Routers with Very Small Buffers

Mihaela Enachescu*, Yashar Ganjali†, Ashish Goel‡, Nick McKeown†, and Tim Roughgarden*
*Department of Computer Science, Stanford University
{mihaela, tim}@cs.stanford.edu
†Department of Electrical Engineering, Stanford University
{yganjali, nickm}@stanford.edu
‡Dept of Management Science and Engineering, Stanford University
ashishg@stanford.edu

Abstract— **Internet routers require buffers to hold packets during times of congestion. The buffers need to be fast, and so ideally they should be small enough to use fast memory technologies such as SRAM or all-optical buffering. Unfortunately, a widely used rule-of-thumb says we need a bandwidth-delay product of buffering at each router so as not to lose link utilization. This can be prohibitively large. In a recent paper, Appenzeller *et al.* challenged this rule-of-thumb and showed that for a backbone network, the buffer size can be divided by $\sqrt{N}$ without sacrificing throughput, where N is the number of flows sharing the bottleneck. In this paper, we explore how buffers in the backbone can be significantly reduced even more, to as little as a few dozen packets, if we are willing to sacrifice a small amount of link capacity. We argue that if the TCP sources are not overly bursty, then fewer than twenty packet buffers are sufficient for high throughput. Specifically, we argue that $O(\log W)$ buffers are sufficient, where W is the window size of each flow. We support our claim with analysis and a variety of simulations. The change we need to make to TCP is minimal—each sender just needs to pace packet injections from its window. Moreover, there is some evidence that such small buffers are sufficient even if we don't modify the TCP sources so long as the access network is much slower than the backbone, which is true today and likely to remain true in the future.**

We conclude that buffers can be made small enough for all-optical routers with small integrated optical buffers.

I. Motivation and Introduction

Until quite recently, Internet routers were widely believed to need large buffers. Commercial routers today have huge packet buffers, often storing millions of packets, under the assumption that large buffers lead to good statistical multiplexing and hence efficient use of expensive long-haul links. A widely-used rule-of-thumb states that, because of the dynamics of TCP's congestion control mechanism, a router needs a bandwidth-delay product of buffering, $B = \overline{\mathrm{RTT}} \times C$, in order to fully utilize bottleneck links [5], [6], [16]. Here, C is the capacity of the bottleneck link, B is the size of the buffer in the bottleneck router, and $\overline{\mathrm{RTT}}$ is the effective round-trip propagation delay of a TCP flow through the bottleneck link. Recently, Appenzeller *et al.* proposed using the rule $B = \overline{\mathrm{RTT}} \times C/\sqrt{N}$ instead, where N is the number of flows through the bottleneck link [3]. In a backbone network today, N is often in the thousands or the tens of thousands, and so the sizing rule $B = \overline{\mathrm{RTT}} \times C/\sqrt{N}$ results in significantly fewer buffers.

In this paper, we explore if and how we could build a network with much smaller buffers still—perhaps with only a few dozen packet buffers in each router, and perhaps at the expense of 100% link utilization. While this is an interesting intellectual exercise in its own right, there would be practical consequences if it were possible.

First, it could facilitate the building of all-optical routers. With recent advances [8], [9], [13], it is now possible to perform all-optical switching, opening the door to routers with huge capacity and lower power than electronic routers. Recent advances in technology make possible optical FCFS packet buffers that can hold a few dozen packets in an integrated opto-electronic chip [13]. Larger all-optical buffers remain infeasible, except with unwieldy spools of optical fiber (that can only implement delay lines, not true FCFS packet buffers). We are interested in exploring the feasibility of an operational all-optical network with just a few dozen optical packet buffers in each router.

Second, if big electronic routers required only a few dozen packet buffers, it could reduce their complexity, making them easier to build and easier to scale. A typical

An earlier, shorter, incomplete, invited and unreviewed version of this paper appeared in ACM CCR, July 2005. The editor encouraged us to submit to Infocom so that a complete, reviewed version would be available to the community.

This work was supported under DARPA/MTO DOD-N award no. W911NF-04-0001/KK4118 (LASOR PROJECT) and the Buffer Sizing Grant no. W911NF-05-1-0224. Ashish Goel's work was also supported by an NSF career grant and an Alfred P. Sloan faculty fellowship, and the Tim Roughgarden's work was also supported in part by ONR grant N00014-04-1-0725.

10Gb/s router linecard today contains about one million packet buffers, using many external DRAM chips. The board space the DRAMs occupy, the pins they require, and the power they dissipate all limit the capacity of the router [3]. If a few dozen packet buffers suffice, then packet buffers could be incorporated inside the network processor (or ASIC) in a small on-chip SRAM; in fact, the buffers would only occupy a tiny portion of the chip. Not only would external memories be removed, but it would allow the use of fast on-chip SRAM, which scales in speed much faster than DRAM.

Our main result is that minor modifications to TCP would indeed allow us to reduce buffer-sizes to dozens of packets with the expense of slightly reduced link utilization. We obtain this result in a succession of steps. We will start by adopting two strong assumptions: (1) That we could modify the way packets are transmitted by TCP senders, and (2) That the network is over-provisioned. However, we will soon relax these assumptions.

We start by asking the following question: What if we kept the AIMD (Additive Increase Multiplicative Decrease) dynamics of TCP window control, but changed the TCP transmission scheme to "space out" packet transmissions from the TCP sender, thereby making packet arrivals less bursty? We assume that each TCP flow determines its window size using the standard TCP AIMD scheme. However, if the current window size at time t is W and the current round-trip estimate is RTT, then we assume the TCP sender sends according to a Poisson process of rate W/RTT at time t. This results in the same average rate as sending W packets per RTT. While this is a slightly unrealistic assumption (it can result in the window size being violated and so might alter TCP behavior in undesirable ways), this scenario yields important clues about the feasibility of very small buffers.

We are also going to assume that the network is over-provisioned—even if each flow is sending at its maximum window size, the network will not be congested.[1] Under these assumptions, we show that a buffer size of $O(\log W_{\max})$ packets is sufficient to obtain close to peak throughput, where $W_{\max}$ is the maximum window size in packets. Some elements of the proof are interesting in their own right.[2] The exact scenario is explained in Section II and the proof itself is in Appendix I.

To get some feel for these numbers, consider the scenario where 1000 flows share a link of capacity 10Gbps. Assume that each flow has an RTT of 100ms, a maximum window size of 64KB, and a packet size of 1KB. The peak rate is roughly 5Gbps. The bandwidth-delay product rule-of-thumb suggests a buffer size of 125MB, or around 125,000 packets. The $\overline{\text{RTT}} \times C/\sqrt{N}$ rule suggests a buffer size of around 3950 packets. Our analysis suggests a buffer size of twelve packets plus some small additive constant, which brings the buffer size down to the realm where optical buffers can be built in the near future.

