Battling Bufferbloat with CeroWrt (and some updates on Linux 3.3)

D. Täht

Bufferbloat.net

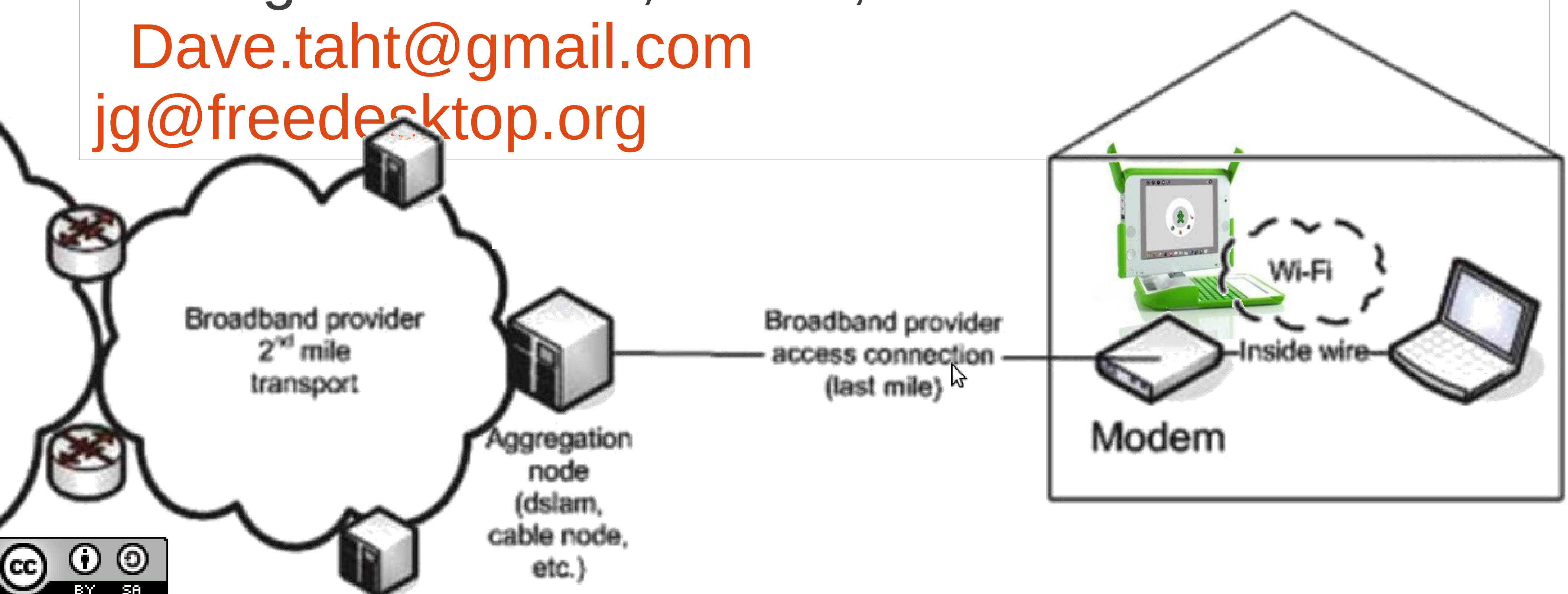
Visiting Researcher, LINCs, Paris

Dave.taht@gmail.com

J. Gettys

Alcatel-Lucent

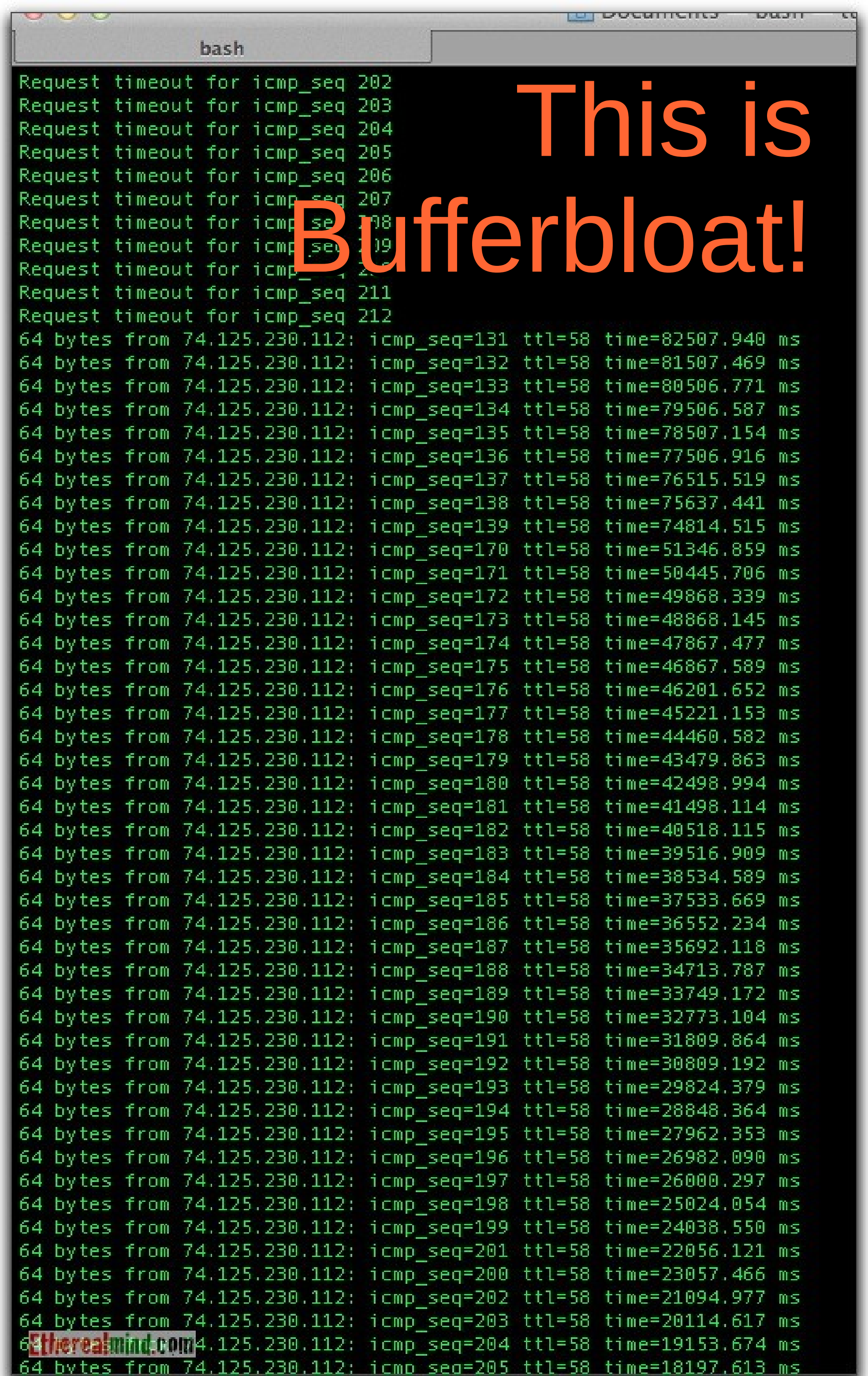
jg@freedesktop.org

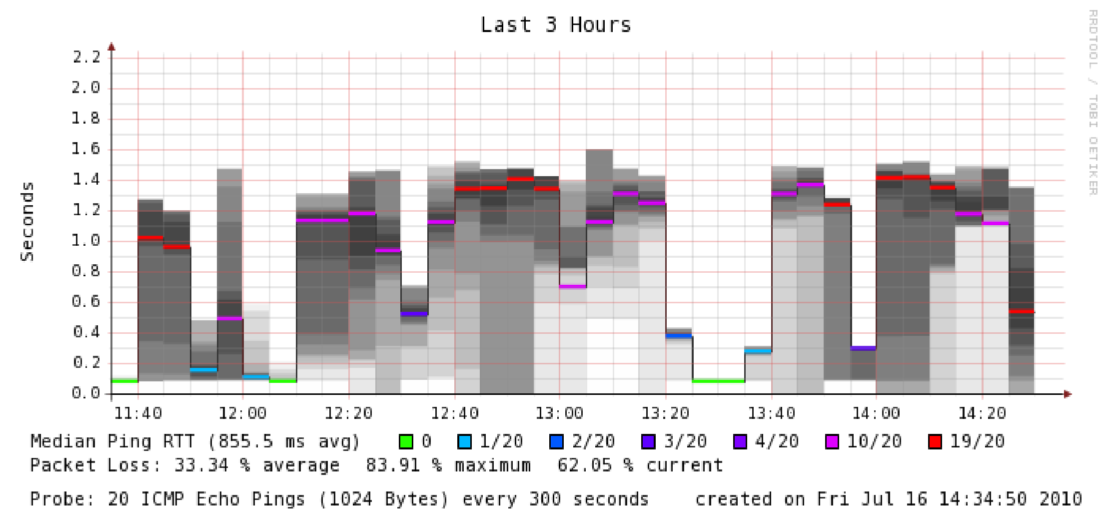


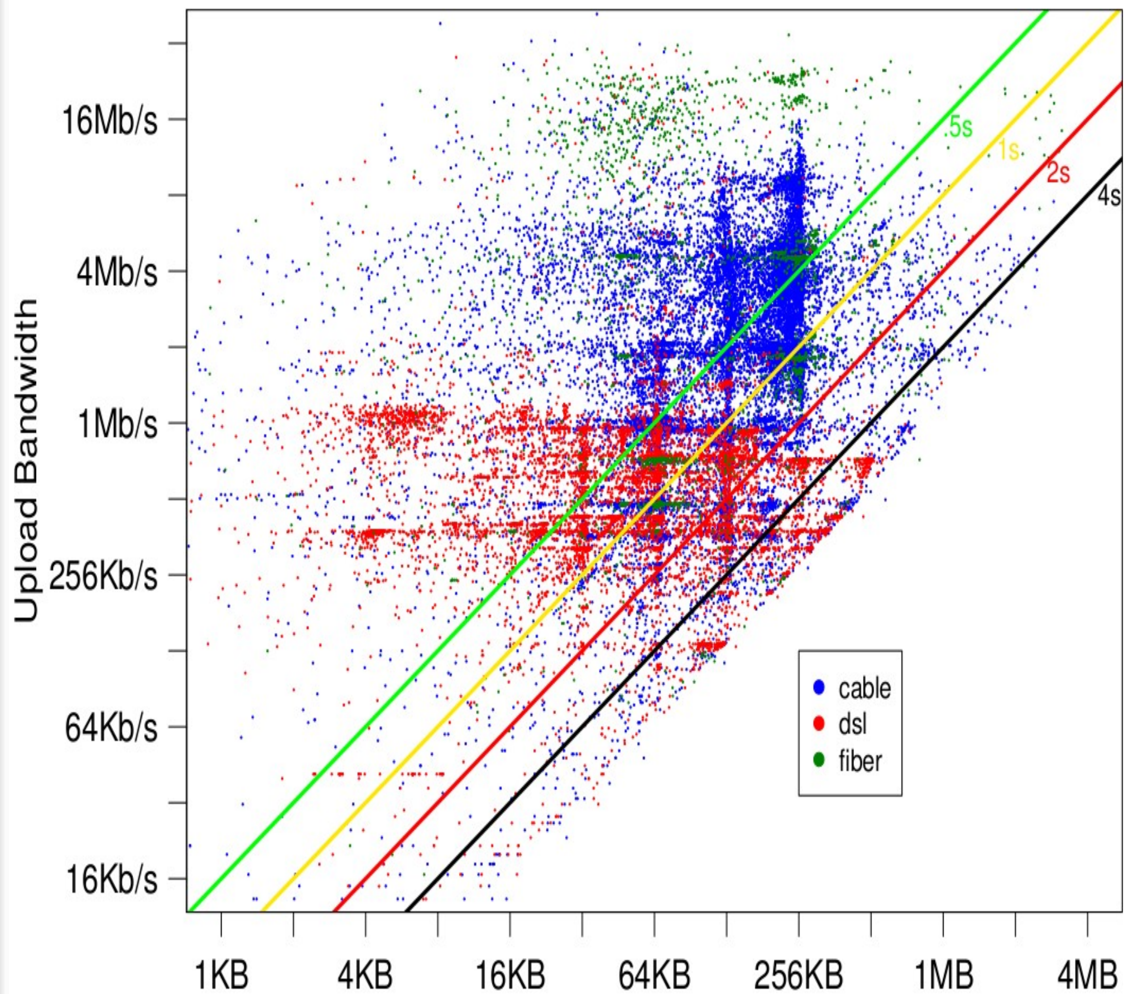
Hi! I'm Dave Täht

- I (along with hundreds of others) am trying to reduce latencies across the internet...
- Just finished up a stint as a guest researcher at the LINCs lab in Paris...
    - I co-run bufferbloat.net with Jim Gettys
    - Give talks, write code, do experiments
    - Design, Analyze networks, do embedded work
    - I work on the CeroWrt test router project
    - And I'm trying to beat the huge internet latencies induced by bufferbloat...
    - What's bufferbloat?

This is Bufferbloat!







Bufferbloat Canonical reference

- ACM Queue: “Bufferbloat”

