How internet congestion control actually works in the bufferbloated age

Dave Taht (dave.taht@gmail.com)
Nagger in Chief – Bufferbloat.net
CTO - TekLibre

About this talk

- An introduction to network congestion control
- With a demo of TCP and Queuing covering:
  - Windowed protocols, TCP Reno/Cubic, Voip and Gaming Traffic, Fair Queuing, AQMs, BBR, and ECN...
- Using people as packets! I will need about 12 volunteers a couple slides from now for 20 minutes....

About dave...

- Co-founder (with Jim Gettys) of bufferbloat.net.
  - In 1991, my dream was to have an internet where I could plug my guitar into the wall, and play with a drummer across town. The speed of light across SF was only 36us! Playing on a stage, live, is 8ms! I thought it would be **easy**.

https://www.patreon.com/dtaht

  - I run the make-wifi-fast, cerowrt, and ecn-sane projects and also contribute to “cake”, & embedded linux in general.
  - I work in 9 layers of the ISO network stack (including the political and financial layers in the IETF) to make a faster, more reliable, less jittery internet more widely deployed.
  - I work off of contracts (fixing wifi and cpe) and donations.
  - https://www.patreon.com/dtaht

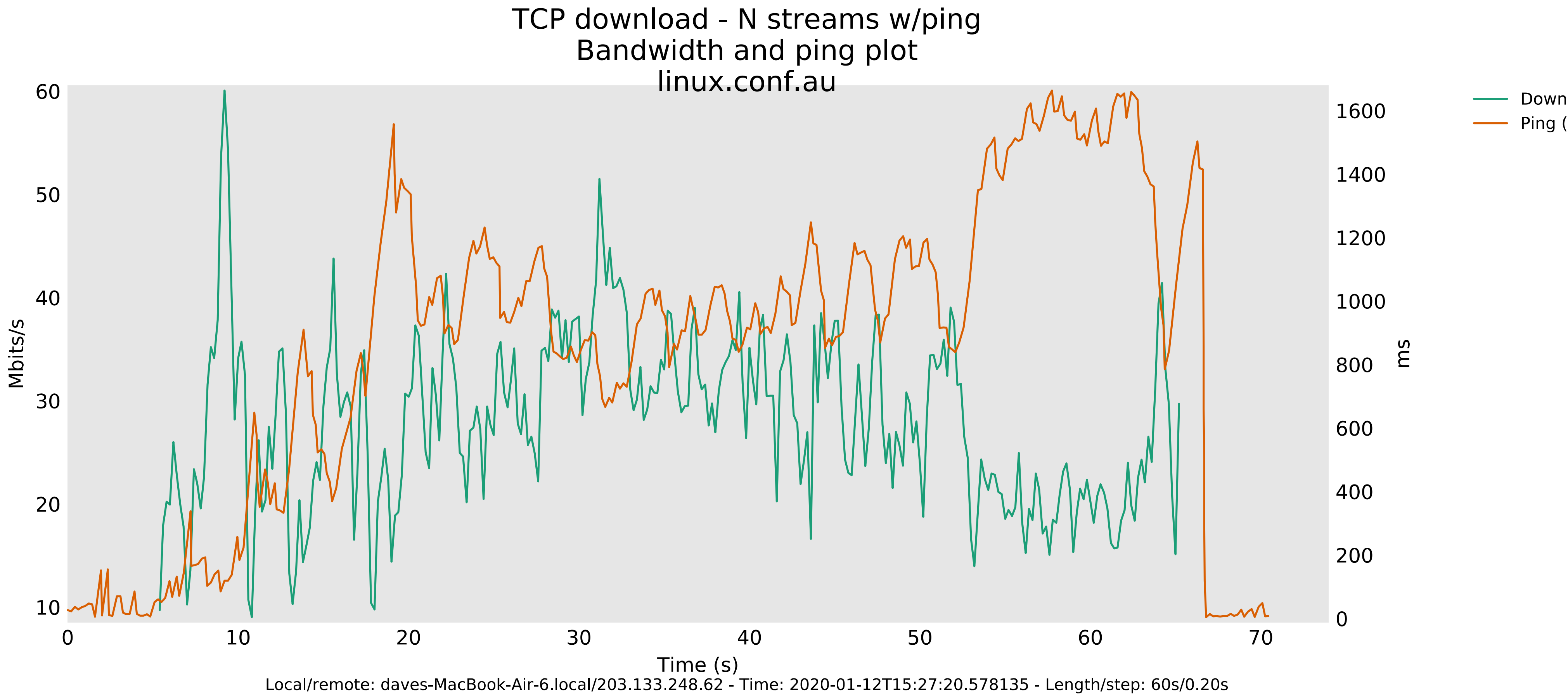
What is congestion control?

- Internet congestion control algorithms govern how multiple flows from multiple sources and destinations ultimately share the network more or less fairly when crossing bottleneck links.
- Without congestion control, the internet would stop working. (it already did, once)
- Many networks today teeter on the brink of congestion collapse, degrading to near uselessness under load due to excessive buffering and **not enough packet loss**.
- https://hpbn.co/building-blocks-of-tcp/

About bufferbloat.net

- Gang of internet originals and 500+ volunteers on the mailing lists who noticed that all our edge technologies – 3g, wifi, cable, dsl - all had a fixed amount of buffering, generally configured for the highest rate the hardware could do, NOT what users actually had – inducing seconds of excess latency under load.
- Starting in 2011, we invented new algorithms to manage buffering better, slammed them into linux, OSX, IOS and freebsd, made them into IETF standards, and many have appeared in new products!
- But we still have a billion routers and other devices to upgrade….