We then systematically remove the two assumptions we made above, using a combination of simulations and analysis. We first tackle the assumption that TCP sends packets in a locally Poisson fashion. Intuitively, sending packets at fixed (rather than random) intervals should give us the same benefit (or better) as sending packets at a Poisson rate. Accordingly, we study the more reasonable case where the TCP sending agent "paces" its packets deterministically over an entire RTT. Paced TCP has been studied before [2], and does not suffer from the problem of overshooting the window size. We perform an extensive simulation study of paced TCP with small buffers. When the network is over-provisioned, the performance of paced TCP closely mirrors our analytical bound of $O(\log W_{\max})$ for Poisson sources. This holds for a wide range of capacities and number of flows, and not just in the regime where one might expect the aggregate arrival process at the router to resemble a Poisson process [4]. These results are presented in Section III. In Appendix II, we provide additional intuition for this result: if many paced flows are superimposed after a random jitter, then the packet drop probability is as small as with Poisson traffic.

The next assumption we attempt to remove is that of the network being over-provisioned. We consider a single bottleneck link, and assume that if each flow were to send at its maximum window size, then the link would be severely congested. In our simulations (presented in Section IV), Paced TCP results in high throughput (around 70-80%) with the relatively small buffers (10-20) predicted by the simple Poisson-transmissions analysis. While we have not been able to extend our formal analysis to the under-provisioned network case, some analytical intuition can also be obtained: if we assume

[1]This assumption is less restrictive than it might appear. Current TCP implementations usually cap window sizes at 32 KB or 64 KB [10], and it is widely believed that there is no congestion in the core of the Internet. All optical networks, in particular, are likely to be significantly over-provisioned. Later we will relax this assumption, too.

[2]For example, we do not need to assume the TCP equation [12] or aggregate Poisson arrivals [14]—hence we do not rely on the simplifying assumptions about TCP dynamics and about a large number of flows that are required for these two results.

that the TCP equation [12] holds and that the router queue follows the M/M/1/B dynamics, then buffers of size $O(\log W_{\max})$ packets suffice to utilize a constant fraction of the link capacity.

Our results are qualitatively different from the bandwidth-delay rule-of-thumb or from the results of Appenzeller *et al.* On the positive side, we have *completely removed* the dependence of the buffer size on the bandwidth-delay product. To understand the importance of this, consider the scaling where the RTT is held fixed at τ, but the maximum window size $W_{\max}$, the number of flows N, and the capacity C all go to ∞ such that $C = NW_{\max}/\tau$. This is a very reasonable scaling since τ is limited by the speed of light, whereas C, N, and $W_{\max}$ are all expected to keep growing as Internet traffic scales. Under this scaling, the sizing rule of Appenzeller *et al.* suggests that the buffer size should grow as $\sqrt{N}W_{\max}$, whereas our results suggest that the buffer size needs to grow only at the much more benign rate of $\log W_{\max}$. On the negative side, unlike the result of Appenzeller *et al.* , our result is a trade-off result—to obtain this large decrease in buffers, we have to sacrifice some fixed fraction (say around 20%) of link capacity. This is a good trade-off for an all-optical network where bandwidth is plentiful and buffers are scarce. But for electronic routers, this might possibly be a sub-optimal trade-off.

We give evidence that our result is tight in the following sense.

1) Under the scaling described above, buffers must at least grow in proportion to $\log W_{\max}$ to obtain a constant factor link utilization. In Section IV-A, we present simulation evidence that constant sized buffers are not adequate as the maximum window size grows to infinity. We also perform a simple calculation that shows the necessity of the log-scaling assuming the TCP equation and M/M/1/B queueing.
2) When we run simulations without using Paced TCP, we can not obtain reasonable link utilizations with log-sized buffers, even in the over-provisioned case (Section III).

While TCP pacing is arguably a small price to pay for drastic reduction in buffer sizes, it does require a change to end-hosts. Fortunately, we suspect this may not be necessary, as two effects naturally provide some pacing in current networks. First, the access links are typically much slower than the core links, and so traffic entering the core from access links is automatically paced; we call this phenomenon "link-pacing". We present simulations showing that with link-pacing we only need very small buffers, because packets are spaced enough by the network. Second, the ACK-clocking scheme of TCP paces packets [2]. The full impact of these two phenomena deserves further study.

Other interesting directions for further study include the impact of packet sizes, the interaction of switch scheduling algorithms and small buffers, the impact of short flows, and the stability properties of TCP with our log-scaling rule. (Significant progress in analyzing stability was made recently by Raina and Wischik [14].)

Of course, significant additional work—including experimental verification, more detailed analysis, and larger simulation studies—is required before we undertake a drastic reduction in buffer sizes in the current Internet.

II. Intuition: Poisson Injections and an Over-provisioned Network

The intuition behind our approach is quite simple. Imagine for a moment that each flow is an independent Poisson process. This is clearly an unrealistic (and incorrect) assumption, but it serves to illustrate the intuition. Assume too that each router behaves like an M/M/1 queue. The drop-rate would be ρ^B, where ρ is the link utilization and B is the buffer size. At 75% load and with 20 packet buffers, the drop rate would be 0.3%, independent of the RTT, number of flows, and link-rate. This should be compared with a typical 10Gb/s router line-card today that maintains 1,000,000 packet buffers, and its buffer size is dictated by the RTT, number of flows and link-rate. In essence, the cost of not having Poisson arrivals is about five orders of magnitude more buffering! An interesting question is: How "Poisson-like" do the flows need to be in order to reap most of the benefit of very small buffers?

To answer our question, assume N long-lived TCP flows share a bottleneck link. Flow i has time-varying window size $W_i(t)$ and follows TCP's AIMD dynamics. In other words if the source receives an ACK at time t, it will increase the window size by $1/W_i(t)$, and if the flow detects a packet loss it will decrease the congestion window by a factor of two. In any time interval $(t, t']$ when the congestion window size is fixed, the source will send packets as a Poisson process at rate $W_i(t)/\text{RTT}$. Note that this is different from regular TCP, which typically sends packets as a burst at the start of the window.

