Sizing Router Buffers

Stanford HPNG Technical Report TR04-HPNG-06-08-00

Guido Appenzeller
Stanford University
appenz@cs.stanford.edu

Isaac Keslassy
Stanford University
keslassy@yuba.stanford.edu

Nick McKeown
Stanford University
nickm@stanford.edu

ABSTRACT

All Internet routers contain buffers to hold packets during times of congestion. Today, the size of the buffers is determined by the dynamics of TCP's congestion control algorithm. In particular, the goal is to make sure that when a link is congested, it is busy 100% of the time; which is equivalent to making sure its buffer never goes empty. A widely used rule-of-thumb states that each 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. And queueing delays can be long, have high variance, and may destabilize the congestion control algorithms. In this paper we argue that the rule-of-thumb ($B = \overline{RTT} \times C$) is now outdated and incorrect for backbone routers. This is because of the large number of flows (TCP connections) multiplexed together on a single backbone link. Using theory, simulation and experiments on a network of real routers, we show that a link with n flows requires no more than $B = (\overline{RTT} \times C)/\sqrt{n}$, for long-lived or short-lived TCP flows. 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.

Categories and Subject Descriptors

C.2 [**Internetworking**]: Routers

*The authors are with the Computer Systems Laboratory at Stanford University. Isaac Keslassy is now with the Technion (Israel Institute of Technology), Haifa, Israel. This work was funded in part by the Stanford Networking Research Center, the Stanford Center for Integrated Systems, and a Wakerly Stanford Graduate Fellowship.

Technical Report TR04-HPNG-06-08-00

This is an extended version of the publication at SIGCOMM 04

The electronic version of this report is available on the Stanford High Performance Networking group home page http://yuba.stanford.edu/

General Terms

Design, Performance.

Keywords

Internet router, buffer size, bandwidth delay product, TCP.

1. INTRODUCTION AND MOTIVATION

1.1 Background

Internet routers are packet switches, and therefore buffer packets during times of congestion. Arguably, router buffers are the single biggest contributor to uncertainty in the Internet. Buffers 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].Using experimental measurements of at most eight TCP flows on a 40 Mb/s link, they concluded that — because of the dynamics of TCP's congestion control algorithms — 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 [2]. The rule is found in architectural guidelines [3], 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.

The goal of our work is to find out if the rule-of-thumb 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 [4].

One thing is for sure: It is not well understood how much buffering is actually needed, or how buffer size affects network performance [5]. In this paper we argue that the rule-of-thumb is outdated and incorrect. We believe that sig-

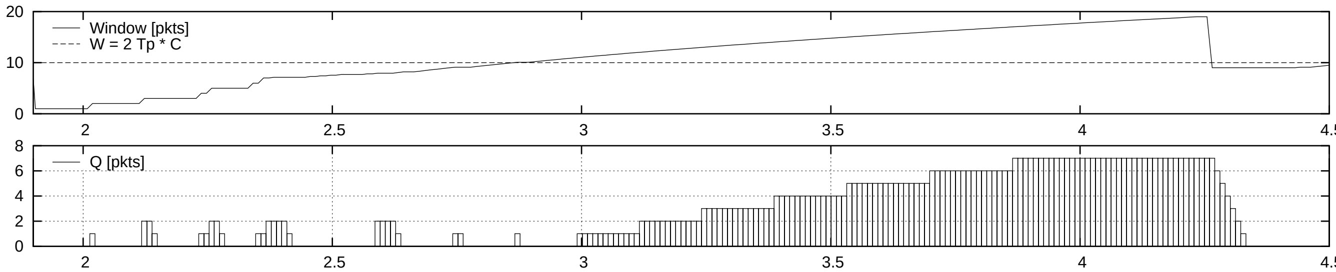

Figure 1: Window (top) and router queue (bottom) for a TCP flow through a bottleneck link.

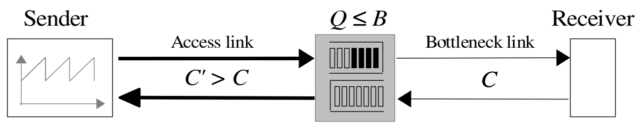

Figure 2: Single flow topology consisting of an access link of latency l_{Acc} and link capacity C_{Acc} and a bottleneck link of capacity C and latency l.

nificantly smaller buffers could be used in backbone routers (e.g. by removing 99% of the buffers) without a loss in network utilization, and we show theory, simulations and experiments to support our argument. At the very least, we believe that the possibility of using much smaller buffers warrants further exploration and study, with more comprehensive experiments needed 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? We believe 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). In some cases large delays can make congestion control algorithms unstable [6] and applications unusable.

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.

Router buffers are sized so that when TCP flows pass through them, they don't *underflow* and lose throughput; and this is where the rule-of-thumb comes from. The metric we will use is throughput, and our goal is to determine the size of the buffer so as to maximize throughput of a bottleneck link. The basic idea is that when a router has packets buffered, its outgoing link is always busy. If the outgoing link is a bottleneck, then we want to keep it busy as much of the time as possible, and so we just need to make sure the buffer never underflows and goes empty.

Fact: The rule-of-thumb is the amount of buffering needed by a single TCP flow, so that the buffer at the bottleneck link never underflows, and so the router doesn't lose throughput.

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 2 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.[1] 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

[1]We assume the reader is familiar with the dynamics of TCP. A brief reminder of the salient features can be found in Appendix A.

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, doesn'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. Figure 1 illustrates the effect of bursts on the queue size for a typical single TCP flow. Clearly the queue is absorbing very short term bursts in the slow-start phase, while it is accommodating a slowly changing window size in the congestion avoidance phase. We will examine the effect of burstiness caused by short-flows in Section 4. We'll find that the short-flows play a very small effect, 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 [7]. 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 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

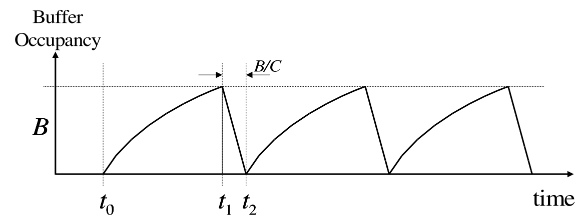

Figure 3: Schematic evolution of a router buffer for a single TCP flow.

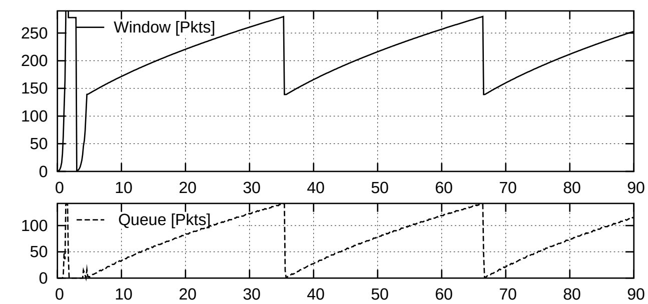

Figure 4: A TCP flow through a single router with buffers equal to the delay-bandwidth product. The upper graph shows the time evolution of the congestion window $W(t)$. The lower graph shows the time evolution of the queue length $Q(t)$.

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 happens when many flows are multiplexed together.