Bufferbloat on the LCA 2020 WiFi network



Other Bufferbloat “solutions”

- Rather than fix the routers, the internet has devolved…
  - Most traffic is in small bursts that are dependent on the rtt to scale up (web)
  - Rate limited streaming (e.g. netflix) is on the rise. “Buffering…” is really annoying so most buy WAY more bandwidth than needed… when they can
  - “old” applications like scp, voip, games, are on the decline…
- Still seems better to improve the routers…

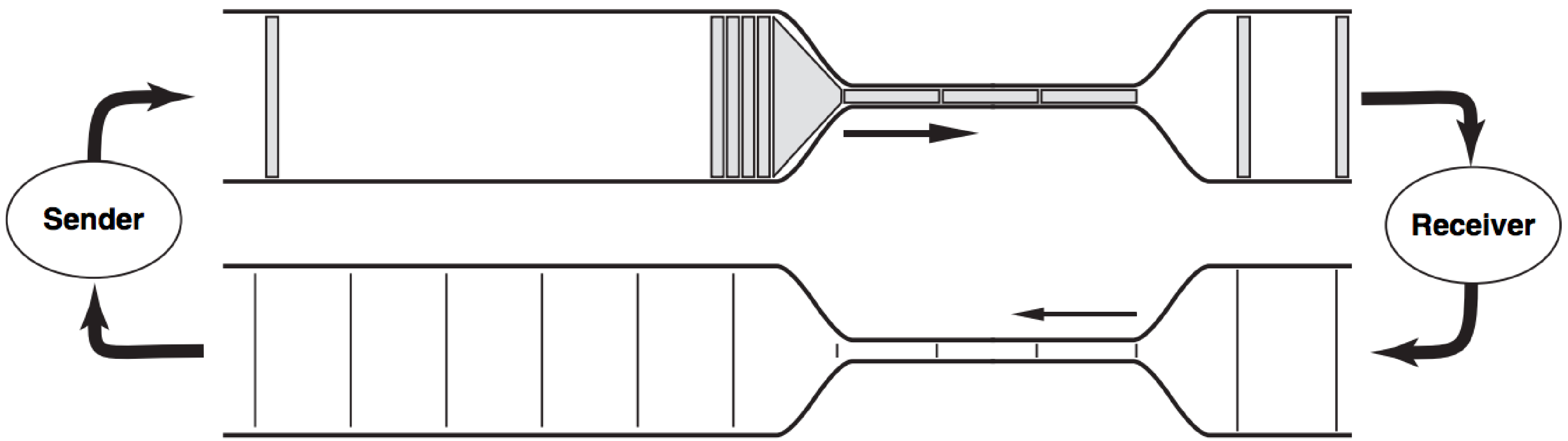
Congestion control demos

- I’m going to do a bunch of demos now on how the internet was designed to work…
- At each point, after I get our “packets” to demonstrate what is actually happening, I’ll cry out “Packets, Freeze” in order to explain each algorithm in play.
- Feel free to overact. You are on video, however, so don’t do anything you don’t want your mum to see.
- Come on up!!!!

Data transfer without windows

- Setup: a short link
- A simple request response protocol (like “xmodem”) only works on short links
- An efficient file transfer protocol has to “fill the pipe, not the queue” when there is capacity in the pipe between the two locations.

To fill the pipe- care about the RTT

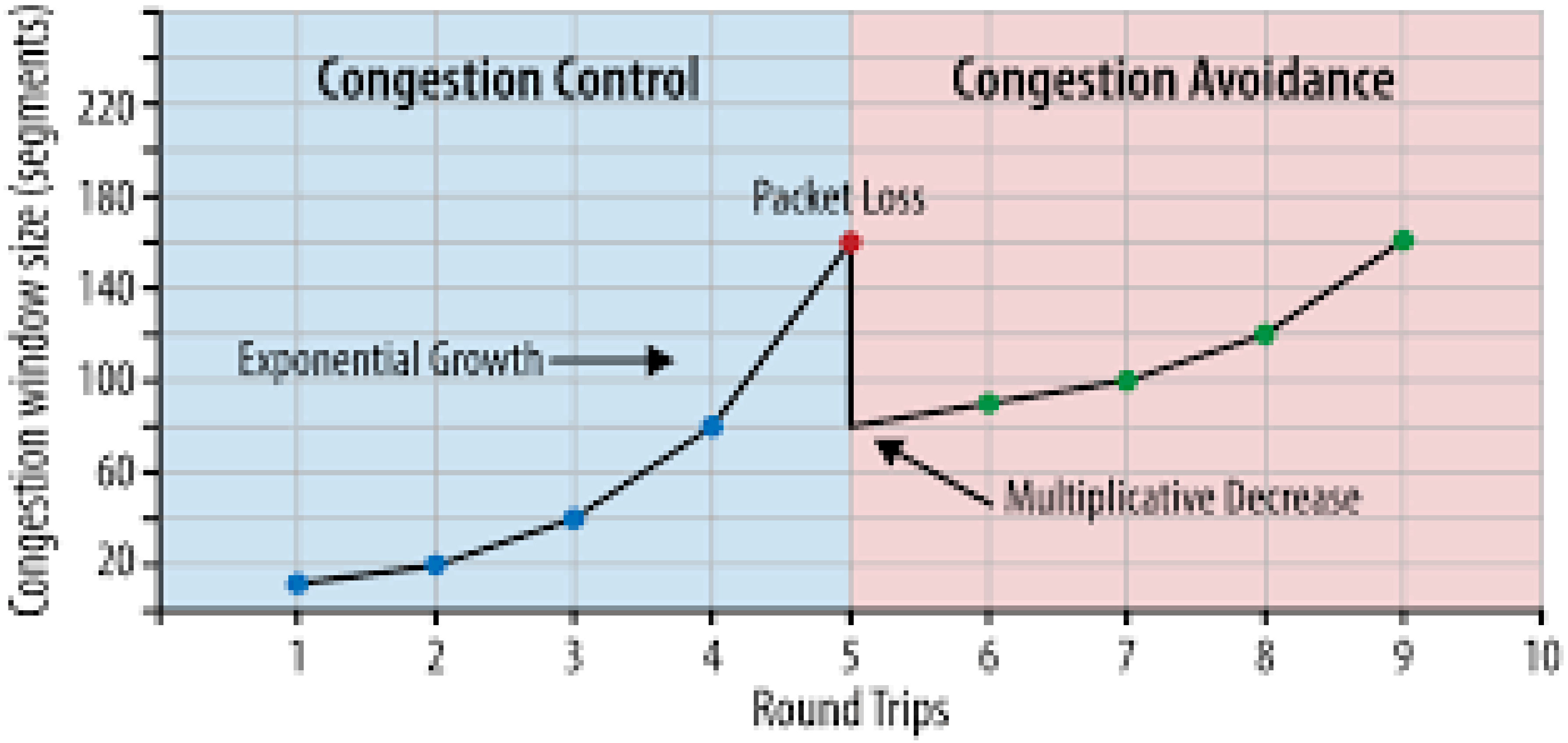


Typical Cablemodem latencies
UNLOADED UP: 16ms   DOWN: 22ms
LOADED   UP: 280ms DOWN: 680ms