We will assume that the window size is bounded by $W_{\max}$. Implementations today typically have a bound imposed by the operating system (Linux defaults to $W_{\max} = 64$kbytes), or the window size is limited by the speed of the access link. We'll make the simplifying assumption that the two-way propagation delay of each flow is RTT. Having a different propagation delay for each flow leads to the same results, but the analysis is more complicated. The capacity C of the shared link is assumed to be at least $(1/\rho) \cdot NW_{\max}/\text{RTT}$ where ρ is some constant less than 1. Hence, the network is over-provisioned by a factor of $1/\rho$, i.e. the peak throughput is ρC. The effective utilization, θ, is defined as the achieved throughput divided by ρC.

In this scenario, the following theorem holds:

Theorem 1: To achieve an effective utilization of θ, a buffer of size

$$B \geq \log_{1/\rho}\left(\frac{W_{\max}^2}{2(1-\theta)}\right) \tag{1}$$

suffices.

Proof: See Appendix I. □

As an example of the consequences of this simple model, if $W_{\max} = 64$ packets, $\rho = 0.5$, and we want an effective utilization of 90%, we need a buffer size of 15 packets *regardless of the link capacity*. In other words, the AIMD dynamics of TCP don't necessarily force us to use larger buffers, if the arrivals are well-behaved and non-bursty. So what happens if we make the model more realistic? In the next section we consider what happens if instead of injecting packets according to a Poisson process, each source uses Paced TCP in which packets are spread uniformly throughout the window.

III. Paced TCP, Over-provisioned Network

It should come as no surprise that we can use very small buffers when arrivals are Poisson: arrivals to the router are benign and non-bursty. Queues tend to build up—and hence we need larger buffers—when large bursts arrive, such as when a TCP source sends all of its outstanding packets at the start of the congestion window. But we can prevent this from happening if we make the source spread the packets over the whole window. Intuitively, this modification should prevent bursts and hence remove the need for large buffers. We now show that this is indeed the case. Throughout this section, we assume that the bottleneck link is over-provisioned in the same sense as in the previous section. In the next section we remove this assumption.

First, suppose N flows, each with maximum window size $W_{\max}$, share a bottleneck link. Then the following is true.

Theorem 2: Assume that: (1) The buffer size at the bottleneck link is at least $c_B \log W_{\max}$ packets, where $c_B > 0$ is a sufficiently large constant; (2) The rate of each flow is at least a $c_S \log W_{\max}$ factor slower than that of the bottleneck link, where c_S is a sufficiently large constant; and (3) Random jitter prevents a priori synchronization of the flows. Then the packet loss probability during a single RTT is $O(1/W_{\max}^2)$.

Proof: See Appendix II. □

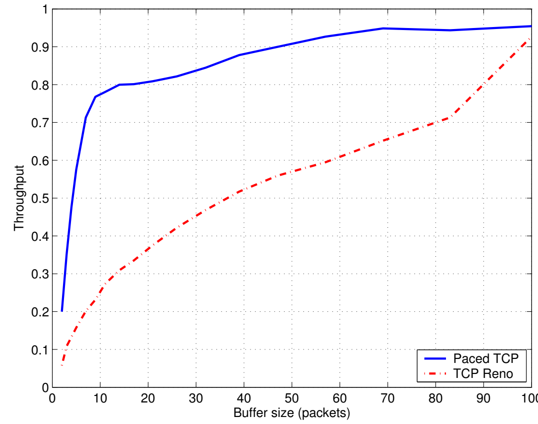

Fig. 1. Bottleneck link utilization for different buffer sizes (TCP Reno vs. Paced TCP)

The packet loss probability in Theorem 2 is comparable to that for Poisson traffic with the same buffer size. The buffer size requirement for Theorem 2 (Assumption (1)) is comparable to that in Theorem 1—a few dozen packets for present-day window sizes, independent of the link capacity, number of flows, and RTT. This requirement appears to be necessary to achieve constant throughput, even with Paced TCP (see Section IV-A). The second assumption states that flows should be "sufficiently non-bursty". Note that some assumption of this form is necessary, since if flows can send at the same rate as the bottleneck link, then there is no pacing of traffic whatsoever and our simulations indicate that constant throughput is not achievable with log-sized buffers. Precisely, Theorem 2 requires that all flows send at a rate that is roughly a $\log W_{\max}$ factor slower than that of the bottleneck link, and obtains a Poisson-like throughput-buffer size trade-off under this requirement. This slowdown factor is only a few dozen for present-day window sizes, while access links are often orders of magnitude slower than backbone links. This huge difference in access link and backbone link speeds also seems likely to persist in the near future (especially with an all-optical backbone). The final assumption of Theorem 2 simply ensures that the flows are not initialized in a synchronized state.

To explore the validity of Theorem 2, we performed simulations using the popular $ns2$ simulation tool [1]. We implemented Paced TCP by adding a new timer to TCP Reno which regulates the injection time of packets. This is a high granularity timer, since using low granularity timers might generate bursts, which is undesirable[3]. Each flow is generated at a source node, goes through an individual access link, and then through a shared link. The capacity and the propagation delay of the shared and access links vary for different simulations. We add a random jitter to the propagation delay of the access links to model different RTTs in the network.

In Figure 1 we compare the buffer size needed by TCP Reno with Paced TCP. We plot the throughput of the system as a function of the buffer size used in the router. The capacity of the bottleneck link is 100Mb/s, there are 40 flows in the system, and the average RTT is 100ms. In this experiment, the maximum congestion window size is set to 32 packets, and the size of packets is 1,000 bytes, thus the maximum offered load is 100Mb/s with 40 flows. The simulation is run for 1,000 seconds, and we start recording the data after 200 seconds.

As we can see, with 40 unmodified TCP (Reno) flows, we need to buffer about 100 packets to achieve a throughput above 80%. However, in the same setting, Paced TCP achieves 80% throughput with just 10 packet buffers.

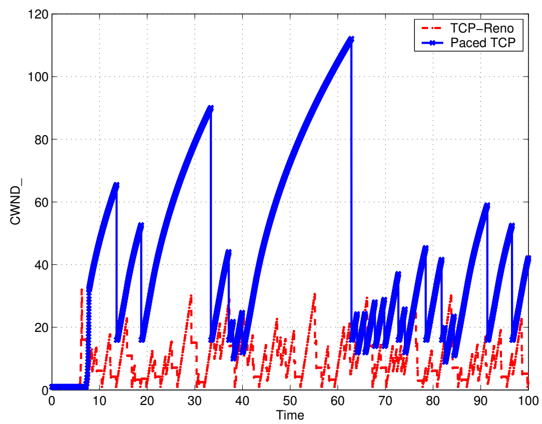

Fig. 2. Congestion window size (TCP Reno vs. Paced TCP)

To understand the impact of pacing, we take a closer look at the congestion window (CWND_) of TCP Reno and Paced TCP in Figure 2. In this experiment, 500 flows share a bottleneck link with a capacity of 1.5Gb/s; the buffer size is 10 packets; and each flow is limited to a maximum window size of 32 packets[4] and the average RTT is 100ms. We observe that TCP Reno rarely reaches the maximum window size of 32 packets, whereas Paced TCP has a larger congestion window at almost all times. The bursty nature of TCP Reno makes the flows experience packet drops more frequently when the buffer size is small, while Paced TCP flows experience fewer drops, and so CWND_ grows to larger values. Consequently, Paced TCP sends with its maximum possible capacity of 32 packets per RTT most of the time.

