Sizing Router Buffers (Redux)

Nick McKeown
Stanford University
nickm@stanford.edu

Guido Appenzeller
Yubico, Inc.
guido@appenzeller.net

Isaac Keslassy
Technion, Israel
isaac@ee.technion.ac.il

This article is an editorial note submitted to CCR. It has NOT been peer reviewed.
The authors take full responsibility for this article's technical content. Comments can be posted through CCR Online.

ABSTRACT

The queueing delay faced by a packet is arguably the largest source of uncertainty during its journey. It therefore seems crucial that we understand how big the buffers should be in Internet routers. Our 2004 Sigcomm paper revisited the existing rule of thumb that a buffer should hold one bandwidth-delay product of packets. We claimed that for long-lived TCP flows, it could be reduced by $\sqrt{N}$, where N is the number of active flows, potentially reducing the required buffers by well over 90% in Internet backbone routers. One might reasonably expect that such a result, which supports cheaper routers with smaller buffers, would be embraced by the ISP community. In this paper we revisit the result 15 years later, and explain where it has succeeded and failed to affect how buffers are sized.

CCS CONCEPTS

• **Networks** → **Intermediate nodes**;

KEYWORDS

Router, Buffer Sizing, Switch ASIC

1 A BRIEF HISTORY OF BUFFER SIZES

We first started wondering "*How big should a router buffer be?*" in 2002. We were curious for a few reasons. First, every packet-switched router must have a buffer to hold packets during times of congestion, and queueing delay through the buffer is the biggest cause of uncertainty in the end-to-end latency. It seems important we understand how big the buffer should be, else how can network owners correctly design and configure their routers? Second, the size has significant implications on how routers are designed. If the buffer is small enough to fit on a single switch ASIC, the router is smaller, simpler, cheaper and consumes less power. Third, at the time, most router vendors claimed you need a buffer equal to the bandwidth-delay product, which led to a separate class of routers (some costing millions of dollars) for Internet service providers. Curious, we polled five well-known researchers to ask them why a router needs a bandwidth-delay product of buffering ... and we received five different explanations! Clearly, as a community, the buffer size was not well understood.

At the time, the popular rule-of-thumb came from the 1994 paper by Villamizar and Song [19]. They experimentally evaluated the buffers needed to keep a 40 Mb/s link fully utilized, using a small number of TCP flows. They concluded that a router with port speed C, carrying TCP flows with average minimum round-trip time RTT_{min},[1] needs a buffer

$$B \geq C \cdot RTT_{min} \tag{1}$$

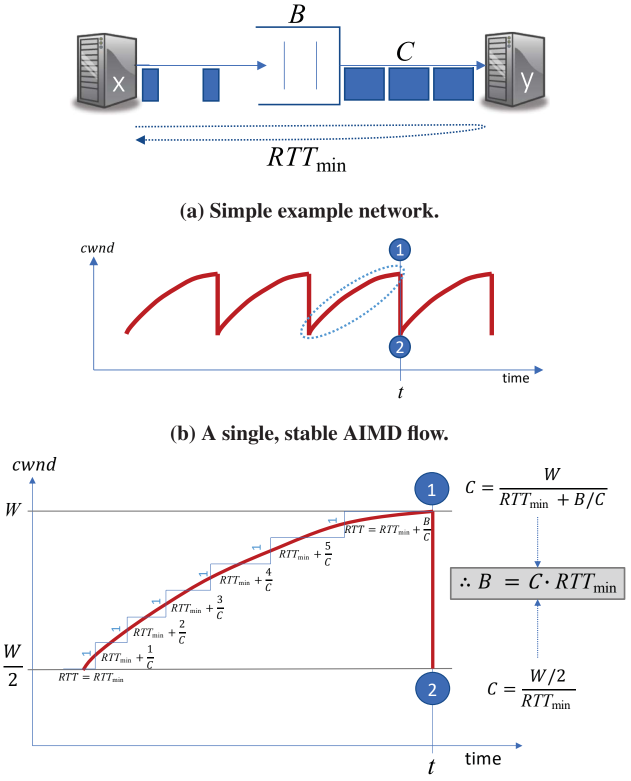
in order to maintain full utilization of the bottleneck link. This rule was widely used by Internet service providers and router manufacturers when configuring and designing routers, who argued that any less buffering would be too risky. But the cost of the rule was substantial; for example, a 100Gb/s link carrying flows with an average RTT of 200ms would, according to the rule, require 2.4GBytes of buffer memory.[2] Not only does this mean a forwarding ASIC has to devote half its capacity to accessing off-chip memory, it means packets can potentially be delayed by an extra 200ms.