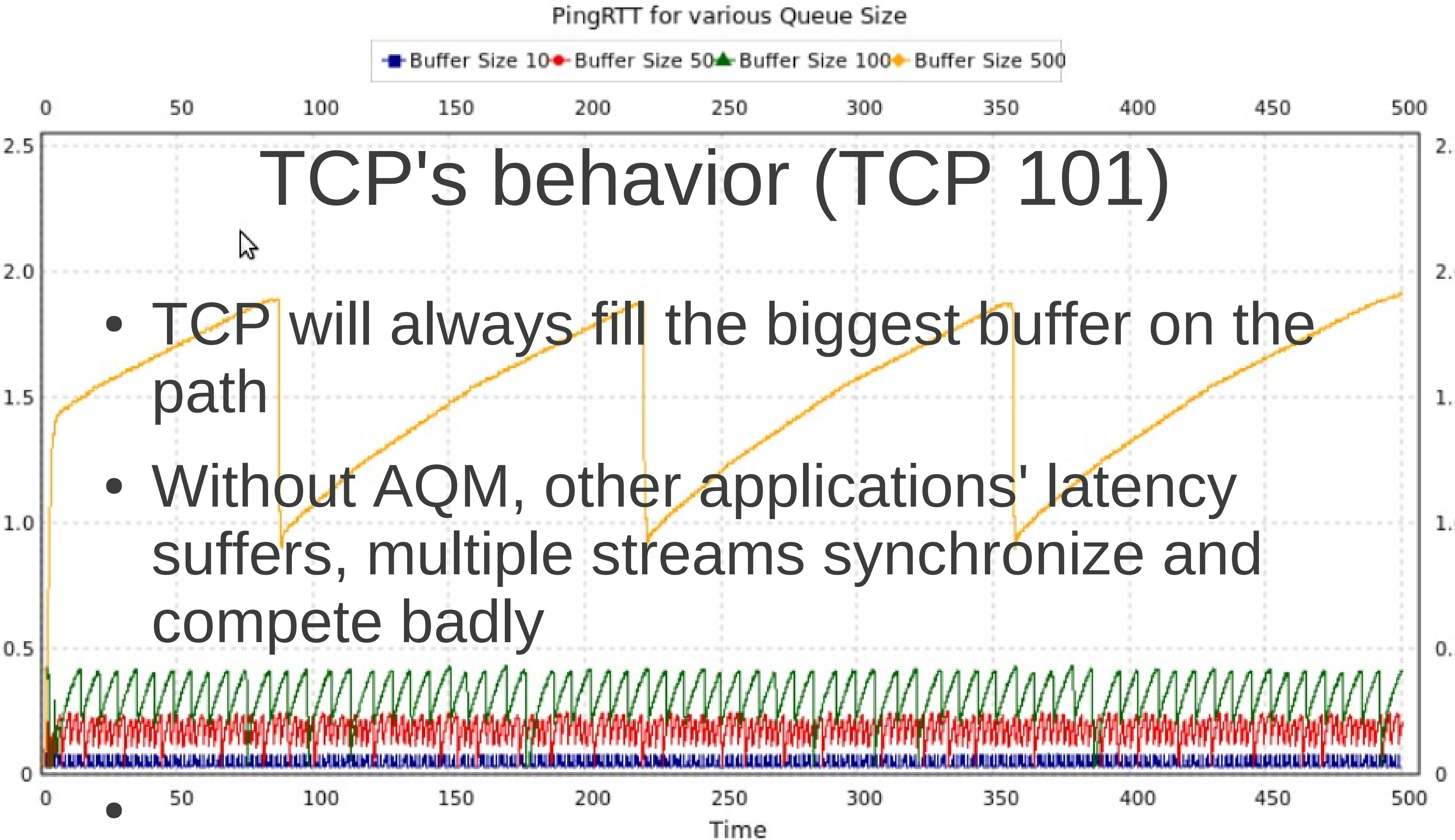
  – Jim Gettys, Kathie Nichols

  http://queue.acm.org/detail.cfm?id=2071893
- ACM Queue – “What's wrong with the Internet?” Vint Cerf, Van Jacobson, Jim Gettys, Nick Weaver

  http://queue.acm.org/detail.cfm?id=2076798
- Jim Gettys' Google Video
