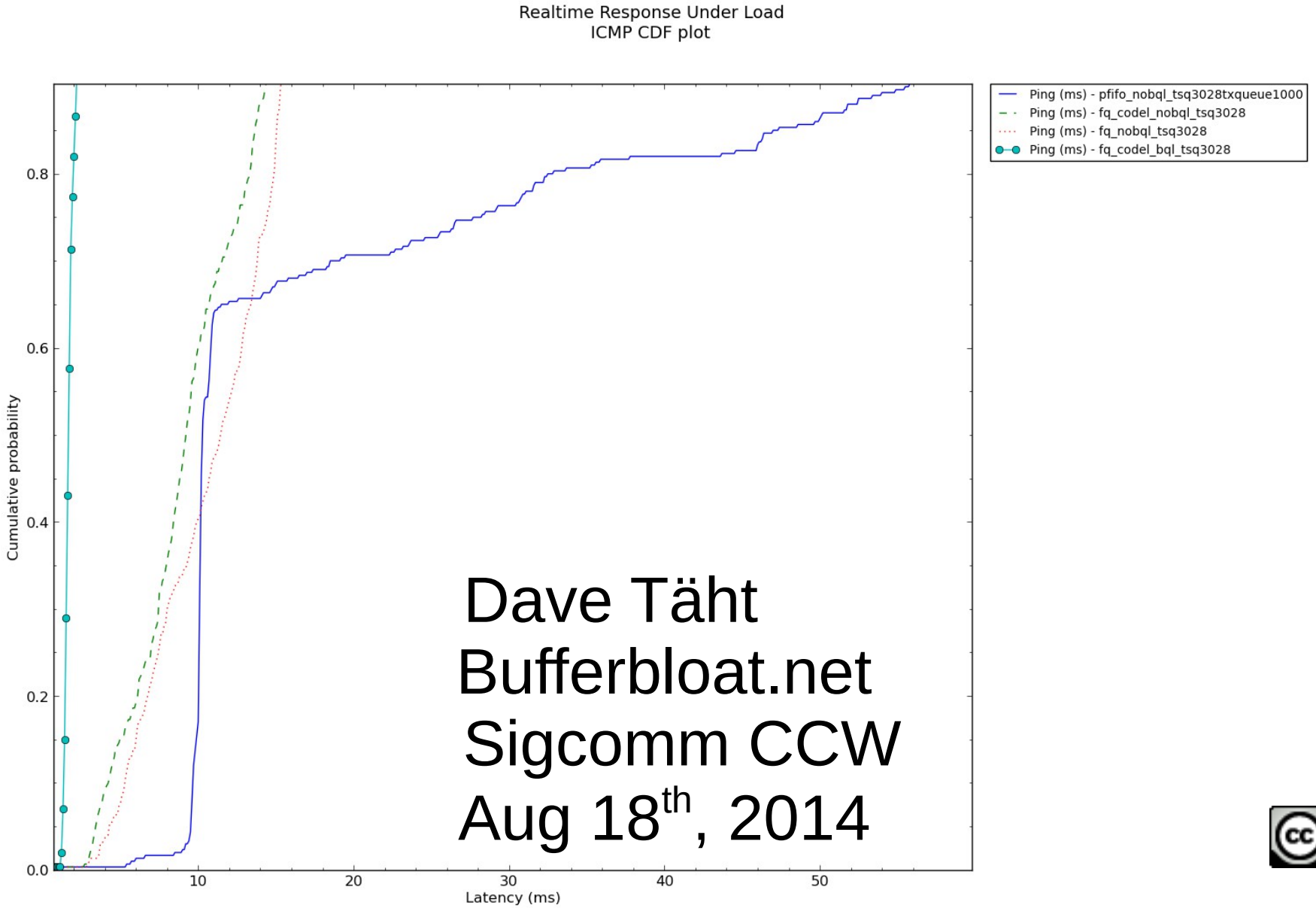
The value of repeatable experiments and negative results



Dave Täht
Bufferbloat.net
Sigcomm CCW
Aug 18th, 2014

Installing netperf-wrapper

- Linux (debian or ubuntu)
  - sudo apt-get install python-matplotlib python-qt4 fping subversion git-core
  - git clone https://github.com/tohojo/netperf-wrapper.git
  - cd netperf-wrapper; sudo python setup.py install
  - See the readme for other dependencies.(netperf needs to be compiled with –enable-demo)
- OSX
  - Requires macports or brew, roughly same sequence
- Data Set: http://snapon.lab.bufferbloat.net/~d/datasets/
- This slide set: http://snapon.lab.bufferbloat.net/~d/sigcomm2014.pdf

Bufferbloat.net Resources

Reducing network delays since 2011...

Bufferbloat.net: http://bufferbloat.net
Email Lists: http://lists.bufferbloat.net (codel, bloat, cerowrt-devel, etc)
IRC Channel: #bufferbloat on chat.freenode.net
Codel: https://www.bufferbloat.net/projects/codel
CeroWrt: http://www.bufferbloat.net/projects/cerowrt
Other talks: http://mirrors.bufferbloat.net/Talks
Jim Gettys Blog: http://gettys.wordpress.com
Google for bloat-videos on youtube...
Netperf-wrapper test suite: https://github.com/tohojo/netperf-wrapper

“Reproducible” vs “Repeatable” Experiments

- Researcher MIGHT be able to repeat experiment in light of new data.
- Reviewer typically lacks time to reproduce.
- Reader MIGHT be able, from the descriptions in the paper, reproduce the result, by rewriting the code from scratch.

- Researcher MUST be easily repeat the experiment in light of new data.
- Reviewer SHOULD be able to re-run the experiment and inspect the code.
- Reader SHOULD be able from the code and data supplied by the research, repeat the experiment quickly and easily.

What's wrong with reproducible?

- Coding errors are terribly common – code review is needed...
- Reproduction takes time I don't have.
- Science in physical law is constant. Gravity will be the same tomorrow, as today...
- In computer science, the laws are agreements, and protocols, and they change over time.
- You have 30 seconds to get my attention.
- 5 minutes, tops, to make a point.
- There are thousands of papers to read.
- Show me something that *works*... that I can fiddle with, or change a constant to match another known value... anything....