In Figure 1 we increased the system load as we increased the number of flows since the capacity of the shared link is kept fixed. It is interesting to see what happens if we keep the system load constant (at 80% in this case) while increasing the number of flows. This is illustrated in Figure 3, for flows with a maximum congestion window of 32 packets. As we increase the number of flows from one to more than a thousand, we also increase the bottleneck link capacity from 3.2Mb/s to 3.4Gb/s to keep the peak load at about 80%. The buffer size is still set to 10 packets. The graph shows that regardless of the number of flows, throughput is improved by Paced TCP. The throughput of Paced TCP is around 70% (i.e., the effective utilization is more than 85%) while the throughput of the TCP Reno is around 20% (with an effective utilization of around 25%) regardless of the number of flows in the system.

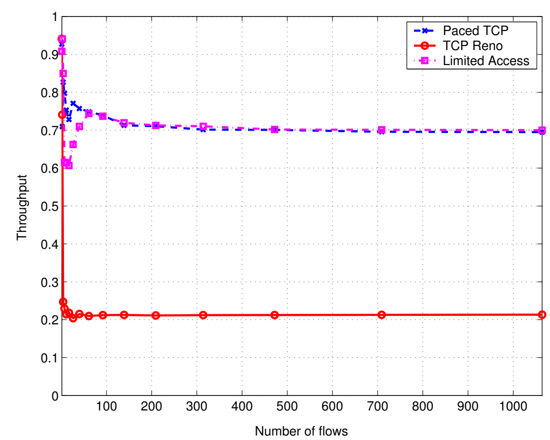

Fig. 3. Throughput of Paced TCP vs. TCP Reno; the capacity of shared link is increased as we increase the number of flows.

We have shown that Paced TCP can gain a very high throughput even with very small buffers. As we have already noted, if the capacity of the access links is much smaller than the core link, then packets entering the core will automatically have spacing between them even without modifying TCP. To verify this we repeat the previous experiment except instead of using Paced TCP, we limit the capacity of the access links to 5Mb/s. As shown in Figure 3, the spacing resulting from limited access link capacities has the same effect as using Paced TCP as the two curves follow each other very closely. In practice, the capacity of the access links is usually several orders of magnitude smaller than the capacity of the core links, which means even without

[3] The implementation of Paced TCP that comes with $ns2$ is not accurate and needs some modifications.

[4] Note that CWND_ can go beyond 32 packets in practice, and the source can have up to min(32,CWND_) unacknowledged packets at any point of time.

any modification to TCP, we can gain high throughput with very small buffers.

It is important to note that this significant discrepancy between the throughput of paced and regular TCP is observed only with small buffers. If we use the bandwidth-delay rule for sizing buffers, this discrepancy vanishes.

IV. PACED TCP, UNDER-PROVISIONED NETWORK

So far we have assumed that the network is over-provisioned and we do not have congestion on the link under study. Even though this is true for most links in the core of the Internet, it is also interesting to relax this assumption. We next study, via simulations, how congestion affects link utilization.

We repeat an experiment similar to that depicted in Figure 1 (this time with Paced TCP only). We increase the number of flows to up to 200 so that the bottleneck link becomes congested. The average RTT is 100ms, and the maximum window size is 32 packets. Each packet is 1000 bytes, which means each flow can contribute a load of $32*1000*8/0.1 \simeq 2.5$Mb/s. The capacity of the core link is 100Mb/s, which mean if we have more than 40 flows, the core link will become congested. For 100 and 200 flows the bottleneck link is highly overloaded.

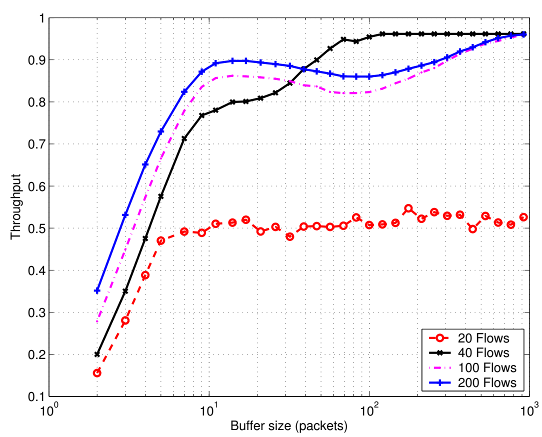

Fig. 4. Bottleneck link utilization vs. the buffer size. With only 40 flows the core link becomes saturated, but even if we increase the number up to 200 flows, the throughput does not go below 80%.

Figure 4 shows the throughput of the bottleneck link as a function of the buffer size for various number of flows. We can see that as we increase the number of flows from 20 to 40 (at which point the link starts to be saturated) the throughput goes from around 50% to about 80-90%. As we increase the number of flows to 100 and 200, for small buffers (1-12 packets) the system throughput slightly improves. For middle size buffers (15-100 packets), however, we see a degradation in throughput, although the throughput never goes below 80%. This minor degradation in performance is due to the partial synchronization between congestion windows of flows, which occurs for middle sized buffers. This phenomenon has been explained by Raina and Wischik [14] in more detail. We note that the throughput of the system remains above 80% even when the buffer size is 10 packets.

Figure 5(a) depicts the throughput of the bottleneck link as a function of the number of flows when the access link capacity is fixed to 5Mb/s and the buffer size is 10 packets. We change the offered load to the system by adjusting the capacity of the bottleneck link relative to the total capacity of the access links. To keep the maximum offered load fixed at a given value, we increase the capacity of the bottleneck link as the number of flows is increased. As we can see in this graph, as we increase the offered load beyond 100%, the overall throughput increases. In other words, even when the network is under-provisioned we expect reasonable throughput with very small buffers.

