Experimental Study of Router Buffer Sizing [*] [†]

Neda Beheshti
Department of Electrical Engineering, Stanford University, Stanford, CA, USA
nbehesht@stanford.edu

Yashar Ganjali
Department of Computer Science, University of Toronto, Toronto, ON, Canada
yganjali@cs.toronto.edu

Monia Ghobadi
Department of Computer Science, University of Toronto, Toronto, ON, Canada
monia@cs.toronto.edu

Nick McKeown
Department of Electrical Engineering, Stanford University, Stanford, CA, USA
nickm@stanford.edu

Geoff Salmon
Department of Computer Science, University of Toronto, Toronto, ON, Canada
geoff@cs.toronto.edu

ABSTRACT

During the past four years, several papers have proposed rules for sizing buffers in Internet core routers. Appenzeller *et al.* suggest that a link needs a buffer of size $O(C/\sqrt{N})$, where C is the capacity of the link, and N is the number of flows sharing the link. If correct, buffers could be reduced by 99% in a typical backbone router today without loss in throughput. Enachecsu *et al.*, and Raina *et al.* suggest that buffers can be reduced even further to 20-50 packets if we are willing to sacrifice a fraction of link capacities, and if there is a large ratio between the speed of core and access links. If correct, this is a five orders of magnitude reduction in buffer sizes. Each proposal is based on theoretical analysis and validated using simulations. Given the potential benefits (and the risk of getting it wrong!) it is worth asking if these results hold in real operational networks. In this paper, we report buffer-sizing experiments performed on real networks - either laboratory networks with commercial routers as well as customized switching and monitoring equipment (UW Madison, Sprint ATL, and University of Toronto), or operational backbone networks (Level 3 Communications backbone network, Internet2, and Stanford). The good news: Subject to the limited scenarios we can create, the buffer sizing results appear to hold. While we are confident that the $O(C/\sqrt{N})$ will hold quite generally for backbone routers, the 20-50 packet rule should be applied with extra caution to ensure that network components satisfy the underlying assumptions.

Categories and Subject Descriptors

C.2 [**Computer-Communication Networks**]: Internetworking

General Terms

Experimentation, Measurement, Performance

Keywords

NetFPGA, Network Test-beds, Router Buffer Size, TCP

1. MOTIVATION AND INTRODUCTION

Most routers in the backbone of the Internet have a bandwidth delay product worth of buffering for each link; *i.e.* $B = 2T \times C$, where C is the bottleneck link capacity, and $2T$ is the effective two-way propagation delay (RTT) of TCP flows through the core router [16, 17, 27]. This value is recommended by Internet RFCs [6], architectural guidelines, and network operators.

On the other hand, several recent papers propose considerably reducing the buffers in backbone routers [4, 9, 23]. For example, Appenzeller *et al.* [4] propose that the buffers can be reduced to $2T \times C/\sqrt{N}$, where N is the number of long-lived flows sharing the link. Throughout the paper we will refer to this as the *small buffer model*. The basic idea follows from the observation that the buffer size is, in part, determined by the sawtooth window size process of the TCP flows. The bigger the sawtooth, the bigger the buffers need to be in order to guarantee 100% throughput. As the number of flows increases, variations in the aggregate window size process (the sum of all the congestion window size processes for each flow) decrease, following the central limit theorem. The result relies on several assumptions: (1) that flows are sufficiently independent of each other to be desynchronized, (2) that the buffer size is dominated by the long-lived flows, and, perhaps most importantly, (3) that there are no other significant unmodeled reasons for buffering more packets. If the result is correct, then a backbone link carrying 10,000 long-lived flows could have its buffer size reduced by a factor of 100 without loss in throughput.

[*]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 and W911NF-07-1-0024. University of Toronto's work was supported by NSERC Discovery, NSERC RTI as well as a grant from Cisco Systems.

[†]The experiments' data is available online at: http://sysweb.cs.toronto.edu/bsizing/

IMC'08, October 20–22, 2008, Vouliagmeni, Greece.

If, though, the result is wrong, then the consequences of reducing the buffer sizes in a router, or in an operational commercial network, could be quite severe. The problem is, how to decide if the result is correct, without trying it in an operational network? But who would reduce buffers in an operational network, and risk losing customers' traffic, before knowing if the result is correct?

It is, therefore, not surprising that, apart from the results we present here, we are not aware of any backbone commercial network in which the buffers have been reduced to anywhere close to $2T \times C/\sqrt{N}$. So the first goal of our work is to experiment with *small buffers* in laboratory and operational networks.

More recently, Enachescu *et al.* [9] and Raina *et al.* [23] propose reducing buffers much further to $O(\log W)$ packets in backbone routers, where W is the congestion window size. This translates to a few dozen packets for present-day window sizes [9]. We will refer to this as the *tiny buffer model*. In [9], the authors reach their conclusion by considering the tradeoff between reducing buffers and losing some throughput – assumed to be 10-20%. In other words, when congested, links behave as if they run at 80-90% of their nominal rates. This could be an interesting assumption in networks with abundant link capacity, or in future optical networks where link capacity might be cheaper than buffers. The results depend on the network traffic being non-bursty, which they propose can happen in two ways: (1) if the core links run much faster than the access links (which they do today), then the packets from a source are spread out and bursts are broken, or (2) TCP sources are changed so as to pace the delivery of packets. If the results are correct, and relevant, then a backbone link could reduce its buffers by five orders of magnitude.

Again, it is difficult to validate these results in an operational network, and we are not aware of any other comprehensive laboratory or network experiments that test the $O(\log W)$ results. So it is the second goal of our work to experiment with *tiny buffers* in laboratory and operational networks.

In the remainder of this paper we describe a number of laboratory and network experiments performed between 2003 and 2008. The laboratory experiments were performed in the WAIL laboratory at University of Wisconsin Madison, in the Sprint Advanced Technology Laboratory, and in University of Toronto's Advanced Packet Switch and Networking Laboratory. Experiments were also performed on the following operational networks: Level 3 Communications' operational backbone network, Internet2 and the Stanford University dormitory network. We should make clear that our results are necessarily limited; while a laboratory network can use commercial backbone routers and accurate TCP sources, it is not the same as a real operational backbone network with millions of real users. On the other hand, experiments on an operational network are inevitably limited by the ability to control and observe the experiments. Commercial routers do not offer accurate ways to set the buffer size and do not collect real-time data on the occupancy of their queues. Real network experiments are not repeatable for different buffer sizes, making apples-with-apples comparisons difficult.

In laboratory experiments, we generate live TCP traffic (ftp, telnet, or http) using a cluster of PCs or commercial traffic generators. We measure the performance from either real-time or statistical traces collected from the system. On one hand, we have a lot of control over the experiments, and can observe almost anything. On the other hand, the traffic is synthetic and might not represent real users. We note that we cannot simply use traces gathered from operational backbone networks for buffer sizing experiments because TCP uses a feedback loop to control congestion, thus live traffic is needed so that we can measure the reaction of flow sources to network conditions.

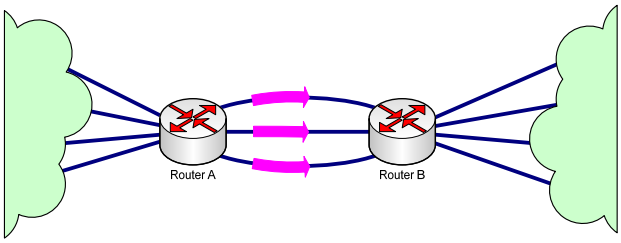


Figure 1: Setup used for buffer sizing experiments in Level 3 Communications' backbone network. The incoming traffic to Router A is divided amongst the three links connecting Router A to Router B using a static hash function to balance the flows.

In our experiments on operational backbone networks we can test the results with real user traffic. However, we have no control over the traffic pattern or the system load. For example, Internet2 has very low load (about $20-30\%$), which means congestion does not occur naturally. Fortunately, at the time of our experiments, part of the Level 3 Communications network included some links facing a 3rd party non-customer network which ran at very high link utilization (up to 96%). We report results from both networks. Where possible, we run experiments over a wide range of operating conditions for both the *small buffer* and *tiny buffer* models, including system load ranging from 25% up to 100%, different number of users, various traffic patterns and flow sizes, different propagation delays, access link capacities, and congestion window limits.

The rest of the paper is organized as follows: Section 2 describes the small buffer experiments. We focus on experiments performed on Level 3 Communications' operational commercial backbone network, and give a brief overview of other experiments. Section 3 is on tiny buffer size experiments performed at University of Toronto and Sprint ATL. Section 4 concludes the paper.

2. SMALL BUFFER EXPERIMENTS

We start with, perhaps, the most interesting experiments, which are performed on Level 3 Communications' operational commercial backbone network. We follow these experiments with a brief overview of others we have conducted in different networks.

2.1 Experiment Setup and Characteristics

Although we have limited control of an operational network, these experiments have several interesting properties. First, the links under study are highly utilized with real live traffic. Their utilization varies between 28.61% and 95.85% during a 24 hour period, and remains above 85% for about four hours every day (an exceptionally high value - new link

capacity was added right after the experiments were completed).

