the situation after the sender has resumed transmission. The window size is now $W_{max}/2$, and the buffer is empty. The sender has to send packets at rate C or the link will be underutilized. It is well known that the sending rate of TCP is $R = W/RTT$. Since

[3]While TCP measures window size in bytes, we will count window size in packets for simplicity of presentation.

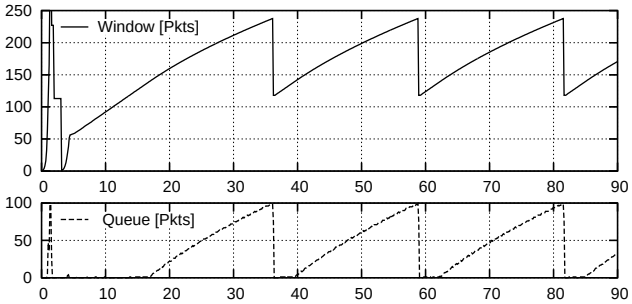

Figure 4: A TCP flow through an underbuffered router.

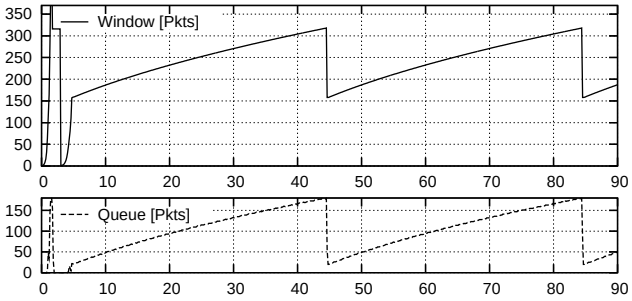

Figure 5: A TCP flow through an overbuffered router.

the buffer is empty, we have no queueing delay. Therefore, to send at rate C we require that

$$C = \frac{W}{RTT} = \frac{W_{max}/2}{2T_P}$$

or $W_{max}/2 = 2T_P \times C$ which for the buffer leads to the well-known rule-of-thumb

$$B \geq 2T_p \times C = \overline{RTT} \times C.$$

Figure 3 illustrates the evolution of a single TCP flow, using the topology shown in Figure 1. The buffer size is exactly equal to the rule-of-thumb, $B = \overline{RTT} \times C$. The window size follows the familiar sawtooth pattern, increasing steadily until a loss occurs and then halving the window size before starting to increase steadily again. Notice that the buffer occupancy *almost* hits zero once per packet loss, but never stays empty. This is the behavior we want for the bottleneck link to stay busy.

Figures 4 and 5 show what happens if the link is underbuffered or overbuffered. In Figure 4, the router is *underbuffered*, and the buffer size is less than $\overline{RTT} \times C$. The congestion window follows the same sawtooth pattern as in the sufficiently buffered case. However, when the window is halved and the sender pauses waiting for ACKs, there is insufficient reserve in the buffer to keep the bottleneck link busy. The buffer goes empty, the bottleneck link goes idle, and we lose throughput.

On the other hand, Figure 5 shows a flow which is *overbuffered*. It behaves like a correctly buffered flow in that it fully utilizes the link. However, when the window is halved, the buffer does not completely empty. The queueing delay of the flows is increased by a constant, because the buffer always has packets queued.

In summary, if $B \geq 2T_p \times C = \overline{RTT} \times C$, the router buffer (just) never goes empty, and the bottleneck link will never go idle.

