The State of the Art in Bufferbloat Testing and Reduction on Linux

Toke Høiland-Jørgensen

Roskilde University

IETF 86, 12th March 2013

Outline

Introduction

Recent changes in the Linux kernel

Testing methodology and best practices

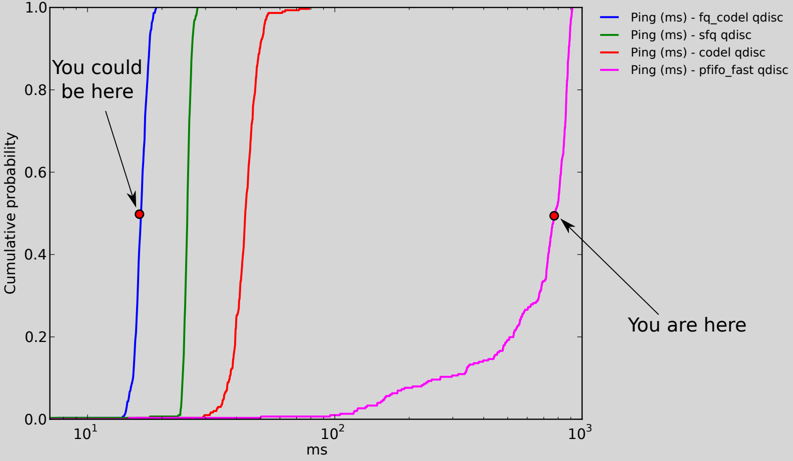
Test results

Introduction

Spoiler

Effects of bufferbloat mitigation - RRUL test

Latency during four TCP streams in each direction.

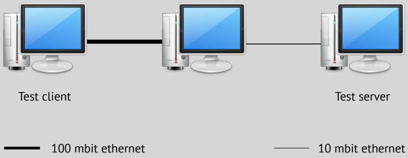


Note the log scale.

The research behind this

- Experiments done as part of university project.
- Three computers networked in lab setup.
- Switch the active `qdisc` and compare results.
- Goal: Real-world measurements on shipped Linux kernel.

Test setup



Recent changes in the Linux kernel

Byte Queue Limits (BQL)

- Introduced in Linux 3.3, by Tom Herbert of Google.
- Sits between traffic control subsystem and device drivers.
  - Requires driver support (ongoing effort).
- Keeps track of number of *bytes* queued in the driver.
- Addresses variability of packet sizes (64 bytes up to 4KiB w/TSO).
- Unneeded in the presence of software rate limiting.

TCP Small Queues (TSQ)

- Introduced in Linux 3.6 by Eric Dumazet.
- Enhancement to the TCP stack (i.e. *above* the traffic control layer).
- Makes the TCP stack aware of when packets leave the system.
  - Sets a configurable limit (default 128KiB) of bytes in transit in lower layers.
  - After this limit, keeps the packets at the TCP layer.
- This allows for more timely feedback to the TCP stack.

New queueing disciplines

- Straight CoDel implementation in the `codel` qdisc.
- Enhancements to the Stochastic Fairness Queueing (`sfq`) qdisc.
  - Optional head drop, more hash buckets, no permutation.
- Combination of CoDel and DRR fairness queueing in the `fq_codel` qdisc.
  - Prioritises thin flows.
  - This is currently the best bufferbloat mitigation qdisc in mainline Linux.

Testing methodology and best practices

Testing methodology

- Basically: Load up the bottleneck link, measure latency.

- Useful tools: `netperf`, `iperf`, `ping`, `fping`.
- Use `mtr` to locate bottleneck hop.

- Or use `netperf-wrapper` to automate tests!

The `netperf-wrapper` testing tool

- Python wrapper to benchmarking tools (mostly `netperf`).
- Runs concurrent tool instances, aggregates the results.
- Output and intermediate storage is JSON.
  - Exports to CSV.
- Graphing through python `matplotlib`.
- Tests specified through configuration files (in Python).
  - Common tests included (such as RRUL).
- Developed and tested on Linux.
  - One or two issues on FreeBSD (WiP).
- Install: `pip install netperf-wrapper`. Netperf 2.6+.