The link under study consists of three physical, load-balanced links (Figure 1). Traffic at the upstream router is divided equally among the three physical links. Each incoming flow is assigned to one of the three links using a static hash function based on the source-destination IP and port numbers of the flow. Ideally, there is equal traffic on each link (particularly as there are thousands of flows)[1]. If we give each physical link a different amount of buffering, we can perform an apples-with-apples comparison between different buffer sizes under almost identical conditions.

The three physical links are OC-48 (2.5Gbps) links facing a 3rd party non-customer network, carrying Internet mix traffic with an emphasis toward high speed consumer access. Assuming an average rate of 250Kbps per flow, each link carries about 10,000 flows when highly utilized[2]. The default buffer size is 190ms per-link (60MB or 125,000 packets assuming an average packet size of 500B). We reduce the buffer sizes to 10ms (about 3MB or 6,000 packets), 5ms (1.5MB or 3,000 packets), 2.5ms (750KB or 1,500 packets) and 1ms (300KB or 600 packets). Based on the small buffer size model, we can expect to need a buffer size of about 2-6ms (depending on the actual value of N).

The buffer sizes are set for 5 days, so they capture the impact of daily and weekly changes in traffic. The whole experiment lasts two weeks and was performed in March, 2005. We gather link throughput and packet drop statistics which are collected by the router every 30 seconds from each of the three links. It would be preferable to capture all packets and recreate the time series of the buffer occupancy in the router, but the network does not have the facility to do this. Still, we are able to infer some interesting results.

We also actively inject test flows, measuring the throughput and drops, and compare the performance of the flows going through different links to find out the impact of buffer size reduction. The amount of test flow traffic is kept small. It is worth noting that the network does not use traffic shaping at the edges or in the core. The routers do not use RED, and packets are dropped from the tail of the queue.

2.2 Experiment Results

During the course of the experiments, we always keep the buffer size on one link at its original size of 190ms and reduce the buffer size on the other two links. Figure 2 shows the packet drop rate as a function of system load for various buffer sizes. As explained before, both the load and drop rates are measured in time intervals of 30 seconds, and each dot in the graph represents one such interval. Figure 2(a) shows that we do not see a single packet drop using a buffer size of 190ms. Similarly, we observe no drops using either 10ms or 5ms of buffering during the course of the experiments, which last more than 10 days for the 190ms buffer size, and about 5 days for each of the 10ms and 5ms buffer sizes.

It is quite surprising that the buffer size can be reduced by a factor of forty without dropping packets even though the utilization frequently exceeds 95%. It suggests that the backbone traffic is very smooth. Others also report smoothness in traffic in core networks [13,14], despite somewhat older results which show self-similarity in core traffic [10, 11, 24]. Whether this is a result of a shift in traffic patterns and network characteristics over time, or simply the consequence of huge amounts of multiplexing, remains to be determined. We find traffic to be extremely smooth in laboratory and backbone networks when there are a large number of flows.

Figure 2(b) shows the drop rate as a function of load, when the buffer sizes is reduced to 2.5ms. This is in the lower part of the range defined by the small buffer sizing model, and, as expected, we start to observe some packet drops. However, packet drops are still rare; less than 0.02% for the majority of samples. In two samples over five days, we see a packet drop rate between 0.02% and 0.04% and in one sample the drop rate is close to 0.09% [3].

Figure 2(c) shows what happens when we reduce the buffer size to 1ms. Here there are a lot more packet drops, which we expect because the the buffer size is now about half the value suggested by the small buffer sizing model. It is interesting to note that almost all of the drops occur when the load is above 90%, even though the load value is averaged over a period of 30 seconds. The instantaneous load is presumably higher, and the link must be the bottleneck for some of the flows. We conclude that having some packet drops does not lead to a reduction in available throughput; it appears that TCP's congestion control algorithms are functioning well.

Next we take a closer look at link utilizations over time. Figure 3(a) compares the link utilization for the links with 190ms and 1ms of buffering, over three days, by plotting their ratio (i.e. relative utilization) as a function of time. Ideally, the utilization on both links would be equal at all times. However, the differences are not symmetric. The link with 1ms buffering has a slightly higher utilization for the majority of the time.

To further investigate the cause of this asymmetry, we plot link utilizations as a function of time in Figure 3(b). By comparing this graph with Figure 3(a) we observe that during the periods when the overall load of the system is high, the link with 1ms has a slightly higher utilization than the link with 190ms. Figure 3(c) suggests the same, in a different yet more precise way. Each dot in this figure, represents the utilization of the two links in a period of 30 seconds. We can see that the majority of the dots fall below the 45 degree line in the graph, which suggests the link with 1 ms of buffering has a higher utilization.

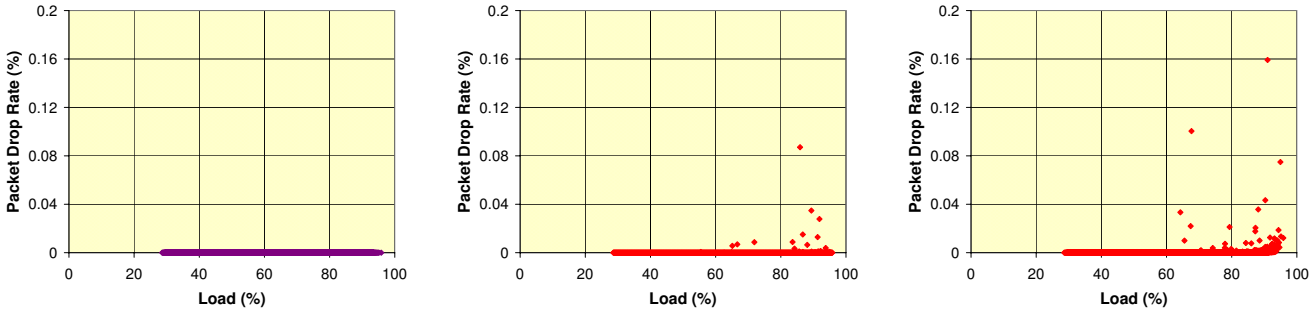
The higher utilization with smaller buffers can be attributed to one of the following two reasons: (1) It might be due to higher loads on the link with 1ms of buffering. Since we have more drops on this link, sources need to send duplicates of the dropped packets, and that might be why we see a higher load. Or, (2) the load balancing scheme might be skewed and might divide the traffic somewhat unevenly among the three links, thus directing more traffic to one of the links.

The question is which of the two reasons is the cause of higher link utilization on one of the two links? The easiest way to answer this question would be swapping the buffer sizes on the two links under study. Unfortunately, immedi-

[1]In this case, the load balancing was not ideal, as we will explain later.

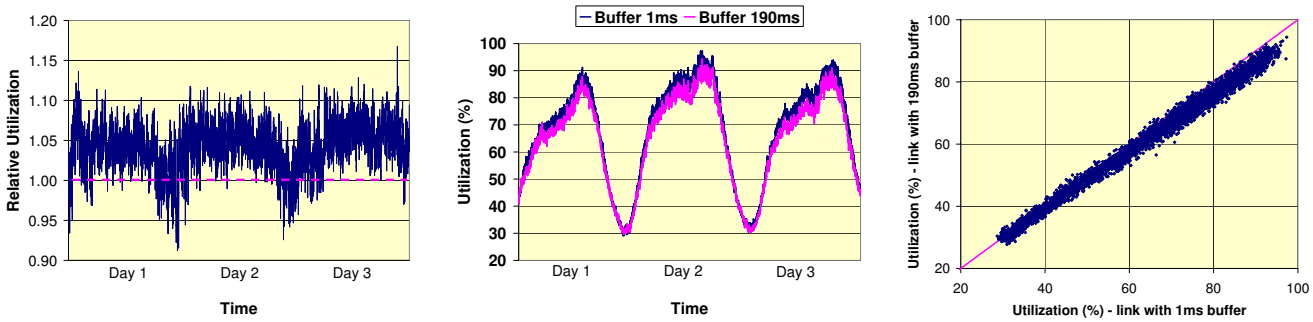
[2]This assumption is arbitrary based on what we assume to be the average end-user bandwidth.

[3]It would be very interesting to take a closer look at how packet drops are distributed over time. Unfortunately, given our limited measurement capabilities in a commercial infrastructure, we cannot study the distribution of packet drops over time granularities finer than 30 seconds.



(a) (b) (c)

Figure 2: **Packet drop rate as a function of load for different buffer sizes. (a) For a buffer size of 190ms we observe no packet drops. Although not shown here, there are also no drops with either 10ms or 5ms of buffering. (b) For a buffer size of 2.5ms packet drops occur in only a handful of cases. (c) When the buffer size is set to 1ms we observe packet drops during high utilization time periods.**



(a) (b) (c)

Figure 3: **Comparing the utilization of the links with 1ms and 190ms of buffering. (a) Link utilization using 1ms buffering relative to 190ms buffering. (b) Individual link utilizations over time. (c) Utilization of 1ms buffer link vs. the utilization of the 190ms buffer link.**

ately after our experiments, Level 3 upgraded the network and added extra capacity to reduce the load on the links we were studying, which means we could not repeat the experiment with the same conditions.