It is fairly easy to see that the rule-of-thumb is only correct in very limited circumstances, for example in a network with a single TCP flow (for versions of TCP using additive-increase and multiplicative-decrease, or AIMD, *e.g.,* New Reno). To understand why, we need to understand how buffers and TCP interact.

Consider the simple network in Figure 1(a) carrying one TCP flow through a router with a bottleneck link of rate C and buffer of size B. The evolution of the TCP congestion window sawtooth is shown in Figure 1(b) after the flow has settled into the familiar AIMD mode; we expand the view of one sawtooth in Figure 1(c). When the window size is at its maximum, W, the buffer is full and $RTT = RTT_{min} + B/C$ (point (1) on the figure). When the acknowledgments are successfully received, the sender increases the window size by one and the buffer overflows in the next round-trip time. A packet is lost and the client halves its window, hence the window size becomes $W/2$. With the new, smaller window, the sender must stop sending until it has received enough acknowledgements to catch up to the new, smaller window size of $W/2$. The buffer serves as a reservoir of packets during the pause. For the bottleneck link to remain busy during the pause, the buffer must hold enough packets so that it doesn't go empty before the client resumes sending again and the next packet arrives to the queue. If the buffer is *just* big enough, then at the moment the next packet arrives, the buffer will have *just* gone empty. Hence at point (2) on the figure, $RTT = RTT_{min}$, and given that the bottleneck link is busy at both times, the sending rate satisfies $C = W/RTT$, hence $C = \frac{W}{RTT_{min}+B/C} = \frac{W/2}{RTT_{min}}$, which means $B = C \cdot RTT_{min}$. If we choose a smaller buffer, the bottleneck link will go idle after a packet drop and we lose utilization of the link; if we make the buffer bigger, the bottleneck link will always be busy, but the buffer will never go empty and packets will unnecessarily encounter a fixed additional delay. Put another way, there is no

[1] The minimum round-trip time is the fixed component when the queues are empty. The average minimum is the average across the flows' minimum round-trip times.

[2] For comparison, the largest on-chip SRAM buffer today is less than 100MBytes.


B
C
x
y
RTT_{min}
cwnd
1
2
t
time
cwnd
W
$\frac{W}{2}$
$RTT = RTT_{min}$
$RTT_{min} + \frac{1}{C}$
$RTT_{min} + \frac{2}{C}$
$RTT_{min} + \frac{3}{C}$
$RTT_{min} + \frac{4}{C}$
$RTT_{min} + \frac{5}{C}$
$RTT = RTT_{min} + \frac{B}{C}$
1
1
1
1
1
1
1
$C = \frac{W}{RTT_{min} + B/C}$
$\therefore B = C \cdot RTT_{min}$
$C = \frac{W/2}{RTT_{min}}$
1
2
t
time

(a) Simple example network.

(b) A single, stable AIMD flow.

(c) The sender's rate is always C because the link is fully utilized. Therefore, it is C both just before the drop occurs at (1) as well as when the sender resumes sending at (2), when the buffer is empty.

Figure 1: An example of a network carrying a single TCP flow, illustrating why we need $B \geq C \cdot RTT_{min}$ to keep the bottleneck link busy all the time.

benefit to making the buffer bigger: It will increase latency and the congestion control algorithm will be more sluggish because it has to deal with a longer response time.[3]

A network carrying only one flow is obviously not very common. What happens when we have multiple flows? If TCP flows are synchronized (*i.e.,* they all experience loss events within the same RTT) then the same rule of thumb applies. The synchronization behavior of TCP is not well understood, but simulation as well as experimental evidence indicates that for a small number of TCP flows with similar RTTs and no external source of randomization, synchronization can occur, which helps explain the origin of the rule of thumb [19].

But an Internet core router, or a data-center switch, carries thousands (sometimes hundreds of thousands) of flows and many external sources of randomization break the synchronization between the TCP flows. During any RTT interval, packets belonging to a small fraction of the flows will happen to arrive to a full queue, be dropped and cause their sender to halve its window size. Because only a small fraction of flows reduce their window size, the occupancy of the buffer changes little; in fact, under congestion, unlike the single flow case, it will remain almost completely full all the time.