Aaron Swartz

Open Access

- Cathedral and the Bazaar
- Why programmers don't bother joining ACM
- Artifacts

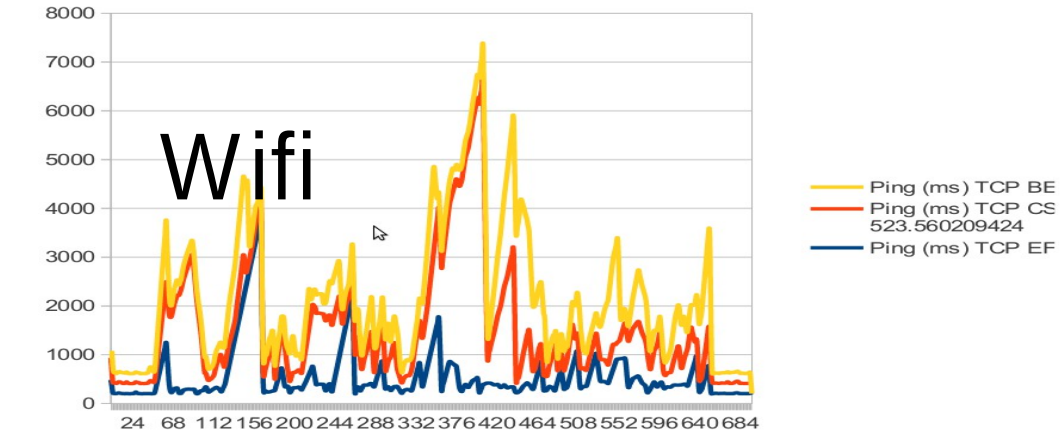
A working engineer's Plea: Help me out here!

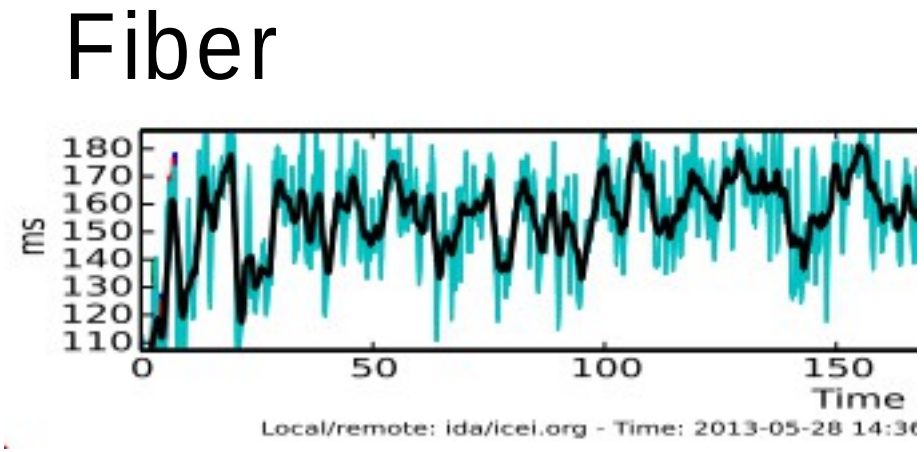
- First, lose TeX:
  - HTML and mathjax work just fine now.
  - I forgot everything I knew about TeX in 1992....
- Math papers
  - Please supply a link to a mathematica, wolfram language, or spreadsheet of your analysis
    - You had to do it for yourself anyway...
- CS Papers
  - Supply source code in a public repository like github
  - Put raw results somewhere
  - Keep a vm or testbed around that can repeat your result
  - Make it possible for others to repeat your experiment
  - It would be great if there was a way to comment on your paper(s) on the web
- And do Code Review, before Peer Review
  - Your code might suck. News for you: everyone's code sucks.
  - Defect rates even in heavily reviewed code remain high, why should yours be considered perfect?

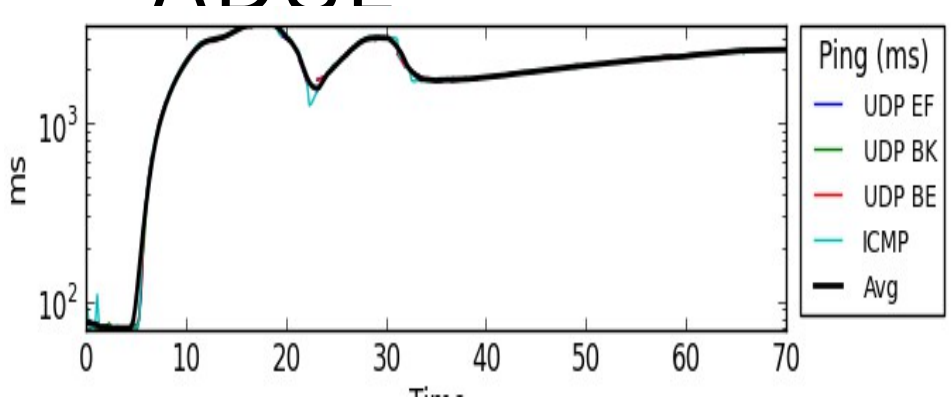
Several Fundamental Laws of Computer Science

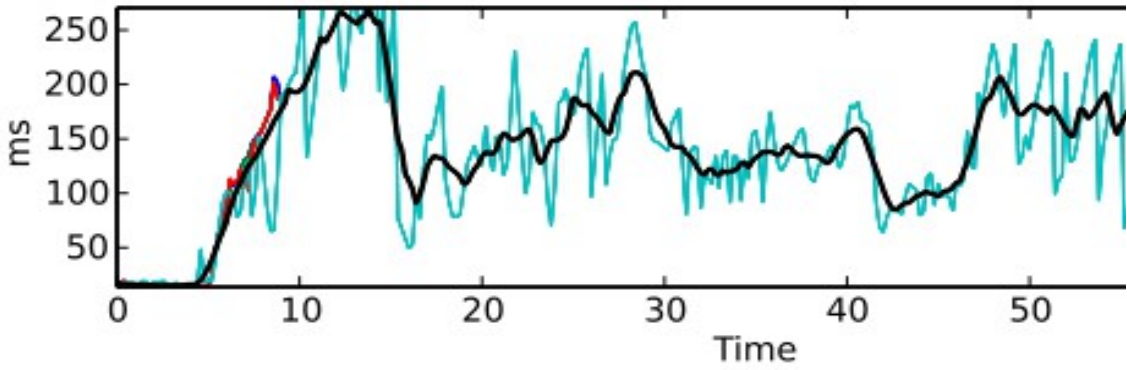
- Murphy's Law: “Anything that can go wrong, will...”
  - At the worst possible time...
  - At the demo...
  - And in front of your boss.
- Linus's Law: “With enough eyeballs, all bugs are shallow”
  - “With enough bugs, all eyeballs are shallow” - Post-Heartbleed corollary
- Everything evolves – getting faster, smaller, more complex, with ever more interrelationships...
- And if assumptions are not rechecked periodically...
- You get:

Network Latency with Load: 2011









March, 2011: **no** AQM or Fair Queue algorithm worked right on Linux

- Turn on red, sfq, what have you...
- On ethernet at line rate (10mbit, 100mbit, GigE) – nearly nothing happened
- On wifi – nothing happened
- On software rate limited qos systems – weird stuff happened (on x86) that didn't happen on some routers
- Everything exhibited exorbitant delays... wifi especially, but ethernet at 100mbit and below was also terrible...

And we didn't know why...

Bufferbloat was everywhere...And the fq/aqm field was dead, dead, dead.