The RRUL test

- Runs four concurrent TCP streams in each direction.
  - Each stream with different diffserv marking.
- Simultaneously measures UDP and ICMP ping times.
- Supports IPv4 and IPv6.
  - Variants that measure v4 vs v6 and RTT fairness.

- The four streams pretty reliably loads any link to capacity.
- This is a simple and effective way of finding bufferbloat.
  - `netperf-wrapper -H <test server> rrul`

- Works well as a backdrop for testing other stuff.
  - The Chrome benchmark works well for websites.

Best configuration practices

- Disable offloads (esp. TSO/GSO).
  - Modern CPUs can handle up to gigabit speeds without it.
  - No offloads means better interleaving ⇒ lower latency.

- Lower BQL limit.
  - BQL defaults developed and tuned at 1Gbit/s+.
  - 1514 (ethernet MTU + header) works well up to ≃10Mbit/s.
  - 3028 up to ≃100Mbit/s.
  - But further work is needed in this area.

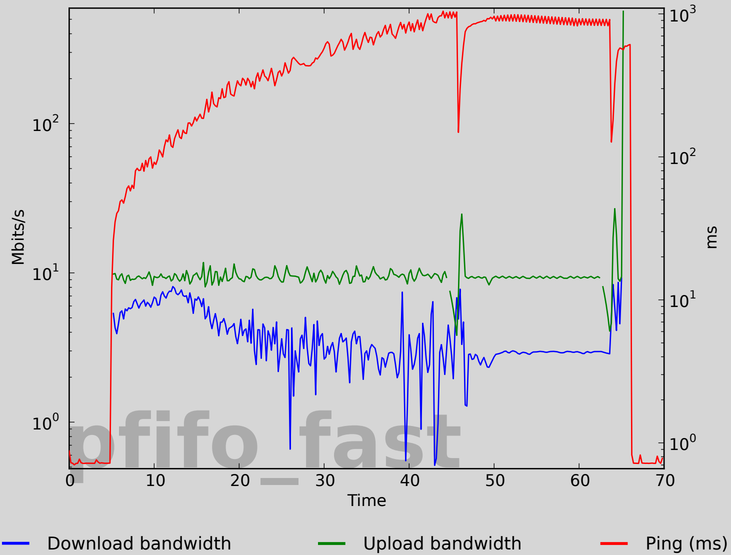
- Make sure driver(s) are BQL-enabled.
  - BQL is Ethernet only, and not all drivers are updated.
  - Esp. many SOCs have drivers without BQL.

Best configuration practices (cont.)

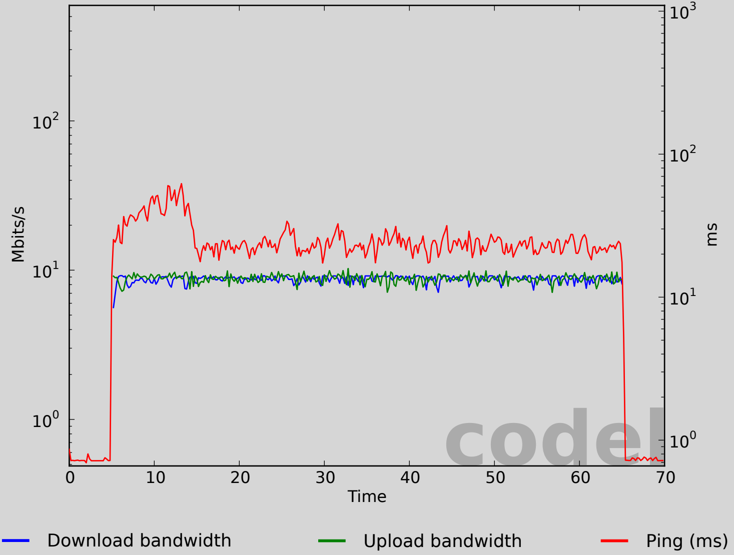
- If using *netem* to introduce latency, use a separate middlebox.
    - In particular, netem does not work in combination with other qdiscs.
- Change qdiscs at the right place - at the bottleneck!
    - Or use software rate limiting (e.g. `htb`) to move the bottleneck.
- Beware of buffers at lower layers.
    - Non-Ethernet drivers (DSL etc).
    - Buffering in error correction layers (e.g. 802.11n, 3g, LTE).
    - Even `htb` buffers an extra packet.
    - (fq)CoDel doesn't know about buffers at lower levels.
- Beware the cheap switches
    - Pause frames and/or excess buffering.