The main observation of our Sigcomm 2004 paper [2] is that with many flows a central limit theorem applies. The paper proves that for N desynchronized flows, the bottleneck link can be kept fully utilized with a buffer size of approximately

$$B \geq \frac{C \cdot RTT_{min}}{\sqrt{N}} \qquad (2)$$

and verifies it with multiple simulations, and a small test network in the lab. In this paper we refer to Eqn 2 as the *Small Buffers* result.[4]

If correct, this result can have quite staggering consequences on the design of Internet routers. For example, a 1 Tb/s ISP router carrying one TCP flow with an RTT_{min} of 100ms would require 12.5 GB of buffer and off-chip buffering. If it carries 100,000 flows, then the buffer can be safely reduced to less than 40MB, reducing the buffering and worst-case latency by 99.7%. With Small Buffers, the buffer would comfortably fit on a single chip switch ASIC.

1.1 Are Small Buffers OK in practice?

There was natural skepticism about our 2004 claim, and Eqn. 2 in particular; after all, it was surprising to think that we could eliminate almost all the buffers in a big public network and potentially *improve* application behavior. There were of course some worried employees at a few router manufacturers who felt the justification of a separate class of ISP routers was under threat.[5] We knew that big network operators would need more solid evidence before adopting the new guideline in their networks. So we ran some experiments.

Level 3 Backbone. The first measurement paper, led by Neda Beheshti and Yashar Ganjali, reported results collected in the Level 3[6] public Internet backbone [5]. Fortuitously (for us, not them), Level 3's network was extremely congested and in urgent need of faster links, and they were about to upgrade links from 2.5Gb/s to 10Gb/s. They approached us to see if we could collaborate to try and understand how big their buffers needed to be. The financial consequences were large: If they could justify purchasing lower-priced enterprise switches with smaller on-chip buffers, they could save hundreds of millions of dollars in their network by avoiding expensive routers with large (and possibly unnecessary) packet buffers. Level 3 had the clever idea of an apples-to-apples comparison across the three links shown in Figure 2. Three 2.5Gb/s links carried nominally identical traffic loads, which was achieved using a static hash function. During the busiest period of the day, their 2.5Gb/s links were sustaining over 95% load and dropping packets. If they set different buffer sizes just

[3] As an interesting intellectual aside, it is interesting to note that Eqn 1 is a somewhat arbitrary artifact of the design decision for TCP to divide its window in half when it detects a lost packet. If instead the designers had chosen the TCP window to divide by $k > 1$ instead, then Eqn 1 becomes: $B \geq (k-1) \cdot C \cdot RTT_{min}$. Furthermore, if a sender knows or measures RTT_{min} and picks $k = 1 + a/RTT_{min}$ for constant a, then the buffer size becomes $B \geq aC$ and is independent of RTT. For example, for a 10Gb/s link and $RTT_{min} = 100$ms, then Eqn 1 says we need 1Gbit of buffering. If we pick $k = 1.14$, it drops to 140Mbits. With $k = 1 + 0.1/RTT_{min}$, then it drops even further to 50Mbits.

[4] Some other authors refer to it as the *Stanford Model*.

[5] A VP at one router vendor even approached us, offering to fund us to discredit our own results. It was one of those occasions when you know you have hit a raw nerve and are onto something.

[6] Level 3 was one of the largest commercial Tier-1 ISPs, acquired by CenturyLink in 2016.

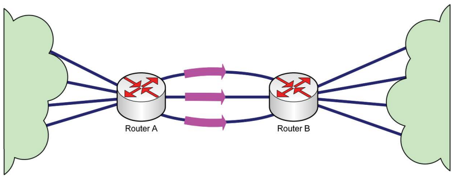


Figure 2: (Figure 1 reproduced from [5]). Network for buffer sizing experiments in Level 3 Communications' backbone network. Traffic from Router A to Router B was divided equally among the three 2.5Gb/s links using a static hash function to balance TCP flows.

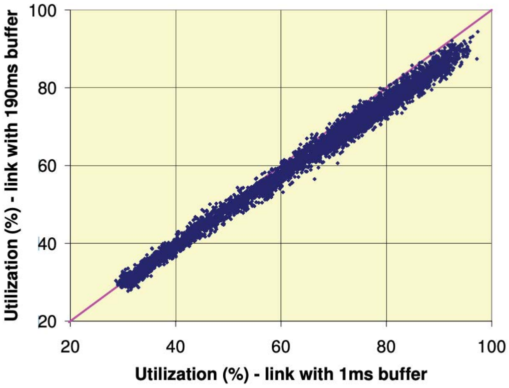
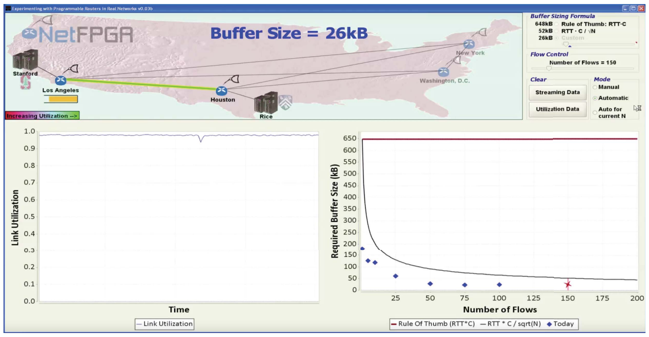