Interestingly, we see the same phenomena (higher utilization in one link than the others) when the buffer sizes are set to 190ms, and 5ms on two links. Since we do not have any packet drops in these cases, buffer occupancies in both links cannot be affected by packet drops, the RTT of flows must be the same in both links, and therefore (1) cannot be the reason here, *i.e.* any difference in utilization is most probably not a result of the reaction of TCP sources to packet drops[4]. In other words, we associate these slight differences between link utilizations with imperfections in the load balancing scheme deployed in the system, rather than the changes in buffer sizes. We conclude that reducing buffer sizes in this network does not have a significant impact on the performance.

[4]Based on our experiment results, the load balancing scheme used in this system was changed to one which is believed to be more fair in distributing the load.

2.3 Other Small Buffer Experiments

Other than the experiments on Level 3 Communications' backbone network, we have also conducted some other experiments with the small buffer sizing model. These experiments include University of Wisconsin Madison's Advanced Internet Laboratory (WAIL), and Stanford University's dormitory network.

In the WAIL experiment, already reported by Appenzeller *et al.* [4], a cluster of PCs is used to generate up to 400 live TCP flows, and the traffic is directed to a Cisco GSR router. The buffer sizes on the router is reduced by a factor of 10-20, which does not result in any degradation in the system throughput.

In the Stanford University experiment, we use a Cisco VXR 7200 which connects the dormitories to the Internet via a 100Mbps link. Traffic is from a mix of different applications including web, ftp, games, peer-to-peer, streaming and others, and we have 400-1900 flows at any given time. The buffer sizes on the router are reduced by a factor of 20 with no impact on network throughput. We omit the details of these experiments due to limited space and refer the interested reader to [15] for more details. All of these experiments are inline with the small buffers theory, which is based on loose assumptions on the number of flows and their

independence. We are fairly confident that the $O(C/\sqrt{N})$ will hold quite generally for backbone routers.

3. TINY BUFFER EXPERIMENTS

In this section we describe our experiments performed in the context of the tiny buffer model: *i.e.* we consider a single point of congestion, assume core links run much faster than the access links, and expect a 10-20% reduction in network throughput. Without a guarantee that these conditions hold in an operational backbone network, it is not feasible to test the *tiny buffer model*, and, therefore, we have to content ourselves to laboratory experiments. We understand this is a limiting factor, and view our work as a first pass in a more comprehensive experimental study of the tiny buffer sizing model by us and others.

3.1 University of Toronto Experiments

We perform an extensive set of experiments on tiny buffer sizing at University of Toronto's Advanced Packet Switch and Networking Laboratory. The goal is to study the impact of tiny buffers on network performance and to identify conditions under which tiny buffers are sufficient. During the course of our experiments, we vary several network parameters such as buffer size in routers, packet injection times, and hardware level parameters. The performance metrics that we study are link utilization, an important factor from Internet Service Providers' point of view, as well as loss and flow completion times, which are major concerns of end-users.

3.1.1 Experiment Setup

Performing time-sensitive network experiments is extremely difficult, especially in the context of tiny buffers, mainly because creating a network with a topology that is representative of a real backbone network requires significant resources. During the course of our experiments, we encounter several challenges, including generating realistic network traffic, emulating delay, approximating large topologies, collecting high-resolution packet-level measurements, and accounting for scaling approximations. In general, these factors, along with hardware/software configuration and limitations, can have a large influence on an experiment's outcome. In [5] we study the challenges associated with performing time-sensitive network experiments in a test-bed. We provide guidelines for setting up test-beds, paying particular attention to those factors that may affect the accuracy of experimental results, and describe obstacles encountered during our own experiments. Below we summarize the important factors for our tiny buffers experiments.

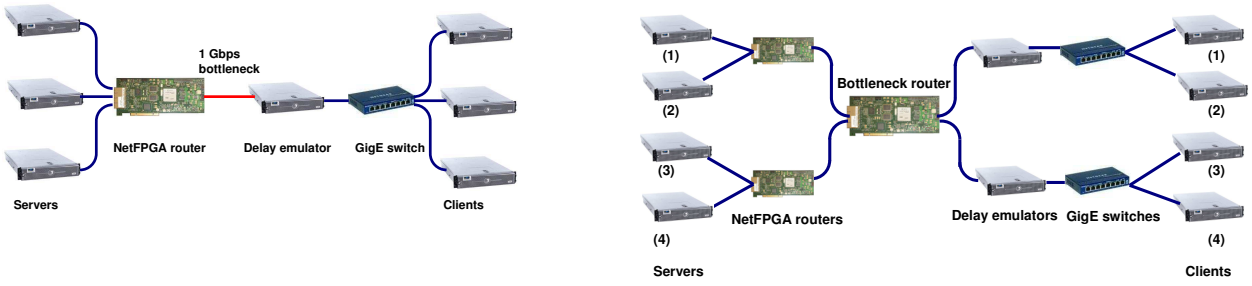
Traffic Generation: As mentioned above, generating realistic traffic is one of the key challenges in modeling a network. Experiments in a laboratory setup often use multiple hosts as traffic generators. However, creating a large number of connections, in order to model traffic in networks closer to the core of the Internet, with thousands of flows, is not a trivial task. In our experiments, the traffic is generated using the open-source Harpoon traffic generator [25]. We use a closed-loop version [22] of Harpoon, modified by researchers at the Georgia Institute of Technology. It is shown in [21] that most Internet traffic (60-80%) conforms to a closed-loop flow arrival model. In this model, a given number of users (running at the client hosts) perform successive TCP requests from the servers. The size of each TCP transfer follows a specified random distribution. After each download, the user stays idle for a thinking period which follows another distribution. We also made several further modifications to the closed-loop Harpoon: each TCP connection is immediately closed once the transfer is complete, the thinking period delay is more accurately timed, and client threads with only one TCP socket use blocking instead of non-blocking sockets. For the transfer sizes, we use a Pareto distribution with mean 80KB and shape parameter 1.5. These values are realistic, based on comparisons with actual packet traces [22]. The think periods follow an exponential distribution with a mean duration of one second. We perform extensive experiments to evaluate Harpoon's TCP traffic; the results are provided in the Appendix.

Switching and Routing: One of the major problems we encounter while performing buffer sizing experiments is that commercial routers do not allow precise adjustment of their buffer sizes. Moreover, they are not able to provide a precise buffer occupancy time-series, which is essential for studying buffer sizing. To address these issues, we use a programmable network component called NetFPGA [1] as the core element of our test-bed. NetFPGA is a PCI form factor board that contains reprogrammable FPGA elements and four Gigabit Ethernet interfaces. Incoming packets to a NetFPGA board can be processed, possibly modified, and sent out on any of the four interfaces. Using a NetFPGA as a router allows us to precisely set the buffer sizes to a specific number of either bytes or packets, and the openness of the NetFPGA platform avoids the dangers of hidden buffers that may exist in commercial routers.

Traffic Monitoring: Obtaining the exact queue occupancy, the packet loss rate and the bottleneck link utilization over time is vital for experiments involving small packet buffers, which are extremely sensitive to packet timings. Unfortunately, to the best of our knowledge, no commercial router provides these metrics today. However, we added a module to the NetFPGA-based router that records an event each time a packet is written to, read from or dropped by an output queue [5]. Each event includes the packet size and the precise time that it occurred, with an 8 nanosecond granularity. These events are gathered together into event packets, which can be received and analyzed by the computer hosting the NetFPGA board or another computer on the network. The event packets contain enough information to reconstruct the exact queue occupancy over time and to determine the packet loss rate and bottleneck link utilization. The resulting data is at a previously unobtainable level of precision, which is invaluable for our experiments.

Packet Pacing: The *tiny buffer model* assumes network traffic is paced. This happens naturally if we have slow access links. As packets of a TCP flow cross the boundaries of slow access links (usually operating at a few Mbps) to high capacity core links (operating at tens of Gbps) they are automatically spaced out. However, if access links are fast, sources need to implement Paced TCP [23], which spaces out packets as they leave the source.

To study the necessity of the pacing assumption in the tiny buffer model, we perform experiments with *paced* and *non-paced* traffic. We use the Precise Software Pacer (PSPacer) [26] package to create the *paced* traffic by emulating multiple slower access links at each server. The PSPacer package is installed as a loadable kernel module for the Linux platform and provides precise network bandwidth control and traffic smoothing. In the *non-paced* experiments, we make no ef-



(a) (b)

Figure 4: Topologies of the network used in our experiments. The capacity of all links is 1Gbps.

fort to pace the traffic. Both experiments use an unmodified version of TCP, as implemented by the Linux network stack.

Packet Delay: To emulate the long Internet paths in a test-bed it is necessary to artificially delay every packet. We route all traffic through a host running NISTNet [7], a network emulator, to introduce propagation delays in the packets that flow from the clients to the servers. The NISTNet host is neither a client nor a server in the experiment.