- Bufferbloat Project: http://bufferbloat.net

TCP's behavior (TCP 101)

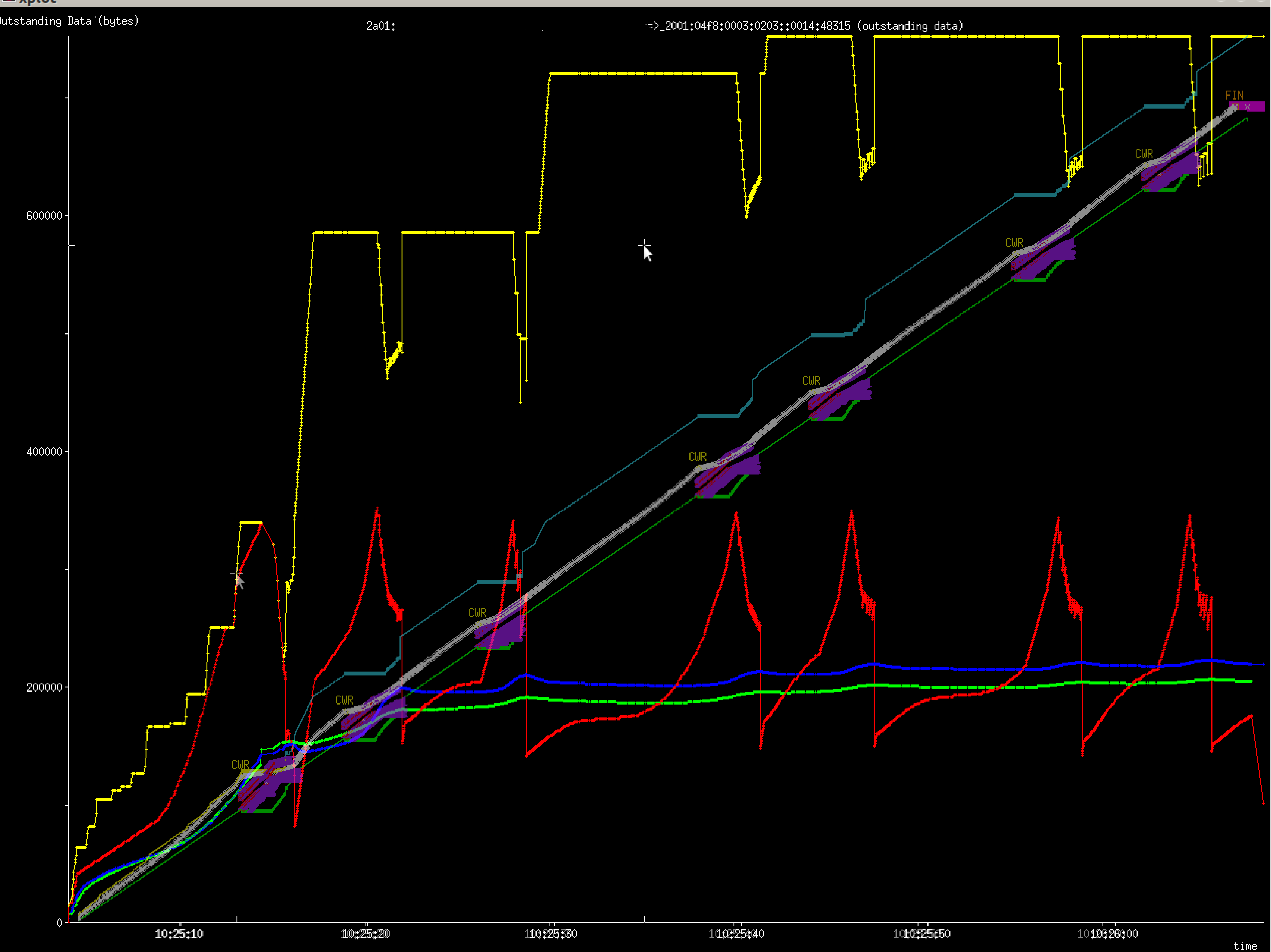
- TCP will always fill the biggest buffer on the path
- Without AQM, other applications' latency suffers, multiple streams synchronize and compete badly