Starting with a single flow, consider again the topology in Figure 2 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 3. 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 times-out at time t_1, because it is waiting for an ACK for the dropped packet. It immediately 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

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

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

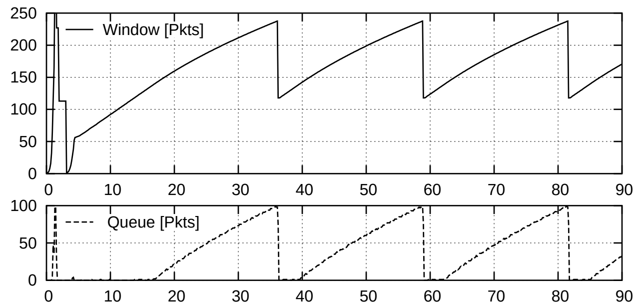

Figure 5: A TCP flow through an underbuffered router.

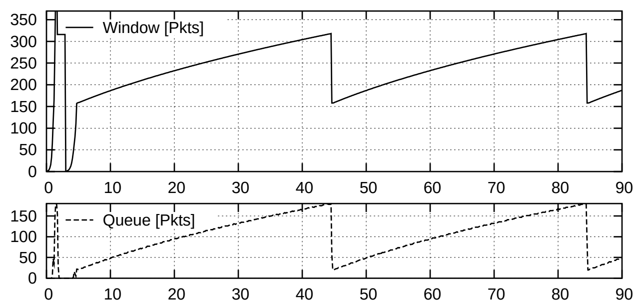

Figure 6: A TCP flow through an overbuffered router.

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 3). 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 $R = W/RTT$ (see equation 7 in AppendixA). Since 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

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

While not widely known, similar arguments have been made previously [8, 9], and our result can be easily verified using $ns2$ [10] simulation and a closed-form analytical model (Appendix B). Figure 4 illustrates the evolution of a single TCP Reno flow, using the topology shown in Figure 2. 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.

Appendix B presents an analytical fluid model that provides a closed-form equation of the sawtooth, and closely matches the $ns2$ simulations.

Figures 5 and 6 show what happens if the link is underbuffered or overbuffered. In Figure 5, 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 6 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.

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 [4].[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 drops sharply.

3.1 Synchronized Long Flows

Consider the evolution of *two* TCP Reno flows through a bottleneck router. The evolution of the window sizes, sending rates and queue length is shown in Figure 7.

Although the two flows start at different times, they quickly synchronize to be perfectly in phase. This is a well-documented and studied tendency of flows sharing a bottleneck to become synchronized over time [9, 11, 12, 13].

A set of precisely synchronized flows has the same buffer requirements as a single flow. Their aggregate behavior is still a sawtooth; as before, the height of the sawtooth is

[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.

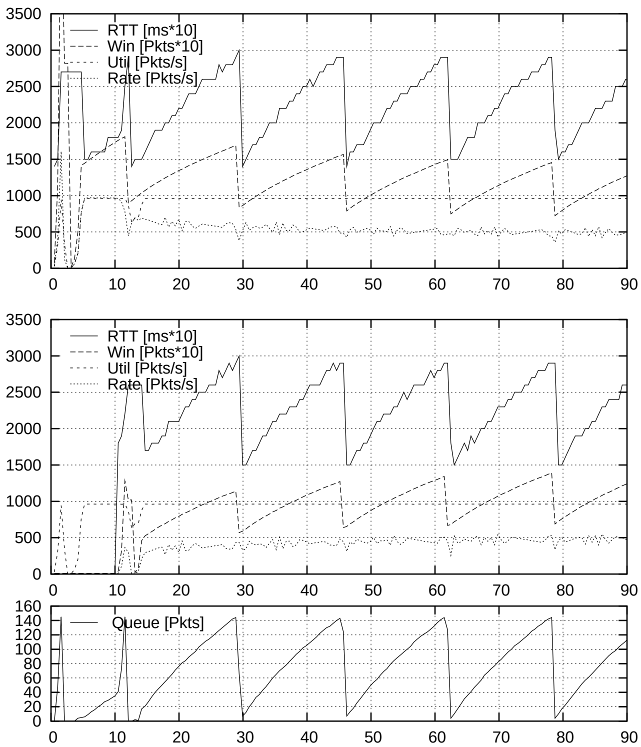

Figure 7: Two TCP flows sharing a bottleneck link. The upper two graphs show the time evolution of the RTT of the flows, the congestion window of the senders, the link utilization and the sending rate of the TCP senders. The bottom graph shows the queue length of the router buffer.

dictated by the maximum window size needed to fill the round-trip path, which is independent of the number of flows. Specifically, assume that there are n flows, each with a congestion window $W^i(t)$ at time t, and end-to-end propagation delay T_p^i, where $i = [1, ..., n]$. The window size is the maximum allowable number of outstanding bytes, so

$$\sum_{i=1}^{n} W^i(t) = 2\overline{T}_p \cdot C + Q(t) \quad (1)$$

where $Q(t)$ is the buffer occupancy at time t, and $\overline{T}_P$ is the average propagation delay. As before, we can solve for the buffer size by considering two cases: just before and just after packets are dropped. First, because they move in lock-step, the flows all have their largest window size, W_{max} at the same time; this is when the buffer is full, so:

$$\sum_{i=1}^{n} W^i_{max} = W_{max} = 2\overline{T}_p \cdot C + B. \quad (2)$$

Similarly, their window size is smallest just after they all drop simultaneously [9]. If the buffer is sized so that it just goes empty as the senders start transmitting after the pause, then

$$\sum_{i=1}^{n} W^i_{min} = \frac{W_{max}}{2} = 2\overline{T}_p \cdot C. \quad (3)$$

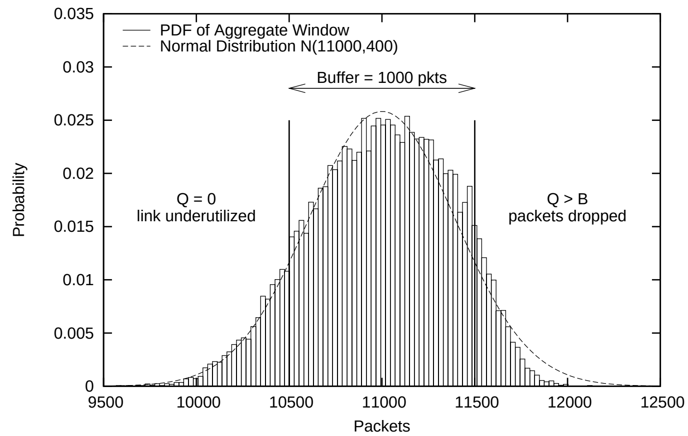

Figure 8: The probability distribution of the sum of the congestion windows of all flows passing through a router and its approximation with a normal distribution. The two vertical marks mark the boundaries of where the number of outstanding packets fit into the buffer. If sum of congestion windows is lower and there are less packets outstanding, the link will be underutilized. If it is higher the buffer overflows and packets are dropped.

Solving for B we find once again that $B = 2\overline{T}_p \cdot C = \overline{RTT} \cdot C$. Clearly this result holds for any number of synchronized in-phase flows.

3.2 Desynchronized Long Flows

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 [14]; and the absence of synchronization has been demonstrated in real networks [4, 15]. 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 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

[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 $ns2$ 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.