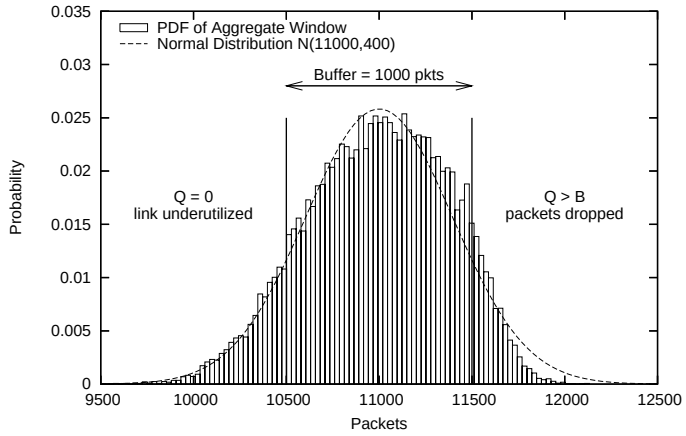

Figure 6: The probability distribution of the sum of the congestion windows of all flows passing through a router and its approximation with a normal distribution. The two vertical marks mark the boundaries of where the number of outstanding packets fit into the buffer. If sum of congestion windows is lower and there are less packets outstanding, the link will be underutilized. If it is higher the buffer overflows and packets are dropped.

3 When many long TCP flows share a link

In a backbone router many flows share the bottleneck link simultaneously, and so the single long-lived flow is not a realistic model. For example, a 2.5Gb/s (OC48c) link typically carries over 10,000 flows at a time [7].[4] So how should we change our model to reflect the buffers required for a bottleneck link with many flows? We will consider two situations. First, we will consider the case when all the flows are synchronized with each other, and their sawtooths march in lockstep perfectly in-phase. Then we will consider flows that are not synchronized with each other, or are at least not so synchronized as to be marching in lockstep. When they are sufficiently desynchronized — and we will argue that this is the case in practice — the amount of buffering required drops sharply.

It is well-documented that if multiple TCP flows share a bottleneck link, they can become synchronized with each other [8, 9]. They are coupled because they experience packet drops at roughly the same time, and so their "sawtooths" become synchronized and in-phase. If a number of flows share a bottleneck link, they each halve their window size at the same time; and so the aggregate window process (the sum of all the window size processes), looks like an amplified version of a single flow. As with a single flow, the buffer needs to be as large as the distance from the peak to the trough of the aggregate window size process, which is still equal to the bandwidth-delay product.

Flows are not synchronized in a backbone router carrying thousands of flows with varying RTTs. Small variations in RTT or processing time are sufficient to prevent synchronization [10]; and the absence of synchronization has been demonstrated in real networks [7, 11]. Likewise, we found in our simulations and experiments that while in-phase synchronization is common for under 100 concurrent flows, it is very rare above 500 concurrent flows [5]. Although we don't precisely understand when

[4]This shouldn't be surprising: A typical user today is connected via a 56kb/s modem, and a fully utilized 2.5Gb/s can simultaneously carry over 40,000 such flows. When it's not fully utilized, the buffers are barely used, and the link isn't a bottleneck. So we should size the buffers for when there are a large number of flows.

[5]Some out-of-phase synchronization (where flows are synchronized but scale down their window at different times during a cycle) was visible in some *ns*2 simulations with up to 1000 flows. However, the buffer requirements are very similar for out-of-phase synchronization as they are for no synchronization at all.

and why synchronization of TCP flows takes place, we have observed that for aggregates of over 500 flows, the amount of in-phase synchronization decreases. Under such circumstances we can treat flows as being not synchronized at all.

To understand the difference between adding synchronized and desynchronized window size processes, recall that if we add together many synchronized sawtooths, we get a single large sawtooth, and the buffer size requirement doesn't change. If on the other hand the sawtooths are not synchronized, the more flows we add, the less their sum will look like a sawtooth; they will smooth each other out, and the distance from the peak to the trough of the aggregate window size will get smaller. Hence, given that we need as much buffer as the distance from the peak to the trough of the aggregate window size, we can expect the buffer size requirements to get smaller as we increase the number of flows. This is indeed the case, and we will provide an intuitive argument. For a more detailed mathematical treatment see [5].

Consider a set of TCP flows with random (and independent) start times and propagation delays. We'll assume that they are desynchronized enough that the window size processes are independent of each other. We can model the total window size as a bounded random process made up of the sum of these independent sawtooths. We know from the central limit theorem that the aggregate window size process will converge to a Gaussian process. Figure 6 shows that indeed the aggregate window size does converge to a Gaussian process. The graph shows the probability distribution of the sum of the congestion windows of all flows $W = \sum W_i$, with different propagation times and start times as explained in Section 5.1.