The traffic at the delay emulator machine is monitored using tcpdump, which captures the headers of every packet. In some circumstances, under high-load tcpdump may not capture a packet, however we observe that the number of such missed packets is negligible: less than 0.1% of the total packets. We use these packet traces to measure the flow completion times and per-flow packet interarrival times.

Topologies: Due to limitations in a laboratory environments, we have to content ourselves with a limited set of topologies. Our experiments are conducted in two different topologies. In the first one, shown in Figure 4(a), a single point of congestion is formed, where packets from multiple TCP flows go through the NetFPGA router, and share a bottleneck link toward their destinations. Throughout different sets of experiments, we change the size of the output buffer in the NetFPGA router, and study how the buffer size affects the utilization and loss rate of the bottleneck link as well as the flow completion times. As mentioned before, the necessity of smooth traffic is investigated by changing the bandwidth of access links in the network. Our second topology is illustrated in Figure 4(b), where the main traffic (traffic on the bottleneck link) is mixed with some cross traffic, and is separated from the cross traffic before going through the bottleneck link. The goal is to determine how the main traffic is affected by the cross-cut traffic.

In our experiments, we mostly use the dumb-bell shaped topology. This is typical in the congestion control literature for two main reasons. First, there is a very small chance of having more than one point of congestion along any source-destination path. Once a link on the path is congested, flows on that path are limited in rate, and thus cannot increase their rates to congest another link along the path. Second, the client/server nodes and the links connecting them to the bottleneck link in the dumb-bell shaped topology can represent a path connecting source nodes to the bottleneck link in a real topology. Based on these two reasons, as well as the difficulties associated with performing accurate time-sensitive network experiments, we mainly consider dumb-bell shaped topology as a representative of more generic networks. However, we understand the limitations of this approach and that it may suffer from problems such as those noted in [12]. We hope to study the impact of other topologies with multiple points of congestion in the future.

Host Setup: In all sets of the experiments, we use TCP New Reno with the maximum advertised TCP window size set to 20MB, so that data transferring is never limited by the window size. The path MTU is 1500 bytes and the servers send maximum-sized segments. The aggregated traffic goes to the NetFPGA router over the access links, and from the router to the client network over the 1Gbps bottleneck link. The Linux end-hosts and the delay emulator machines are Dell Power Edge 2950 servers running Debian GNU/Linux 4.0r3 (codename Etch) and use Intel Pro/1000 Dual-port Gigabit network cards.

3.1.2 *Experiment Results*

In this section, we provide the results of our experimental studies on tiny buffers. We report network performance, including utilization of the bottleneck link, loss rate, and flow completion times for TCP traffic.

To study the impact of tiny buffers on performance, we run two sets of experiments, *paced* and *non-paced*, using the topology shown in Figure 4(a) with various router output-buffer sizes. Slower access network bandwidths are emulated using PSPacer with flows on a single machine grouped into classes. All flows belonging to the same class share a single queue with a service rate set to 200 Mbps. The size of the queue is chosen to be large enough (5000 packets) that there are no drops at these queues. Because the router's input/output links are 1Gbps, the emulated slow access links spread out the bursts in the router's aggregate ingress traffic. Initially, we observed an increased RTT during the *paced* experiments because our method of pacing forces packets to wait in queues which are serviced at a slower rate. To make the two cases more comparable we increase the delay

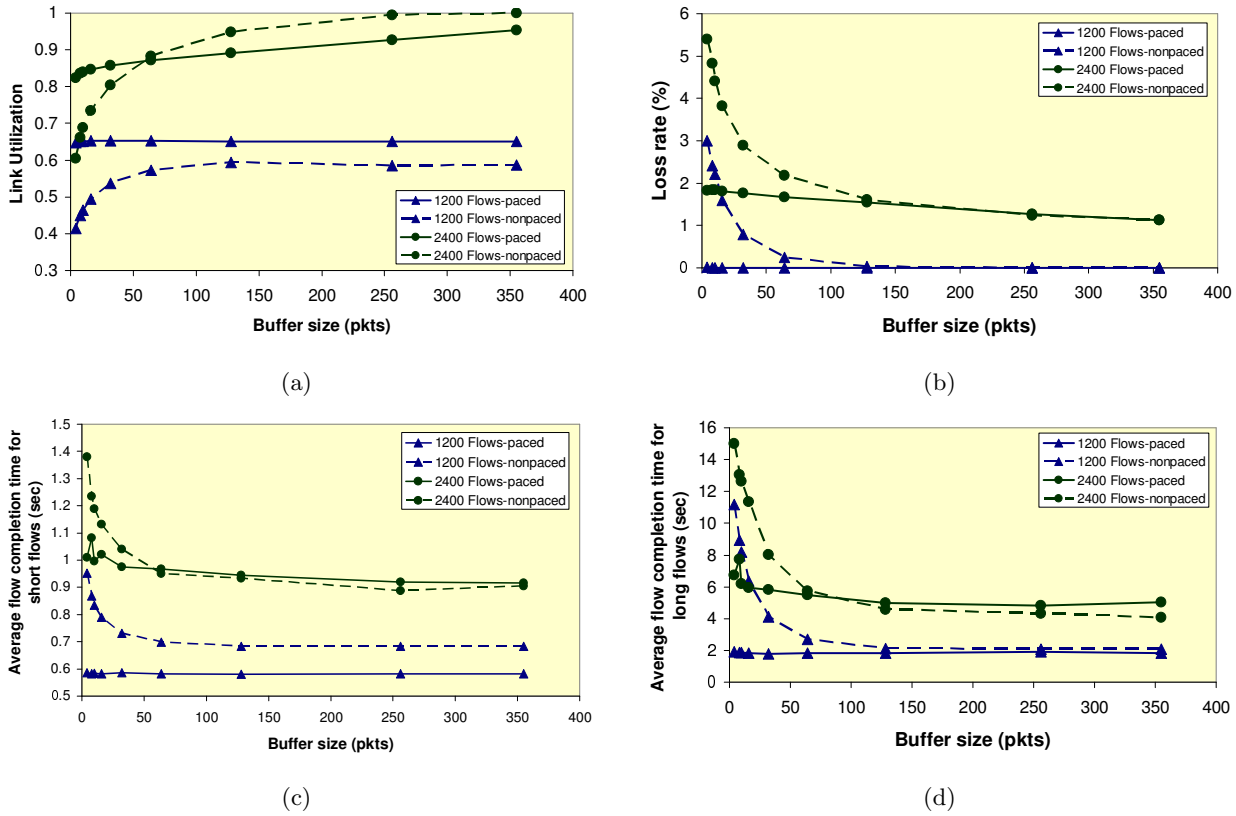


Figure 5: **(a) Link utilization as a function of buffer size for paced and non-paced experiments. (b) Loss rate as a function of buffer size for paced and non-paced experiments. (c) Average flow completion times for short-lived flows. (d) Average flow completion times for long-lived flows.**

introduced by NISTNet in the *non-paced* case so that the RTT is roughly 130 ms in both cases. The results shown in Figure 5 and analyzed in the following sections are the average of ten runs. The run time for each experiment is two minutes. To avoid transient effects, we analyze the collected traces after a warm-up period of one minute.

For each experiment we change the size of the output buffer in the NetFPGA router and investigate how the buffer size affects both the utilization and loss rate of the bottleneck link as well as the flow completion times.

Performance: Figure 5(a) shows the effect of changing the buffer size on the average utilization of the bottleneck link. The total number of flows sharing the bottleneck link is either 1200 or 2400. In both cases, pacing results in significantly larger utilization when the buffer size is very small – smaller than 10-50 packets. For example, with 1200 flows, pacing increases the utilization from about 45% to 65% when the buffer size is only 10 packets and it increases the utilization from about 65% to over 80% when we have 2400 flows in the system. This difference gets smaller as the buffer size increases.

Note that with 1200 flows, in both the paced and non-paced cases the link utilization does not achieve higher than 65%. This is the maximum offered load to the link since in this case the think time and file size distribution parameters do not create enough active users to saturate the bottleneck link (we observed that the number of active users is roughly half of the total users). Most of these active users are setting up many short connections which have very small throughput (about 6 packets per RTT) and thus they cannot necessarily saturate the link. Assuming that the average transmission rate is 6 packets per RTT, the total utilization would be roughly 600 (active) × 6 (packets) × 1500 × 8 (MTU) / 0.130 (RTT) which is only about 330 Mbps. The plot shows larger amount (about 650 Mbps) because not all of the flows are this small and hence there are some flows that are sending at a higher rate.

Another interesting observation is that with paced traffic, the link utilization appears to be independent of the buffer size while this is not the case for non-paced traffic. Thus, with paced traffic, there is no need to increase the buffering to achieve a certain link utilization.

Interestingly, we observe that in the 2400 flows case, increasing the buffer size makes the paced traffic's link utilization lower than the non-paced traffic's. This has already been observed and reported in [3] and [28]. The main reason is the trade-off between fairness and throughput. In the case where there are many paced flows and the buffer size is large, at the point where the buffer is almost full, each flow is sending paced packets at each RTT. These packets get mixed with other flows in a paced manner before arriving at the bottleneck router, and, if the buffer is already full and number of flows in the network is large, many of flows will experience a loss event and reduce their congestion window

by half. Although it seems fair that the packet drops should be spread among many flows, it may cause the bottleneck link's utilization to drop. However, with non-paced traffic, since packets arrive at the bottleneck link in large bursts, a smaller number of unlucky flows will hit the full buffer, will experience the loss event, and will reduce their rate. This is unfair for the few flows that reduce their rate, but it will keep the link utilization high.