In a related experiment we set the core link bandwidth to 1Gb/s, and vary the capacity of the access links. The maximum window size is very large (set to 10,000), the buffer size is set to 10 packets, and the average RTT is set to 100ms. Figure 5(b) shows the throughput of the system as a function of the capacity of the access links. We can see that at the beginning the throughput increases (almost) linearly with access link capacity, until the shared link becomes congested. For example, with 100 flows, this happens when the access link capacity is below 8-9Mb/s. Note that the normalized throughput is close to 100% when the offered load is less than the capacity of the shared link since the core link is not the bottleneck. As we increase the access link capacity, the throughput gradually decreases. This is because we lose the natural spacing between packets as the capacity of access links is increased.

A. *The necessity of logarithmic scaling of buffer-sizes*

We have not been able to extend our proof of theorem 1 to the case when the network is under-provisioned. However, the TCP equation [12] gives interesting insights if we assume that the router queue can be modeled as an M/M/1/B system [15]. Consider the scaling (described in the introduction) where the RTT is held fixed at τ, but the maximum window size $W_{\max}$, the number of flows N, and the capacity C all go to ∞. To capture the fact that the network is under-provisioned, we will assume that $C = \frac{NW_{\max}}{2\tau}$ i.e. the link can only support half the peak rate of each flow. Similarly, $C = \frac{2NW_{\max}}{\tau}$ represents the under-provisioned case.

Let p be the drop probability, and ρ the link utilization.

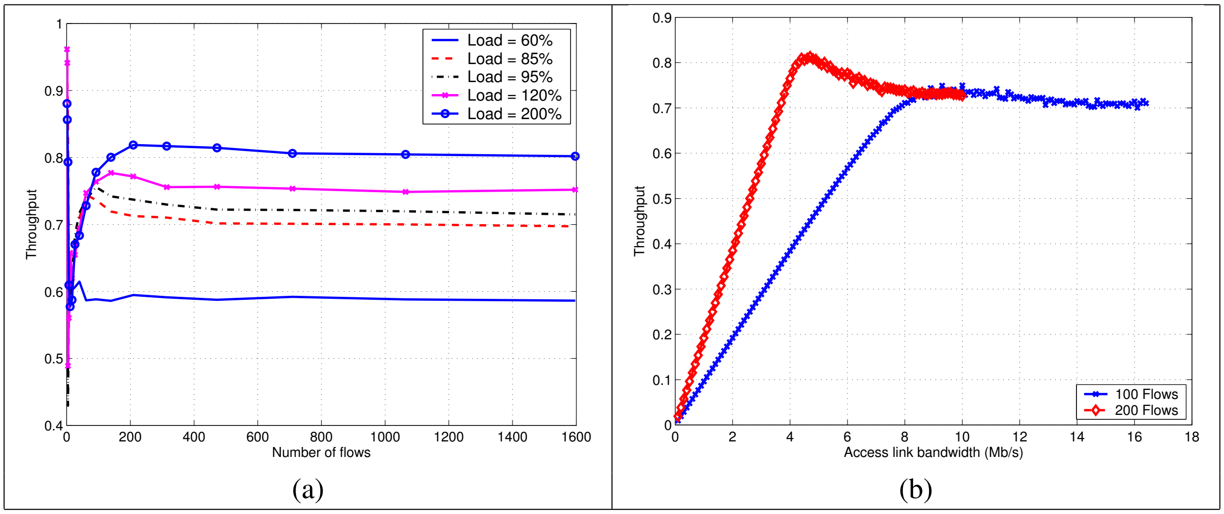

Fig. 5. (a) Throughput as a function of number of flows for various values of the offered load to the system; (b) Throughput as a function of access link capacity.

Clearly, $\rho = RN/C$, where R is the average throughput of each flow. Then, the TCP equation states:

$$R = \frac{1}{\tau}\sqrt{\frac{3}{2p}} + o(1/\sqrt{p}) \simeq \frac{1}{\tau}\sqrt{\frac{3}{2p}}. \tag{2}$$

The M/M/1/B assumption yields [7]:

$$p = \rho^B \frac{1-\rho}{1-\rho^B}\frac{\rho}{1+\rho} \simeq \rho^{B+1}. \tag{3}$$

Equations 2, 3 immediately yield the following:

1) Assume $C = \frac{NW_{\max}}{2\tau}$. For every constant $\alpha < 1$, there exists another constant β such that setting $B = \beta \log W_{\max}$ yields $\rho > \alpha$. In other words, logarithmic buffer-sizes suffice for obtaining constant link utilization even when the network is underprovisioned.
2) Assume $C = \frac{2NW_{\max}}{\tau}$. If $B = o(\log W_{\max})$ then $\rho = o(1)$. In other words, if the buffer size grows slower than $\log W_{\max}$ then the link utilization drops to 0 even in the over-provisioned case.

Obtaining formal proofs of the above statements remains an interesting open problem. Simulation evidence supports these claims, as can be seen in Figure 6 which describes the throughput for a constant vs. a logarithmic sized buffer. For this simulation we are using Paced TCP, N is held fixed at 10, $W_{\max}$ varies from 10 to 1529, and C varies as follows. Initially, C is chosen so that the peak load is constant and a little over 50%, and this choice determines the initial value for the ratio $\frac{C\overline{\mathrm{RTT}}}{NW_{max}}$; then, since we fix N and $\overline{\mathrm{RTT}}$, C varies proportionally to W_{max} to keep the above ratio constant as in our theoretical modeling. The buffer size is set to 5 packets when $W_{\max}$ = 10. Thereafter, it increases in proportion with $\log W_{\max}$ for the log-sized-buffer case,

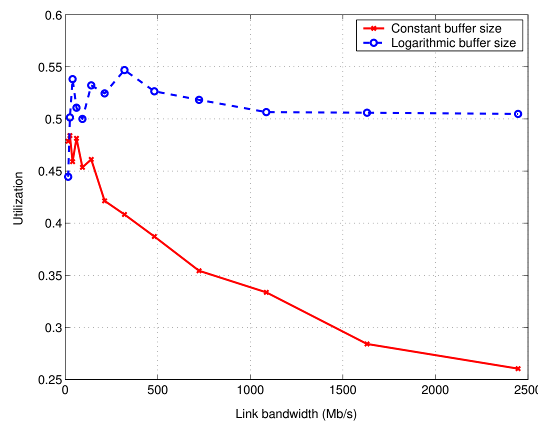

Fig. 6. Constant vs. logarithmic buffers

and remains fixed at 5 for the constant buffer case. Here, initially the throughput is around 50% for both buffer sizing schemes. However, the throughput for the constant sized buffer drops significantly as C and $W_{\max}$ increase, while for the logarithmic sized buffer the throughput remains approximately the same, just as predicted by our theoretical model.