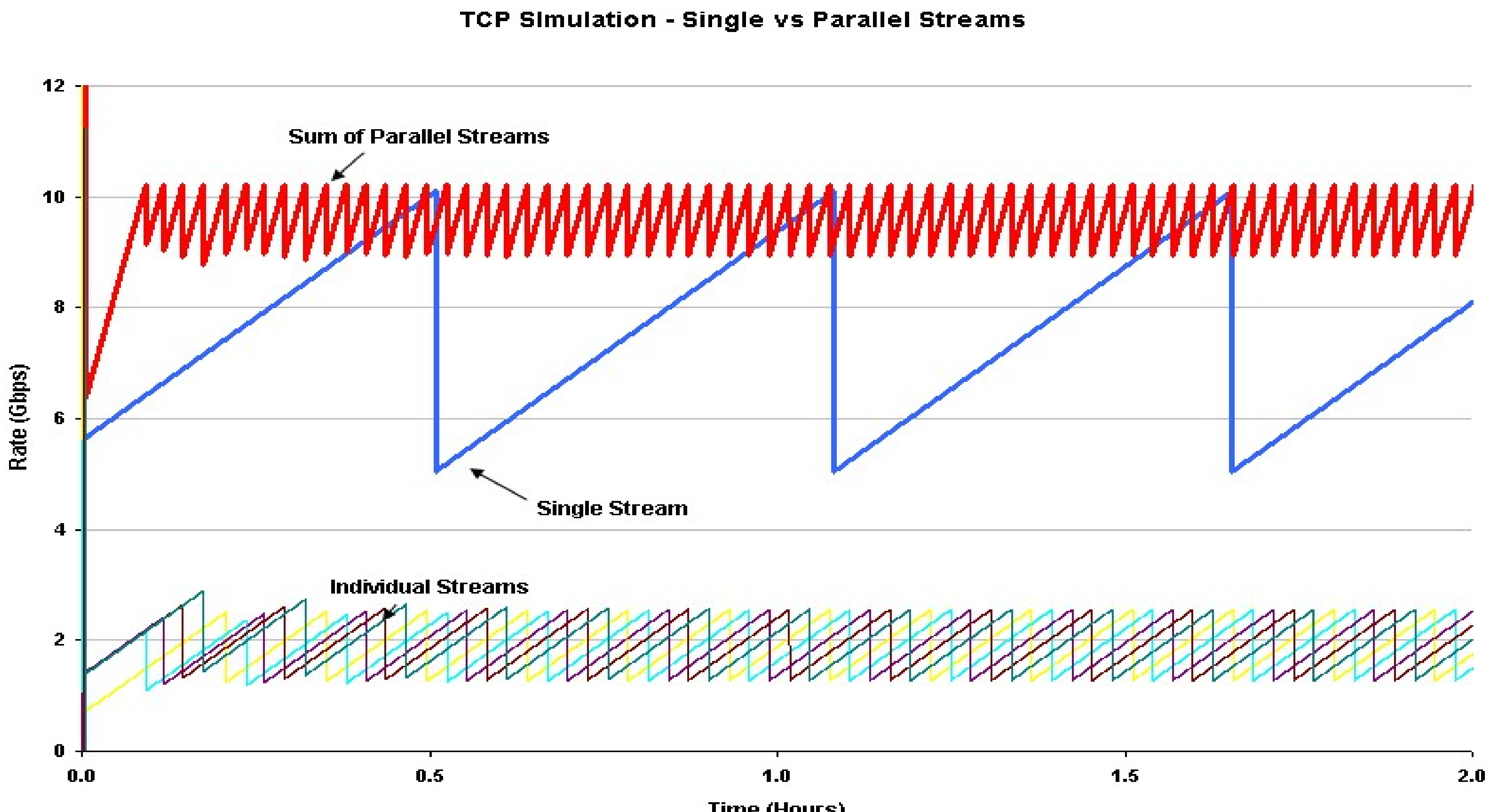
The TCP Initial Window

- An initial burst of packets MAY fill the pipe
- That burst also MAY complete the transaction
- Linux TCP has an IW of 10. Most other OSes have an IW of 4. Although we published IW10 Considered Harmful in 2010...
  - Much web traffic is IW10 + 22 paced packets nowadays... because that completes the transaction for the server.

TCP's Slow Start and Congestion Avoidance Algos



The TCP Reno/Cubic sawtooth



The role of loss and marking

- Most TCPs rely on a loss to back off by ½.
- Buffering, particularly along the edge and in wifi and 4G can be oft measured in **seconds**.
- Going from 1 second to ½ second of induced latency doesn’t help much.
- *Timely* loss or marking is essential for the network’s correct operation.

Ack Clocking

- TCP connections rely on packet acknowledgements (acks) for reliable transmission.
- Each ack contains cumulative data as to loss and timing. Many can be lost and your network still work. A loss is retransmitted after it gets acks indicating that a loss happened.
- The absence of ack (eventually) triggers a retransmit.
- Loss and marking patterns determine the size of the bursts emitted by the sender reno/cubic TCPs.
- Acks ALSO provide a “clock”, a very unreliable one, subject to noise on either the forward or reverse path.
- Delays in either the forward (data) or reverse (ack) direction make accurate round trip estimation **hard**.

