Sizing Router Buffers

June 1, 2012 · by brandonheller · Bookmark the permalink. ·

i
6 Votes

Team: Vimalkumar Jeyakumar, Nikhil Handigol, Brandon Heller, and Bob Lantz; also every student in CS244 who did Programming Assignment 2 (http://www.stanford.edu/class/cs244/2012/pa2.html).

Key Result(s): Buffers in Internet routers that see large numbers of flows can be significantly reduced, without sacrificing throughput – in fact, by a √N factor, where N is the number of flows.

Source(s):

1. Original Paper: Sizing Router Buffers (http://guido.appenzeller.net/pubs/sigcomm.pdf). Guido Appenzeller, Isaac Keslassy, Nick McKeown. SIGCOMM 2004.
2. Later Paper: Experimental study of router buffer sizing (http://yuba.stanford.edu/~nbehesht/publications/BufferSizing-Experiments---IMC08). N. Beheshti, Y. Ganjali, M. Ghobadi, N. McKeown, and G. Salmon. IMC 2008.
3. Related YouTube video: Buffer Sizing in Internet Routers (http://www.youtube.com/watch?v=ykga6N_x27w). Neda Beheshti, Yashar Ganjali, Guido Appenzeller, Nick McKeown.
4. Related Web Site: Buffer Sizing in the Internet (http://yuba.stanford.edu/buffersizing/).

Introduction

All Internet routers contain buffers to hold packets during times of congestion. The size of the buffers is dictated by the dynamics of TCP's congestion control algorithm. The goal is to make sure that when a link is congested, it is busy 100% of the time, which is equivalent to making sure the buffer never goes empty. The common assumption was that each link needs a buffer of size RTT x C, where RTT is the average round-trip time of a flow passing across the link, and C is the data-rate of the bottleneck link, to ensure full throughput for TCP.

The Sizing Router Buffers (http://guido.appenzeller.net/pubs/sigcomm.pdf) paper by Appenzeller et al. [1], outlines a new rule of thumb to decide the amount of buffering needed to maintain high link utilization. The authors showed that a link with N flows requires a buffer size of not more than RTT * C /√(N). The original paper included results from simulation and measurements from a real router, but not for a real network. Later, Neda Beheshti created a hardware testbed to test the buffer sizing results in the Internet2 backbone, and demonstrated it at Sigcomm 2008 [3]. We highly recommend watching the five-minute YouTube video of the experiment (http://www.youtube.com/watch?v=ykga6N_x27w).

The graph to replicate is shown below. For 98% utilization, the simulation result closely tracks RTT*BW/sqrt(N). We would like to compare this result against both Beheshti's hardware tests and our own done on Mininet-HiFi.

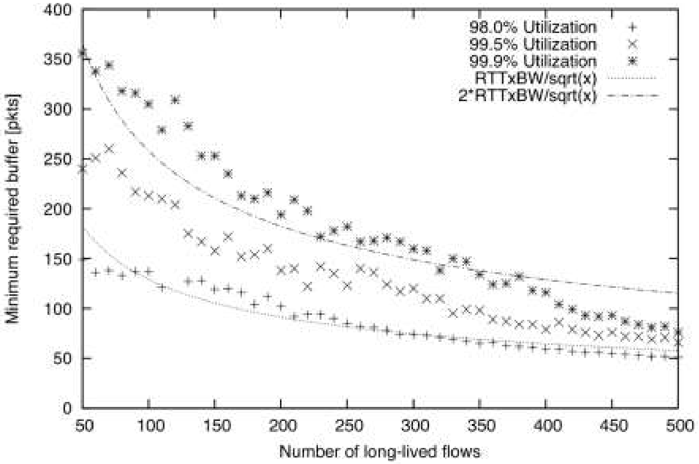


Figure 10: Minimum required buffer to achieve 98, 99.5 and 99.9 percent utilization for an OC3 (155 Mb/s) line with about 80ms average RTT measured with *ns*2 for long-lived TCP flows.

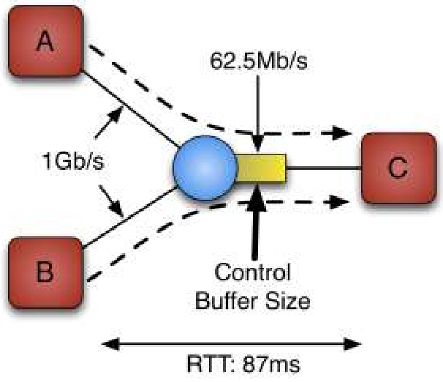
(https://reproducingnetworkresearch.files.wordpress.com/2012/06/sigcomm_result.png)

Methods

We contacted Neda and obtained her results from the hardware testbed. In the hardware experiments, a number of TCP flows start from an end-host at Stanford University (California) to a server at Rice University (Houston, Texas), via a NetFPGA IPv4 router in the Internet2 POP in Los Angeles. The link from LA to Houston is constrained to 62.5Mb/s to create a bottleneck, and the end-to-end RTT was measured to be 87ms. Once the flows are established, a script runs a binary search to find the buffer size needed for 99% utilization on the bottleneck link.