Test results

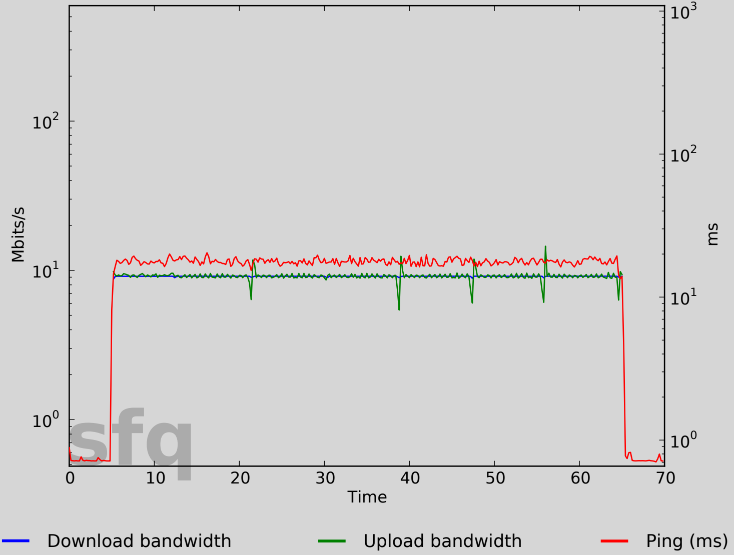
Two TCP streams + ping - pfifo_fast



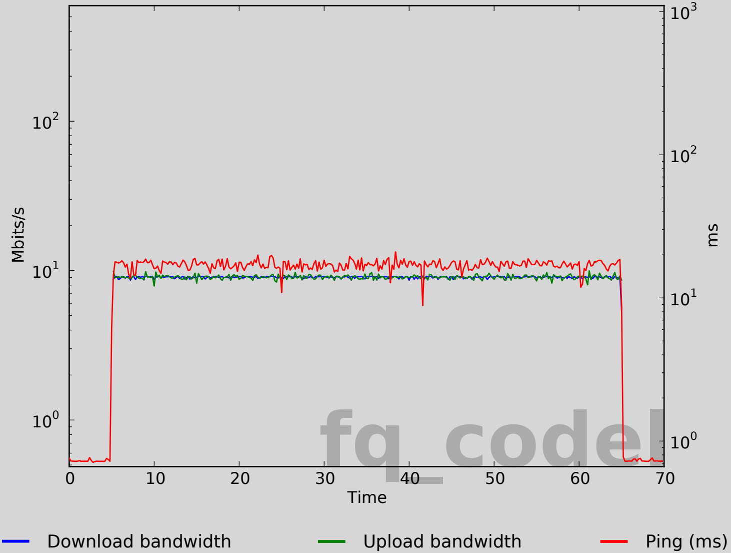
Two TCP streams + ping - codel



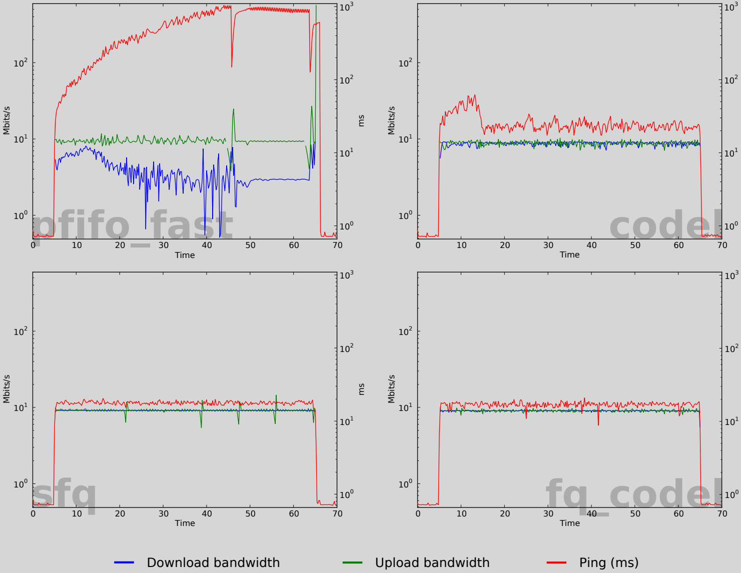
Two TCP streams + ping - sfq



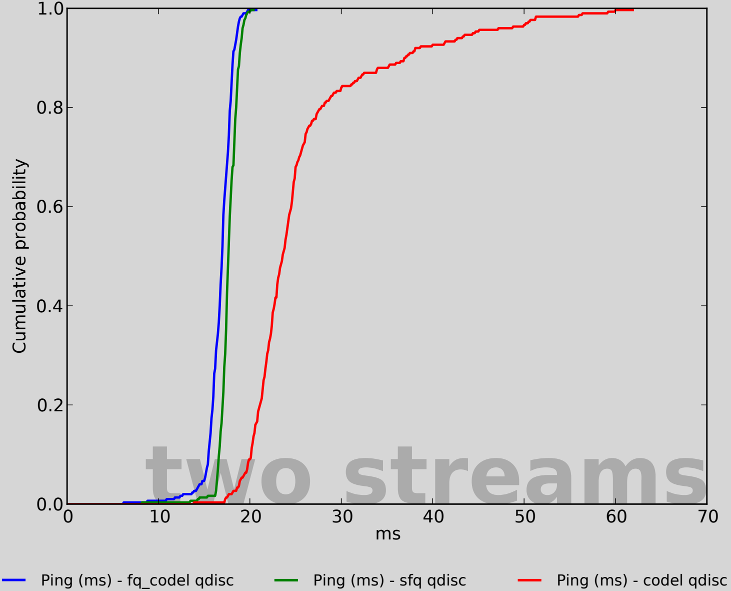
Two TCP streams + ping - `fq_codel`



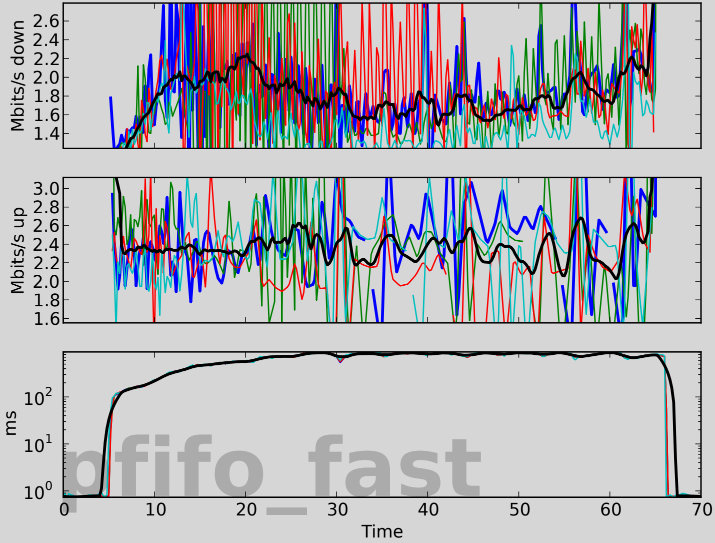