Figure 5(b) compares the loss rate as a function of buffer size for paced and non-paced traffic. We can see that similar to link utilization, for paced traffic the loss rate is almost independent of buffer size, whereas it decreases exponentially with non-paced traffic. For tiny buffers, there is a notable reduction in the loss rate for both 1200 and 2400 flow cases. With 1200 flows, the link is not saturated and hence the loss rate for paced traffic is always less than 0.01%. However, with tiny buffers, non-paced traffic experiences around 2% drop on average.

Flow Completion Time: To address the question of how tiny buffers may affect an individual flow's completion time, we collect the start and finish times of all the flows going through the bottleneck link, and find the average completion time separately for short and long-lived flows. We define a short-lived flow to be a flow which never exits the slow-start mode. Long-lived flows are those which enter the congestion avoidance mode. We consider flows smaller than 50 KB (roughly 33 packets) as short-lived flows and flows larger than 1000 KB (roughly 600 packets) as long-lived flows. If there is no loss, it takes less than 6 RTTs for the short flows to be completed.

Figures 5(c) and 5(d) show the average flow completion times for short-lived and long-lived flows, respectively. As the plots show, with 1200 paced flows, the flow completion time is independent of the buffer size for both short and long-lived flows, whereas with non-paced traffic, increasing the buffer size reduces the flow completion times. In this case, the flow completion time of the paced traffic is always smaller than that of the non-paced traffic (for both short- and long-lived flows) and there is a notable difference between flow completion times for long-lived flows with tiny buffers. With 2400 flows, the average flow completion time of the paced traffic is always less than that of the non-paced traffic in the tiny buffers region and they are almost equal with buffers larger than 50 packets.

Cross-cut Traffic: To examine the effect of cross-cut traffic, we perform a set of experiments with the topology depicted in Figure 4(b). Each of the four servers communicate with exactly one of the four clients. Figure 6 shows the four network paths and identifies two of the paths as cross-cut traffic paths and two as main traffic paths. In the direction of data transmission, each of the bottleneck router's two incoming links are shared by one cross-cut traffic path and one main traffic path, whereas the outgoing links are not shared. Over different sets of experiments, we change the characteristics of main and cross-cut flows as well as the size of the output buffer in the NetFPGA bottleneck router, and we study the effect of cross-cut traffic on the interarrival time of packets at the bottleneck queue.

Figure 7(a) and (b) show four CDFs of packet interarrival times at the output queue of the bottleneck router, as reported by the NetFPGA router. Only packets that are stored in the queue are included in the statistics; packets dropped at the queue are ignored. In each experiment, there

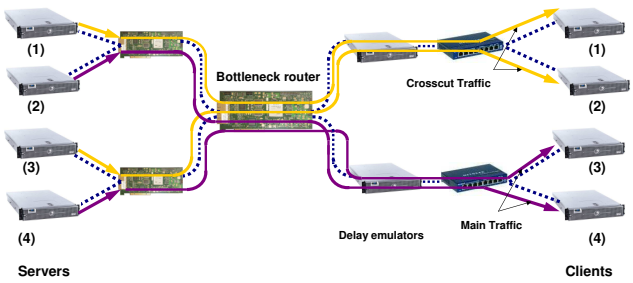


Figure 6: Illustration of cross-cut traffic and main traffic flows.

are at most 2400 simultaneous flows: 1200 main and 1200 cross-cut. We run the experiments with the router buffer size set to 32, 64, 128, 256 and 355 packets, but only the 32 packet case is shown in the figure. The results for the other buffers sizes are very similar and the minor differences do not affect the following analysis. Note that the x-axis in the figure is logarithmic.

The basic case in which no cross-cut traffic nor pacing is present , illustrated in Figure 7(a), shows that most of the interarrival times are 12µs, which is roughly the transfer time of MTU sized packets at 1Gbps. Because there are actually two input links with packets that may be destined to the same output link, this suggests the traffic on the two input links is bursty and unsynchronized. The 30% of the interarrival times that are less than 12µs correspond to packets arriving nearly simultaneously at the two input links.

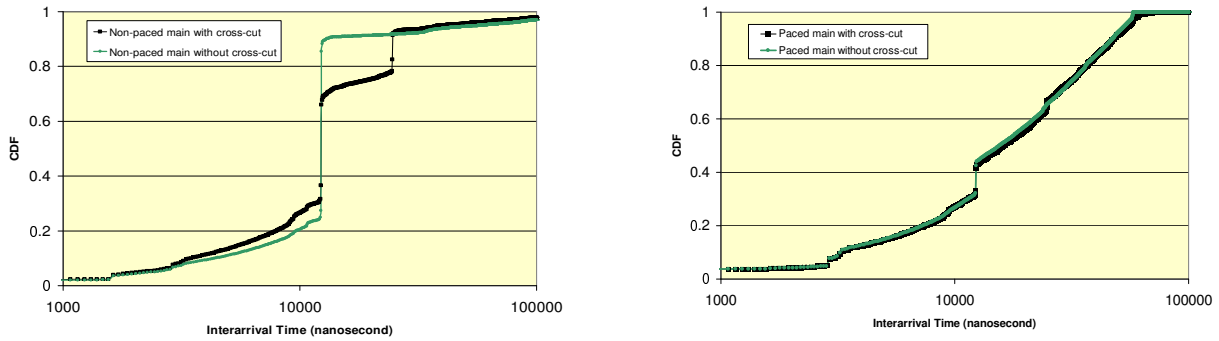
Comparing Figures 7(a) and 7(b) shows that the addition of cross-cut traffic is noticeable only when the main traffic is not paced: the two CDFs for paced main traffic in Figure 7(b) are very similar implying that presence of cross-traffic has little effect on the distribution of interarrival times of paced traffic while introducing cross-cut traffic when the main traffic is not paced, Figure 7(a), produces a pronounced second step in the CDF.

In the results shown here, the cross-traffic is not artificially limited or paced. Repeating the experiments and limiting the transmission rate of the cross-cut senders - to 200Mbps - does not result in any noticeable change of the results.

3.1.3 Sensitivity to Parameters

In network test-beds, where a handful of computers generate traffic representing the communication of hundreds or thousands of actual computers, the configuration of each traffic generator is critically important. Small changes to the software or hardware in a test-bed can have a large impact on the generated traffic and the experiment results, whereas changes to individual machines are unlikely to affect the aggregate traffic at an Internet core router. For an experiment's results to be relevant to the Internet's core, the validity of the traffic is paramount.

We investigate the effects of various software and hardware parameters in the context of buffer sizing experiments, and believe some of these parameters require careful tuning so that the artificially generated traffic mimics the properties of core network flows. For instance, recent network in-



(a) (b)

Figure 7: CDF of interarrival times at the output queue of the bottleneck router in cross-cut topology: (a) main traffic is non-paced, b) main traffic is paced.

terface cards have many advanced features that can impact the shape of the output traffic, or measurement of various performance metrics. Due to space limitations we describe only two such parameters here, which we believe have the highest impact on our results, and refer the interested reader to [5].

TCP Segmentation Offload (TSO): With TSO enabled, the task of chopping big segments of data into packets is done on the network card, rather than in software by the Operating System. The card sends out the group of packets that it receives from the kernel back to back, creating bursty and un-mixed traffic. Clearly, this makes the traffic bursty and highly impacts the results of buffer sizing experiments in a test-bed. Also, TSO *must* be disabled if packets are being paced in software. If TSO is enabled during our experiments, the gaps between packets added by PSPacer, described in Section 3.1.1, are only added between the large groups of packets sent to the network card, and the resulting traffic on the wire does not have a gap between each packet. Instead, it contains a group of packets back to back followed by a small gap, which is drastically different from the intended traffic pattern.

Interrupt Coalescing (IC): To lower the CPU's interrupt servicing overhead, network cards can coalesce the interrupts caused by multiple events into a single interrupt. With receiver IC enabled, the interarrival time of packets are changed. The network card will delay delivering packets to the operating system while waiting for subsequent packets to arrive. Not only does this affect packet timing measurements, but, due to the feedback in network protocols like TCP, it can also change the traffic's shape [19].

3.2 Sprint ATL Experiments

Our second set of tiny buffers experiments is conducted in collaboration with Sprint ATL. Figure 8 shows the topology of the emulated network, which is similar to the setting considered in tiny buffer sizing model [9]. The core of the experiments is a Juniper T640 router, whose buffers are modified throughout the study[5]. The router is connected to four different networks through four Gigabit Ethernet interfaces. Each cloud in Figure 8 represents one of these networks. The cloud on the left contains all the users/clients, and the three clouds on the right hold the servers. Each server belongs to one of the 99 different subnets (33 for each of the three server networks). The capacity of the access link connecting each server to the rest of the network is set to 15Mbps by default. The requests for file downloads flow from left to right (from clients to servers), and the actual files are sent back from right to left. In this direction, the router has three ingress and one egress line, which means by increasing the load we are able to create congestion on the link connection T640 router to the client network.