How does the shape of the Gaussian, and thus our buffer, depend on the number of flows? If we have more flows, we would expect more statistical multiplexing and thus a narrower Gaussian. In fact, the central limit theorem tells us that if we increase the number of flows n, the width of the Gaussian (or, more formally, its standard deviation) should decrease with $\frac{1}{\sqrt{n}}$. The role of the buffer is to absorb the fluctuation in the total window size. If the standard deviation of the total window size decreases with $\frac{1}{\sqrt{n}}$, we would expect the required amount of buffer to do the same.

We know that the correct amount of buffering for a single flow ($n = 1$) is $2T_p \times C$ as for a single flow the old rule-of-thumb still holds. We would thus expect the required amount of buffering for desynchronized flows to be:

$$B_{min} = \frac{2\overline{T}_p \cdot C}{\sqrt{n}}.$$

This result has important practical implications for building routers. A core router currently has from 10,000 to over 100,000 flows passing through it at any given time. While the vast majority of flows are short (e.g., flows with fewer than 100 packets), the flow length distribution is heavy tailed and the majority of packets at any given time belong to long flows. As a result, such a router would achieve close to full utilization with buffer sizes that are only $\frac{1}{\sqrt{10000}} = 1\%$ of the delay-bandwidth product. We will verify this result experimentally in Section 5.2.

4 Sizing the Router Buffer for Short Flows

Not all TCP flows are long-lived; in fact many flows last only a few packets, never leave slow-start, and so never reach their equilibrium sending rate [7]. Up until now we've only considered long-lived TCP flows, and so now we'll consider how short TCP flows affect the size of the router buffer. We're going to find that short flows (TCP and non-TCP) have a much smaller effect than long-lived TCP flows, particularly in a backbone router with a large number of flows.

We will define a short-lived flow to be a TCP flow that never leaves slow-start (e.g., any flow with fewer than 90 packets, assuming a typical maximum window size of 65kB).

Consider again the topology in Figure 1 with multiple senders on separate access links. As has been widely reported from measurement, we can assume that new short flows arrive according to a

Poisson process [12, 13]. In slow-start, each flow first sends out two packets, then four, eight, sixteen, etc. This is the slow-start algorithm in which the sender increases the window-size by one packet for each received ACK. If the access links have lower bandwidth than the bottleneck link, the bursts are spread out and a single burst causes no queueing. We assume the worst case where access links have infinite speed, bursts arrive intact at the bottleneck router.

It has been shown [5] that in this case we can model the queue of the buffer using simple queueing theoretical model. From this model we can calculate the probability distribution of the router's queue length. If we know the queue length distribution, we can derive for it an upper bound for the loss rate for any given buffer. We know that if the length of the queue is shorter than the length of the buffer, no packets are dropped. Thus the probability of the queue exceeding a certain length is greater than the drop probability for a router that has a buffer of this length.

Using the model, we can approximate the queue length distribution using effective bandwidth methodology [14]. The result is that the probability of the queue exceeding a certain length (and thus the upper bound for the drop probability for a router having a length of this queue) is

$$P_{Drop}(B = b) = P(Q \geq b) = e^{-b\frac{2(1-\rho)}{\rho}\cdot\frac{E[X_i]}{E[X_i^2]}}$$

A complete derivation of this result can be found in [5].

Our goal is to drop very few packets (if a short flow drops a packet, the retransmission significantly increases the flow's duration). In other words, we want to choose a buffer size B such that $P(Q \geq B)$ is small.

A key observation is that — for short flows — the size of the buffer does not depend on the line-rate, the propagation delay of the flows, or the number of flows; it only depends on the load of the link, and length of the bursts. In practice the length of the bursts is limited by the maximum window size of TCP. Current operating systems have maximum window sizes of 12 (most flavors of Windows) to 43 (default on most UNIX hosts). Window sizes above 43 require the use of scaling options which are rarely used today. As a result, the amount of buffering needed to absorb bursts is almost fixed and is typically in the order of hundreds of packets.

The methodology used to model TCP flows can also be used for UDP flows and other traffic that does not react to congestion.