Figure 3: (Figure 3c reproduced from [5]). When Level 3 reduced buffers from 190ms to 1ms, there was no measurable loss in throughput. In fact, a slight increase.

before the three links leaving Router A, then they could look at the consequences for the same (or at least very similar) load. The router vendor recommended a default of 190ms per-link (60MB). We reduced the buffer sizes to 10ms (3MB), 5ms (1.5MB), 2.5ms (750KB) and 1ms (300KB) and ran experiments lasting several weeks. Based on Eqn. 2, we expected to need a buffer size of 2-6ms (based on an estimate of 10,000 flows).

Figure 3 reproduces a graph from the 2008 measurement paper [5], comparing the utilization of the link with the default buffers (190ms) against the link with the smallest experimental buffers (1ms). The most important thing to notice is that there is no reduction in throughput when reducing the buffer by 190-fold. In fact, there is a small *increase* in throughput, which was not explained by packet loss or imbalance in the hash function. Our suspicion (which we could not verify with passive observations at the router) is that the TCP control loop became tighter and faster, because of lower latency, and therefore could achieve *higher* link utilization with smaller buffers. If true, it means that TCP works *better* with small buffers in this case. As we will see later, we came across additional, anecdotal evidence in 2018 to support this observation in a big video streaming service.

Internet 2. In order to accurately measure how throughput varies with the number of flows in a real production network, we ran experiments in the Internet2 backbone interconnecting US universities. To make sure we could precisely control the buffer size (and change it in real-time), and accurately measure N and packet loss, we built our own NetFPGA-based 4-port 1 Gb/s routers [14] and deployed them in four Internet2 POPs (LA, Houston, New York and Washington DC). Our routers carried a mix of production and experimental traffic cross-country. We conducted a number of different experiments to test whether Eqn. 2 holds, one of which is shown in Figure 4. In this particular screenshot of an experiment, a 100Mb/s dedicated link between LA and Houston carried 150 TCP flows. The old rule of thumb in Eqn. 1 predicts we need 648-kB buffers, shown in the horizontal red line on the bottom right graph. Our tool automatically measures, by successive approximation, how big the router buffer needs to be in order to sustain 100% utilization of the bottleneck link, then plots it as a blue diamond on the graph. The tool controls N and measures the required B; at the time the screenshot was taken, it was evaluating the required buffer for $N = 150$ flows. In the particular case shown, the system concludes we need 26kB of buffering (about half of the 56kB predicted by Eqn. 2). As we can see in the graph (and as we found in many more experiments), the amount of buffer needed is consistently bounded above by our new rule of thumb in Eqn. 2. This is explained in detail in a short video [3].


Experimenting with Programmable Routers in Real Networks v0.03b

Buffer Size = 26kB

Buffer Sizing Formula
648kB ○ Rule of Thumb: RTT·C
52kB ○ RTT · C / √N
26kB ● Custom

Flow Control
Number of Flows = 150

Mode
○ Manual
● Automatic
○ Auto for current N

Figure 4: Screenshot of an experiment in Internet2 to verify whether the new rule of thumb in Eqn. 2 holds. The screenshot is from a video of the experiment [3].

Stanford University Backbone. Several experiments involving real routers are described in Chapter 6 of [1]. We came to realize that the documented configurations of buffer size settings of commercial routers were often incorrect or at least very confusing. We carefully calibrated the settings of several types of routers from different vendors using external packet capture devices.[7] In every router we looked at, it took considerable effort to understand and calibrate its internal behavior.

One experiment involved a gateway—a Cisco VXR 7200 shared-memory router—forwarding Internet traffic to and from Stanford's student dormitories. In the experiment, conducted in 2004, we could control the data rate C, measure the link utilization and count the number of flows, N, using Netflow.