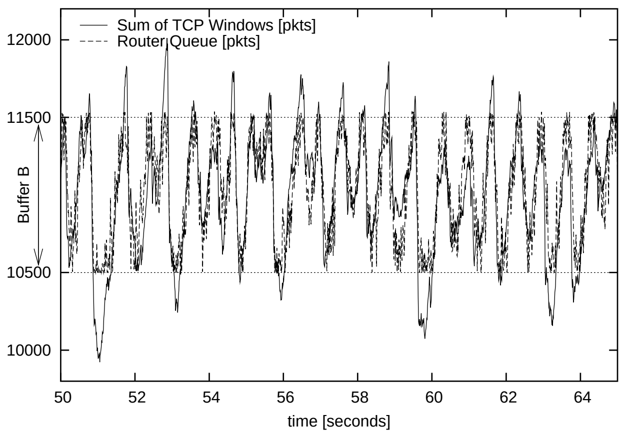

Figure 9: Plot of $\sum W_i(t)$ of all TCP flows, and of the queue Q offset by 10500 packets.

requirements to get smaller as we increase the number of flows. This is indeed the case, and we will explain why, and then demonstrate via simulation.

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 8 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.

From the window size process, we know that the queue occupancy at time t is

$$Q(t) = \sum_{i=1}^{n} W_i(t) - (2T_P \cdot C) - \epsilon. \qquad (4)$$

In other words, all outstanding bytes are in the queue ($Q(t)$), on the link ($2T_p \cdot C$), or have been dropped. We represent the number of dropped packets by ϵ. If the buffer is large enough and TCP is operating correctly, then ϵ is negligible compared to $2T_P \cdot C$. Therefore, the distribution of $Q(t)$ is shown in Figure 9, and is given by

$$Q \stackrel{d}{=} W - 2T_P \cdot C. \qquad (5)$$

Because W has a normal distribution, Q has the distribution of a normal shifted by a constant (of course, the normal distribution is restricted to the allowable range for Q). This is very useful, because we can now pick a buffer size and know immediately the probability that the buffer will underflow and lose throughput.

Because it is gaussian, we can determine the queue occupancy process if we know its mean and variance. The mean is simply the sum of the mean of its constituents. To find the variance, we'll assume for convenience that all sawtooths have the same average value (having different values would still yield the same results). Each TCP sawtooth can be modelled as oscillating with a uniform distribution around its average congestion window size $\overline{W}_i$, with minimum $\frac{2}{3}\overline{W}_i$ and maximum $\frac{4}{3}\overline{W}_i$. Since the standard deviation of the uniform distribution is $\frac{1}{\sqrt{12}}$-th of its length, the standard deviation of a single window size σ_{W_i} is thus

$$\sigma_{W_i} = \frac{1}{\sqrt{12}}\left(\frac{4}{3}\overline{W}_i - \frac{2}{3}\overline{W}_i\right) = \frac{1}{3\sqrt{3}}\overline{W}_i$$

From Equation (5),

$$\overline{W}_i = \frac{\overline{W}}{n} = \frac{2\overline{T}_p \cdot C + \overline{Q}}{n} \leq \frac{2\overline{T}_p \cdot C + B}{n}.$$

For a large number of flows, the standard deviation of the sum of the windows, W, is given by

$$\sigma_W \leq \sqrt{n}\sigma_{W_i},$$

and so by Equation (5) the standard deviation of $Q(t)$ is

$$\sigma_Q = \sigma_W \leq \frac{1}{3\sqrt{3}} \frac{2\overline{T}_p \cdot C + B}{\sqrt{n}}.$$

Now that we know the distribution of the queue occupancy, we can approximate the link utilization for a given buffer size. Whenever the queue size is below a threshold, b, there is a risk (but not guaranteed) that the queue will go empty, and we will lose link utilization. If we know the probability that $Q < b$, then we have an upper bound on the lost utilization. Because Q has a normal distribution, we can use the error-function [6] to evaluate this probability. Therefore, we get the following lower bound for the utilization.

$$Util \geq erf\left(\frac{3\sqrt{3}}{2\sqrt{2}} \frac{B}{\frac{2\overline{T}_p \cdot C + B}{\sqrt{n}}}\right). \qquad (6)$$

Here are some numerical examples of utilization, using $n = 10000$.

Router Buffer Size	Utilization
$B = 1 \cdot \frac{2\overline{T}_p \cdot C}{\sqrt{n}}$	Util $\geq$ 98.99 %
$B = 1.5 \cdot \frac{2\overline{T}_p \cdot C}{\sqrt{n}}$	Util $\geq$ 99.99988 %
$B = 2 \cdot \frac{2\overline{T}_p \cdot C}{\sqrt{n}}$	Util $\geq$ 99.99997 %

This means that we can achieve full utilization with buffers that are the delay-bandwidth product divided by square-root of the number of flows, or a small multiple thereof. As the number of flows through a router increases, the amount of required buffer decreases.

This result has 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.

[6] A more precise result could be obtained by using Chernoff Bounds instead. We here present the Guassian approximation for clarity of presentation.

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 [4]. 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). In Section 5.3 we will see that our results hold for short non-TCP flows too (e.g. DNS queries, ICMP, etc.).

Consider again the topology in Figure 2 with multiple senders on separate access links. As has been widely reported from measurement, we assume that new short flows arrive according to a Poisson process [16, 17]. 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.

We will model bursts arriving from many different short flows at the bottleneck router. Some flows will be sending a burst of two packets, while others might be sending a burst of four, eight, or sixteen packets and so on. There will be a distribution of burst-sizes; and if there is a very large number of flows, we can consider each burst to be independent of the other bursts, even of the other bursts in the same flow. In this simplified model, the arrival process of bursts themselves (as opposed to the arrival of flows) can be assumed to be Poisson. One might argue that the arrivals are not Poisson as a burst is followed by another burst one RTT later. However under a low load and with many flows, the buffer will usually empty several times during one RTT and is effectively "memoryless" at this time scale.

For instance, let's assume we have arrivals of flows of a fixed length l. Because of the doubling of the burst lengths in each iteration of slow-start, each flow will arrive in n bursts of size

$$X_i = \{2, 4, ...2^{n-1}, R\},$$

where R is the remainder, $R = l \ mod \ (2^n - 1)$. Therefore, the bursts arrive as a Poisson process, and their lengths are i.i.d. random variables, equally distributed among $\{2, 4, ...2^{n-1}, R\}$.

The router buffer can now be modelled as a simple M/G/1 queue with a FIFO service discipline. In our case a "job" is a burst of packets, and the job size is the number of packets in a burst. The average number of jobs in an M/G/1 queue is known to be (e.g. [18])

$$E[N] = \frac{\rho}{2(1-\rho)} E[X^2].$$

Here ρ is the load on the link (the ratio of the amount of incoming traffic to the link capacity C), and $E[X]$ and $E[X^2]$ are the first two moments of the burst size. This model will overestimate the queue length because bursts are processed packet-by-packet while in an M/G/1 queue the job is only dequeued when the whole job has been processed. If the queue is busy, it will overestimate the queue length by half the average job size, and so

$$E[Q] = \frac{\rho}{2(1-\rho)} \frac{E[X^2]}{E[X]} - \rho \frac{E[X]}{2}$$

It is interesting to note that the average queue length is independent of the number of flows and the bandwidth of the link. It only depends on the load of the link and the length of the flows.