Bufferbloat was at all layers of the network stack

- Virtual machines
- Applications
- TCP
- CPU scheduler
- FIFO Queuing systems (qdiscs)
- The encapsulators (PPPoe and VPNs)
- The device drivers (tx rings & buffers)
- The devices themselves
- The mac layer (on the wire for aggregated traffic)
- Switches, routers, etc.

We went back to the beginning of Internet Time...

- 1962 Donald Davies “packet” = 1000 bits (125 bytes)

  "The ARPANET was designed to have a reliable communications subnetwork. It was to have a transmission delay between hosts of less than ½ second. Hosts will transmit messages, each with a size up to 8095 bits. The IMPs will segment these messages into packets, each of size up to 1008 bits."

  -- http://www.cs.utexas.edu/users/chris/think/ARPANET/Technical_Tour/overview.shtml
- 70s-80s packet size gradually grew larger as headers grew
- 80s ethernet had a maximum 1500 MTU
- 90s internet ran at a MTU of 584 bytes or less
- IPv6 specified minimum MTU as 1280 bytes
- 00's internet hit 1500 bytes (with up to 64k in fragments)
- 10's internet has TSO/GSO/GRO offloads controlling flows with up to 64k bursts – TSO2 has 256k bursts...
- Were these giant flows still “packets”? Average packet size today is *still* ~300 bytes... VOIP/gaming/signalling packets generally still less than 126 bytes

Fundamental reading: Donald Davies, Leonard Kleinrock & Paul Baran

- http://www.rand.org/content/dam/rand/pubs/research_memoranda/2006/RM3420.pdf
- Kleinrock - “Message delay in communication nets with storage” http://dspace.mit.edu/bitstream/handle/1721.1/11562/33840535.pdf
- Are Donald Davies 11 volumes on packet switching not online??
- Van Jacobson & Mike Karels - “Congestion Avoidance and Control”

  http://ee.lbl.gov/papers/congavoid.pdf
- Nagle: “Packet Switches with infinite storage” http://tools.ietf.org/html/rfc970

Fair Queuing Research, revisited

- RFC970
- Analysis and simulation of a Fair Queuing Algorith
- Stochastic Fair Queueing (SFQ) – used in wondershaper
- Deficit Round Robin- had painful setup
- Quick Fair Queuing – crashed a lot
- Shortest Queue First – only in proprietary firmware
- FQ_CODEL – didn't exist yet

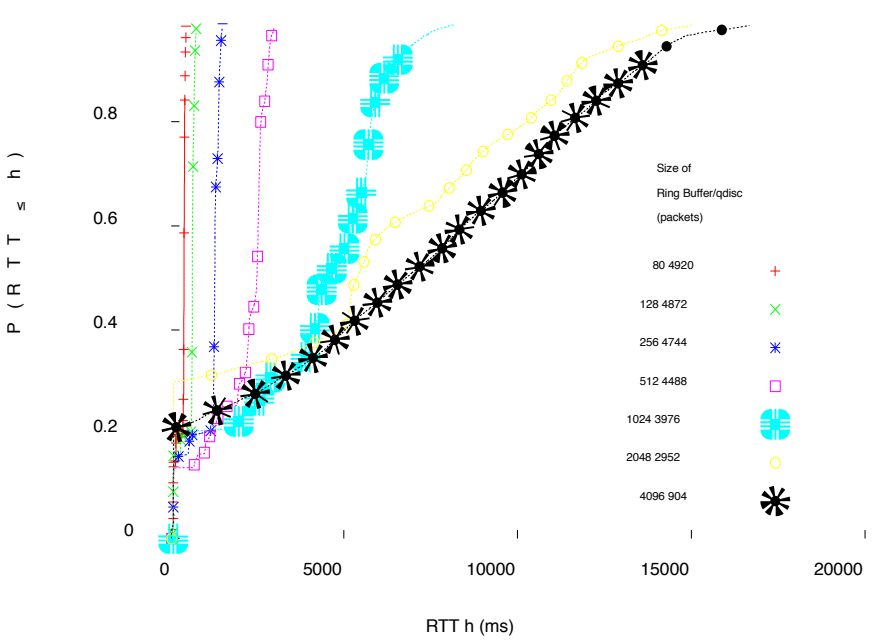
AQM research, revisited

- RED – Didn't work at all like the ns2 model
- RED in a different Light – not implemented
- ARED - unimplemented
- BLUE – couldn't even make it work right in ns2
- Stochastic Fair Blue – implementation needed the sign reversed and it still didn't work
  - We talked to Kathie Nichols, Van Jacobson, Vint Cerf, John Nagle, Fred Baker, and many others about what had transpired in the 80s and 90s.
  - And established a web site and mailing list to discuss what to do...

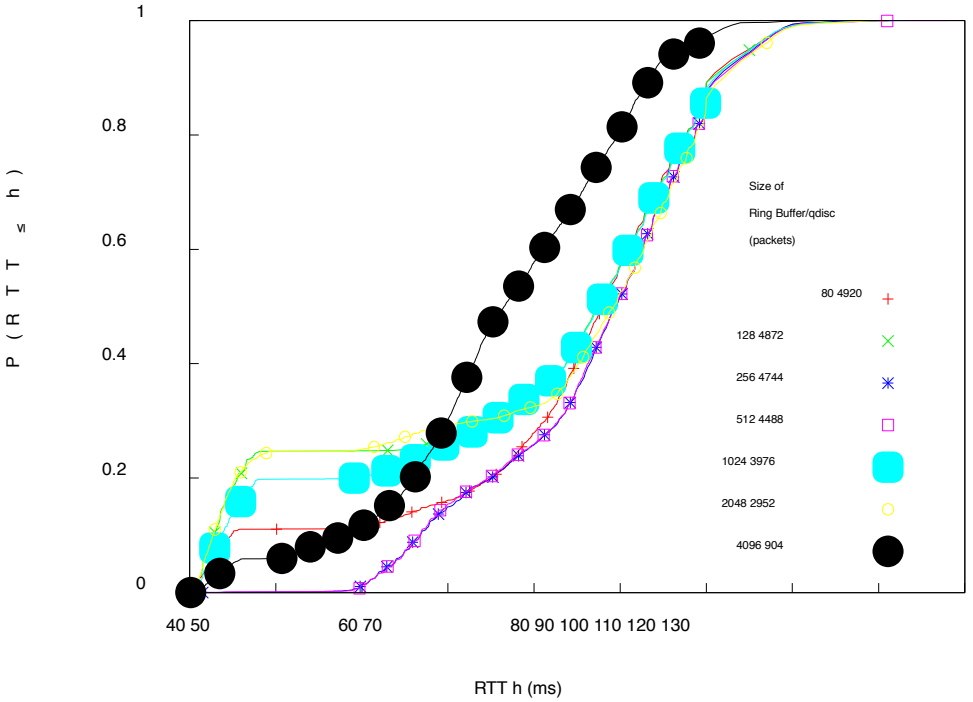
Some debloating progress against all these variables...