V. Conclusions

The main conclusion, of course, is that our results suggest packet buffers can be made much smaller; perhaps as small as 10-20 packets, if we are prepared to sacrifice some of the link capacity. It appears from simulation - though we have not been able to prove it - that the buffer size dictates directly how much link capacity is lost, however congested the network is. For example, a 40Gb/s link with 15 packet buffers could be considered to operate like a 30Gb/s link. Of course, this loss in link capacity could be eliminated by making the router run faster than the link-rate. In a future network with abundant link capacity, this could be a very good trade-off: Use tiny buffers so that we can process packets optically. In the past, it was reasonable to assume that packet buffers were cheap, while long-haul links were expensive and needed to be fully utilized. Today, fast, large packet buffers are relatively painful to design and deploy; whereas link capacity is plentiful and it is common for links to operate well below capacity. This is even more so in an all-optical network where packet buffers are extremely costly and capacity is abundant.

The buffer size we propose depends on the maximum window size. Today, default settings in operating systems limit window size, but this limitation will probably go away over time. However, even if the maximum window size were to increase exponentially with time according to some form of "Moore's law", the buffer size would only need to increase linearly with time, which is a very benign scaling given recent technology trends.

Our results also assume that packets are sufficiently spaced out to avoid heavy bursts from one flow. Again, slow access links help make this happen. But if this is not true - for example, when two supercomputers communicate - the TCP senders can be modified to use Paced TCP instead.

Our results lead to some other interesting observations. First, it seems that TCP dynamics have very little effect on buffer-sizing, and hence these results should apply to a very broad class of traffic. This is surprising, and counters the prevailing wisdom (and our own prior assumption) that buffers should be made large because of TCP's sawtooth behavior.

Acknowledgement

The authors would like to thank Neda Beheshti, Peter Glynn, and Damon Mosk-Aoyama for helpful discussions.

References

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

[2] A. Aggarwal, S. Savage, and T. Anderson. Understanding the performance of TCP pacing. In *Proceedings of the IEEE INFOCOM*, pages 1157–1165, Tel-Aviv, Israel, March 2000.

[3] G. Appenzeller, I. Keslassy, and N. McKeown. Sizing router buffers. In *SIGCOMM '04*, pages 281–292, New York, NY, USA, 2004. ACM Press.

[4] 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.

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

[6] V. Jacobson. Congestion avoidance and control. *ACM Computer Communications Review*, pages 314–329, Aug. 1988.

[7] F. P. Kelly. *Chapter 3: Queueing Networks*, pages 57–94. Wiley, Chichester, 1979.

[8] V. Lal, J. A. Summers, M. L. Masanovic, L. A. Coldren, and D. J. Blumenthal. Novel compact inPbased monolithic widely-tunable differential Mach-Zehnder interferometer wavelength converter for 40Gbps operation. In *Indium Phosphide and Related Materials*, Scotland, 2005.

[9] M. L. Masanovic, V. Lal, J. S. Barton, E. J. Skogen, J. A. Summers, L. Rau, L. A. Coldren, and D. J. Blumenthal. Widely-tunable monolithically-integrated all-optical wavelength converters in InP. *Journal of Lightwave Tehcnology*, 23(3), 2005.

[10] Microsoft. TCP/IP and nbt configuration parameters for windows xp. Microsoft Knowledge Base Article - 314053, November 4, 2003.

[11] R. Motwani and P. Raghavan. *Randomized Algorithms*. Cambridge University Press, 1995.

[12] J. Padhye, V. Firoiu, D. Towsley, and J. Kurose. Modeling tcp throughput: a simple model and its empirical validation. In *SIGCOMM '98*, pages 303–314, New York, NY, USA, 1998. ACM Press.

[13] H. Park, E. F. Burmeister, S. Bjorlin, and J. E. Bowers. 40-gb/s optical buffer design and simulations. In *Numerical Simulation of Optoelectronic Devices (NUSOD)*, 2004.

[14] G. Raina and D. Wischik. Buffer sizes for large multiplexers: Tcp queueing theory and instability analysis. http://www.cs.ucl.ac.uk/staff/D.Wischik /Talks/tcptheory.html.

[15] K. Ramanan and J. Cao. A poisson limit for buffer overflow probabilities. In *INFOCOM*, 2002.

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

Appendix I
Proof of the Main Theorem

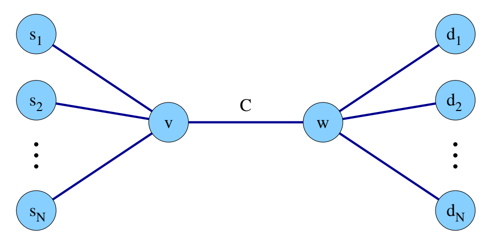

Fig. 7. Topology

We will use the topology of Figure 7. The capacity of the shared link (v, w), which is denoted by C, is assumed to be at least $(1/\rho) \cdot NW_{\max}/\text{RTT}$ where ρ is some constant less than 1. Hence, the network is over-provisioned by a factor of $1/\rho$, i.e. the peak throughput is ρC. The effective utilization, θ, is defined as the achieved throughput divided by ρC. We also assume that node v is an output queued switch, and has a buffer size of B.

The flow going through the link (v, w) is the superposition of the N long-lived TCP flows. Since the packet injection rate of the i-th flow is $W_i(t)/\text{RTT}$, its flow injection is dominated by a Poisson process of rate $W_{\max}/\text{RTT}$. More specifically, we can consider a virtual system in which flow i has an injection process which is Poisson and of rate $W_{\max}$/RTT. We can couple the packet injection processes in the real system and this virtual system such that the packets injected by the real system are a subset of the packets injected by the virtual system. Therefore, the aggregate of the N flows is dominated by a Poisson process of rate $NW_{\max}$/RTT.

Now, let us consider the output queue at node v. We assume this is a drop-tail queue, and prove the following lemma.

Lemma 1: The number of packet drops in the real system is less than or equal to the number of packet drops in the virtual system.

Proof: At a given point of time t, let us denote the residual amount of data (queue occupancy plus part of the packet being served which has not left the system yet) in the real system with $Q_R(t)$ and the amount of data residing in the virtual system with $Q_V(t)$. We also denote the accumulative number of packet drops for the real system by $D_R(t)$ and the number of packet drops for the virtual system by $D_V(t)$. We claim that for any time t,

$$[Q_R(t) - Q_V(t)]^+ \leq (D_V(t) - D_R(t)). \qquad (4)$$

Clearly this is true when both queues are empty at the beginning. Now we consider the following cases:

1) If we have no arrival and just time passes, the right hand side does not change, while the left hand side can only decrease or remain the same. This case also includes when packets depart either system.
2) If we have arrivals or drops at both queues at the same time, the inequality still holds.
3) If we have an arrival to the virtual system, and no arrivals to the real system no matter if we have a drop or not, the LHS doesn't increase, and the RHS might increase, which means the inequality still holds.
4) If we have an arrival to both of the queues and the real system drops the packet but the virtual system doesn't, we consider two cases. If $[Q_R(t) - Q_V(t)]^+ \geq 1$, then both sides go down by one unit. Otherwise, since the real system drops the packet but the virtual one doesn't, we can conclude that $Q_R(t) > Q_V(t)$ (t is the time right before this last arrival), which means $[Q_R(t) - Q_V(t)]^+$ is strictly greater than zero, and therefore $D_V(t)$ is greater than $D_R(t)$. Now, after the arrival and the drop, the LHS will become zero, while the RHS is greater than or equal to zero.