Two TCP streams + ping - comparison



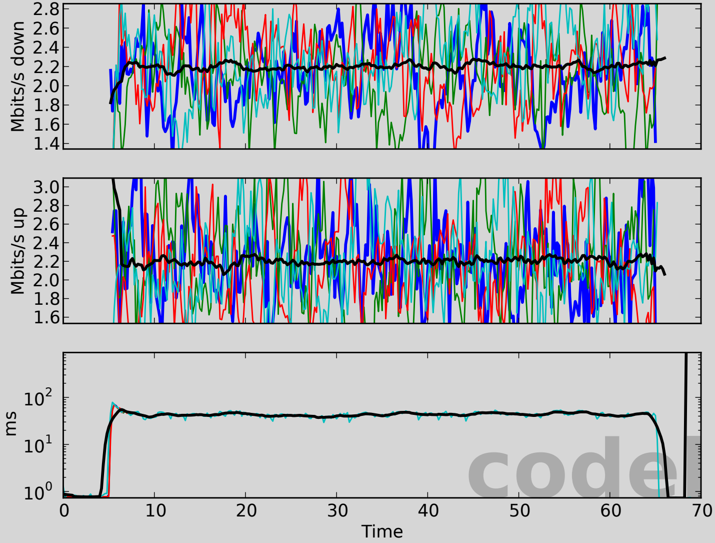
Two TCP streams + ping - CDF



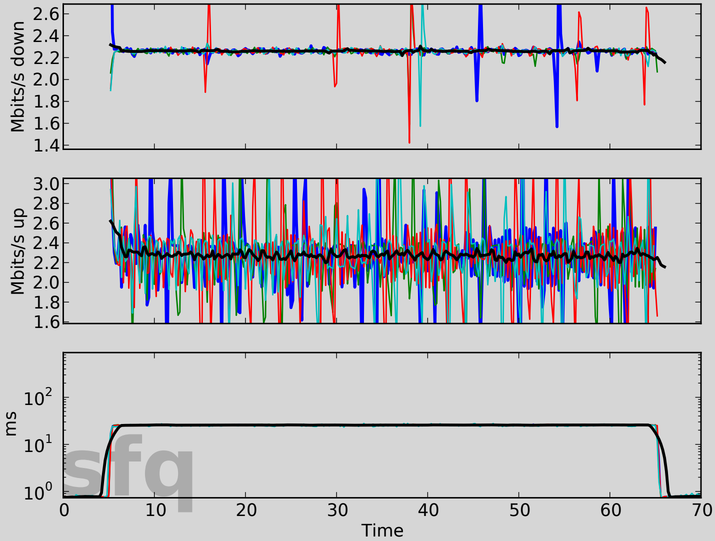
RRUL test - pfifo_fast



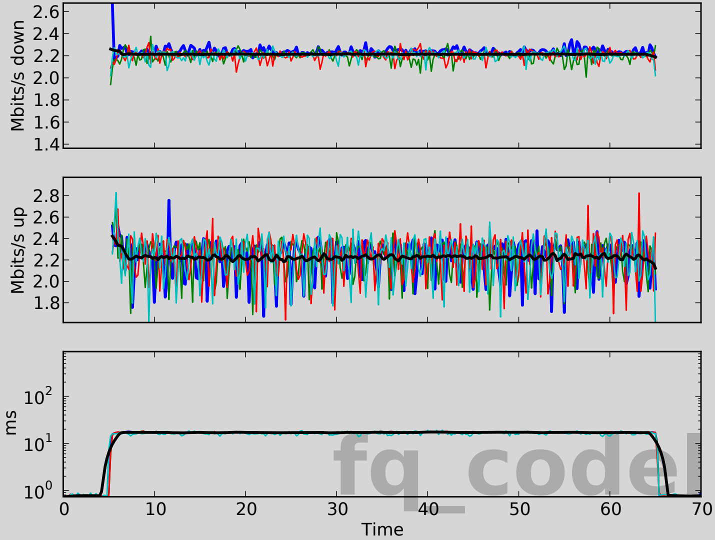
RRUL test - codel



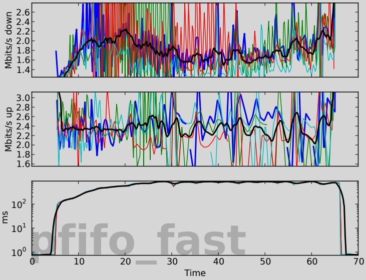
RRUL test - `sfq`

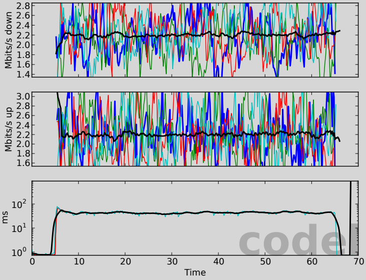


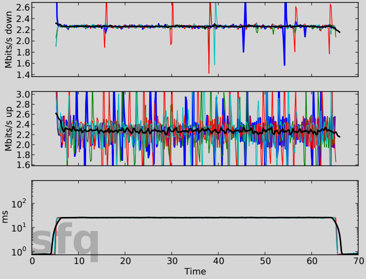