In practice, buffers can be made even smaller. For our model and simulation we assumed access links that are faster than the bottleneck link. There is evidence [7, 15] that highly aggregated traffic from slow access links in some cases can lead to bursts being smoothed out completely. In this case individual packet arrivals are close to Poisson, resulting in even smaller buffers. The buffer size can be easily computed with an M/D/1 model by setting $X_i = 1$.

In summary, short-lived flows require only small buffers. When there is a mix of short- and long-lived flows, we will see from simulations and experiments in the next section, that the short-lived flows contribute very little to the buffering requirements, and so the buffer size will usually be determined by the number of long-lived flows[6].

5 Simulation and Experimental Results

Up until now we've described only theoretical models of long- and short-lived flows. We now present results to validate our models. We use three validation methods: simulation (using *ns2*), experiments on physical routers in a lab setting and an experiment on a live production network.

[6]For a distribution of flows we define short flows and long flows as flows that are in slow-start and congestion avoidance mode respectively. This means that flows may transition from short to long during their existence.

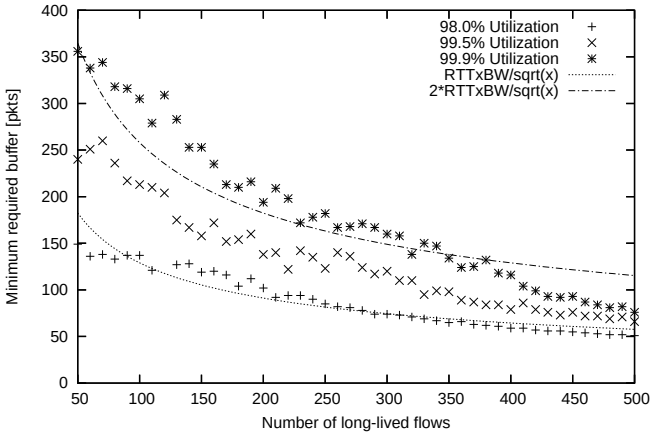

Figure 7: Minimum required buffer to achieve 98, 99.5 and 99.9 percent utilization for an OC3 (155 Mb/s) line with about 80ms average RTT measured with *ns*2 for long-lived TCP flows.

5.1 NS2 Simulations

In our simulations we assume a network with only one congested link in the core. Network operators usually run backbone links at loads of 10%-30% and as a result packet drops are rare in the Internet backbone. If a single point of congestion is rare, then it is unlikely that a flow will encounter two or more congestion points. We assume that the router maintains a single FIFO queue with drop-tail. However we expect our results to be valid for other queueing disciplines (e.g., RED) as well. The average propagation delay of a TCP flow varied from 25ms to 300ms.

5.1.1 Long-lived TCP Flows

Figure 7 simulates an OC3 (155Mb/s) line carrying long-lived TCP flows. The graph shows the minimum required buffer for a given utilization of the line, and compares it with the buffer size predicted by the model. For example, our model predicts that for 98% utilization a buffer of $\frac{\overline{RTT}\times C}{\sqrt{n}}$ should be sufficient. When the number of long-lived flows is small the flows are partially synchronized and the result doesn't hold. However, when the number of flows exceeds 250, the model holds well, as shown in the graph. We found that in order to attain 99.9% utilization we needed buffers twice as big, just as the model predicts.

In our simulations and experiments we also looked at the packet loss rate. Decreasing the latency of a TCP flow will always increase the loss rate. The loss rate of a TCP flow is a function of the flow's window size and can be approximated to $l = \frac{0.76}{W^2}$ (see [16]). If we reduce buffers, we decrease the RTT of the flow, therefore decrease the average W and thus increase loss. A positive side-effect is that with smaller buffers, TCP performance improves, as we will see below.