We can validate our model by comparing it with simulations. Figure 10 shows a plot of the average queue length for a fixed load and varying flow lengths, generated using $ns2$. Graphs for three different bottleneck link bandwidths (40, 80 and 200 Mb/s) are shown. The model predicts the relationship very closely. Perhaps surprisingly, the average queue length peaks when the probability of large bursts is highest, not necessarily when the average burst size is highest. For instance, flows of size 14 will generate a larger queue length than flows of size 16. This is because a flow of 14 packets generates bursts of $X_i = \{2, 4, 8\}$ and the largest burst of size 8 has a probability of $\frac{1}{3}$. A flow of 16 packets generates bursts of sizes $X_i = \{2, 4, 8, 4\}$, where the maximum burst length of 8 has a probability of $\frac{1}{4}$. As the model predicts, the bandwidth has no effect on queue length, and the measurements for 40, 80 and 200 Mb/s are almost identical. The gap between model and simulation is due to the fact that the access links before the bottleneck link space out the packets of each burst. Slower access links would produce an even smaller average queue length.

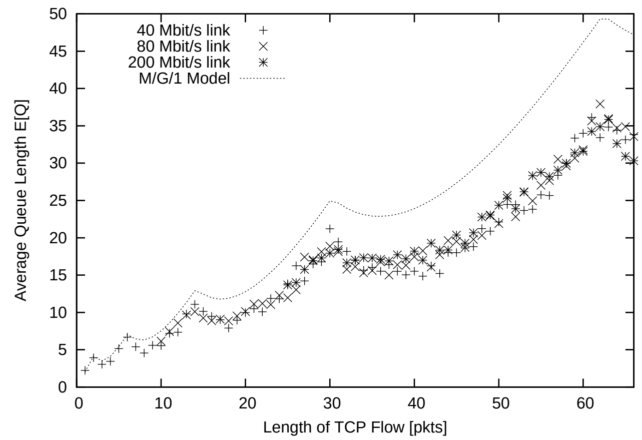

Figure 10: The average queue length as a function of the flow length for $\rho = 0.8$. The bandwidth has no impact on the buffer requirement. The upper bound given by the M/G/1 model with infinite access link speeds matches the simulation data closely.

To determine the buffer size we need the probability distribution of the queue length, not just its average. This is more difficult as no closed form result exists for a general M/G/1 queue length distribution. Instead, we approximate its tail using the effective bandwidth model [19], which tells us that the queue length distribution is

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

This equation is derived in Appendix C.

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 flows. Therefore, a backbone router serving highly aggregated traffic needs the same amount of buffering to absorb short-lived flows as a router serving only a few clients. Furthermore, because our analysis doesn't depend on the dynamics of slow-start (only on the burst-size distribution), it can be easily extended to short unresponsive UDP flows.

In practice, buffers can be made even slower. For our model and simulation we assumed access links that are faster than the bottleneck link. There is evidence [4, 20] 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[7].

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 two validation methods: simulation (using $ns2$), and a network of real Internet routers. The simulations give us the most flexibility: they allow us to explore a range of topologies, link speeds, numbers of flows and traffic mixes. On the other hand, the experimental network allows us to compare our results against a network of real Internet routers and TCP sources (rather than the idealized ones in $ns2$). It is less flexible and has a limited number of routers and topologies. Our results are limited to the finite number of different simulations and experiments we can run, and we can't prove that the results extend to any router in the Internet [21]. And so in Section 5.3 we examine the scope and limitations of our results, and what further validation steps are needed.

Our goal is to persuade a network operator to test our results by reducing the size of their router buffers by approximately 99%, and checking that the utilization and drop rates don't change noticeably. Until that time, we have to rely on a more limited set of experiments.

5.1 NS2 Simulations

We ran over 15,000 $ns2$ simulations, each one simulating several minutes of network traffic through a router to verify our model over a range of possible settings. We limit our simulations to cases where flows experience only one congested link. 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.

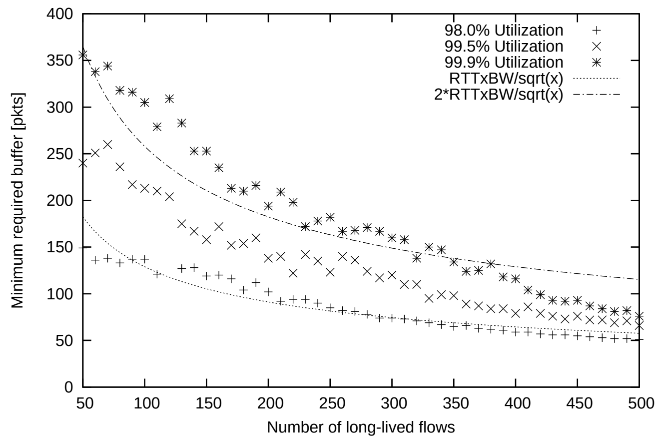

Figure 11: Minimum required buffer to achieve 98, 99.5 and 99.9 percent utilization for an OC3 (155 Mb/s) line with about 80ms average RTT measured with $ns2$ for long-lived TCP flows.

We assume that the router maintains a single FIFO queue, and drops packets from the tail only when the queue is full (i.e. the router does not use RED). Drop-tail is the most widely deployed scheme today. We expect the results to extend to RED, because our results rely on the desynchronization of TCP flows — something that is more likely with RED than with drop-tail. We used TCP Reno with a maximum advertised window size of at least 10000 bytes, and a 1500 or 1000 byte MTU. The average propagation delay of a TCP flow varied from 25ms to 300ms.

5.1.1 Simulation Results for Long-lived TCP Flows

Figure 11 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 – and as can be seen in the graph – when the number of flows exceeds 250, the model holds well. We found that in order to attain 99.9% utilization, we needed buffers twice as big; just as the model predicts.

We found similar results to hold over a wide range of settings whenever there are a large number of flows, there is little or no synchronization, and the average congestion window is above two. If the average congestion window is smaller than two, flows encounter frequent timeouts and more buffering is required [22].

In our simulations and experiments we looked at three other commonly used performance metrics, to see their effect on buffer size:

[7]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.

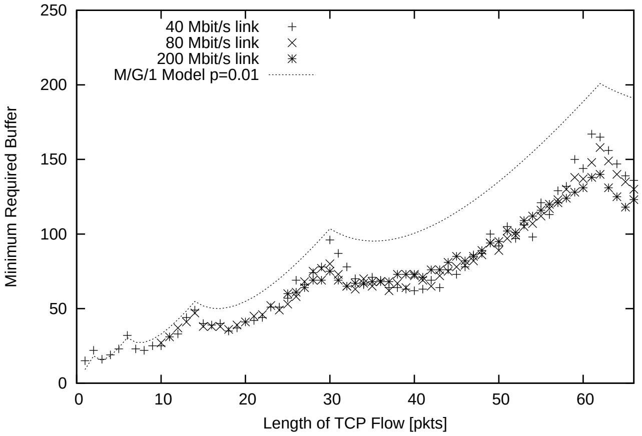

Figure 12: The minimum required buffer that increases the Average Flow Completion Time (AFCT) by not more than 12.5% vs infinite buffers for short flow traffic.