After calibrating the router, we ran experiments varying C (using a packet shaper to throttle the link) while measuring N and the link utilization. The traffic was not idealized, but a complex mix

[7] For one widely deployed backbone router we discovered that the actual buffer size was *inversely* proportional to the console setting!

of long flows, short flows, UDP and TCP and a variety of non-congestion-aware applications. While Eqn. 2 predicts we would need 557 packets, in our experiments the link sustained 98.5% utilization with 85 packets, close to the amount predicted by Eqn. 1.

Experiments by others. We are aware of a small number of large network operators who have experimented with, and in some cases adopted, smaller buffers in their networks, inspired by these results. For example, Google's B4 private backbone network [11, 12, 18] is based on single-chip ASICs with less than 48MB of buffer. They don't report N, so we cannot evaluate whether or not Eqn. 2 holds, but published numbers for C and RTT_{min} suggest their buffers are at least 25-times smaller than the old rule of thumb.

Similarly, Microsoft's software-defined WAN network was originally built from switches with "queue sizes of 9-16MB" [10], about two orders of magnitude smaller than the old rule. In neither case do the operators report sizing buffers on the newer rule, nor do they report the number of flows, N, but their results support the basic idea that much smaller buffers are sufficient.

We are also aware, anecdotally, of two other large datacenter owners who use similarly small buffers for their inter-DC WANs, and a large streaming video service that found buffers of 25MB per 100Gb/s link are sufficient, once again about 25-fold smaller than the old rule of thumb. We are encouraging all of them to present their work at the Buffer Sizing Workshop, to be held at Stanford University in December 2019 [13].

1.2 Small buffers and packet loss

A common and understandable concern is that smaller buffers increase packet loss. Large networks often operate under the strict requirements of a Service Level Agreement (SLA) with hard limits on packet loss. Making the buffers smaller means risking violating the SLA and paying a penalty.

As we know, packet loss is a double-edged sword for TCP: It is good (it is the primary congestion control signal) and it is bad (it leads to retransmissions). Even our well-behaved single TCP flow in Figure 1 drops one out of every $3/8 * W^2$ packets; *i.e.,* to meet a specific loss rate SLA, W must exceed a threshold. To control congestion, TCP requires senders to decrease their sending rate, W/RTT. But they can't reduce W too far, and so instead the operators must increase RTT, which is effectively what they do. They use huge buffers that fill up (and never go empty), increasing RTT to a point where W is large enough to meet the SLA and W/RTT is low enough to be sustainable. This is bad because it doesn't actually help with throughput, creates unnecessarily large queueing delay, and is very challenging for real-time applications. If you run simple ping tests across the public Internet today, you often see RTTs much larger than the fixed propagation time, suggesting the ISPs are (inadvertently) using large buffers to increase RTT and thereby indirectly control congestion.

You may be wondering, as we have, why so few ISPs have publicly tested the buffer size in Eqn. 2. After all, they could potentially lower their costs by reducing buffers and removing the need for special-purpose routers in their networks. There seem to be two reasons. First, their understandable concern about SLAs makes them nervous, for the reasons we described above. Among some large ISPs we have observed a reluctance to even run brief experiments in their production network, for fear of violating an SLA. Second, several router vendors have tried to dismiss our results, for fear of losing revenue for their special-purpose routers. In our opinion, this is an example of the perils of a networking industry dominated by a small number of vendors; we believe a coordinated push-back by the ISPs is appropriate, based on a set of sound, commonly accepted and reproducible experimental results. We expand on this in Section 4.

2 ALTERNATIVE BUFFER SIZE THEORIES

Our 2004 paper prompted several authors to publish refinements, clarifications, and limitations of Eqn. 2, with many interesting proposals and results. We are big fans of more debate, particularly on an important, but poorly understood, topic such as this. We doubt Eqn. 2 is the final answer, and we definitely do not think it is the only answer — the result is for a specific context of a network carrying many long-lived TCP New Reno flows.

The team at Hamilton Institute published results from a measurement study of their Internet uplink [20] making the good point that the value of N is hard to pin-down, particularly because it is time-varying. Their link contained a mix of long- and short-lived flows, UDP and TCP traffic. Their adaptive buffer tuning algorithm (ADT [20]) is a particularly interesting way to dynamically reap the rewards of smaller buffers, while giving headroom for times when larger buffers are needed. We agree that caution is required when using Eqn. 2 in access networks, where the number and type of flows fluctuate. As some of the work on buffer bloat has argued [15], there are good reasons to reduce buffer size at the edge, and ADT might provide a good way to do so automatically. However, if a large carrier network has a relatively stable or predictable number of flows during times of congestion (*e.g.,* a large CDN carrying video flows during peak viewing hours), then Eqn. 2 should still hold.