In practice, the clients and servers are emulated by two different boxes: Spirent Communications' Avalanche box plays the role of clients, and the Reflector box plays the role of servers (Figure 9). Each box has four Gigabit Ethernet Interfaces. Obviously, we use only one interface from the Avalanche box, and three interfaces from the Reflector box to connect the boxes to the T640 router. These links correspond to core links, and the link connecting the router to the Avalanche box is the target link. The access links are emulated by the Reflector box, which allows us to change the access link capacity to any desired value[6]. The delay associated with each access link is also emulated by the Reflector box. Since all other link delays are negligible, we can control the two-way propagation delay of packets by modifying these values. In the Appendix, we explain the results of evaluation tests on Avalanche's TCP traffic

Throughout the experiments, we use IPMon systems [2] to capture the headers of all the packets which go through the links connecting the router to the Avalanche and Reflector boxes. These headers are recorded along with high precision time-stamps. By matching the packet traces on ingress and egress lines of the router, we can measure the time each packet has spent inside the router, and thus, we can calculate the time-series representing the queue occupancy of

[5]We tried several other routers including Cisco GSR 12000, and Juniper M160. For the buffer sizes we were interested in this experiment, Juniper T640 seemed to be the most suitable choice (for details see [15]).

[6]In the Internet, access links are slow on client side. We found out Avalanche does not enforce rate limitations for incoming traffic, and had to push slow accesses to the server side in this experiment so that we can emulate the impact of slow access links. Avalanche has some other minor timing issues which are described in the Appendix.

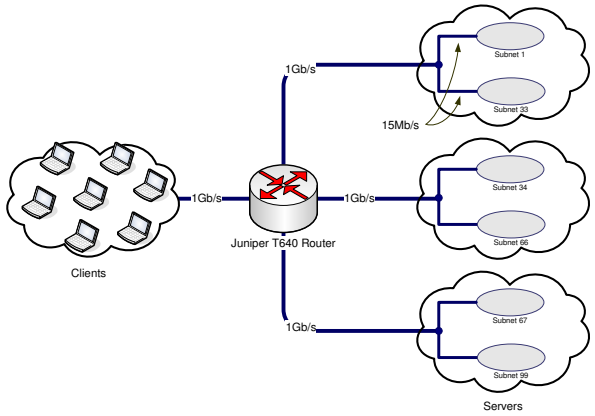


Figure 8: **Topology of the network used in experiments. The capacity of core links is 1Gbps, and the capacity of access links is 15Mbps.**

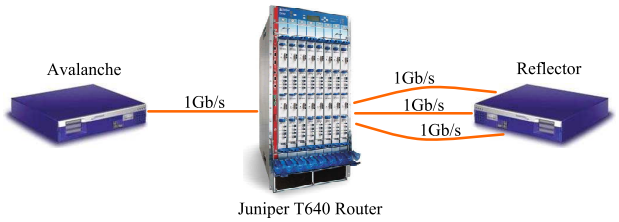


Figure 9: **Sprint ATL's tiny buffer experiment setup.**

the router. This also helps us identify any undocumented buffers inside the router. Such buffers could be fixed-delay buffers (*e.g.* part of a pipeline, or staging buffers), or could be additional FIFO queues.

3.2.1 Experiment Results

In this section we study the impact of changing buffer sizes on network performance. When allowed by our testing equipment, we also study the effect of changing some other network properties (like traffic patterns, access link properties, number of flows, and others) on buffer sizing results. Due to lack of space, we review some of our results here, and refer the interested reader to [15] for details.

Performance: We reduce the buffer sizes on the router from 8500 packets to just 50 packets, and measure the throughput, drop rate, and delay observed by individual packets as performance metrics. At 1Gbps line speed, with an RTT of 50ms, 8500 packets is about twice the bandwidth-delay product, and 50 packets lies in the range of the tiny buffer sizing model.

In this experiment we increase the number of users from 0 to 600 during a period of 50 seconds, and keep the number of users at 600 for 5 minutes, measuring throughput, delay, and drop rate during this time interval[7]. Each user downloads a 1MB file from an ftp server. Once the file download is completed the user immediately starts downloading another file. The average RTT of the system is 50ms (more precisely $15 + U[0, 20]$ on each of the forward and reverse paths), and the capacity of access links connecting servers to the system is 15Mbps. Both the server and clients have an advertised congestion window size of 16KB.

Figure 10(a) illustrates throughput as a function of time for various buffer sizes, and Figure 10(b) represents the average throughput for different buffer sizes. If we consider the overhead of packet headers, the maximum throughput we can get is about 950Mbps. We can see that a buffer size between 8500 and 922 packets, gives a throughput of about 100%. This is the range between the rule-of-thumb and the small buffer model. When we push the buffer size to 192, 63, and 50 packets, which is in the range of tiny buffers model, the throughput goes down by 10%, as predicted theoretically. The average level of throughput is maintained very smoothly throughout the experiments, as seen in Figure 10(a).

Figure 10(c) shows that on average packets go through a delay of 155μs to 4.5ms (equivalent to 13 and 375 packets) for buffer sizes between 50 and 8500 packets. The average packet delay is considerably smaller than the maximum buffer size when it is set to 8500 packets. This is very similar to what we observed in Level 3 Communications' network. The average delay increases as the buffer size is increased from 50 to 1017 packets, and is slightly reduced for 8500 packets. Since the packet drop rate is close to zero when buffer size is set to 1017 or 8500 packets, we expect these two to have similar average delays, and the observed reduction in average delay might be a result of activation/deactivation of some hidden buffer inside the router.

For buffer sizes between 922 and 8500 packets, the drop rate is very close to zero (Figure 10(d)). As expected, in these cases utilization is close to 100%. For smaller buffers we see a packet drop rate of up to 0.75%; only 0.25% more than a M/D/1 queue of similar size and arrival rate, confirming once more the smoothness of traffic going through the router.

The impact of increasing network load: In the previous experiment, parameters were chosen so that the effective system load is very close to 100%. What happens if we keep increasing the load? Does the throughput of the network collapse as a result of congestion? This is a valid concern, and to find out the answer we perform another set of experiments. This time, we vary the *potential load* of the system between 25% and 150% [8]. We control the system load by limiting the access link rates and advertised congestion window, and by changing the number of end users from 150 to 1200.

Figure 11(a) plots the throughput of the system as a function of load and various buffer sizes in this scenario. For any given buffer size increasing the potential load monotonically increases the throughput. For large buffers, the throughput reaches 100% (950Mbps) when the potential load is 100%, and remains at that level for increased potential load. For smaller buffers, the throughput reaches 90%-95% as we increase the potential load from 25% to 100%, and remains

[7]The number 600 of users is chosen so that the effective load of the system is about 100%.

[8]The potential load of the system is defined as the utilization achieved when the bottleneck link capacity is increased to infinity, and when the throughput is limited by other factors (like the maximum congestion window size, RTT, and access link capacities)

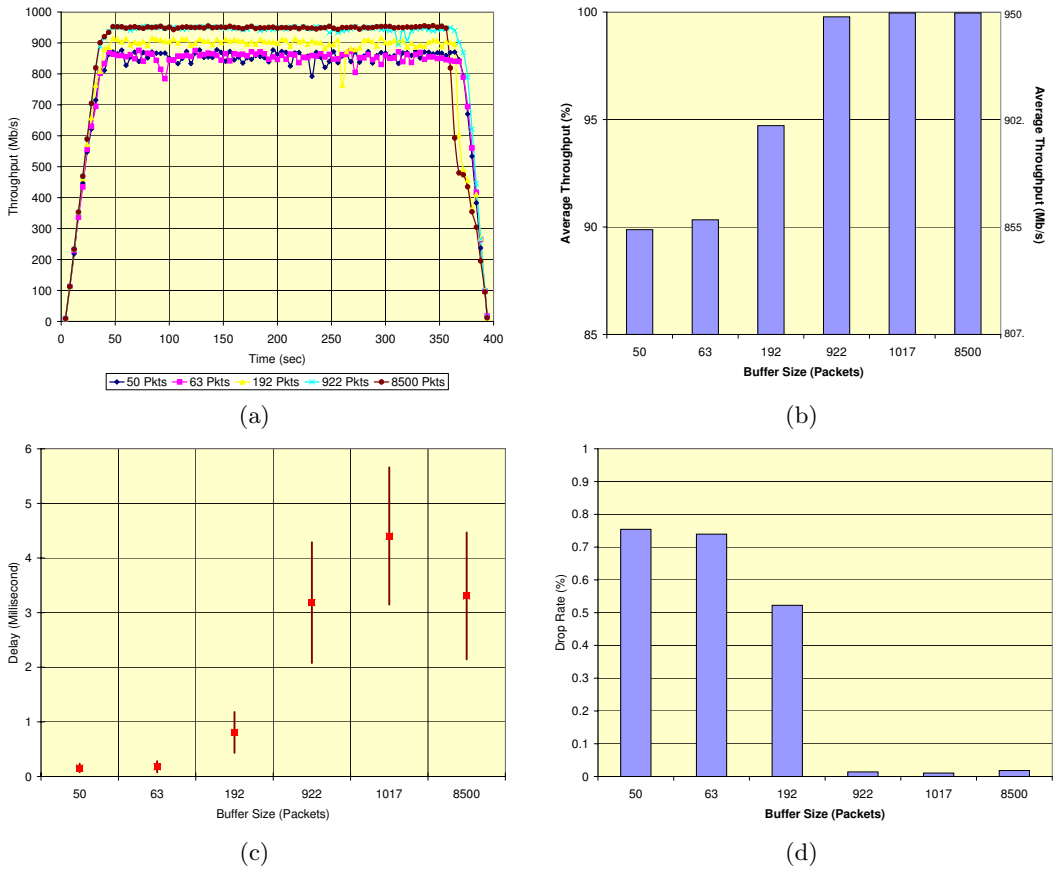


Figure 10: (a) Throughput vs. time for various buffer sizes. (b) Average throughput vs. buffer size. (c) Delay statistics vs. the buffer size. The square represents the average delay and the bar represents the standard deviation. (d) Drop rate vs. buffer size.