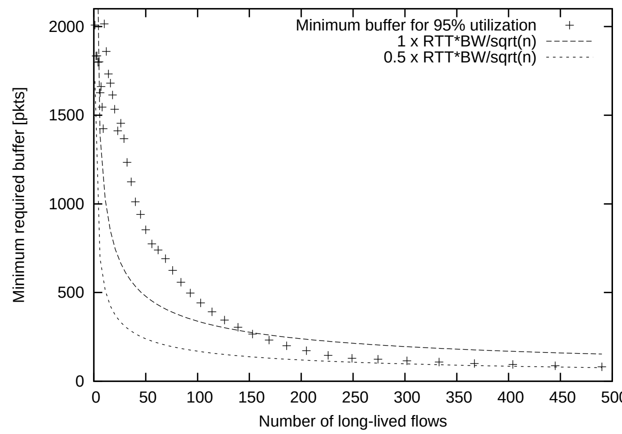

Figure 13: Buffer requirements for traffic mix with different flow lengths, measured from a $ns2$ simulation.

- **Packet loss** If we reduce buffer size, we can expect packet loss to increase. 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 [22]). From Equation 1, we know the sum of the window sizes is $RTT \times C + B$. If B is made very small, then the window size halves, increasing loss by a factor of four. This is not necessarily a problem. TCP uses loss as a useful signal to indicate congestion; and TCP's loss rate is very low (one packet per multiple round-trip times). More importantly, as we show below, flows complete sooner with smaller buffers than with large buffers. One might argue that other applications that do not use TCP are adversely affected by loss (e.g. online gaming or media streaming), however these applications are typically even more sensitive to queueing delay.

- **Goodput** While 100% utilization is achievable, goodput is always below 100% because of retransmissions. With increased loss, goodput is reduced, but by a very small amount, as long as we have buffers equal or greater than $\frac{RTT \times C}{\sqrt{n}}$.

- **Fairness** Small buffers reduce fairness among flows. First, a smaller buffer means all flows have a smaller round-trip time, and their sending rate is higher. With large buffers, all round-trip times increase and so the relative difference of rates will decrease. While over-buffering would increase fairness, it also increases flow completion times for all flows. A second effect is that timeouts are more likely with small buffers. We did not investigate how timeouts affect fairness in detail, however in our $ns2$ simulations it seemed to be only minimally affected by buffer size.

5.1.2 Short Flows

We will use the commonly used 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). We are interested in the tradeoff between buffer size and AFCT. In general, for a link with a load $\rho \ll 1$, the AFCT is minimized when we have infinite buffers, because there will be no packet drops and therefore no retransmissions.

We take as a benchmark the AFCT with infinite buffers, then find the increase in AFCT as a function of buffer size. For example, Figure 12 shows the minimum required buffer so that the AFCT is increased by no more than 12.5%. Experimental data is from $ns2$ 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 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 practice, routers transport a mix of short and long flows; the exact distribution of flow lengths varies from network to network, and over time. This makes it impossible to measure every possible scenario, and so we give a general idea of how the flow mix influences the buffer size. The good news is that long flows dominate, and a buffer size of $\overline{RTT} \times C/\sqrt{n}$ will suffice when we have a large number of flows. Better still, we'll see that the AFCT for the short flows is lower than if we used the usual rule-of-thumb.

In our experiments the short flows always slow-down the long flows because of their more aggressive multiplicative increase, causing the long flows to reduce their window-size. Figures 13 and 14 show a mix of flows over a 400 Mb/s link. The total bandwidth of all arriving short flows is about 80 Mb/s or 20% of the total capacity. The number of long flows was varied from 1 to 500. During the time of the experiment, these long flows attempted to take up all the bandwidth left available by short flows. In practice, they never consumed more than 80% of the bandwidth as the rest would be taken by the more aggressive short flows.

As we expected, with a small number of flows, the flows are partially synchronized. With more than 200 long-lived flows, the synchronization has largely disappeared. The

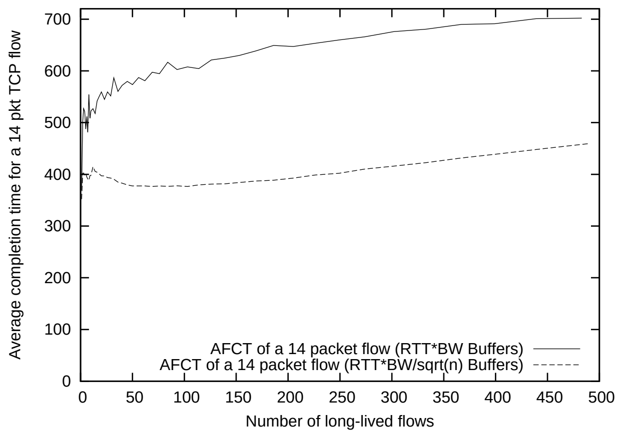

Figure 14: Average flow completion times with a buffer size of $(\overline{RTT} \times C)/\sqrt{n}$, compared with a buffer size $\overline{RTT} \times C$.

graph shows that the long flows dominate the flow size. If we want 95% utilization, then we need a buffer close to $\overline{RTT} \times C/\sqrt{n}$. [8] This means we can ignore the short flows when sizing the buffer. Of course, this doesn't tell us how the short-lived flows are faring — they might be shutout by the long flows, and have increased AFCT. But Figure 14 shows that this is not the case. 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.

5.1.4 Pareto Distributed Flow Lengths

Real network traffic contains a broad range of flow lengths. The flow length distribution is known to be heavy tailed [4] and in our experiments we used Pareto distributions to model it. As before, we define short flows to be those still in slow start.

For Pareto distributed flows on a congested router (i.e. $\rho \approx 1$), the model holds and we can achieve close to 100% throughput with buffers of a small multiple of $(\overline{RTT} \times C)/\sqrt{n}$.[9] For example in an $ns2$ simulation of a 155 Mb/s line, $\bar{R}TT \approx 100ms$) we measured 100-200 simultaneous flows and achieved a utilization of 99% with a buffer of only 165 packets.

It has been pointed out [23] that in a network with low latency, fast access links and no limit on the TCP window size, there would be very few concurrent flows. In such a network, a single very heavy flow could hog all the bandwidth for a short period of time and then terminate. But this is unlikely in practice, unless an operator allows a single user to saturate their network. And so long as backbone networks are orders of magnitude faster than access networks, few users will be able to saturate the backbone anyway. Even if they could, TCP is not capable of utilizing a link quickly due to its additive increase behavior above a certain window size. Traffic transported by high-speed routers on commercial networks today [4, 24] has 10's of 1000's of concurrent flows and we believe this is unlikely to change in the future.

An uncongested router (i.e. $\rho \ll 1$) can be modeled using the short-flow model presented in section 4 which often leads to even lower buffer requirements. Such small buffers may penalize very long flows as they will be forced into congestion avoidance early even though bandwidth is still available. If we want to allow a single flow to take up to $1/n$ of the bandwidth, we always need buffers of $(\overline{RTT} \times C)/\sqrt{n}$, even at a low link utilization.

We found that our general result holds for different flow length distributions if at least 10% of the traffic is from long flows. Otherwise, short flow effects sometimes dominate. Measurements on commercial networks [4] suggest that over 90% of the traffic is from long flows. It seems safe to assume that long flows drive buffer requirements in backbone routers.

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 [25] 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 [26] on Linux and BSD machines, and aggregated using a second Cisco GSR 12410 router with Gigabit Ethernet line cards. Utilization measurements were done using SNMP on the receiving end, and compared to Netflow records [27].

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 15: Comparison of our model, $ns2$ simulation and experimental results for buffer requirements of a Cisco GSR 12410 OC3 linecard.