RRUL test - `fq_codel`

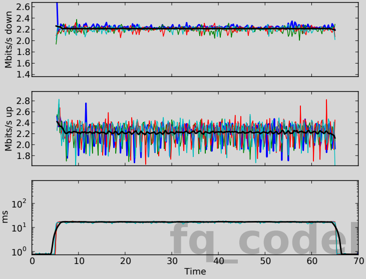


RRUL test - comparison

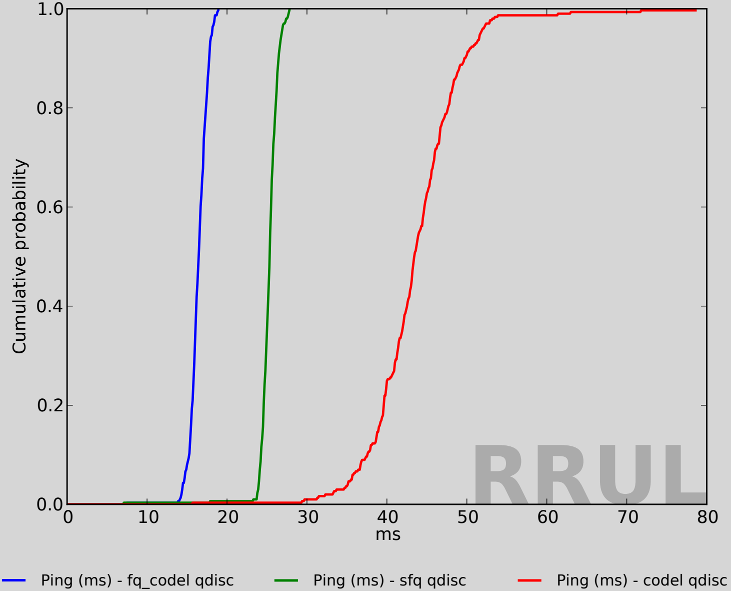




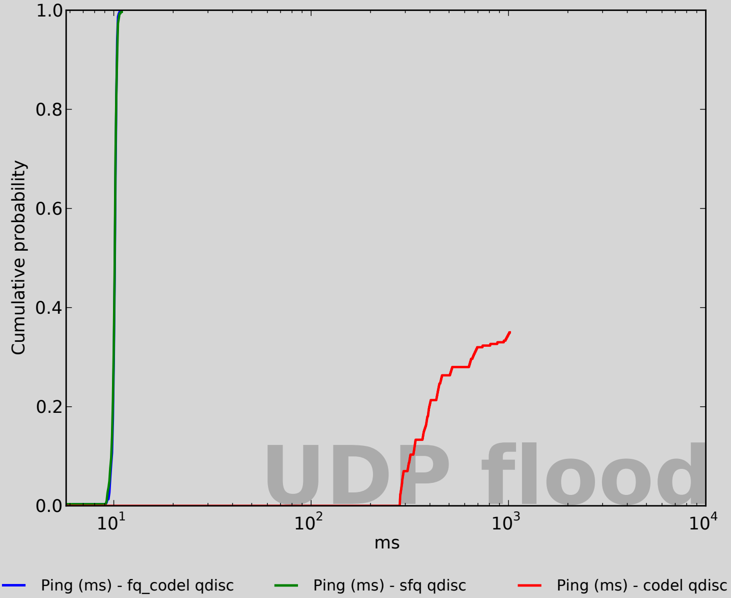




RRUL test - CDF



CDF UDP flood



References

- BQL: `https://lwn.net/Articles/454390/`
- netperf: `http://www.netperf.org/netperf/`
- netperf-wrapper: `https://github.com/tohojo/netperf-wrapper`
- Paper on experiments: `http://akira.ruc.dk/~tohojo/bufferbloat/bufferbloat-paper.pdf`
- RRUL test spec draft: `https://github.com/dtaht/deBloat/blob/master/spec/rrule.doc`
- Best practices: `https://www.bufferbloat.net/projects/codel/wiki/Best_practices_for_benchmarking_Codel_and_FQ_Codel`
- My email address: `toke@toke.dk`

Questions?

Questions? Comments?