- 2011 Timeline:
  - April: TSO/GSO exposed as causing htb breakage
    - Turning it off gave saner results on x86
  - June: Infinite retry bug in wifi fixed
    - :whew: no congestion collapse here
  - August: “Byte Queue Limits” BQL developed
    - Finally, multiple AQM and FQ algorithms had bite
    - Even better, they started scaling up to 10GigE and higher!
  - October: eBDP algorithm didn't work out on wireless-n
  - November: BQL deployed into linux 3.3
    - Multiple drivers upgraded
    - 2 bugs in RED implementation found and fixed
- Finding these problems required throwing out all prior results (including all of yours) and repeating the experiments...
  - And I stopped believing in papers...

The value of Byte Queue Limits
2 orders of magnitude lag reduction



Latency without BQL



9:Latency with BQL.

Src: Bufferbloat Systematic Analysis (published this week at its2014)

New starting Point: WonderShaper

- Published 2002:
  - Combines fair queuing + a poor man's AQM (permutation) on outbound
  - 3 levels of service via classification
  - 12 lines of code...
- Worked, really well, in 2002, at 800/200kbit speeds.
  - I'd been using derivatives for 9 years...
  - Most open source QOS systems are derived from it
  - But it didn't work well in this more modern era at higher speeds
- What else was wrong?

  (google for “Wondershaper must die”)

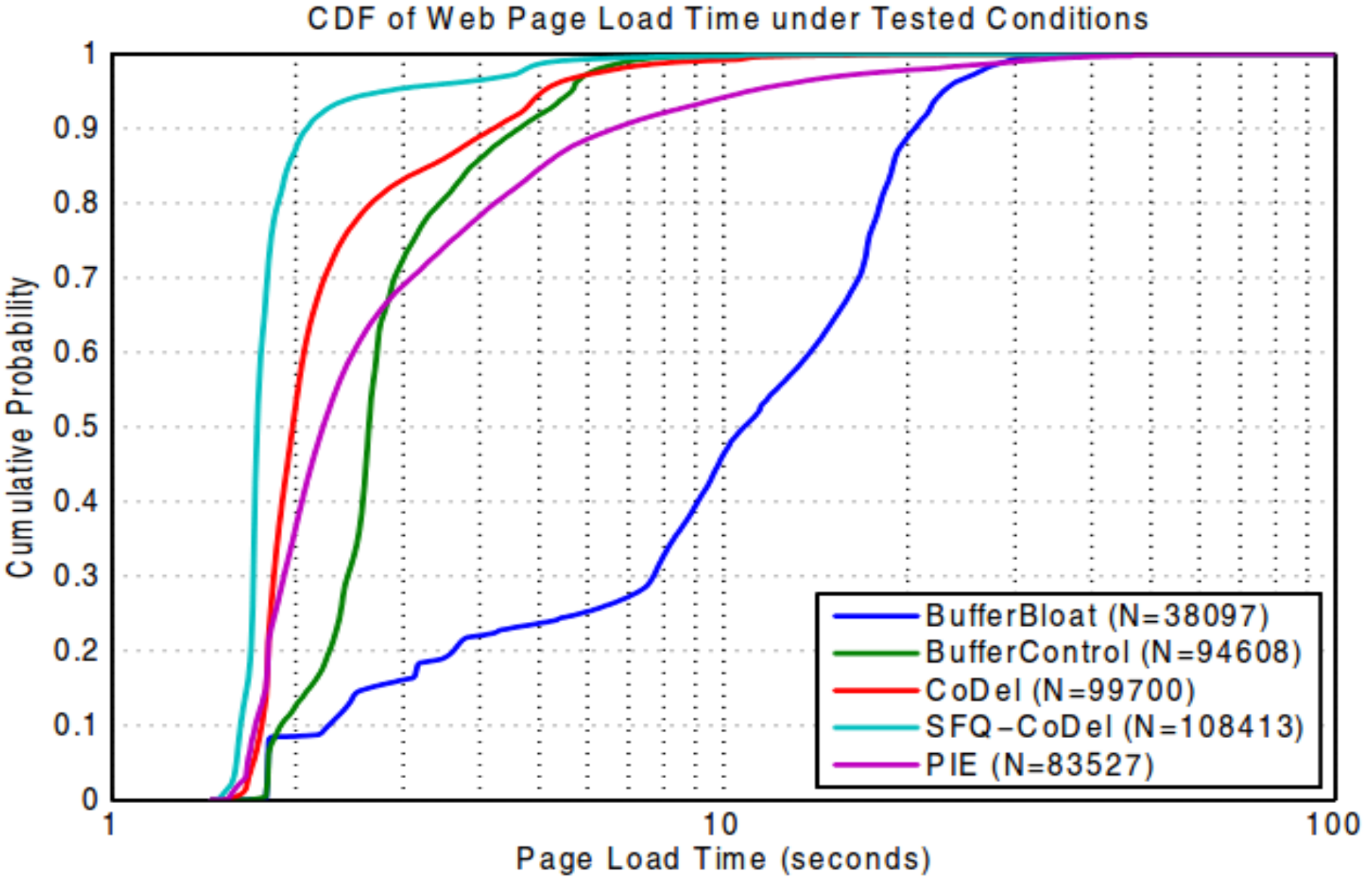
2012 progress in Linux

- Jan: SFQRED (hybrid of SFQ + RED) implemented
  - ARED also
- May: Codel algorithm published in ACM queue
- June: codel and fq_codel (hybrid of DRR + codel) slammed into the linux kernel
- Aug: pie published
- And....
  - The new FQ and AQM tech worked at any line rate without configuration! (so long as you had BQL,
  - Or HTB with TSO/GSO/GRO turned off.
- SQM (CeroWrt's Smart Queue Management) developed to compare them all...
- Netperf-wrappers started to test them all...

FQ_Codel

- A hybrid of
  - DRR (Deficit Round Robin) Fair Queueing
  - Codel (for queue length management)
  - With some SQF-like features for the sparsest flows
- Achieves near-0 queue delay for sparse flows, and under 20ms of queue delay for heavier loads
- It and variants are currently winner among the congestion management on-the-router tools in OpenWrt, dd-wrt, CeroWrt, Ipfire, and free.fr's deployment
- NS2 and NS3 models available
- http://tools.ietf.org/html/draft-hoeiland-joergensen-aqm-fq-codel-00

Web page load time wins



Video at: http://www.youtube.com/watch?v=NuHYOu4aAqg

“FQ_Codel provides great isolation... if you've got low-rate videoconferencing and low rate web traffic they never get dropped. A lot of issues with IW10 go away, because all the other traffic sees is the front of the queue. You don't know how big its window is, but you don't care because you are not affected by it. FQ_Codel increases utilization across your entire networking fabric, especially for bidirectional traffic...”