5.2.1 Long Flows

Figure 15 shows the results of measurements from the

[8] Here n is the number of active long flows at any given time, not the total number of flows.

[9] The number of long flows n for sizing the buffer was found by measuring the number of flows in congestion avoidance mode at each instant and visually selecting a robust minimum.

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. We subtracted the size of the internal FIFO on the line-card (see Section 5.2.2). *Model* is the lower-bound on the utilization predicted by the model. *Sim.* and *Exp.* are the utilization as measured by a simulation with $ns2$ 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 $\pm 0.1\%$. $ns2$ 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).

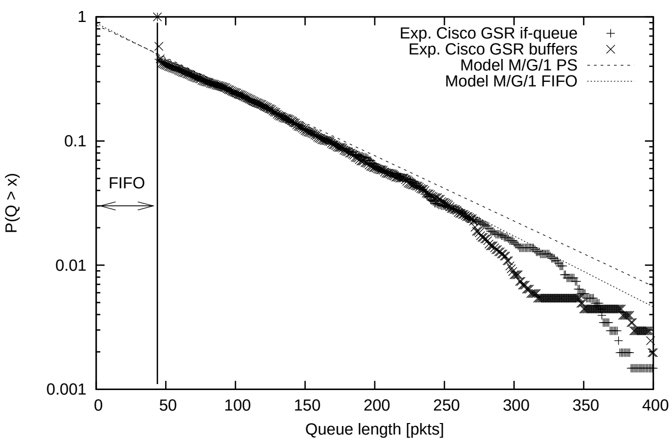

Figure 16: Experimental, Simulation and Model prediction of a router's queue occupancy for a Cisco GSR 12410 router.

5.2.2 Short Flows

In Section 4 we used an M/G/1 model to predict the buffer size we would need for short-lived, bursty TCP flows. To verify our model, we generated lots of short-lived flows and measured the probability distribution of the queue length of the GSR 12410 router. Figure 16 shows the results and the comparison with the model, which match remarkably well.[10]

5.3 Scope of our Results and Future Work

The results we present in this paper assume only a single point of congestion on a flow's path. We don't believe our results would change much if a percentage of the flows experienced congestion on multiple links, however we have not investigated this. A single point of congestion means there is no reverse path congestion, which would likely have an effect on TCP-buffer interactions [28]. With these assumptions, our simplified network topology is fairly general. In an arbitrary network, flows may pass through other routers before and after the bottleneck link. However, as we assume only a single point of congestion, no packet loss and little traffic shaping will occur on previous links in the network.

We focus on TCP as it is the main traffic type on the internet today. Constant rate UDP sources (e.g. online games) or single packet sources with Poisson arrivals (e.g. DNS) can be modelled using our short flow model and the results for mixes of flows still hold. But to understand traffic composed mostly of non-TCP packets would require further study.

Our model assumes there is no upper bound on the congestion window. In reality, TCP implementations have maximum window sizes as low as 6 packets [29]. Window sizes above 64kByte require use of a scaling option [30] which is rarely used. Our results still hold as flows with limited window sizes require even smaller router buffers [1].

We did run some simulations using Random Early Detection [12] and this had an effect on flow synchronization for a small number of flows. Aggregates of a large number (> 500) of flows with varying RTTs are not synchronized and RED tends to have little or no effect on buffer requirements. However early drop can slightly increase the required buffer since it uses buffers less efficiently.

There was no visible impact of varying the latency other than its direct effect of varying the bandwidth-delay product.

Congestion can also be caused by denial of service (DOS) [31] attacks that attempt to flood hosts or routers with large amounts of network traffic. Understanding how to make routers robust against DOS attacks is beyond the scope of this paper, however we did not find any direct benefit of larger buffers for resistance to DOS attacks.

6. RELATED WORK

Villamizar and Song report the $RTT \times BW$ rule in [1], in which the authors measure link utilization of a 40 Mb/s network with 1, 4 and 8 long-lived TCP flows for different buffer sizes. They find that for FIFO dropping discipline and very large maximum advertised TCP congestion windows it is necessary to have buffers of $RTT \times C$ to guarantee full link utilization. We reproduced their results using $ns2$ and can confirm them for the same setup. With such a small number of flows, and large congestion windows, the flows are almost fully synchronized and have the same buffer requirement as a single flow.

Morris [32] investigates buffer requirements for up to 1500 long-lived flows over a link of 10 Mb/s with 25ms latency. He concludes that the minimum amount of buffering needed is a small multiple of the number of flows, and points out that for a bandwidth-delay product of 217 packets, each flow has only a fraction of a packet in transit at any time. Many flows are in timeout, which adversely effects utilization and fairness. We repeated the experiment in $ns2$ and obtained similar results. However for a typical router used by a carrier or ISP, this has limited implications. Users with fast access links will need several packets outstanding to achieve adequate performance. Users with very slow access links (e.g. 32kb/s modem users or 9.6kb/s GSM mobile access) need additional buffers in the network so they have sufficient

[10] The results match very closely if we assume the router under-reports the queue length by 43 packets. We learned from the manufacturer that the line-card has an undocumented 128kByte transmit FIFO. In our setup, 64 kBytes are used to queue packets in an internal FIFO which, with an MTU of 1500 bytes, accounts exactly for the 43 packet difference.

packets outstanding. However this additional buffer should be at the ends of the access link, e.g. the modem bank at the local ISP, or GSM gateway of a cellular carrier. We believe that overbuffering the core router that serves different users would be the wrong approach, as overbuffering increases latency for everyone and is also difficult to implement at high line-rates. Instead the access devices that serve slow, last-mile access links of under 1Mb/s should continue to include a few packets worth of buffering for each link.

With line speeds increasing and the MTU size staying constant, we would also assume this issue to become less relevant in the future.

Avrachenkov et al [33] present a fixed point model for utilization (for long flows) and flow completion times (for short flows). They model short flows using an M/M/1/K model that only accounts for flows but not for bursts. In their long flow model they use an analytical model of TCP that is affected by the buffer through the RTT. As the model requires fixed point iteration to calculate values for specific settings and only one simulation result is given, we can not directly compare their results with ours.

Garetto and Towsley [34] describe a model for queue lengths in routers with a load below one that is similar to our model in section 4. The key difference is that the authors model bursts as batch arrivals in an $M^{[k]}/M/1$ model (as opposed to our model that models bursts by varying the job length in a M/G/1 model). It accommodates both slow-start and congestion avoidance mode, however it lacks a closed form solution. In the end the authors obtain queue distributions that are very similar to ours.

7. CONCLUSION

We believe that the buffers in backbone routers are much larger than they need to be — possibly by two orders of magnitude. If our results are right, they have consequences for the design of backbone routers. While we have evidence that buffers can be made smaller, we haven't tested the hypothesis in a real operational network. It is a little difficult to persuade the operator of a functioning, profitable network to take the risk and remove 99% of their buffers. But that has to be the next step, and we see the results presented in this paper as a first step towards persuading an operator to try it.

If an operator verifies our results, or at least demonstrates that much smaller buffers work fine, it still remains to persuade the manufacturers of routers to build routers with fewer buffers. In the short-term, this is difficult too. In a competitive market-place, 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.

8. ACKNOWLEDGMENTS

The authors would like to thank Joel Sommers and Professor Paul Barford from the University of Wisconsin-Madison for setting up and running the measurements on a physical router in their WAIL testbed; and Sally Floyd and Frank Kelly for useful disucssions. Matthew Holliman's feedback on long flows led to the central limit theorem argument.