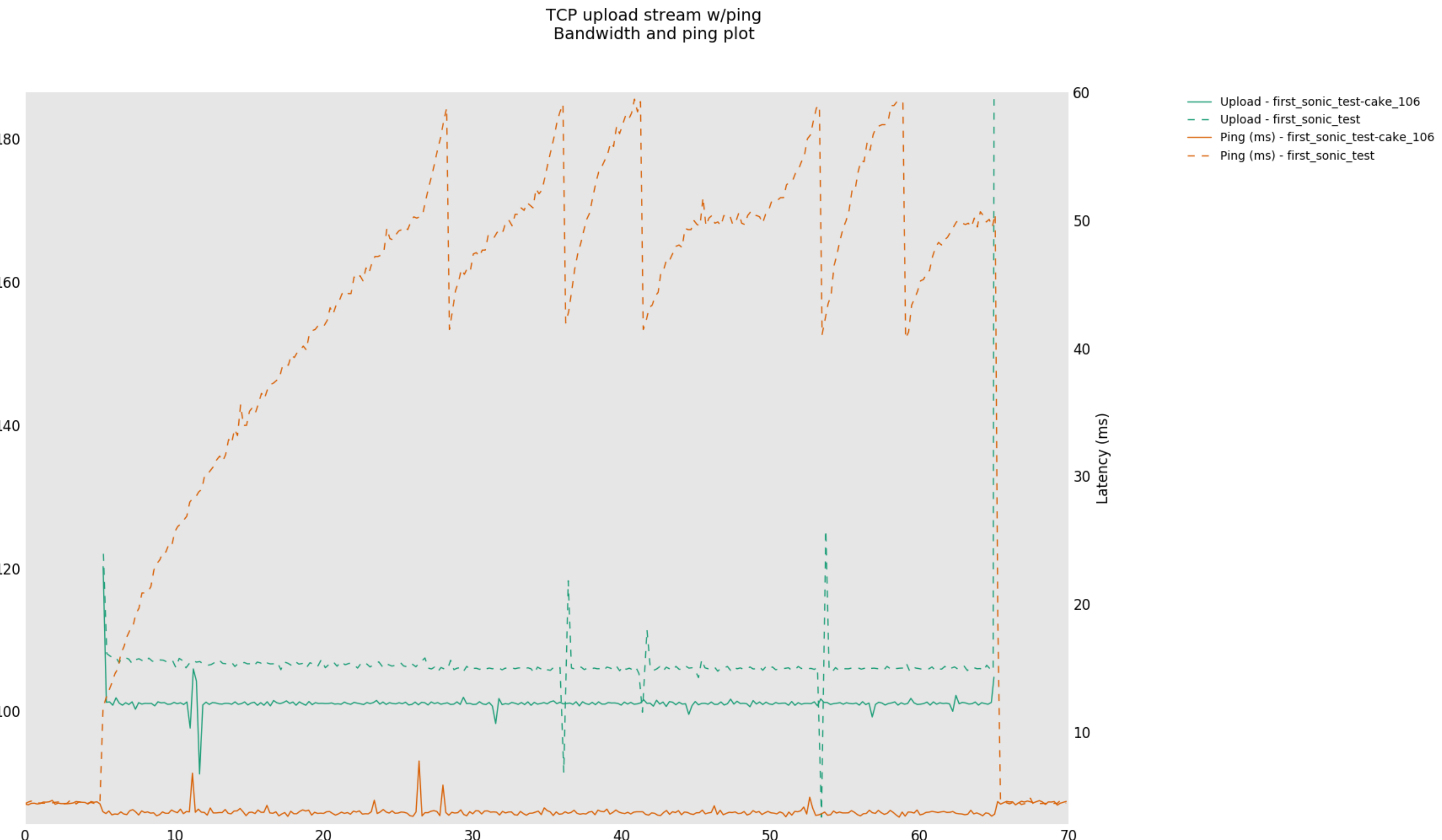
Receive and send windows

- Given bufferbloat, where packets might take a detour around the moon, many tcp clients are actually bound by the native send or receive window… (how many packets can the host keep track of) which is in the 200-400ms range, or worse, measured by bytes rather than time.
- Many clients and algorithms exist to clamp these to hold the RTT down.
- The scope is usually on a single tcp flow only.