-

- This is a ns2 model from:

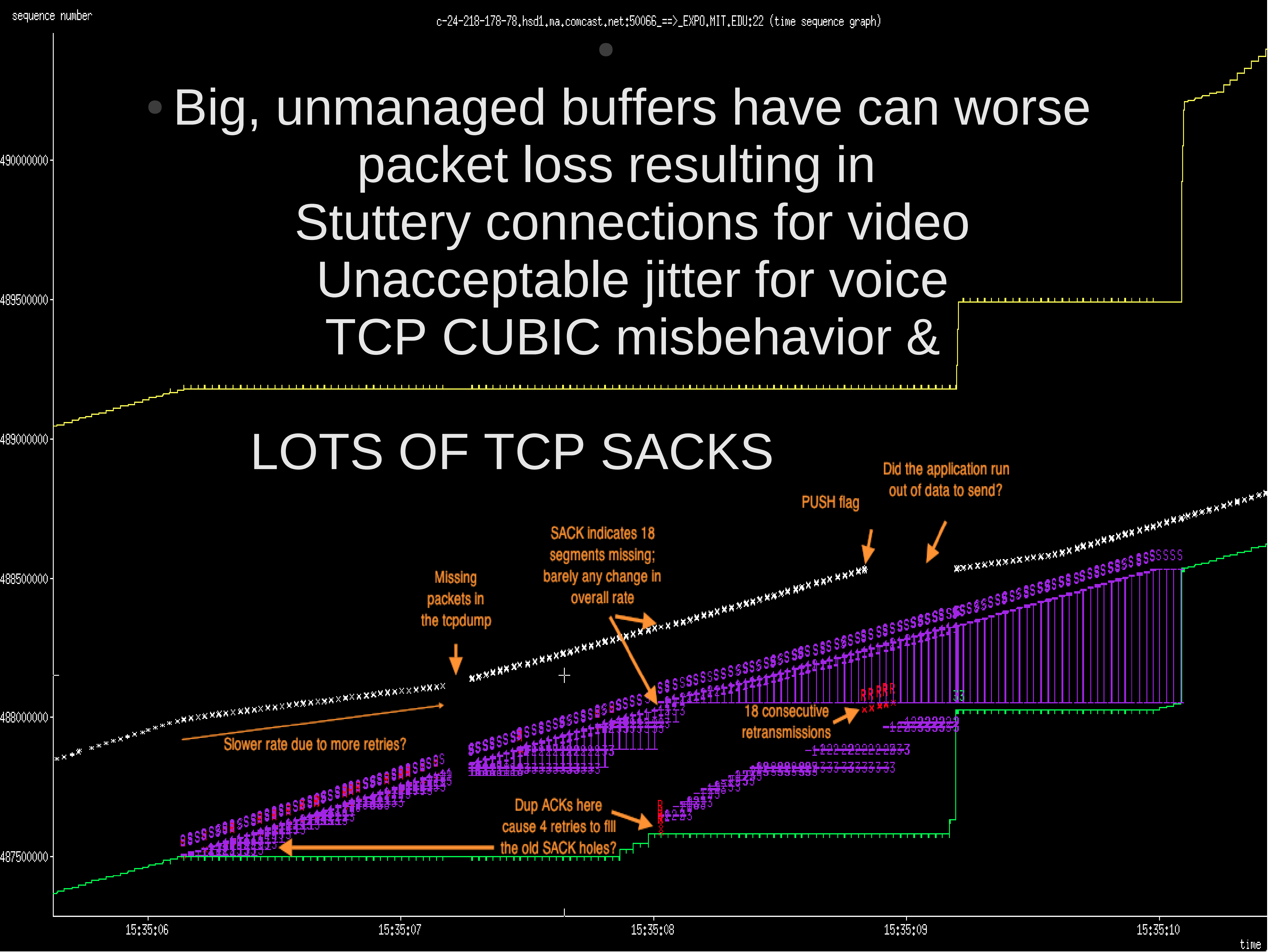
http://staff.science.uva.nl/~delaat/netbuf/bufferbloat_BG-DD.pdf



xplot
Outstanding Data (bytes)
2a01:
->_2001:04f8:0003:0203::0014:48315 (outstanding data)
600000
400000
200000
0
CWR
FIN
10:25:10
10:25:20
10:25:30
10:25:40
10:25:50
10:26:00
time

- Big, unmanaged buffers have can worse packet loss resulting in
  Stuttery connections for video
  Unacceptable jitter for voice
  TCP CUBIC misbehavior &