We repeated this experiment on Mininet-HiFi with fresh binary search code written in Python that measures buffer sizes and link bandwidths using interface counters. On Mininet-HiFi, the average CPU utilization did not exceed 5%. Both Mininet-HiFi and hardware results are averaged over four runs. Both sets of results do a sweep from 2 to 800 flows, with flows split evenly across 400 hosts. We used a slightly modified version of iperf that (1) waits 5 seconds after connecting to start sending traffic, to avoid starved flows from missed SYNs once the link was full, and (2) supports more than 5 concurrent flows to one server.

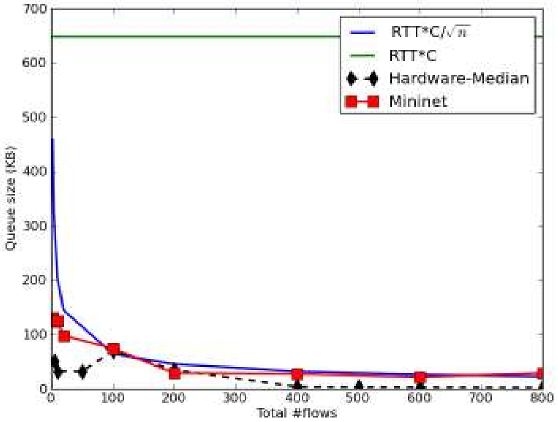
The Mininet-HiFi topology is shown below, which uses a slightly simplified topology compared to the Internet2 experiment (only one switch):



(https://reproducingnetworkresearch.files.wordpress.com/2012/06/buffersizing-topo.png)

Results

The results (summarized below) are almost identical between those generated by Beheshti on a private Internet2 topology and those generated by Mininet, and both are bounded by the theoretical limit in the Appenzeller paper.



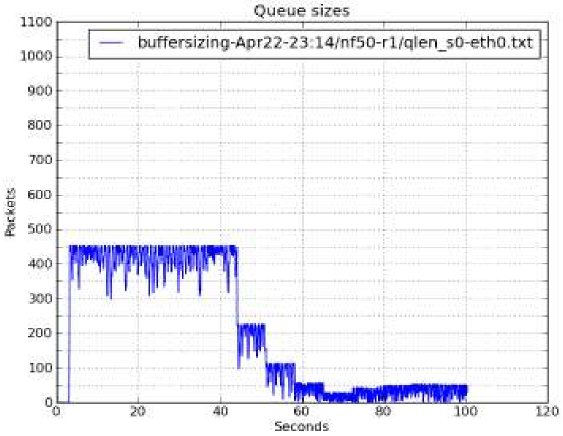
(https://reproducingnetworkresearch.files.wordpress.com/2012/06/emint_result.png)

To our surprise, Mininet-HiFi faithfully reproduces the dynamics when there are only a small number of flows — including the synchronization of flows. If Mininet-HiFi had been available (and widely accepted) the researchers could have avoided building a costly testbed.

Lessons Learned

This experiment was relatively straightforward to implement in stock Mininet HiFi using the slightly modified iperf client, but there are number of seemingly small, non-obvious choices for the binary search process that lead to different results. It is easy to get these choices wrong, and no paper described the complete measurement process in sufficient detail to be exactly replicated. For example, not waiting long enough after changing the queue size can yield invalid bandwidth measurements. In most cases, the effect is detected and the window updated within an RTT (~90 ms). However, when a flow loses its full window – which becomes more likely with higher numbers of flows that individually have smaller CWNDs – the TCP sender must wait the RTO plus many RTTs to reach steady-state again (over a second).

The measurement variability of running on EC2 also had to be addressed before results could be consistently repeatable. To address variability between runs, we used an initial measurement phase, with plenty of buffering, to calibrate the link rate to consider as 100%. Another example: with an average filter, a single bad measurement can yield the conclusion that a given amount of buffering is insufficient, causing the binary search algorithm to turn the wrong way and report a completely invalid result. Median filtering, using more samples, and using a longer total sampling period all led to significantly more consistent results. We have a hypothesis that hypervisor scheduling is causing the low-bandwidth samples, since larger instances (like c1.xlarge) produced better results than smaller (like c1.medium). This variability is evident on a bandwidth trace over the course of the binary search:



(https://reproducingnetworkresearch.files.wordpress.com/2012/06/nf50-r1-q.png)

We are not sure of the scale to which this experiment can be pushed.

Verifying fidelity:

Like the DCTCP experiment, the buffer sizing experiment relies on accurate link emulation at the bottleneck. However, a large number of TCP flows increases the total CPU load. We visualize the effect of system load on the distribution of deviation of inter-packet dequeue times from that of an ideal link. Figure 11 plots the CDF of deviations (in percentage), by varying the total number of flows. Even for 800 flows, more than 99% of all packets in a X second time interval were dequeued within 10% from the ideal dequeue time (of 192µs). It is interesting to note that even though inter-dequeue time intervals were off by 10% for 1% of all packets in a X second time interval, results on Mininet-HiFi qualitatively matched that of hardware. This supports our hypothesis that high-fidelity (if not perfect fidelity) is “good enough” to have results that are reproducible.

Instructions to Replicate This Experiment:

```
git clone https://bitbucket.org/nikhilh/mininet_tests.git
cd mininet_tests/buffersizing
```

Follow the instructions in the README file there. The code has run successfully on the original CS244 Ubuntu 11.10-based AMI in the US East EC2 data center, using a 3.0 kernel with setns and CFS BW patches applied. We suggest running at least a c1.medium instance, and ideally using a c1.xlarge instance.