5.1.2 Short Flows

We will use a commonly used performance metric for short flows: the flow completion time, defined as the time from when the first packet is sent until the last packet reaches the destination. In particular, we will measure the average flow completion time (AFCT). Figure 8 shows the minimum required buffer so that the AFCT is increased by no more than 12.5%. Experimental data is from *ns*2 experiments for 40, 80 and 200 Mb/s and a load of 0.8. Our model, with $P(Q > B) = 0.025$, is plotted in the graph. The bound predicted by the M/G/1 model closely matches the simulation

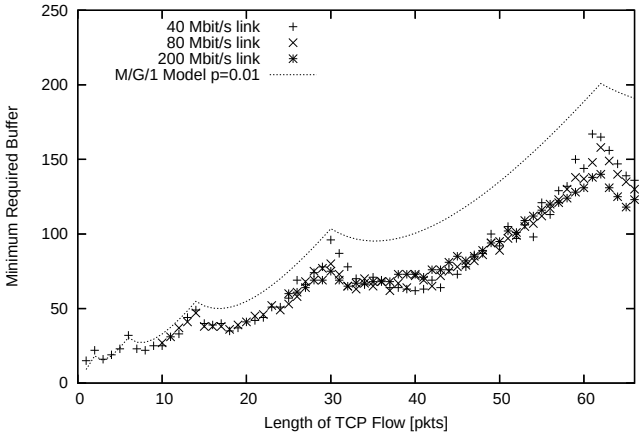

Figure 8: The minimum required buffer that increases the Average Flow Completion Time (AFCT) by not more than 12.5% vs infinite buffers for short flow traffic.

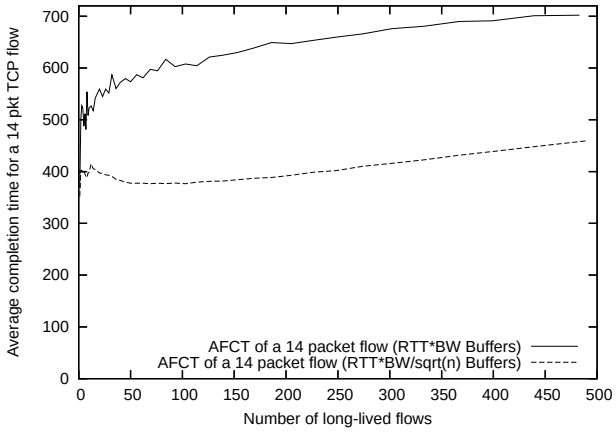

Figure 9: Average flow completion times with a buffer size of $(\overline{RTT} \times C)/\sqrt{n}$, compared with a buffer size $\overline{RTT} \times C$.

results.

The key result here is that the amount of buffering needed does not depend on the number of flows, the bandwidth or the round-trip time. It only depends on the load of the link and the length of the bursts. For the same traffic mix of *only short flows*, a future generation 1 Tb/s core router needs the same amount of buffering as a local 10 Mb/s router today.

5.1.3 Mixes of Short- and Long-Lived Flows

In mixes of short flows and long lived flows, the buffer requirements are driven by the number of long flows. This is even the case if the vast majority of flows, and even up to half of the bandwidth, is generated by short flows.

One might wonder if for mixes of flows, smaller buffers hurt the quality of service for short flows. Figure 9 shows that the opposite is the case — small buffers allow short flows to complete more quickly. In this $ns2$ simulation, the average flow completion time is much shorter with $\overline{RTT} \times C/\sqrt{n}$ buffers than with $\overline{RTT} \times C$ sized buffers. This is because the queueing delay is lower. So by reducing

the buffer size, we can still achieve the 100% utilization *and* decrease the completion times for shorter flows.

In real networks there are no classes of flow length but flow lengths follow a typically heavy-tailed distribution. We ran similar experiments with Pareto distributed flow lengths with essentially identical results.

5.2 Measurements on a Physical Router

While simulation captures most characteristics or router-TCP interaction, we verified our model by running experiments on a real backbone router with traffic generated by real TCP sources.

The router was a Cisco GSR 12410 with a 4 x OC3 POS "Engine 0" line card that switches IP packets using POS (PPP over Sonet) at 155Mb/s. The router has both input and output queues, although no input queueing took place in our experiments, as the total throughput of the router was far below the maximum capacity of the switching fabric. TCP traffic was generated using the Harpoon traffic generator [17] on Linux and BSD machines, and aggregated using a second Cisco GSR 12410 router with Gigabit Ethernet line cards.