9. REFERENCES

[1] C. Villamizar and C. Song. High performance tcp in ansnet. *ACM Computer Communications Review*, 24(5):45–60, 1994 1994.

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

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

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

[5] D. Ferguson. [e2e] Queue size of routers. Posting to the end-to-end mailing list, January 21, 2003.

[6] S. H. Low, F. Paganini, J. Wang, S. Adlakha, and J. C. Doyle. Dynamics of tcp/red and a scalable control. In *Proceedings of IEEE INFOCOM 2002*, New York, USA, June 2002.

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

[8] C. Dovrolis. [e2e] Queue size of routers. Posting to the end-to-end mailing list, January 17, 2003.

[9] 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.

[10] The network simulator - ns-2. http://www.isi.edu/nsnam/ns/.

[11] Y. J. Anna Gilbert and N. McKeown. Congestion control and periodic behavior. In *LANMAN Workshop*, March 2001.

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

[13] L. Zhang and D. D. Clark. Oscillating behaviour of network traffic: A case study simulation. *Internetworking: Research and Experience*, 1:101–112, 1990.

[14] L. Qiu, Y. Zhang, and S. Keshav. Understanding the performance of many tcp flows. *Comput. Networks*, 37(3-4):277–306, 2001.

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

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

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

[18] R. W. Wolff. *Stochastic Modelling and the Theory of Queues*, chapter 8. Prentice Hall, October 1989.

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

[20] 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.

[21] S. Floyd and V. Paxson. Difficulties in simulating the internet. *IEEE/ACM Transactions on Networking*, February 2001.

[22] R. Morris. Scalable tcp congestion control. In *Proceedings of IEEE INFOCOM 2000*, Tel Aviv, USA, March 2000.

[23] S. B. Fredj, T. Bonald, A. Proutière, G. Régnié, and J. Roberts. Statistical bandwidth sharing: a study of congestion at flow level. In *Proceedings of SIGCOMM 2001*, San Diego, USA, August 2001.

[24] Personal communication with stanford networking on characteristics of residential traffic.

[25] Cisco 12000 series routers. http://www.cisco.com/en/US/products/hw/routers/ps167/.

[26] J. Sommers, H. Kim, and P. Barford. Harpoon: A flow-level traffic generator for router and network test. In *Proceedings of ACM SIGMETRICS*, 2004. (to appear).

[27] I. Cisco Systems. Netflow services solution guido, July 2001. http://www.cisco.com/.

[28] L. Zhang, S. Shenker, and D. D. Clark. Observations on the dynamics of a congestion control algorithm: The effects of two-way traffic. In *Proceedings of ACM SIGCOMM*, pages 133–147, September 1991.

[29] Microsoft. Tcp/ip and nbt configuration parameters for windows xp. Microsoft Knowledge Base Article - 314053, November 4, 2003.

[30] K. McCloghrie and M. T. Rose. RFC 1213: Management information base for network management of TCP/IP-based internets:MIB-II, March 1991. Status: STANDARD.

[31] A. Hussain, J. Heidemann, and C. Papadopoulos. A framework for classifying denial of service attacks. In *Proceedings of ACM SIGCOMM*, August 2003.

[32] R. Morris. Tcp behavior with many flows. In *Proceedings of the IEEE International Conference on Network Protocols*, Atlanta, Georgia, October 1997.

[33] K. Avrachenkov, U. Ayesta, E. Altman, P. Nain, and C. Barakat. The effect of router buffer size on the tcp performance. In *Proceedings of the LONIIS Workshop on Telecommunication Networks and Teletraffic Theory*, pages 116–121, St.Petersburg, Russia, Januar 2002.

[34] M. Garetto and D. Towsley. Modeling, simulation and measurements of queueing delay under long-tail internet traffic. In *Proceedings of SIGMETRICS 2003*, San Diego, USA, June 2003.

[35] W. R. Stevens. *TCP Illustrated, Volume 1 - The Protocols*. Addison Wesley, 1994.

APPENDIX

A. SUMMARY OF TCP BEHAVIOR

Let's briefly review the basic TCP rules necessary to understand this paper. We will present a *very* simplified description of TCP Reno. For a more detailed and more comprehensive presentation, please refer to e.g. [35].

The sending behavior of the TCP sender is determined by its state (either slow-start or congestion avoidance) and the congestion window, W. Initially, the sender is in slow-start and W is set to two. Packets are sent until there are W packets outstanding. Upon receiving a packet, the receiver sends back an acknowledgement (ACK). The sender is not allowed to have more than W outstanding packets for which it has not yet received ACKs.

While in slow-start, the sender increases W for each acknowledgement it receives. As packets take one RTT to travel to the receiver and back this means W effectively doubles every RTT. If the sender detects a lost packet, it halves its window and enters congestion avoidance. In congestion avoidance, it only increases W by $1/W$ for each acknowledgment it receives. This effectively increases W by one every RTT. Future losses always halve the window but the flow stays in CA mode.

`Slow-Start:`	`No loss:`	$W_{new} = 2W_{old}$
	`Loss:`	$W_{new} = \frac{W_{old}}{2}$, `enter CA`
`CA:`	`No loss:`	$W_{new} = W_{old} + 1$
	`Loss:`	$W_{new} = \frac{W_{old}}{2}$

In both states, if several packets are not acknowledged in time, the sender can also trigger a timeout. It then goes back to the slow-start mode and the initial window size. Note that while in congestion avoidance, the window size typically exhibits a sawtooth pattern. The window size increases linearly until the first loss. It then sharply halves the window size, and pauses to receive more ACKs (because the window size has halved, the number of allowed outstanding packets is halved, and so the sender must wait for them to be acknowledged before continuing). The sender then starts increasing the window size again.

The sending rate of the TCP flow is the window size divided by the round trip time:

$$R = \frac{W}{RTT} = \frac{W}{2T_p + T_Q} \tag{7}$$

where T_p is the one-way, end-to-end propagation delay of a packet and T_Q is the queueing delay it encounters at the router (we assume a single point of congestion and no queueing delay for ACKs).

B. BEHAVIOR OF A SINGLE LONG TCP FLOW

Consider a single long TCP flow, going through a single router of buffer size equal to the delay-bandwidth product (Figure 2). It is often assumed that the round-trip time of such a flow is nearly constant. Using this assumption, since the window size is incremented at each round-trip time when there is no loss, the increase in window size should be linear with time. In other words, the sawtooth pattern of the window size should be triangular.

However, the round-trip time is *not* constant, and therefore the sawtooth pattern is not exactly linear, as seen in

Figure 4. This becomes especially noticeable as link capacities, and therefore delay-bandwidth products, become larger and larger. Note that if the sawtooth was linear and the round-trip time was constant, the queue size increase would be parabolic by integration. On the contrary, we will see below that it is concave and therefore sub-linear.