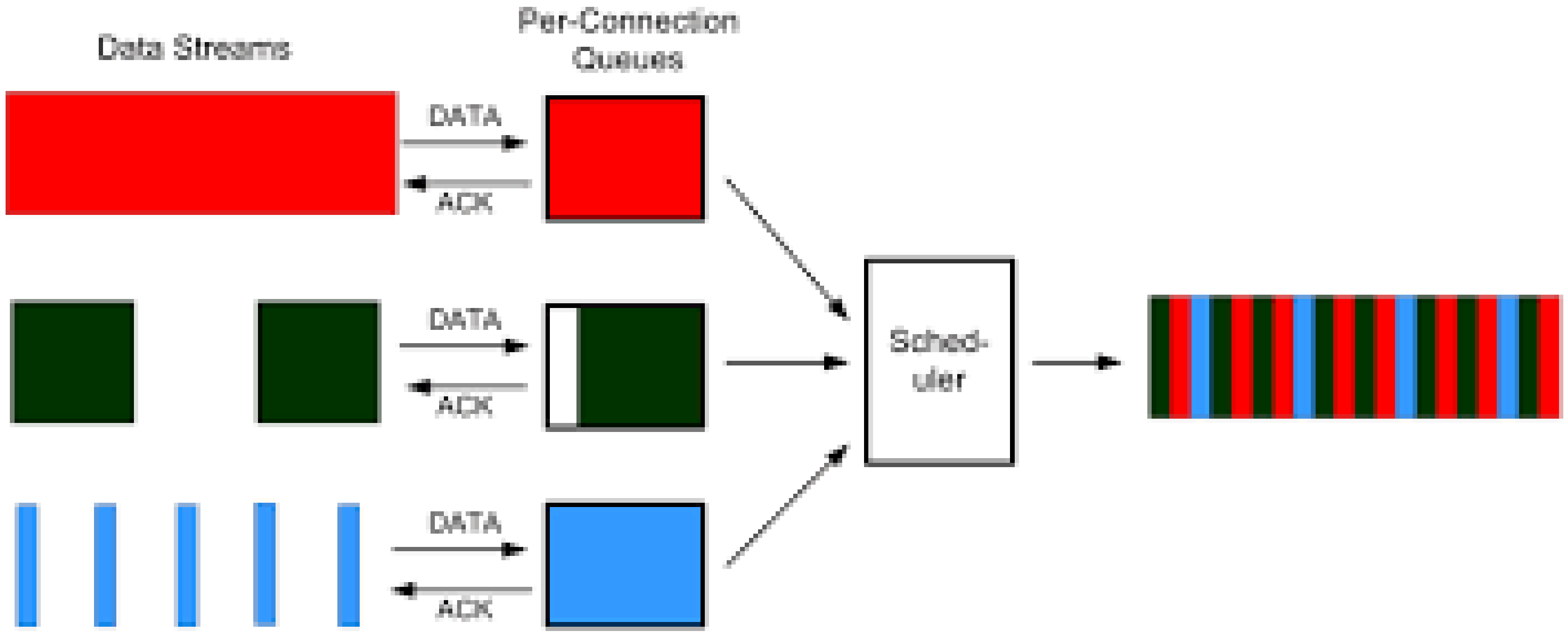
Gaming, DNS, Voip Traffic types

- There exists traffic other than TCP (who knew?)
- Gaming, DNS, Voip, Videoconferencing, request/response protocols generally
  - Want the most recent, not “stale data”
  - Don’t care (much) about loss
  - Care a lot about jitter
- These kinds of traffic **conflict** with how tcp operates, especially on bufferbloated links.

Problem: Riding the sawtooth



Solution: Fair Queuing



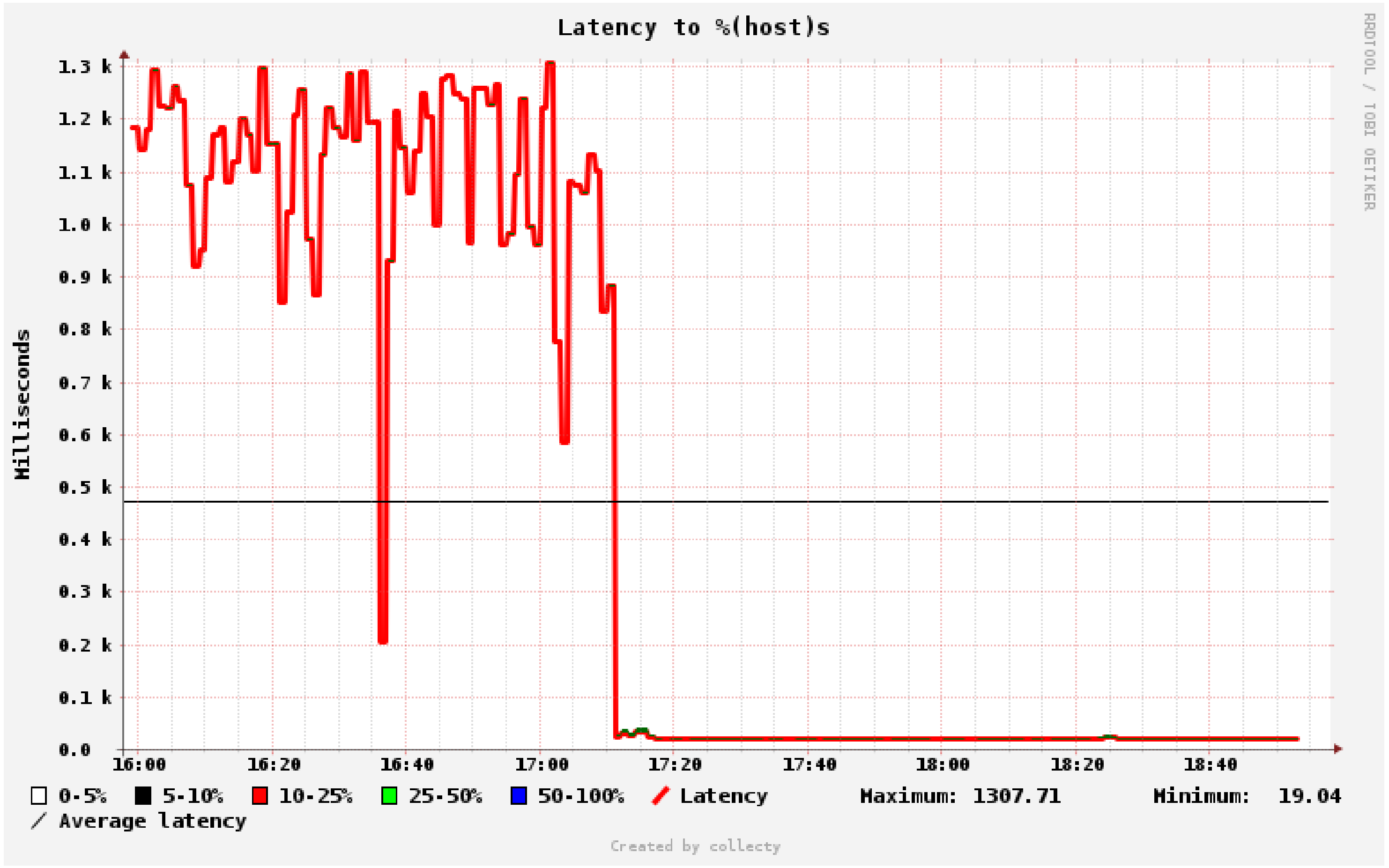
Problem: Keeping Queues Short

- Using shorter buffers in the first place helps a lot – no more than one BDP.
- Modern “Active Queue Managment” AQMs (codel, pie, fq_codel, fq_pie, sch_cake) look at how long packets are spending in the queue, and intelligently drop or mark packets to get the senders to slow down to match the configured rate.
- Typically result in a relatively fixed 5-16ms queuing latency!!