almost fixed beyond that point. This is good news in the sense that we do not see a collapse in throughput as a result of increased congestion. For a core network a potential load beyond 100% is very unlikely given that core networks are usually highly over-provisioned.

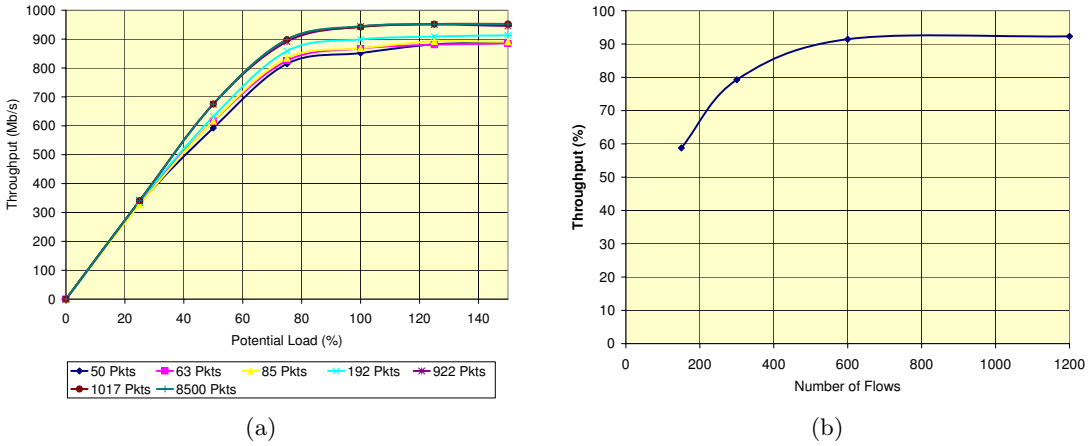
Performance as a function of the number of flows: We would like to see whether the number of flows affects the performance of the system. We cannot simply modify the number of flows, since the potential load to the system changes with the number of flows. To fix this problem, we adjust the maximum congestion window size to keep the potential load fixed, when modifying the number of flows in the network. For 150 flows, the maximum congestion window size is set to 64KB. As we increase the number of flows to 300, 600, and 1200, we reduce the maximum congestion window size accordingly (to 32KB, 16KB, and 8KB). The buffer size is set to 85 packets in all these experiments.

Figure 11(b) illustrates the changes in network throughput as we increase the number of flows. When the number of flows is very low (*i.e.* 150-300) the system throughput is significantly less than 100%. Even when we increase the congestion window size (to increase the potential load), the system throughput is not significantly increased. This can be explained by tiny buffer sizing model as follows: when the number of flows is low, we will not have a natural pacing as a result of multiplexing, and therefore, the throughput will not reach 100%[9]. When the number of flows is large (*i.e.* 600-1200), the system throughput easily reaches 90-95%, independent of the number of flows.

Increasing the number of flows beyond a few thousand can result in a significant reduction in throughput, as the average congestion window size becomes very small (2-3 packets or even less), resulting in a very high drop rate and poor performance [8, 18]. This problem is not associated with tiny buffers, and unless we significantly increase the buffer sizes even more than the rule-of-thumb it would not be resolved. We believe this is a result of poor network design and increasing the buffer sizes is not the right way to address such issues.

We also conducted experiments to study the impact of tiny buffers on performance in the presence of different flow sizes, various access link capacities, and different distributions of RTTs. Our results show the performance of a router with tiny buffers is not highly impacted by changes in these parameters. For the sake of space, we omit the details and refer the interested reader to [15].

[9]This problem can be fixed by modifying traffic sources to use Paced TCP. Here we do not have the tools to test this. The commercial traffic generator which we use does not support Paced TCP.



(a) (b)

Figure 11: (a) Throughput vs. potential load for different buffer sizes. (b) Throughput vs. the number of flows.

3.3 Other Tiny Buffer Experiments

Our tiny buffer experiments have been verified independently in other test-beds at Alcatel-Lucent Technologies and Verizon Communications. The only operational network experiment we have done is performed in collaboration with Internet2. We reduced the buffer size of a core router down to 50 packets, while measuring the performance through active flow injections and passive monitoring tools. Our measurements of throughput and packet drops did not show any degradation. We note that Internet2 operates its network at very low utilization (20-30%); not an ideal setup for buffer sizing experiments. For more details on these experiments, we refer the reader to [15].

Again, all these experiments seem to agree with the tiny buffer sizing model. Clearly, this model has more strict assumptions compared to the small buffer model, and one should be extremely careful to make sure the assumptions hold in an operational network. As discussed in theory, in a network with slow access links the assumptions seem to be satisfied. However, not all backbone traffic comes from slow access links. We conclude that tiny buffer results hold as long as traffic injected to the network is not overly bursty.

4. CONCLUSIONS

The small buffer model ($O(C/\sqrt{N})$) appears to hold in laboratory and operational backbone networks – subject to the limited number of scenarios we can create. We are sufficiently confident in the $O(C/\sqrt{N})$ result to conclude that it is probably time, and safe, to reduce buffers in backbone routers, at least for the sake of experimenting more fully in an operational backbone network. The tiny buffer size experiments are also consistent with the theory. One point that we should emphasize is the importance of the pacing constraint in tiny buffers experiments. As indicated by theory, pacing can happen as a result of slow access links or by modifying sources so as to pace the traffic injected to the network. We find that as long as this constraint is satisfied, we get a good performance. We also find that some network components (like network interfaces cards) might have features that reduce pacing along the path. Therefore, one should be very careful and aware of such details if tiny buffer sizing result is to be applied in practice.

Acknowledgements

We would like to thank Jean Bolot, Ed Kress, Kosol Jintaseranee, James Schneider, and Tao Ye from Sprint Advanced Technology Lab, Stanislav Shalunov from Internet2, Shane Amante, Kevin Epperson, Nasser El-Aawar, Joe Lawrence, and Darren Loher from Level 3 Communications, T.V. Lakshman, Marina Thottan from Alcatel-Lucent, Pat Kush, and Tom Wilkes from Verizon communications for helping us with these experiments. We would also like to thank Guido Appenzeller, Sara Bolouki, and Amin Tootoonchian for discussions and help.

5. REFERENCES

[1] NetFPGA project. http://yuba.stanford.edu/NetFPGA/.

[2] Sprintlink ip backbone measurements. IPMON Sprint Labs IP Monitoring Project, http://ipmon.sprint.com.

[3] 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.

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

[5] N. Beheshti, Y. Ganjali, M. Ghobadi, N. McKeown, J. Naous, and G. Salmon. Time-sensitive network experiments. Technical Report TR08-SN-UT-04-08-00, University of Toronto, April 2008.

[6] R. Bush and D. Meyer. RFC 3439: Some Internet architectural guidelines and philosophy, December 2002.

[7] M. Carson and D. Santay. NIST Net: a Linux-based network emulation tool. *SIGCOMM Comput. Commun. Rev.*, 33(3):111–126, July 2003.

[8] A. Dhamdhere and C. Dovrolis. Open issues in router buffer sizing. *ACM Sigcomm Computer Communication Review*, 36(1):87–92, January 2006.

[9] M. Enachescu, Y. Ganjali, A. Goel, N. McKeown, and T. Roughgarden. Routers with very small buffers. In *Proceedings of the IEEE Infocom*, Barcelona, Spain, April 2006.

[10] A. Erramilli, O. Narayan, A. Neidhardt, and I. Saniee. Performance impacts of multi-scaling in wide area TCP/IP traffic,. In *Proceedings of the IEEE Infocom*, Tel-Aviv, Isreal, March 2000.

[11] A. Feldmann, A. Gilbert, P. Huang, and W. Willinger. Dynamics of IP traffic: A study of the role of variability and the impact of control. In *Proceedings of the ACM Sigcomm*, pages 301–313, Cambridge, Massachusetts, August 1999.