“If we're sticking code into boxes to deploy codel, *don't do that*.

Deploy fq_codel. It's just an across the board win.”

- *Van Jacobson*

IETF 84 Talk

Linux TCP advances: 2010-2014

- Linux 3.0: Proportional Rate Reduction
- Linux 3.3: Byte Queue Limits
- Linux 3.4 RED bug fixes & IW10 & SFQRED
- Linux 3.5 Fair/Flow Queuing packet scheduling (fq_codel, codel)
- Linux 3.6 Stability improvements to fq_codel
- Linux 3.7 TCP small queues (TSQ)
- Linux 3.8 HTB breakage
- Linux 3.11 HTB fixed
- Linux 3.12 TSO/GSO improvements
- Linux 3.13 Host FQ + Pacing
- Linux 3.15 Change to microseconds from milliseconds throughout networking kernel
- The Linux stack is now mostly “pull through”, where it used to be “push”, and looks nothing like it did 4 years ago.
- At least a dozen other improvements I forget

PLEASE don't bother writing any more papers against linux 3.3. Please use the most current kernels you can.
A 3.2.X kernel is a STABLE release containing NONE of these improvements.

Please repeat your experiments!

Host Queue improvments: sch_fq

- Most Fair Queuing systems were developed in the 80s and 90s using approximations for “Time” and “Flows” due to lack of CPU during those decades.
- sch_fq is a new pure fq scheduler by Eric Dumazet, using wallclock time and a red/black tree to manage all flows without any stochastic hashing.
- Parts are inherited from FQ_codel (but no AQM)
- Has sufficient bands to be a drop in replacement for PFIFO_FAST
- Has support for a larger initial quantum (IW burst support)
- Supports “Pacing” of (particularly) ON/OFF flows like DASH with a setsockopt.
- Works with *millions* of active flows
- Will probably become the Linux default.

With BQL, TSQ, and fq_codel, most of us declared victory, and went off to implement them in the real world!

- And (we're sorry...) didn't write a paper on them.
- Van, Eric, & Andrew went off to google
- OpenWrt “Barrier Breaker”, CeroWrt, Ipfire and free.fr adopted it all immediately... followed by dd-wrt and others...
- It went into multiple other Linux distros
- PIE went into DOCSIS 3.1
- And we went to try and refine them further...
  - With tons and tons of negative results
  - And refinements throughout the rest of linux...

Some pending Internet Drafts

- AQM working group
- https://datatracker.ietf.org/doc/charter-ietf-aqm/
- Pending drafts:

-
- http://snapon.lab.bufferbloat.net/~d/draft-taht-home-gateway-best-practices-00.html
- http://tools.ietf.org/html/draft-white-aqm-docsis-pie-00
- http://tools.ietf.org/html/rfc2309 Is beiing revised
- http://tools.ietf.org/html/draft-ietf-aqm-recommendation-03
- http://tools.ietf.org/id/draft-kuhn-aqm-eval-guidelines-00.txt
- http://tools.ietf.org/html/draft-hoeiland-joergensen-aqm-fq-codel-00
- http://tools.ietf.org/html/draft-nichols-tsvwg-codel-02
- http://sandbox.ietf.org/doc/draft-baker-aqm-sfq-implementation/

Fair Queuing Vs AQM Vs E2E

The debate on where congestion control mechanisms should go has raged for 30+ years, going back to: Jan 16-17, 1986: The very first ietf meeting

- IMHO: In order of effectiveness
  - FQ wins over
  - AQM which wins over
  - E2E
  - AND that all these techniques are useful and needed.
- But: I didn't believe that til after thousands of experiments covering a wide range of scenarios, using multiple tcps, multiple aqm systems, and writing tons of tests,
- And... doing multiple test deployments across the cerowrt userbase
- “fq-codel” is the thing to beat (on ethernet/cable/fiber) Not wifi! Yet!
- Since then we have focused on improving various tools to look harder at multiple network scenarios.
- Maybe I can convince you with repeatable experiments. Or you convince me with repeatable experiments...