LOTS OF TCP SACKS



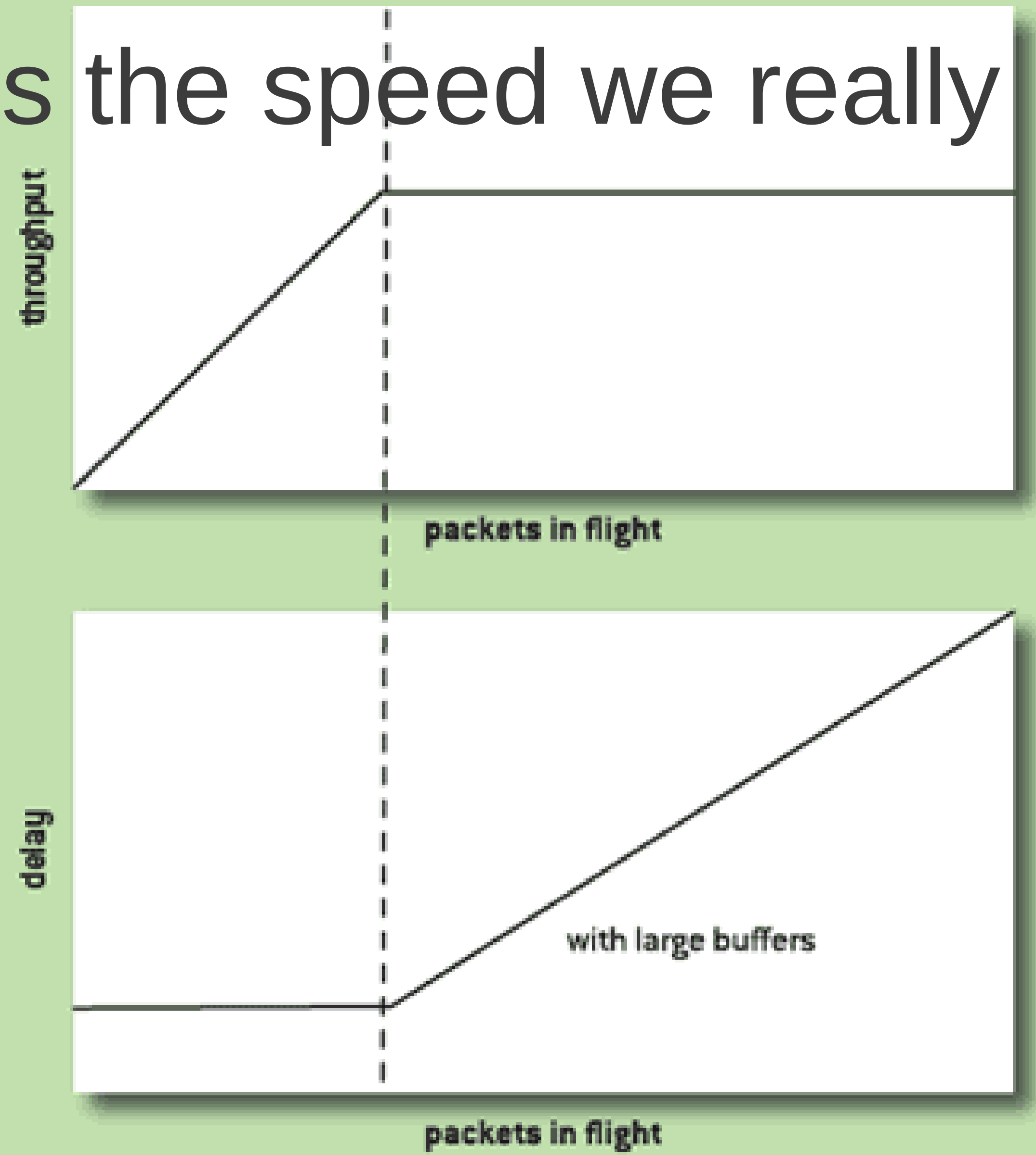
Packet Loss is the only effective TCP signaling mechanism we have

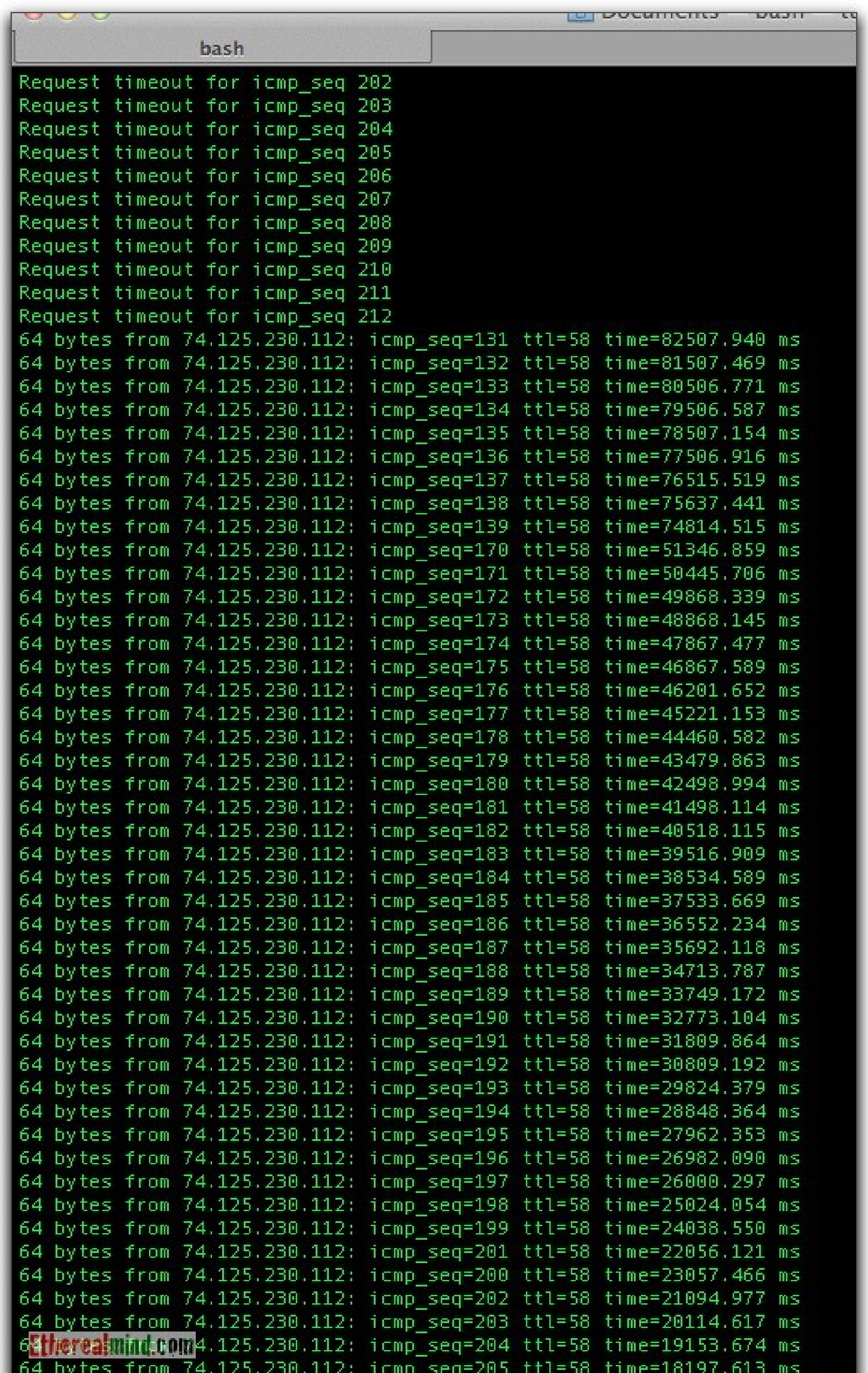
- Without losses, we get congestion collapse
  (See RFC970)
- Without enough & timely packet loss, we get vastly increased latencies and increasingly bad behavior between multiple flows.
- With the rise of multiple users in the home, and overly large buffers, effectively sharing connections grows more important
- Overly big buffers are everywhere in the path.