Our 2004 paper also includes a (less well-known) analysis of short-lived TCP flows (that never leave slow-start). In summary, with lots of short-lived flows, an effective bandwidth result appears to hold:

$$P(Q \geq b) = e^{-b \frac{2(1-\rho)}{\rho} \cdot \frac{E[X_i]}{E[X_i^2]}} \tag{3}$$

with the interesting property that for short flows, the size of the buffer does not depend on the line-rate, the propagation delay of the flows, or the number of flows; it only depends on the load of the link, and length of the flows. And because the analysis doesn't depend on the dynamics of slow-start (only on the burst-size distribution), it can be easily extended to short unresponsive UDP flows.

We found that in environments with a mix of many long- and short-lived flows, the longer flows dominate the buffer requirement and therefore Eqn. 2 is still a good estimation. However, it may not necessarily hold in environments with a small number of flows, or very skewed mixes of short- and long-lived flows. In Internet backbone routers, carrying very large numbers of flows, and with the growing dominance of long-lived video TCP flows, it seems reasonable to assume that Eqn. 2 holds. Still, caution is needed not to overestimate N.

The authors of [7] in 2005 also pointed out that caution is needed to avoid over-estimating N; it should equal the number of flows bottlenecked *at this link*, not those bottlenecked elsewhere. They further argue that to limit the maximum loss rate, the buffer should be proportional to N.

In 2005, Raina *et al.*developed a first control theoretic model of a network with small buffers [16]. They used their model to decide whether or not the TCP congestion control algorithm is stable for a given buffer size, which they likened to whether TCP flows are desynchronized. They confirmed that if flows are desynchronized then Eqn. 2 is sufficient; their theory suggests that small buffers actually promote desynchronization in a virtuous circle.

A 2009 paper [17] provides a comprehensive fluid model of TCP flows in a small-buffer network, providing a strong foundation for exploring the interaction between buffer size and congestion control.

3 FROM SMALL TO TINY BUFFERS

In 2004, DARPA funded two large all-optical router projects under its "Data in the Optical Domain" (DOD-N) program [6, 9]. Initially we argued *against* the program saying that because a router is a packet switch, and a packet switch needs buffers (FIFOs) that are too large to be built using integrated optics, then it seemed a futile endeavor. While some researchers had used fixed delay lines to delay packets, often from unwieldy long spools of fiber, no-one to our knowledge had built a meaningful optical FIFO. And anyway, Eqn. 2 says we still need to store quite a lot of packets in large, high data-rate networks.

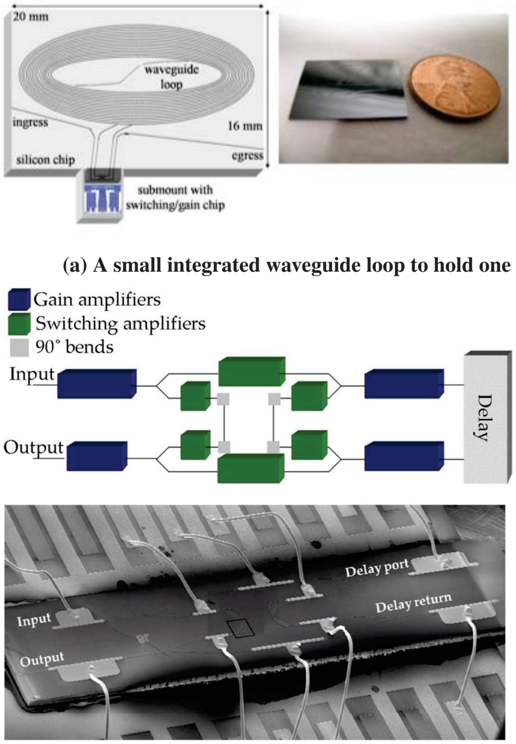
Dan Blumenthal (UCSB) and Jagdeep Shah (DARPA MTO) changed our minds. They told us about the work John Bowers and Emily Burmeister were doing to build an integrated optical FIFO. By the end of the LASOR project, the UCSB team built optical FIFOs capable of delaying, perhaps, 20 short packets [4]. Figure 5(a) shows a building block integrated onto a silicon substrate, and Figure 5(b) shows how they are assembled to make an all-optical FIFO (with electronic control signals).