In all cases the inequality holds. Now, since $[Q_R(t) - Q_V(t)]^+$ is greater than or equal to zero (by definition), our inequality is translated to $D_V(t) \geq D_R(t)$. □

So far we have shown that the number of packet drops in the virtual system is more than the number of packet drops in the real system. The next step is to bound the number of drops in the virtual system. In this system the arrival process is Poisson. If we assume that the packets are all of the same size, the service time will be fixed. Therefore, we have an M/D/1 queue with a service rate of C, and arrival rate of ρC.

Lemma 2: The drop probability of the virtual system is bounded above by ρ^B.

Proof: The queue occupancy distribution of an M/D/1 FCFS queueing system is equal to that of an M/D/1 LCFS-PR (last-come first-served with preemptive resume) queue. This is because both queues have the same arrival process, are work conserving, and have the same service rate. Now, for any M/G/1 LCFS-PR system the steady state queue occupancy distribution is geometric with parameter ρ. Therefore, the drop probability of the M/G/1 LCFS-PR system equals ρ^B, which is an upper bound on the drop probability of the virtual system. □

Note that this is not necessarily an upper bound on the packet drop probability of the real system.

Now that we have an upper bound on the packet loss probability of the virtual system, the next step is to find a lower bound on the throughput of the real system. Without loss of generality, we consider the dynamics of one of the flows, say flow number one. For simplicity, we assume that flow one is in congestion avoidance, *i.e.*, during each RTT the congestion window size is incremented by one if there is no packet loss, and the congestion window goes to zero if a packet loss is detected by the source. Once the congestion window size reaches its maximum value (*i.e.* $W_{\max}$) it will remain fixed.

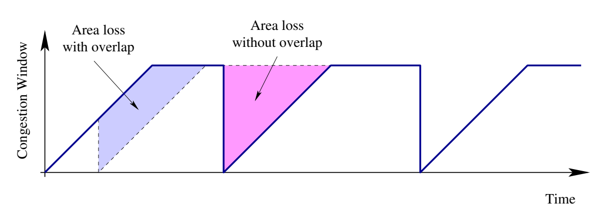

Fig. 8. Dynamics of the congestion window.

Figure 8 depicts an example of the changes in congestion window size. The area under the curve indicates the total amount of data which has been sent by the flow. We can see that by each packet loss some portion of this area is lost, and the amount of loss is maximized when the overlap between the lost regions is minimum. We omit a formal proof. We are ignoring slow-start in this system.

It is not hard to see that considering slow-start can only lead to better bounds (i.e. smaller buffers)—again, we omit the formal proof.

Let us consider a long time interval of length Δ, and let us denote the number of packets injected by the sources in the virtual system during this interval with P_v, and the number of packet drops in the virtual system during this time interval with D_V. Choose an arbitrarily small $\epsilon > 0$. As Δ goes to ∞, we have:

$$Pr\left[P_v > \frac{\Delta N W_{\max}}{RTT}(1+\epsilon)\right] = o(1); \tag{5}$$

and,

$$Pr\left[P_v < \frac{\Delta N W_{\max}}{RTT}(1-\epsilon)\right] = o(1). \tag{6}$$

Since the probability of each packet being dropped is less than ρ^B, using Equation 5, we can bound the total number of packet drops D as follows.

$$Pr\left[D_V > \frac{\rho^B \Delta N W_{\max}}{RTT}(1+\epsilon)\right] = o(1). \tag{7}$$

Based on Lemma 1 the number of packet drops in the virtual system is no less than the number of packet drops in the real system (henceforth denoted by D_R). Therefore, we get the following.

$$Pr\left[D_R > \frac{\rho^B \Delta N W_{\max}}{RTT}(1+\epsilon)\right] = o(1). \tag{8}$$

Now, if none of the flows in the real system encountered any losses during the time interval Δ, the amount of data that could have been sent during this time, U_T, can be bounded below as follows.

$$Pr\left[U_T < \frac{\Delta N W_{\max}}{RTT}(1-\epsilon)\right] = o(1). \tag{9}$$

We will lose some throughput as a result of packet drops in the system. As we can see in Figure 8, the maximum amount of loss occurs when the triangles corresponding to packet losses have the minimum overlap. Therefore, we have

$$U_L \leq \frac{D_R W_{\max}^2}{2}. \tag{10}$$

In Equation 8 we have bounded the number of packet losses in the real system with a high probability. Combining this bound, with Equation 10 we get

$$Pr\left[U_L > \frac{\rho^B \Delta N W_{\max}^3}{2\text{RTT}}(1+\epsilon)\right] = o(1). \tag{11}$$

Now, if we want to guarantee an effective utilization throughput of θ, the following equation must hold.

$$\frac{U_T - U_L}{\rho C \Delta} \geq \theta. \tag{12}$$

Since $\rho C = N W_{\max}/\text{RTT}$, we need to satisfy

$$U_L \leq N \Delta W_{\max}(1-\theta-\epsilon)/\text{RTT}. \tag{13}$$

Combining Equations 9, 11, and 13 if we want to have a throughput of θ, we merely need to ensure

$$\frac{(1+\epsilon)\rho^B \Delta N W_{\max}^3}{2RTT} < \frac{\Delta N W_{\max}(1-\theta-\epsilon)}{RTT}, \tag{14}$$

which, in turn, is satisfied if the following holds:

$$\rho^B < \frac{2(1-\theta-O(\epsilon))}{W_{\max}^2}. \tag{15}$$

Since ϵ is arbitrarily small, it is sufficient for the buffer size B to satisfy

$$B \geq \log_{1/\rho}\left(\frac{W_{\max}^2}{2(1-\theta)}\right), \tag{16}$$

which is $O(\log W_{\max})$ since we assumed that ρ, θ are constants less than 1.