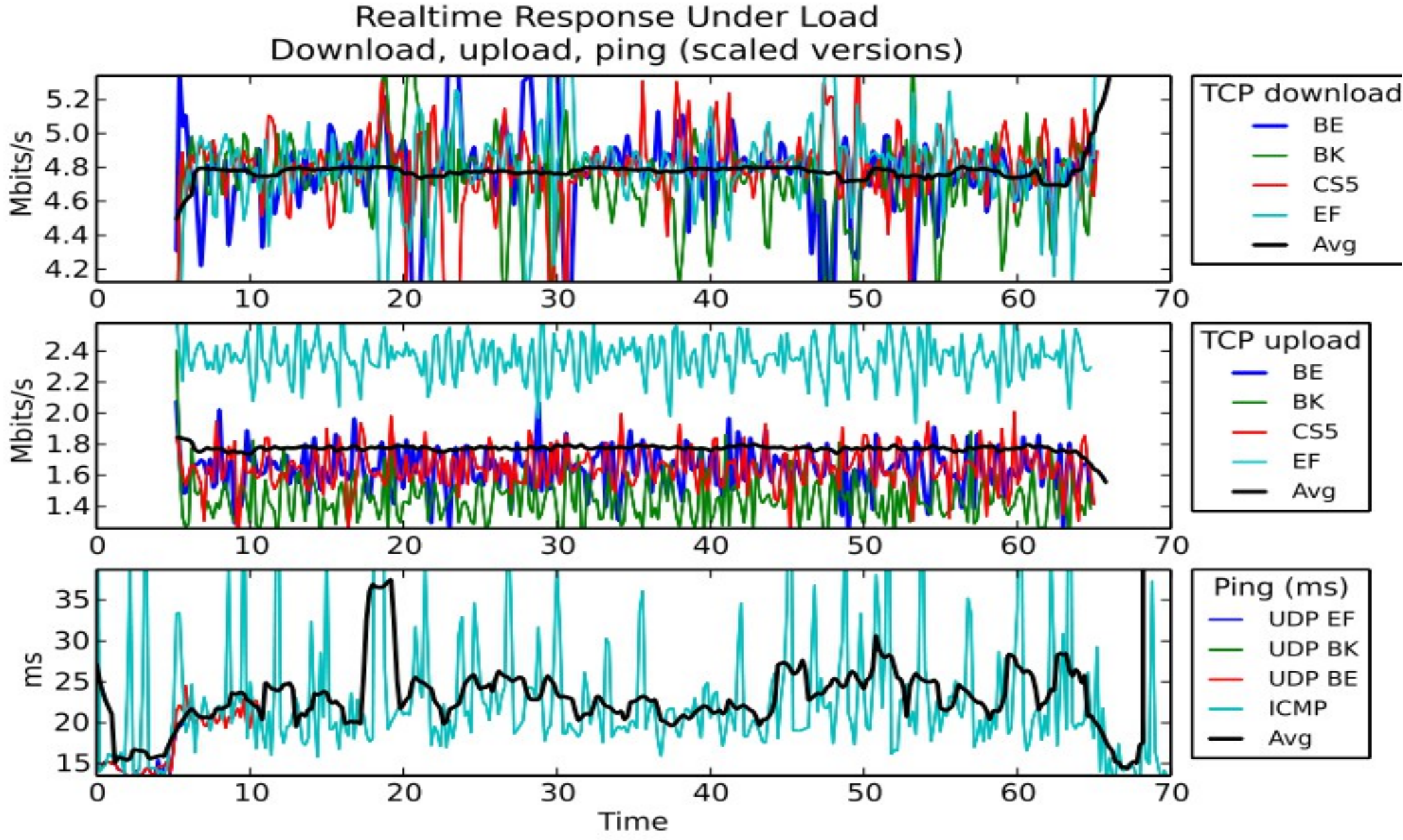
Some Repeatable Experiments

- Tcptrace and xplot.org
- Netperf-wrapper (by Toke Hoeiland-Joergensen)
  - Standardized data format, 30+ network specific tests, 20+ plot types, extensive support for batching and other automation, usage of alternate TCP algorithms, in combination with other web and voip-like traffic
- CeroWrt
  - Inexpensive actual router configurable with nearly every AQM and FQ algorithm using “SQM”
  - Emulations of common CMTS and DSLAM behavior
  - Results proven sane to 50mbits
  - Most of cerowrt is already in openwrt “Barrier Breaker”.
- SQM (“Smart Queue Management”)
  - Drop in replacement for wondershaper portable to all modern linuxes
  - Uses fq_codel by default
- NS2 models of codel, sfq_codel, and pie moving to mainline
- NS3 models of codel, fq_codel, asymmetric edge networks, etc,
  - Google 2014 summer of code
  - Nearly done!

Installing netperf-wrapper

- Linux (debian or ubuntu)
  - sudo apt-get install python-matplotlib python-qt4 fping subversion git-core
  - git clone
  - cd netperf-wrapper; sudo python setup.py install
  - See the readme for other dependencies.(netperf needs to be compiled with –-enable-demo)
- OSX
  - Requires macports or brew, roughly same sequence
- Data Set: http://snapon.lab.bufferbloat.net/~d/datasets/
- This slide set: http://snapon.lab.bufferbloat.net/~d/sigcomm2014.pdf
- Run:
  - netperf-wrapper –gui yourchoiceofdata*.gz
  - Or netperf-wrapper -x -H snapon.lab.bufferbloat.net rrul

20/8Mbit cable modem performance w htb + 3 tier fq_codel (SQM)



Some newer reads

- Controlling Queue Delay : http://queue.acm.org/detail.cfm?id=2209336
- 2002 TCP Pacing paper seemingly refuted http://reproducingnetworkresearch.wordpress.com/2013/03/13/cs244-13-tcp-pacing-and-buffer-sizing/
- On the self similar nature of Network Traffic:

  http://ecee.colorado.edu/~ecen5032/handouts/94LelandSelfSim.pdf
- Google for lots of papers on *“Packet Pairing”.*
- Van Jacobson's Queue Rant: http://www.pollere.net/Pdfdocs/QrantJul06.pdf
- Pfabric and related look a lot like “shortest queue first”:

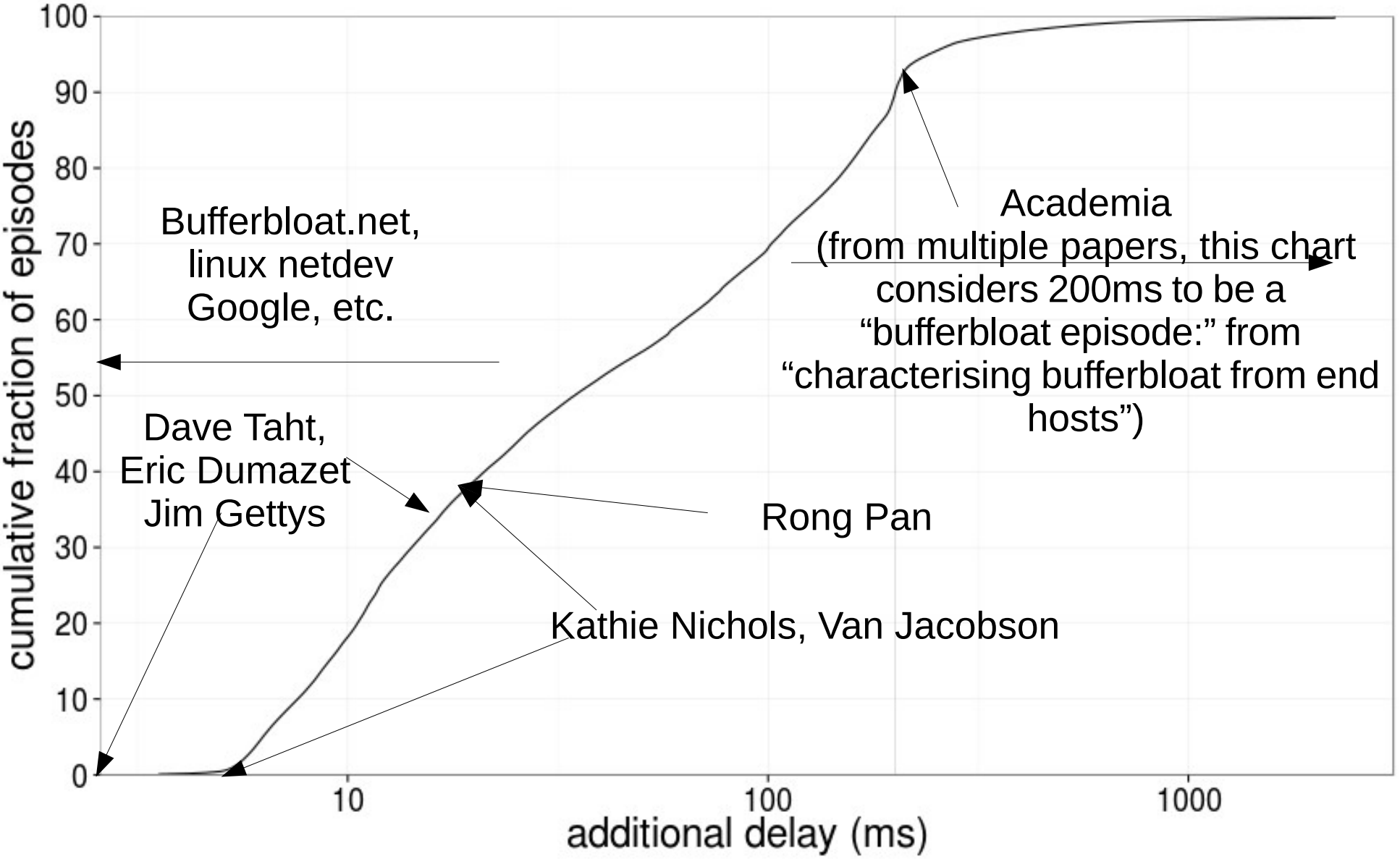
  http://www.stanford.edu/~skatti/pubs/sigcomm13-pfabric.pdf
- On the Co-existence of AQM and Low Priority Congestion Control:

  http://perso.telecom-paristech.fr/~drossi/paper/rossi13tma-b.pdf
  - Note that LPCC has a dirty little secret – 100ms delay target...