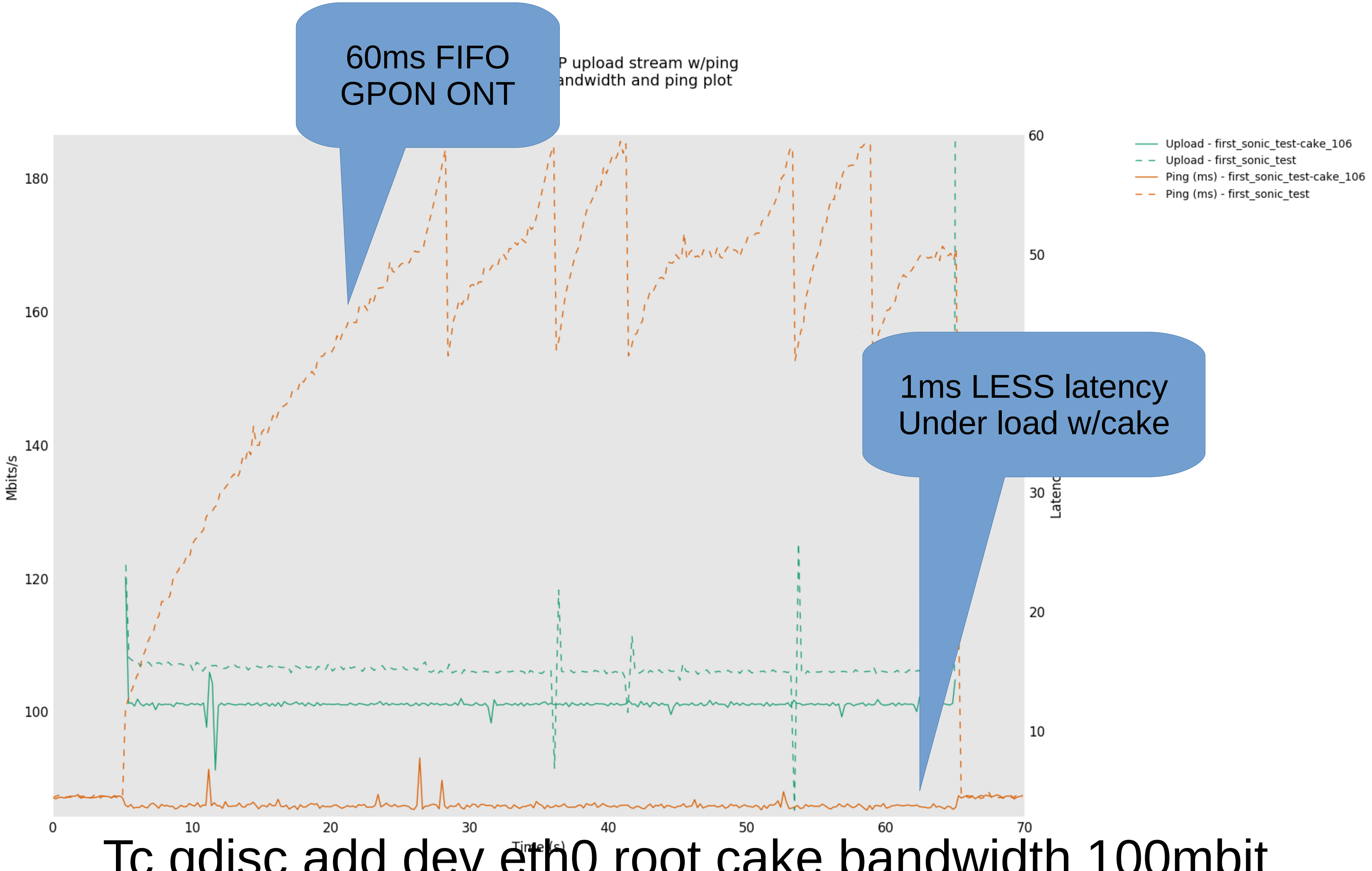
Our Solution: fq_codel (RFC8290)

- 2012: By combining 1024 “fair” queues and the codel AQM algorithm we got home routers minimizing latency and maximizing throughput for **all** forms of internet traffic.
- Implemented in linux and freebsd and well supported by newer devices now in the market under various trade names for QoS and SQM.
- Automatic for **line rate** ethernet, dsl, and 3 wifi chipsets
- (there are other solutions appearing – fq-pie and sch_cake are among the leading alternatives, AFD (from cisco), DOCSIS 3.1 has pie – as much as I like fq_codel, I’m cool with all these)
- The future is already here, it just isn’t evenly deployed yet.