Appendix II
Pacing Analysis

In this Appendix we prove Theorem 2. We will consider the following discrete-time model of the packet arrivals at the bottlelink link during one RTT. There are a total $M = C \cdot RTT$ time slots, where C is the bandwidth of the bottleneck link. We assume that N flows will each send at most $W_{\max}$ packets, and that there are at least S time slots between consecutive packet arrivals of a single flow. The parameter S can be interpreted as a lower bound on the ratio of the bottleneck link speed to the access link speed. Note S is the crucial parameter in this section: when S is small traffic can be arbitrarily bursty and we cannot expect good throughput with small buffers (see also Section IV-A). We thus aim to prove that small buffers permit large throughput provided S is sufficiently large. Finally, we assume that the average traffic intensity $\rho = N W_{\max}/M$ is bounded below 1.

For the rest of this section, we adopt three assumptions.

(1) Buffers are sufficiently large: $B \geq c_B \log W_{\max}$, where $c_B > 0$ is a sufficiently large positive constant.
(2) The distance between consecutive packet arrivals of a single flow is sufficiently large: $S \geq c_S \log W_{\max}$,

where $c_S > 0$ is a sufficiently large positive constant.

(3) Random jitter prevents a priori synchronization of the flows: flow start times are picked independently and uniformly at random from the M time slots.

As discussed in Section III, the first two assumptions are often reasonable and are mathematically necessary for our results. The validity of the third assumption is less clear, especially in the presence of packet drops, which could increase the degree of synchronization among flows. Our simulations in Section IV indicate, however, that our analytical bounds remain valid for long-lived flows that experience packet drops.

Our proof of Theorem 2 will focus on a particular (but arbitrary) packet. If the packet arrives during time slot t, then the probability that it is dropped is at most the probability that for some interval I of l contiguous time slots ending in time slot t, there were at least $l + B$ other packet arrivals. If this event occurs, we will say that the interval I is *overpopulated*. We will bound the probability of overpopulation, as a function of the interval length l, via the following sequence of lemmas. We first state the lemmas, then show how they imply Theorem 2, and finally prove each of the lemmas in turn.

The first lemma upper bounds the overpopulation probability for small intervals (of length at most $\log W_{max}$).

Lemma 3: In the notation above, if $l \leq \log W_{max}$, then the probability that the interval I is overpopulated is at most e^{-cB}, where $c > 0$ is a positive constant that depends only on c_B.

The second lemma considers intervals of intermediate size.

Lemma 4: In the notation above, if $\log W_{max} \leq l \leq SW_{\max}$, then the probability that the interval I is overpopulated is at most e^{-cS}, where $c > 0$ is a positive constant that depends only on c_S.

Finally, we upper bound the probability of overpopulation in large intervals.

Lemma 5: In the notation above, if $l \geq SW_{\max}$, then the probability that the interval I is overpopulated is at most $e^{-cl/W_{\max}}$, where $c > 0$ is a positive constant that depends only on c_S.

We now show how Lemmas 3–5 imply Theorem 2.

Proof of Theorem 2: We consider an arbitrary packet arriving in time slot t, and take the Union Bound over the overpopulation probabilities of all intervals that conclude with time slot t. First, by Lemma 3, the total overpopulation probability for small intervals (length l at most $\log W_{max}$) is $O(e^{-B} \log W_{max})$, which is $O(1/W_{\max}^2)$ provided c_B (in Assumption (1)) is sufficiently large. Next, Lemma 4 implies that the total overpopulation probability of intervals with length l in $[\log W_{max}, SW_{\max}]$ is at most $SW_{\max} e^{-\Omega(S)}$, which is $O(1/W_{\max}^2)$ provided c_S (in Assumption (2)) is sufficiently large. Finally, Lemma 5 implies that the total overpopulation probability of large intervals ($l \geq SW_{\max}$) is at most $\int_{SW_{\max}}^{\infty} e^{-\Omega(x/W_{\max})} dx$. Changing variables ($z = x/W_{\max}$), this quantity equals $W_{\max} \int_S^{\infty} e^{-z} dz$, which is $O(1/W_{\max}^2)$ provided c_S is sufficiently large. Taking the Union Bound over the three types of intervals, we obtain an upper bound of $O(1/W_{\max}^2)$ for the total overpopulation probability, and hence for the probability that the packet is dropped. This completes the proof. $\square$

Proof of Lemma 3: If the length l of interval I is at most $\log W_{max}$, then each flow contributes at most 1 packet to I (assuming that $c_S \geq 1$). This occurs with probability at most $W_{\max} l/M$. Let X_i denote the corresponding indicator random variable for flow i, and define $X = \sum X_i$. Note that $EX \leq \rho l \leq \rho \log W_{max}$. We use the following Chernoff bound (see e.g. [11]) for a sum of indicator random variables with expectation μ: $Pr[X \geq (1+\delta)\mu] < [e^{\delta}/(1+\delta)^{(1+\delta)}]^{\mu}$. Assuming that c_B is sufficiently large (in Assumption (1)), setting $(1+\delta)\mu = B$ gives $Pr[X \geq l + B] \leq Pr[X \geq B] \leq e^{-\Theta(B)}$.

Proof of Lemma 4: Suppose $(k-1)S \leq l \leq kS$ for some $k \in \{1, 2, ..., W_{\max}\}$. Then each flow contributes at most k packets to I. Assume for simplicity that each flow contributes either 0 or k packets to I, with the latter event occurring with probability $W_{\max} S/M$ (so that $E[kX] \approx \rho l$). (Here X_i is the indicator for the latter event, so the total number of arrivals in I is at most kX, where $X = \sum_i X_i$.) A more accurate analysis that permits each flow to contribute any number of packets between 0 and k to I (with appropriate probabilities) can also be made, but the results are nearly identical to those given here.

We use the same Chernoff bound as in the proof of Lemma 3. We are interested in the probability $Pr[X \geq (l+B)/k]$, which we will upper bound by $Pr[X \geq l/k]$. Since $\mu = \Theta(\rho l/k)$, the Chernoff bound gives $Pr[X \geq l/k] \leq \exp\{-\Theta(l/k)\} = e^{-\Theta(S)}$ for fixed $\rho < 1$. $\square$

Proof of Lemma 5: The proof is similar to the previous one. Suppose that $l \geq W_{\max} S$ and assume that each flow i contributes either 0 or $W_{\max}$ packets to I, the latter event occurring with probability l/M. (So $E[W_{\max} X] = \rho l$.) The same Chernoff bound argument as in the proof of Lemma 4 gives $Pr[W_{\max} X \geq l + B] \leq e^{-\Theta(l/W_{\max})}$ for fixed $\rho < 1$. $\square$