I wish I had some red rubber stamps for new papers....

- [Rejected: Ludicrous constants]
- [Rejected: Can't replicate results]
- [Rejected: Incorrect assumptions]
- [Rejected: Tool is broken]
- [Rejected: can't reproduce results]
- [Rejected: doesn't replicate original bufferbloat experiment]
- [Rejected: over-uses statistical legerdemain]
- [Rejected: doesn't use real-world settings for buffering, rtt, etc]
- [Rejected: tool doesn't work]
- [Rejected: Simulation implements outdated TCP]
- [Rejected: Simulation doesn't measure sane loads]
- [Rejected: Analytical TCP models completely disregard slow start]
- [Rejected: Poisson not pareto distributions]
- [Rejected: 95th percentile considered acceptable]

Gripe #1: Acceptable queueing delays?



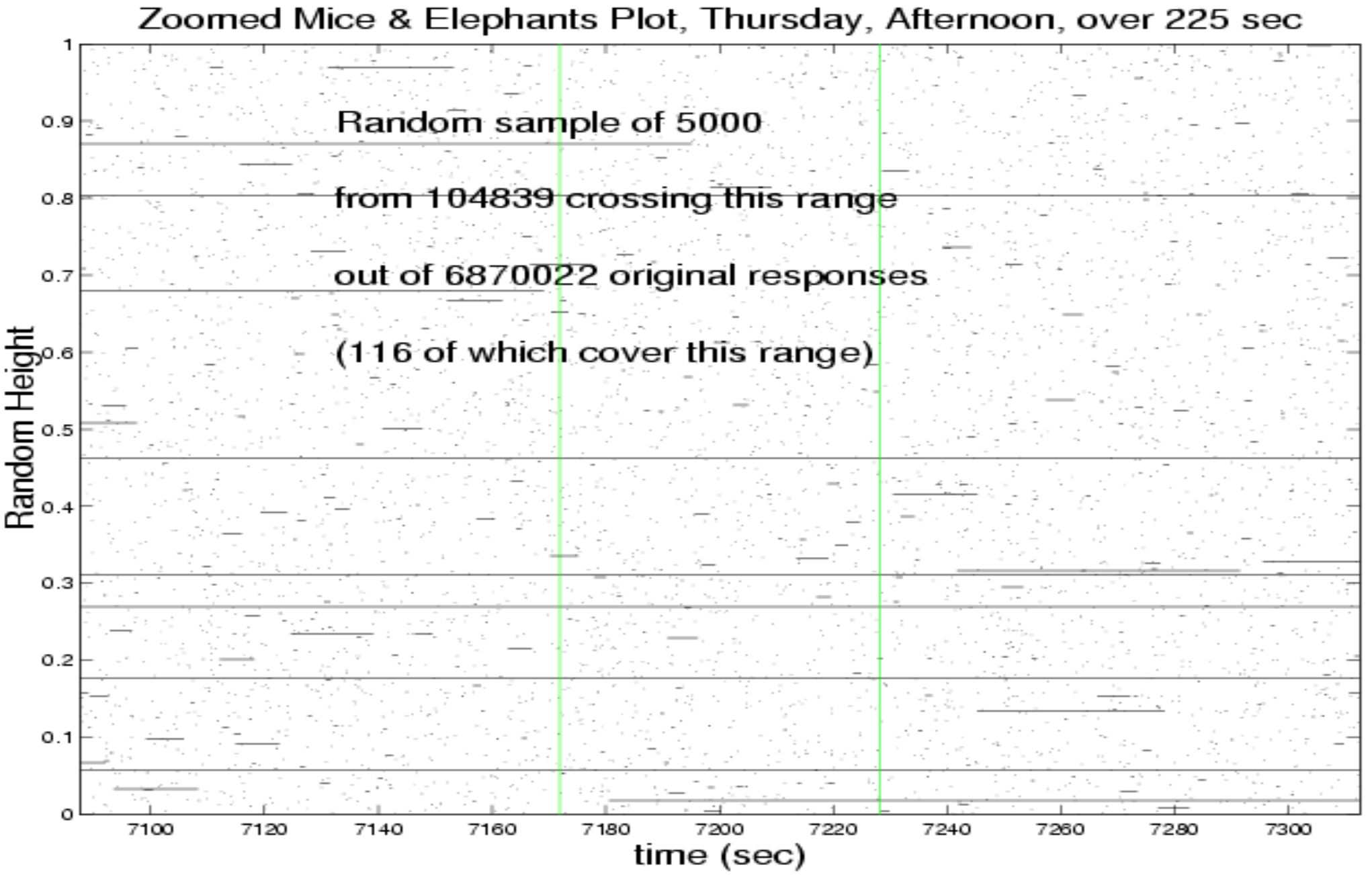
Gripe 2: Bandwidths and Physical RTTs under test

Technology	Bandwidth Down/Up	RTT	Typical buffering
Cable	8M/1M, 20/5, 50/10, 100/20, higher	16-38ms	256k-1Mbyte down 256kup
FIOS	25M/25M, 50/50, higher	8-18ms	512K down
DSL	1M/384k, 8/768,20/1	8-40ms	64k down, 64k up
Wifi	1Mbit-1+gbit	1ms-80s	512 packets
Google Fiber	5M/1M,1G/1G	3-5ms	Unk
Compared	Verses	Academia	
NS2	1-10Mbit	100ms	50-100 packets
NS3	1-10Mbit	4-200ms	50-100 packets
Mininet	1-100Mbit	4-100ms	50 packets

Real Network “Constants”

- Speed of light in the medium
- Media Acquisition time (Cable Request/Grants, Wifi EDCA)
- Min/max Packet size (MTU)
- TCP
  - Initial window (IW4, IW10)
  - Slow Start Threshold
  - RTO timeout
- Aggregation sizes
- These constants matter hugely at low bandwidths (< 10Mbit), increasingly less as you get past 100mbit.
- If you start fixing the endpoints, your choices change

Gripe 3: More realistic loads



Moving forward: Further challenges

- Quest for a full replacement for PFIFO_Fast (diffserv support)
- Adding support for software or hardware rate limiting at higher rates
- Other delay based AQM systems (fq_PIE, etc)
- Further research into the interrelationship of drop mechanisms and fair queuing at high (1gig+ rates) on real traffic
- Developing better tests
- Pouring codel and fq_codel derivatives into hardware and other operating systems
- Coaxing the universe to try it and deploy it
- And there are a few problematic protocols like uTP and DASH, and new ones like webrtc, that need to be looked at harder
- And...