TCP	**Router Buffer**			**Link Utilization (%)**		
Flows	$\frac{RTT \times BW}{\sqrt{n}}$	**Pkts**	**RAM**	**Model**	**Sim.**	**Exp.**
100	0.5 x	64	1 Mbit	96.9%	94.7%	94.9%
100	1 x	129	2 Mbit	99.9%	99.3%	98.1%
100	2 x	258	4 Mbit	100%	99.9%	99.8%
100	3 x	387	8 Mbit	100%	99.8%	99.7%
200	0.5 x	46	1 Mbit	98.8%	97.0%	98.6%
200	1 x	91	2 Mbit	99.9%	99.2%	99.7%
200	2 x	182	4 Mbit	100%	99.8%	99.8%
200	3 x	273	4 Mbit	100%	100%	99.8%
300	0.5 x	37	512 kb	99.5%	98.6%	99.6%
300	1 x	74	1 Mbit	100%	99.3%	99.8%
300	2 x	148	2 Mbit	100%	99.9%	99.8%
300	3 x	222	4 Mbit	100%	100%	100%
400	0.5 x	32	512 kb	99.7%	99.2%	99.5%
400	1 x	64	1 Mbit	100%	99.8%	100%
400	2 x	128	2 Mbit	100%	100%	100%
400	3 x	192	4 Mbit	100%	100%	99.9%

Figure 10: Comparison of our model, *ns*2 simulation and experimental results for buffer requirements of a Cisco GSR 12410 OC3 linecard.

Figure 10 shows the results of measurements from the GSR 12410 router. The router memory was adjusted by limiting the length of the interface queue on the outgoing interface. The buffer size is given as a multiple of $\frac{\overline{RTT} \times C}{\sqrt{n}}$, the number of packets and the size of the RAM device that would be needed. *Model* is the lower-bound on the utilization predicted by the model. *Sim.* and *Exp.* are the utilization as measured by a simulation with *ns*2 and on the physical router respectively. For 100 and 200 flows there is, as we expect, some synchronization. Above that the model predicts the utilization correctly within the measurement accuracy of about $\pm 0.1\%$. *ns*2 sometimes predicts a lower utilization than we found in practice. We attribute this to more synchronization between flows in the simulations than in the real network.

The key result here is that model, simulation and experiment all agree that a router buffer should have a size equal to approximately $\frac{\overline{RTT} \times C}{\sqrt{n}}$, as opposed to $\overline{RTT} \times C$ (which in this case would be 1291 packets).

Buffer	$\frac{RTT \times BW}{\sqrt{n}}$	Bandwidth (measured)	Utilization (measured)	Utilization (model)
500	>> 2×	19.981	99.92 %	100%
85	1.5 ×	19.709	98.55 %	99.9%
65	1.2 ×	19.510	97.55 %	99.5%
46	0.8 ×	19.481	97.41 %	95.9%

Figure 11: Results from the Experiment on a Production Network

5.3 Measurements on a Production Network

We verified the $\frac{\overline{RTT} \times BW}{\sqrt{n}}$ hypothesis on a production network at Stanford University. We throttled a router that serves traffic from the commercial Internet to student dormitories to 20 Mb/s, and measured performance with different buffer sizes. The buffer was serving a wide range of traffic with different protocols and applications. We estimated the number of concurrent flows to be approximately 400 and assumed a maximum RTT of 250ms. Table 11 shows the results. We that buffers of the size that our model predicts achieve close to 100% utilization. Additionally our model can qualitatively predict what the amount of buffering is where utilization drops from full to substantially below 100%.

A second experiment with a 10 Gb/s link on the Internet2 backbone is currently ongoing. Preliminary results indicate that running the router at 0.5% of its default buffer size (5ms compared with the default of 1 second) causes no measurable degradation in quality of service.