DSL w/hw flow control, bql, and fq_codel



Cake v Sonic Fiber @100Mbit



Tc qdisc add dev eth0 root cake bandwidth 100mbit

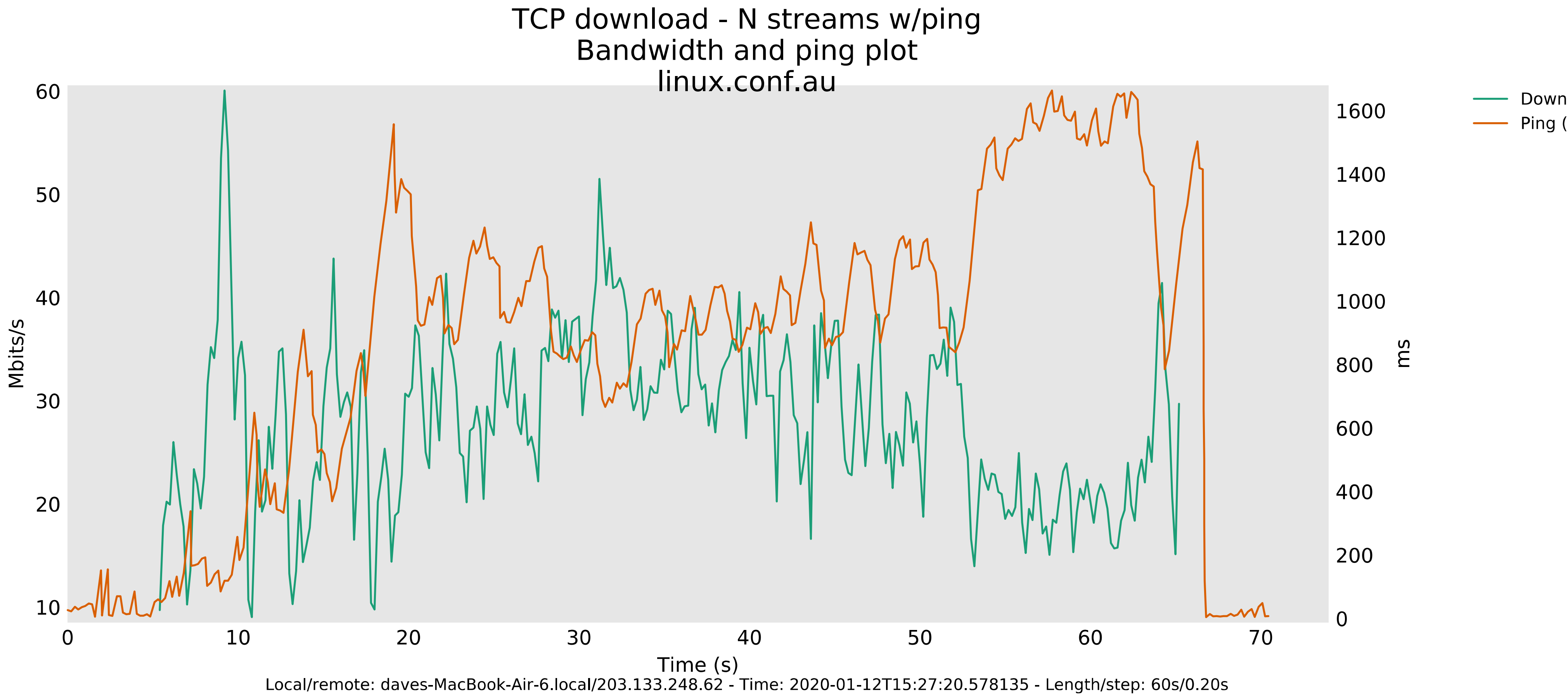
Quick Quiz

- How many of you are running fq_codel?
- To check,
  - on Linux: “tc -s qdisc show | grep fq_”
  - On Apple: netstat -I en0 -qq | less
    - (that’s an “eye” not an “el” and “en0” is one of many potential network interfaces)

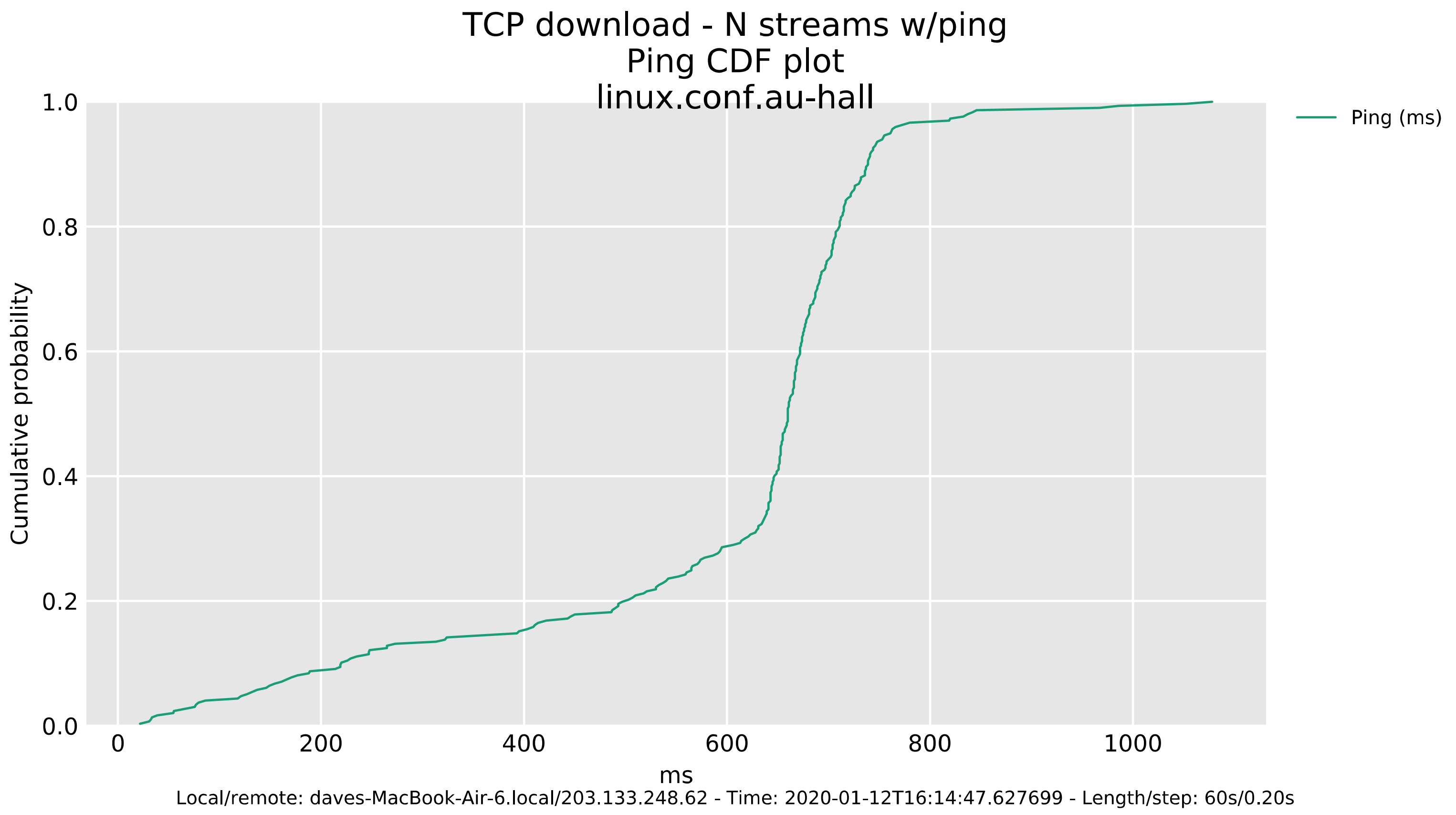
Bufferbloat.net recommendations

- Deploy AQM and Fair Queueing tech everywhere!
  - https://gettys.wordpress.com/2018/02/11/the-blind-men-and-the-elephant/
- Apply SQM to your home and business routers
  - Off the shelf theres dozens – evenroute, ipfire, pfsense, ubnt edgerouters, eero, asus, fritzbox, etc under many, many trade names (but usually fq_codel derived underneath)
  - You can also retrofit older routers with openwrt, dd-wrt, etc – or build your own!
- Pester your ISP to do buffering more right and according to the RFCs.
- Buy gear from vendors that have good solutions
- Deploy advanced algorithms in your datacenters (sch_fq, bbr, ecn, etc)
- And be aware of what works in the DC doesn’t work so well on DSL
- Monitor RTTs in your applications, use tools like irtt and flent to design your networks
- Grok TCP (I hope this talk helped!)
- There’s still a lot of software work to be done – only 3 linux wifi drivers so far (ax200 is in progress), not a lot of movement in dsl, fiber, and few signs the 5G people are paying attention
- The network you save may be your own.