Smarter Queue Management

- Smashing bloat in Device Drivers
- Tighter OS abstractions
- Applications
- Policing
- Rate Shaping
- Active Queue Length Management
- Fair or Flow Queueing
- DiffServ-based Prioritization

What role for ECN? (Explicit congestion notification?)

- Positives:
  - Use a spare 2 bits to mark a packet: lossless signaling of congestion
  - Very useful in Data Centers and in well controlled links
- Negatives:
  - Used to crash a lot of
  - API Breaks “the everything is a file” abstraction
  - No sane way to negotiate end to end and easily abused on the open Internet
  - After 10 years of near-deployment some theorists want to redefine it from “Floyd ECN” (a mark is equivalent to a drop) to “Immediate ECN” (DCTCP) , a multi-bit-over-multi-packet signal.
- Support for it is on by default in FQ_codel, but off in everything else, including TCP endpoints. Mosh is using it as a test, with good results.
- Can it be used in new protocols like Quic, or in Videoconferencing, or routing protocols?

Open questions on FQ_Codel

- It and variants win across the board against pure AQM.
- Still some debate about SFQ-like or DRR-like attributes
  - At lower bandwidths SFQ-like wins
  - Higher bandwidths DRR wins
  - QFQ has “interesting” results
- What's the right number of flows relative to bandwidth and real use?
- Can it be implemented in hardware?
- What's the right target delay?
- Can the random permutation of the hash be detected and abused?
- What's the right things to hash against (5 tuple? Mac addr?)
- What are the effects inline with other Queue management systems?
- Can it apply to tx-op limited layer 2 rather than just packets?
- Can weighting be used?

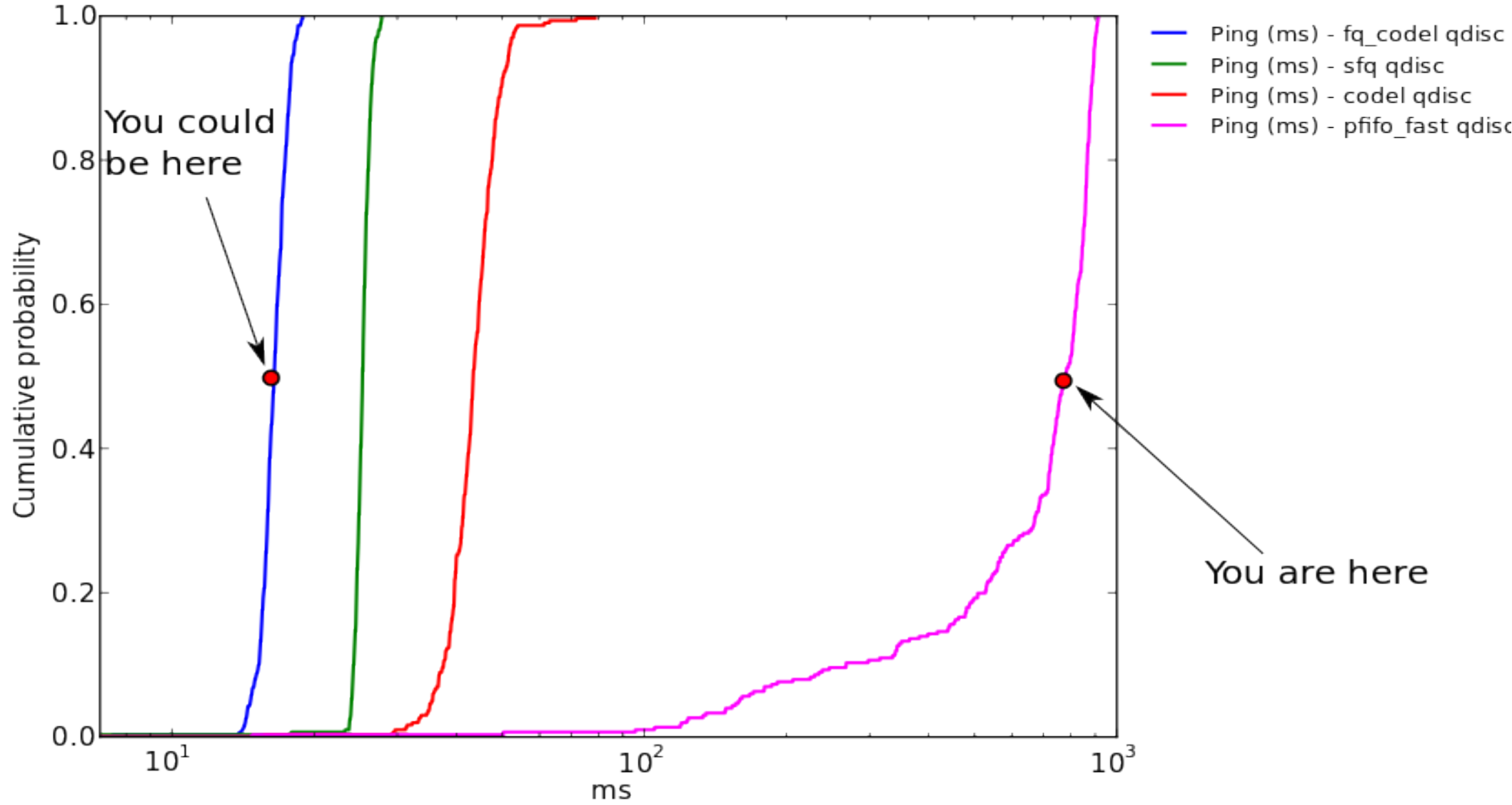
Big Layer 2 problems ahead

- Wireless-n/AC,DOCSIS, GPON, MOCA, all do packet aggregation.
- They also do scheduling (request/grant for DOCSIS, GPON, MOCA, EDCA for wireless)
- All these new technologies aren't very open and move a great deal of “intelligence” into places where they can't be analyzed or improved by open engineers or academia

Bufferbloat.net's next project: MAKE-WIFI-FAST

- We think we can cut queue delay on wifi under load to 20ms or less, at any bandwidth.
- There are hard problems left with aggregation, txop optimization, qos, driver re-organisation, and EDCA scheduling.
- Please come help!
- https://lists.bufferbloat.net/listinfo/make-wifi-fa

Questions?


1.0
0.8
0.6
0.4
0.2
0.0
Cumulative probability
You could be here
You are here
Ping (ms) - fq_codel qdisc
Ping (ms) - sfq qdisc
Ping (ms) - codel qdisc
Ping (ms) - pfifo_fast qdisc
10^1
10^2
10^3
ms