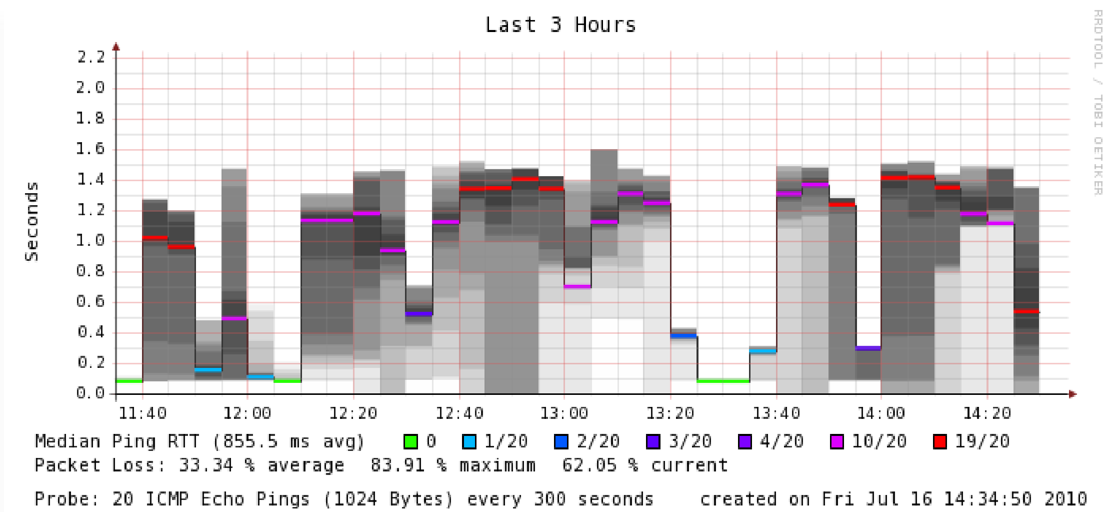
Low Latency

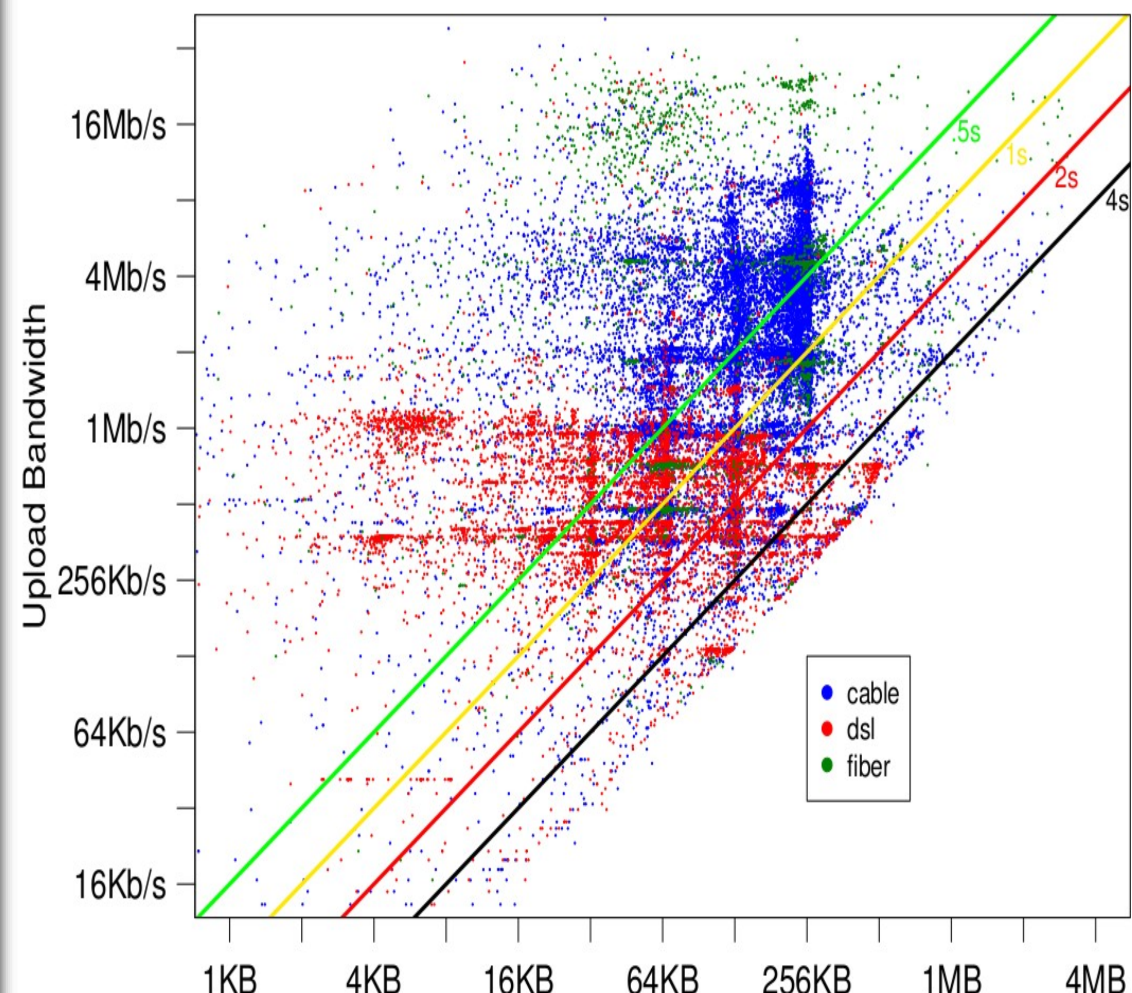
is the speed we really need

FIGURE 1 Throughput and Delay




bash
Request timeout for icmp_seq 202
Request timeout for icmp_seq 203
Request timeout for icmp_seq 204
Request timeout for icmp_seq 205
Request timeout for icmp_seq 206
Request timeout for icmp_seq 207
Request timeout for icmp_seq 208
Request timeout for icmp_seq 209
Request timeout for icmp_seq 210
Request timeout for icmp_seq 211
Request timeout for icmp_seq 212
64 bytes from 74.125.230.112: icmp_seq=131 ttl=58 time=82507.940 ms
64 bytes from 74.125.230.112: icmp_seq=132 ttl=58 time=81507.469 ms
64 bytes from 74.125.230.112: icmp_seq=133 ttl=58 time=80506.771 ms
64 bytes from 74.125.230.112: icmp_seq=134 ttl=58 time=79506.587 ms
64 bytes from 74.125.230.112: icmp_seq=135 ttl=58 time=78507.154 ms
64 bytes from 74.125.230.112: icmp_seq=136 ttl=58 time=77506.916 ms
64 bytes from 74.125.230.112: icmp_seq=137 ttl=58 time=76515.519 ms
64 bytes from 74.125.230.112: icmp_seq=138 ttl=58 time=75637.441 ms
64 bytes from 74.125.230.112: icmp_seq=139 ttl=58 time=74814.515 ms
64 bytes from 74.125.230.112: icmp_seq=170 ttl=58 time=51346.859 ms
64 bytes from 74.125.230.112: icmp_seq=171 ttl=58 time=50445.706 ms