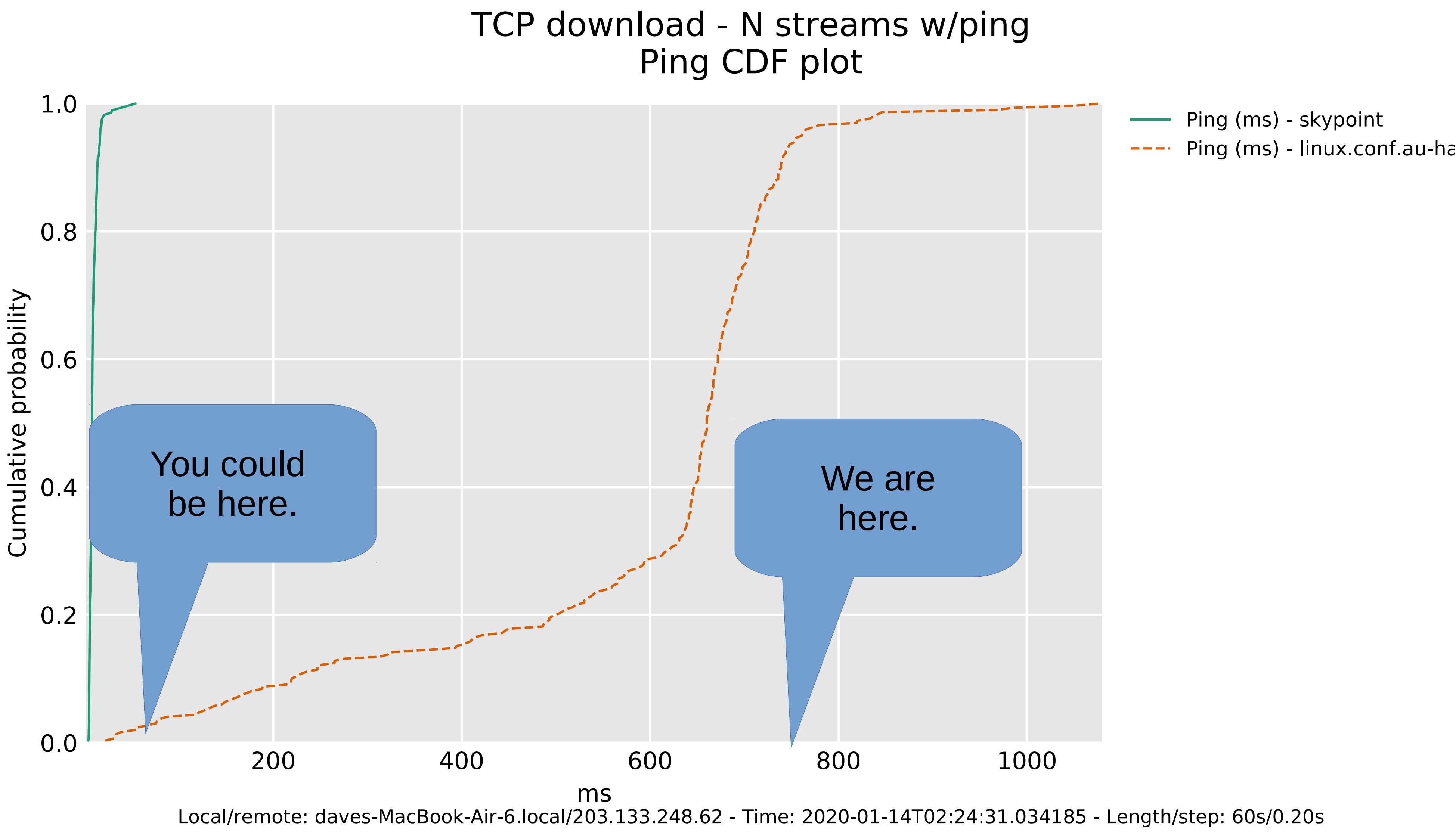
The LCA 2020 WiFi network, redux...



Linux.conf.au wifi cdf plot



Questions?


TCP download - N streams w/ping
Ping CDF plot
1.0
0.8
0.6
0.4
0.2
0.0
Cumulative probability
200
400
600
800
1000
ms
Ping (ms) - skypoint
Ping (ms) - linux.conf.au-ha
You could be here.
We are here.
Local/remote: daves-MacBook-Air-6.local/203.133.248.62 - Time: 2020-01-14T02:24:31.034185 - Length/step: 60s/0.20s

Extra Slides

- Work on better bandwidth continues…

Modern Developments: BBR

- Great paper: “Congestion based Congestion Control”
- KleinRock's BBR paper is also great
- Principal observation: You cannot measure delay and capacity at the same time.
- And: Fixing the endpoints is way easier than replacing the routers
- BBRv2 is in heavy development at the ietf.

Modern Developments: ECN

- The 2 bit ECN field in the IP header lets AQMs mark rather than drop packets to indicate “please slow down”.
- 1.5% of web traffic is now using it.
- fq_codel – billion users - uses - RFC3168 – where a drop = mark.
- There’s a group in the IETF that wants to change the definition of that to make many marks per RTT mean something else.
- There’s another group proposing – “Some Congestion Experienced” which is backward compatible to RFC3168…
- For more details about the current state of that fight , beer is required. After the conference.

Development timeline

- 1983 – Arpanet retired
- 1984 – First internet congestion collapse observed
- 1986 – TCP reno (RFC6582) developed
- 1988 – RED developed
- 1990 – SFQ developed
- 1995 – DRR developed
- 2002 – RFC3168 (ECN) released
- 2010 - “Bufferbloat” coined & categorized
- 2012 – BQL, Codel (RFC8289),, FQ_Codel (RFC8290) & Pie (RFC8033) Developed
- 2015 – Pacing made to work
- 2016 – BBR released, “sch_cake” released
- 2017 – Wifi Fixed on a few chipsets
- 2020 – Where we stand today