Our job was to understand what happens in an all-optical packet switched network with tiny buffers, *much* smaller than Eqn 2. Does throughput fall slowly as we reduce the buffers, or does it fall precipitously? We noticed in earlier simulations that the utilization drops slowly until a threshold, beyond which it drops steeply. Curiously, the threshold often seemed to be below 100 packets. In 2005, Enaschecu *et al.* published a perhaps surprising result arguing that with only

$$B \geq O(\log W) \tag{4}$$

packets of buffering (where W is the average TCP window size and $\log W$ is typically below 100 packets), a backbone network carrying many TCP flows can achieve 90-95% utilization, with the main caveat that packets are paced [8]. Pacing could be explicit (by the sender) or could be a consequence of multiplexing packets from a slower edge network to a faster core. This result is sometimes called the *Tiny Buffers* result.

The intriguing possibility suggested by the *Tiny Buffers* result is that packet buffers might be made much smaller; perhaps as small as 20 packets, if we are prepared to sacrifice some of the link capacity. For example, a 40Gb/s link with 15 packet buffers could be considered to operate like a 30Gb/s link, which could be compensated by making the router run faster than the link-rate. Maybe future networks with abundant link capacity could trade off capacity for all-optical processing and tiny buffers. While in the past we could assume packet buffers were cheap, and long-haul links were expensive and needed to be fully utilized, in an all-optical network packet buffers are extremely costly and capacity is abundant. Only time will tell if this tradeoff makes sense.



(a) A small integrated waveguide loop to hold one packet.

(b) Multiple single packet waveguides are switched to create a FIFO.

Figure 5: Integrated optical buffers built as part of the LASOR project at UCSB [6].

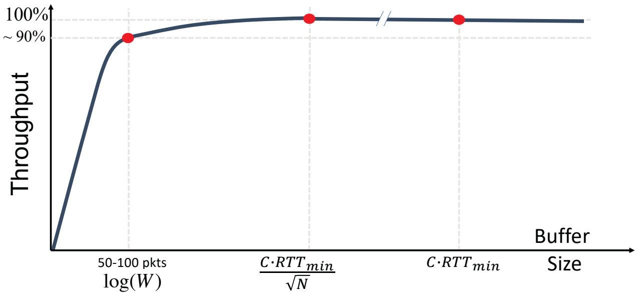


Figure 6: Summary of three theories of buffer size. From right hand side: Old rule of thumb (Eqn. 1), the Small Buffers result (Eqn. 2), and the Tiny Buffers result (Eqn. 4).

4 BUFFER SIZES IN FUTURE

While many papers have been published about buffer sizing since our 2004 paper, without doubt there is much more research to be done. At the very least, the following questions remain open:

(1) Does Eqn. 2 hold for most backbone networks today, carrying large numbers of predominantly long-lived TCP flows? Are

there networks of this type where the results appear *not* to hold?

(2) Are there experimental methods for large network owners to run buffer size experiments, without needing to worry so much about customer SLAs?

(3) What is the correct value of N, the number of active flows, to use in different settings?

(4) What effect do newer congestion control algorithms, such as Timely, DCTCP, BBR and PCC have on buffer size, particularly with large numbers of multiplexed flows? Intuition suggests that a smoothing central limit theorem is still likely, but can we determine the correct buffer size for large N?

(5) It is common in congested networks to understand fairness, or introduce different priorities and classes of service. What happens in networks with small or tiny buffers?

(6) Now that datacenter and cellular networks have become much more important than they were in 2004, are these results relevant to these networks?

(7) How does buffer size affect the application and the user's QoE? There has been some work in this direction, but it is important to understand the relationship.

It seems that the large datacenter and cloud companies are deploying networks with much smaller buffers than most of the ISPs. With the benefit of hindsight, we could have encouraged large ISPs more strongly to run experiments in their networks, to reap the rewards of lower-cost routers with smaller buffers.

For all these reasons, and because we think there is much more work to be done, we are hosting a Workshop on Buffer Sizing at Stanford University in December 2019 [13], and encourage readers to submit new ideas and measurements.

REFERENCES

[1] Appenzeller, G. *Sizing Router Buffers*. PhD thesis, Stanford University, 2005.

[2] Appenzeller, G., Keslassy, I., and McKeown, N. Sizing router buffers. In *ACM SIGCOMM* (2004), pp. 281–292.

[3] Beheshti, N. Buffer sizing in Internet routers. https://youtu.be/ykga6N_x27w?t=170, 2011.