6 Conclusion

Recent results on router buffer sizing suggest that the buffers in backbone routers are significantly larger than they need to be — possibly by two orders of magnitude. If these results hold under a variety of conditions, they have important consequences for the design and operation of backbone routers. While is it difficult to persuade network operators to remove 99% of the buffer capacity from routers in functioning, profitable networks, such experiments must be the next step. We see the results presented in this paper as the first steps toward convincing operators to make such changes.

Even if the network operations community is persuaded to configure smaller buffers in their production routers, manufacturers of routers must also be convinced to build routers with smaller buffer capacities. In the short-term, this is difficult too. In a competitive marketplace, it is not obvious that a router vendor would feel comfortable building a router with 1% of the buffers of its competitors. For historical reasons, the network operator is likely to buy the router with larger buffers, even if they are unnecessary.

Eventually, if routers continue to be built using the current rule-of-thumb, it will become very difficult to build linecards from commercial memory chips. And so in the end, necessity may force buffers to be smaller. At least, if our results are true, we know the routers will continue to work just fine, and the network utilization is unlikely to be affected.

7 Acknowledgments

The authors would like to thank the Stanford Network Team, specifically Sunya Wang, Wayne Sung and Lea Roberts for their help on conducting the Stanford experiments.

References

[1] Curtis Villamizar and Cheng Song. High performance TCP in ANSNET. *ACM Computer Communications Review*, 24(5):45–60, 1994 1994.

[2] Van Jacobson. [e2e] re: Latest TCP measurements thoughts. Posting to the end-to-end mailing list, March 7, 1988.

[3] Cisco line cards. http://www.cisco.com/en/US/products/hw/modules/ps2710/products_data_sheets_list.html.

[4] R. Bush and D. Meyer. RFC 3439: Some internet architectural guidelines and philosophy, December 2003.

[5] Guido Appenzeller, Isaac Keslassy, and Nick McKeown. Sizing router buffers. Technical Report TR04-HPNG-06-08-00, Stanford University, June 2004. Extended version of the paper published at SIGCOMM 2004.

[6] Sundar Iyer, R. R. Kompella, and Nick McKeown. Analysis of a memory architecture for fast packet buffers. In *Proceedings of IEEE High Performance Switching and Routing*, Dallas, Texas, May 2001.

[7] Chuck J. Fraleigh. *Provisioning Internet Backbone Networks to Support Latency Sensitive Applications*. PhD thesis, Stanford University, Department of Electrical Engineering, June 2002.

[8] S. Shenker, L. Zhang, and D. Clark. Some observations on the dynamics of a congestion control algorithm. *ACM Computer Communications Review*, pages 30–39, Oct 1990.

[9] Sally Floyd and Van Jacobson. Random early detection gateways for congestion avoidance. *IEEE/ACM Transactions on Networking*, 1(4):397–413, 1993.

[10] Lili Qiu, Yin Zhang, and Srinivasan Keshav. Understanding the performance of many TCP flows. *Comput. Networks*, 37(3-4):277–306, 2001.

[11] Gianluca Iannaccone, Martin May, and Christophe Diot. Aggregate traffic performance with active queue management and drop from tail. *SIGCOMM Comput. Commun. Rev.*, 31(3):4–13, 2001.

[12] Vern Paxson and Sally Floyd. Wide area traffic: the failure of Poisson modeling. *IEEE/ACM Transactions on Networking*, 3(3):226–244, 1995.

[13] Anja Feldmann, Anna C. Gilbert, and Walter Willinger. Data networks as cascades: Investigating the multifractal nature of internet WAN traffic. In *SIGCOMM*, pages 42–55, 1998.

[14] F. P. Kelly. *Notes on Effective Bandwidth*, pages 141–168. Oxford University Press, 1996.

[15] J. Cao, W. Cleveland, D. Lin, and D. Sun. Internet traffic tends to poisson and independent as the load increases. Technical report, Bell Labs, 2001.

[16] Robert Morris. Scalable TCP congestion control. In *Proceedings of IEEE INFOCOM 2000*, Tel Aviv, USA, March 2000.

[17] Joel Sommers and Paul Barford. Self-configuring network traffic generation. In *Proceedings of the ACM SIGCOMM Internet Measurement Conference, Taormina, Italy*, October 2004.