[12] S. Floyd and E. Kohler. Internet research needs better models. In *Proceedings of HotNets–I*, October 2002.

[13] C. Fraleigh, F. Tobagi, and C. Diot. Provisioning IP backbone networks to support latency sensitive traffic. In *Proceedings of the IEEE Infocom*, San Francisco, California, April 2003.

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

[15] Y. Ganjali. *Buffer Sizing in Internet Routers*. PhD thesis, Stanford University, Department of Electrical Engineering, March 2007.

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

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

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

[19] R. Prasad, M. Jain, and C. Dovrolis. Effects of interrupt coalescence on network measurements. *Passive and Active Measurements (PAM) conference*, April 2004.

[20] R. Prasad and M. K. Thottan. Inconsistencies with Spirent's TCP implementation, 2007.

[21] R. S. Prasad and C. Dovrolis. Measuring the congestion responsiveness of Internet traffic. *PAM*, 2007.

[22] R. S. Prasad, C. Dovrolis, and M. Thottan. Router buffer sizing revisited: the role of the output/input capacity ratio. In *CoNEXT '07: Proceedings of the 2007 ACM CoNEXT conference*, pages 1–12, New York, NY, USA, 2007. ACM.

[23] 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.

[24] V. Ribeiro, R. Riedi, M. Crouse, and R. Baraniuk. Multiscale queuing analysis of long-range dependent network traffic. In *Proceedings of the IEEE Infocom*, Tel-Aviv, Isreal, March 2000.

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

[26] R. Takano, T. Kudoh, Y. Kodama, M. Matsuda, H. Tezuka, and Y. Ishikawa. Design and evaluation of precise software pacing mechanisms for fast long-distance networks. *3rd Intl. Workshop on Protocols for Fast Long-Distance Networks (PFLDnet)*, 2005.

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

[28] M. Wang and Y. Ganjali. The effects of fairness in buffer sizing. In *Networking*, pages 867–878, 2007.

APPENDIX

Harpoon Traffic Generation Evaluation: We run a large set of experiments to evaluate Harpoon's TCP traffic. Mainly, we want to verify whether Harpoon's traffic – which is generated on a limited number of physical machines in our test-bed – can model the traffic coming from a large number of individual TCP sources. If that is the case, then we should expect to see the same traffic pattern when a fixed number of flows are generated on one single machine as when the same number of flows are generated on multiple physical machines. In particular we want to know how the flows are intermixed, if a few physical machines are generating them. Our results show that the aggregate traffic becomes less mixed as the number of physical machines becomes smaller.

Figure 12 shows the topology of an experiment run to compare the traffic generated by four machines versus the traffic generated by two machines. In these experiments, a number of connections are created between each pair of physical machines (denoted by the same numbers in the figure). In the first set of experiments, we create a total number of flows (connections) on four pairs of source-destination machines. In the second set, we repeat this experiment by creating the same total number of flows on only two pairs of source-destination machines (machines numbered 1 and 4 in figure 12), which requires doubling the number of flows generated by each single machine. The goal is to see how alike the traffic of the two experiments are. In this setup all links run at 1Gbps bandwidth, and 100ms delay is added by NISTNet to the ack packets. All packets are eventually mixed on one single link (connecting the NetFPGA router to the NISTNet machine). We do our measurements on this shared link.

Figure 13 compares the percentage of successive packets (in the aggregate traffic) which belong to the same flow versus the total number of flows. The red (darker) bars correspond to the two-source experiment, and the blue (lighter) bars correspond to the four-source experiment. The plot shows the results for four different settings: Buffer size at

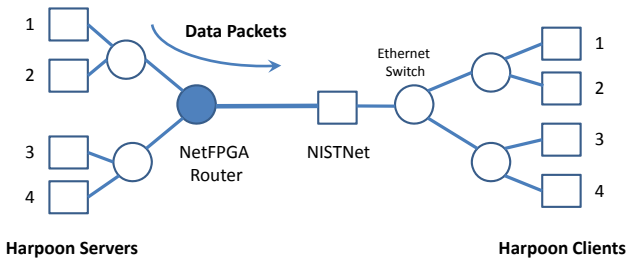


Figure 12: Harpoon Evaluation Topology

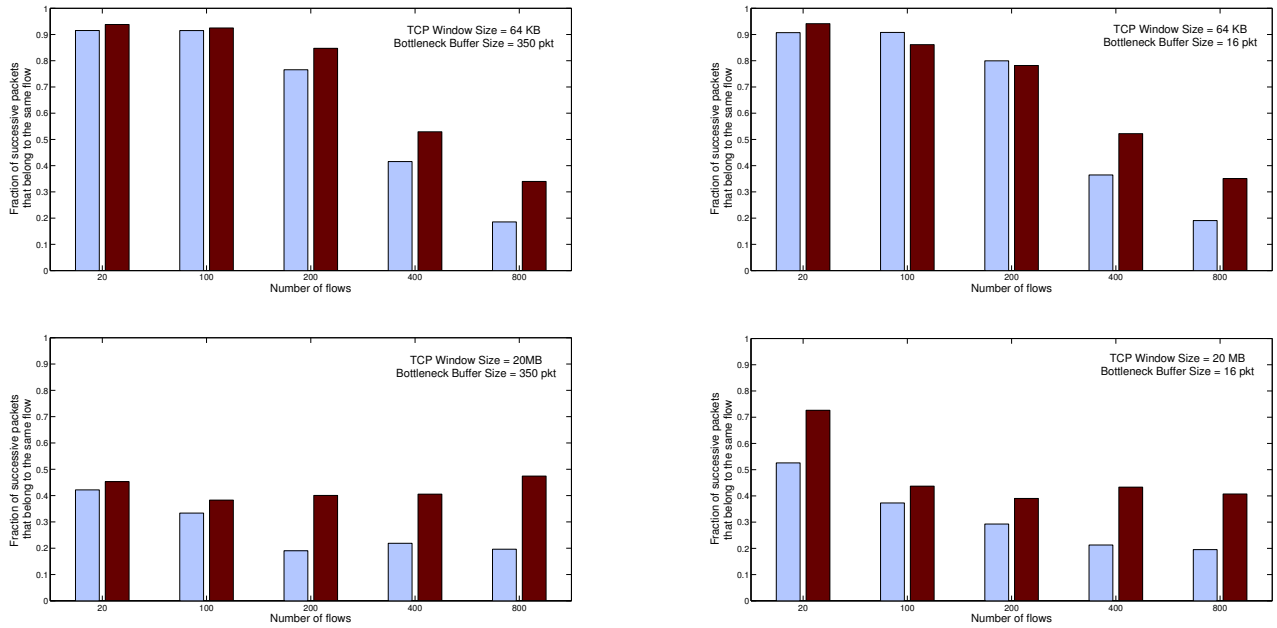


Figure 13: **Fraction of successive packets that belong to the same flow versus the total number of flows. Red (darker) bars correspond to the 2-source experiment, and blue (lighter) bars correspond to the 4-source experiment.**

the shared link being set to 16 and 350 packets, and maximum TCP window size being set to 64KB, and 20MB.

As it can be seen, in all cases packets of individual flows are less likely to appear successively in the aggregate traffic when they are generated by four machines. The difference between the red and the blue bars grows larger as the number of total emulated connections increases. The same result holds when we change the RTT to 50ms and to 150ms.

In our buffer sizing experiments with Harpoon generated traffic, we emulated only a few hundred active users on a single machine. This traffic would have been better mixed (and consequently less bursty), had it come from individual TCP sources. Nevertheless, there is not a noticable difference in the drop rate of packets at the bottleneck link when we change the number of physical machines generating the traffic.

Avalanche Traffic Generator Evaluation: In [20], Prasad *et al.* show some discrepancies with Spirent's TCP implementation. The differences they observe are between flows generated by Spirent's traffic generator and those by NewReno TCP with SACK implemented in Linux 2.6.15. The main problems documented in [20] are the following: Firstly, it has been observed that Spirent's TCP does not do fast retransmit when the receiver window size is larger than a certain value. Secondly, TCP implementation of Spirent does not implement SACK or NewReno. And finally, the RTO estimation is not consistent and does not seem to conform to RFC 2988.

We evaluated the Avalanche/Reflector boxes to see if they generate accurate TCP Reno traffic. We started with a single TCP flow, then changed several parameters (link capacity, delay, congestion window size, packet drop rate, etc.) and manually studied the generated traffic. We concluded that packet injection times are accurate, except for a difference of up to 120μs (which can be attributed to processing times and queueing delays in the traffic generators). We also compared the traffic patterns generated by the Avalanche /Reflector boxes with patterns generated by the *ns-2* simulator in carefully designed scenarios. While *ns-2* is known to have problems of its own, we wanted to identify differences between experimental results and simulation. We found minor differences between *ns-2* and Avalanche output traffic. However, we believe these differences do not have any impact on buffer sizing experiments[10].

[10]These difference were reported to Spirent Communications, and some of them have been resolved in their current systems. We are working with them to resolve the remaining issues.