[4] Beheshti, N., Burmeister, E., Ganjali, Y., Bowers, J. E., Blumenthal, D. J., and McKeown, N. Optical packet buffers for backbone Internet routers. *IEEE/ACM Trans. Netw. 18*, 5 (Oct. 2010), 1599–1609.

[5] Beheshti, N., Ganjali, Y., Ghobadi, M., McKeown, N., and Salmon, G. Experimental study of router buffer sizing. In *ACM IMC* (2008), pp. 197–210.

[6] Blumenthal, D. J., Barton, J., Neda, B., Bowers, J. E., Burmeister, E., et al. Integrated photonics for low-power packet networking. *IEEE Journal of Selected Topics in Quantum Electronics 17* (2011), 458 – 471.

[7] Dhamdhere, A., Jiang, H., and Dovrolis, C. Buffer sizing for congested Internet links. In *IEEE Infocom* (2005), pp. 1072–1083.

[8] Enachescu, M., Ganjali, Y., Goel, A., McKeown, N., and Roughgarden, T. Part iii: Routers with very small buffers. *SIGCOMM Comput. Commun. Rev. 35*, 3 (July 2005), 83–90.

[9] Gripp, J., Simsarian, J. E., LeGrange, J. D., Bernasconi, P., and Neilson, D. T. Photonic Terabit routers: The iris project. In *Optical Fiber Communication Conference* (2010), Optical Society of America, p. OThP3.

[10] Hong, C.-Y., Kandula, S., Mahajan, R., Zhang, M., Gill, V., Nanduri, M., and Wattenhofer, R. Achieving high utilization with software-driven WAN. *SIGCOMM Comput. Commun. Rev. 43*, 4 (Aug. 2013), 15–26.

[11] Hong, C.-Y., Mandal, S., Al-Fares, M., Zhu, M., Alimi, R., B., K. N., Bhagat, C., Jain, S., Kaimal, J., Liang, S., Mendelev, K., Padgett, S., Rabe, F., Ray, S., Tewari, M., Tierney, M., Zahn, M., Zolla, J., Ong, J., and Vahdat, A. B4 and after: Managing hierarchy, partitioning, and asymmetry for availability and scale in Google's software-defined WAN. In *ACM SIGCOMM* (2018), pp. 74–87.

[12] Jain, S., Kumar, A., Mandal, S., Ong, J., Poutievski, L., Singh, A., Venkata, S., Wanderer, J., Zhou, J., Zhu, M., Zolla, J., Hölzle, U., Stuart, S., and Vahdat, A. B4: Experience with a globally-deployed software defined WAN. In *ACM SIGCOMM* (2013), pp. 3–14.

[13] McKeown, N., and Diot, C. Buffer sizing workshop, Dec. 2-3. https://buffer-workshop.stanford.edu/, 2019.

[14] Naous, J., Gibb, G., Bolouki, S., and McKeown, N. NetFPGA: Reusable router architecture for experimental research. In *Proceedings of the ACM Workshop on Programmable Routers for Extensible Services of Tomorrow* (2008), PRESTO '08.

[15] Nichols, K., and Jacobson, V. Controlling queue delay. *Queue 10*, 5 (May 2012), 20:20–20:34.

[16] Raina, G., Towsley, D., and Wischik, D. Part ii: Control theory for buffer sizing. *SIGCOMM Comput. Commun. Rev. 35*, 3 (July 2005), 79–82.

[17] Shifrin, M., and Keslassy, I. Small-buffer networks. *Comput. Netw. 53*, 14 (Sept. 2009), 2552–2565.

[18] Singh, A., Ong, J., Agarwal, A., Anderson, G., Armistead, A., Bannon, R., Boving, S., Desai, G., Felderman, B., Germano, P., Kanagala, A., Provost, J., Simmons, J., Tanda, E., Wanderer, J., Hölzle, U., Stuart, S., and Vahdat, A. Jupiter rising: A decade of Clos topologies and centralized control in Google's datacenter network. In *ACM SIGCOMM* (2015), pp. 183–197.

[19] Villamizar, C., and Song, C. High performance TCP in ANSNET. *SIGCOMM Comput. Commun. Rev. 24*, 5 (Oct. 1994), 45–60.

[20] Vu-Brugier, G., Stanojevic, R. S., Leith, D. J., and Shorten, R. N. A critique of recently proposed buffer-sizing strategies. *SIGCOMM Comput. Commun. Rev. 37*, 1 (Jan. 2007), 43–48.