Let's introduce a fluid model for this non-linear sawtooth behavior. Let $Q(t)$ denote the packet queue size at time t, with $Q(0) = 0$. As defined before, $2T_p$ is the round-trip propagation time, and C is the router output link capacity. For this fluid model, assume that the propagation time is negligible compared to the time it takes to fill up the buffer. Then, in the regularly increasing part of the sawtooth, the window size increases by 1 packet every RTT (Appendix A), and therefore

$$\dot{W}(t) = \frac{1}{RTT} = \frac{1}{2T_p + Q(t)/C}.$$

In addition the window size is linked to the queue size by

$$W(t) = 2T_pC + Q(t).$$

Joining both equations yields

$$\dot{W}(t) = \frac{C}{W(t)}. \tag{8}$$

Hence

$$\frac{d\left(\frac{W^2(t)}{2}\right)}{dt} = C$$

$$W^2(t) = 2Ct + W(0).$$

When the sawtooth pattern starts, $Q(0) = 0$, thus

$$W(0) = 2T_pC + Q(0) = 2T_pC.$$

We finally get a simple model for the increase of the window size in the sawtooth,

$$W(t) = \sqrt{2Ct + (2T_pC)^2}. \tag{9}$$

Therefore, as $t \gg T_p$, the queue size behaves in a square-root fashion, instead of a linear one, as commonly believed. The queue size behaves in a similar fashion,

$$Q(t) = W(t) - 2T_pC = \sqrt{2Ct + (2T_pC)^2} - 2T_pC. \tag{10}$$

As an application, it is now possible to determine the periodicity T of the sawtooth pattern. The maximum buffer size B is reached when $Q(T) = B$, i.e.

$$T = \frac{B^2}{2C} + 2T_pB.$$

When the buffer size is equal to the bandwidth-delay product ($B = 2T_pC$), we get

$$T = 1.5\frac{B^2}{C}.$$

For instance, in Figure 4, the modelled sawtooth period is $T = 1.5\frac{142^2}{1000} = 30.2$, which fits the observed value.

C. QUEUE DISTRIBUTION USING EFFECTIVE BANDWIDTH

The goal of this appendix is to model the behavior of the router queue when there are several senders with short TCP flows. As explained in Section 4, each of these short TCP flows sends a given number of bursts. We assume that all these bursts are independent, and by considering all the independent bursts of all the flows, we model the arrival process to the router as the cumulative arrival of these independent bursts. Let $A(t)$ denote the cumulative amount of traffic arrived to the router at time t, $N(t)$ the number of bursts arrived before t, and X_i the size of each of these bursts. Then:

$$A(t) = \sum_{i=1}^{N(t)} X_i.$$

We assume that the bursts arrive as a Poisson process $N(t)$ of rate ν, and that their burst size follows a distribution function F. The queue is serviced with a capacity C. Therefore, we use an *M/G/1 model* for the job size in the router queue. Of course, the traffic arrival rate λ is the product of the burst arrival rate by the average burst size:

$$\lambda = \nu E[X].$$

The effective bandwidth theory describes the characteristics of a traffic source, and in many cases can be a powerful tool to derive properties that are otherwise hard to compute. For more information we refer the reader to [19], from which most of the results below are adapted. Consider a cumulative traffic arrival function $A(t)$, where $A(t)$ has stationary increments. Its effective bandwidth is defined as:

$$\alpha(s,t) = \frac{1}{st} log E\left[e^{sA(t)}\right].$$

In our case, $A(t)$ has i.i.d. increments. It is a special case of Lévy process, and its effective bandwidth has a simpler form (section 2.4 of [19]):

$$\alpha(s) = \frac{1}{s}\int (e^{sx} - 1)\nu dF(x) = \frac{\lambda}{sE[X]}E\left[e^{sX} - 1\right]. \tag{11}$$

We can now use Cramér's estimate to model the tail of the distribution of the queue length Q (section 3.2 of [19]). This estimate assumes an infinite buffer size. For Cramér's estimate there has to exist a constant κ such that the effective bandwidth is the link capacity:

$$\alpha(\kappa) = C \tag{12}$$

Also the derivative $\alpha'(s)$ has to exist at κ. Both are satisfied in our case as $\lambda < C$ and $\alpha(s)$ is differentiable for any $s > 0$. Cramer's estimate is given by

$$P(Q \geq b) \approx \frac{C - \alpha(0)}{\kappa\alpha'(\kappa)} e^{-\kappa b} \tag{13}$$

To calculate κ we need to simplify Equation (12). We approximate the exponential function by its Taylor Series.

$$e^{sX} = 1 + sX + \frac{s^2}{2}X^2 + \frac{s^3}{6}X^3 + O(X^4) \tag{14}$$

We present solutions for the 2nd and 3rd order approximation. In our experience the 2nd order approximation is sufficient to estimate the required buffer.

Second Order Approximation We substitute Equation (14) into Equation (11) and obtain for the 2nd order approximation:

$$\begin{aligned}\alpha(s) &= \frac{\lambda}{sE[X]}E\left[1+sX+\frac{s^2}{2}X^2-1\right]\\ &= \lambda+\frac{\lambda s}{2}\frac{E[X^2]}{E[X]}\\ \alpha'(s) &= \frac{\lambda}{2}\frac{E[X^2]}{E[X]}\end{aligned}$$

The load ρ is defined as the ratio of the arrival rate λ by the capacity C:

$$\rho=\frac{\lambda}{C}.$$

Using Equation (12), we get:

$$\kappa=\frac{2(1-\rho)}{\rho}\frac{E[X]}{E[X^2]}$$

Finally, for the queue size distribution we obtain

$$P(Q\geq b) \approx \frac{\frac{\lambda}{\rho}-\lambda}{\frac{2(1-\rho)}{\rho}\frac{E[X]}{E[X^2]}\frac{\lambda}{2}\frac{E[X^2]}{E[X]}}e^{-b\kappa}=e^{-b\kappa}$$

Third Order Approximation. We substitute Equation (14) into Equation (11) and obtain for the 3rd order approximation:

$$\begin{aligned}\alpha(s) &= \frac{\lambda}{sE[X]}E\left[1+sX+\frac{s^2}{2}X^2+\frac{s^3}{6}X^3-1\right]\\ &= \lambda+\frac{\lambda s}{2}\frac{E[X^2]}{E[X]}+\frac{\lambda s^2}{6}\frac{E[X^3]}{E[X]}\\ \alpha'(s) &= \frac{\lambda}{2}\frac{E[X^2]}{E[X]}+\frac{\lambda s}{3}\frac{E[X^3]}{E[X]}\end{aligned}$$

Again we now need to solve $\alpha(\kappa)=C$ where $\rho=\frac{\lambda}{C}$. We obtain the quadratic equation.

$$\kappa^2+\kappa\frac{3E[X^2]}{E[X^3]}-\frac{6(1-\rho)}{\rho}\frac{E[X]}{E[X^3]}=0$$

and for κ

$$\kappa=-\frac{3}{2}\frac{E[X^2]}{E[X^3]}+\sqrt{\left(\frac{3}{2}\frac{E[X^2]}{E[X^3]}\right)^2+\frac{6(1-\rho)}{\rho}\frac{E[X]}{E[X^3]}}$$

For the queue size distribution we find.

$$\begin{aligned}P(Q\geq b) &\approx \frac{\frac{\lambda}{\rho}-\lambda}{\frac{\lambda\kappa}{2}\frac{E[X^2]}{E[X]}+\frac{\lambda\kappa^2}{3}\frac{E[X^3]}{E[X]}}e^{-b\kappa}\\ &= \frac{1-\rho}{\rho}\frac{E[X]}{\frac{\kappa}{2}E[X^2]+\frac{\kappa^2}{3}E[X^3]}e^{-b\kappa}\end{aligned}$$