64 bytes from 74.125.230.112: icmp_seq=172 ttl=58 time=49868.339 ms
64 bytes from 74.125.230.112: icmp_seq=173 ttl=58 time=48868.145 ms
64 bytes from 74.125.230.112: icmp_seq=174 ttl=58 time=47867.477 ms
64 bytes from 74.125.230.112: icmp_seq=175 ttl=58 time=46867.589 ms
64 bytes from 74.125.230.112: icmp_seq=176 ttl=58 time=46201.652 ms
64 bytes from 74.125.230.112: icmp_seq=177 ttl=58 time=45221.153 ms
64 bytes from 74.125.230.112: icmp_seq=178 ttl=58 time=44460.582 ms
64 bytes from 74.125.230.112: icmp_seq=179 ttl=58 time=43479.863 ms
64 bytes from 74.125.230.112: icmp_seq=180 ttl=58 time=42498.994 ms
64 bytes from 74.125.230.112: icmp_seq=181 ttl=58 time=41498.114 ms
64 bytes from 74.125.230.112: icmp_seq=182 ttl=58 time=40518.115 ms
64 bytes from 74.125.230.112: icmp_seq=183 ttl=58 time=39516.909 ms
64 bytes from 74.125.230.112: icmp_seq=184 ttl=58 time=38534.589 ms
64 bytes from 74.125.230.112: icmp_seq=185 ttl=58 time=37533.669 ms
64 bytes from 74.125.230.112: icmp_seq=186 ttl=58 time=36552.234 ms
64 bytes from 74.125.230.112: icmp_seq=187 ttl=58 time=35692.118 ms
64 bytes from 74.125.230.112: icmp_seq=188 ttl=58 time=34713.787 ms
64 bytes from 74.125.230.112: icmp_seq=189 ttl=58 time=33749.172 ms
64 bytes from 74.125.230.112: icmp_seq=190 ttl=58 time=32773.104 ms
64 bytes from 74.125.230.112: icmp_seq=191 ttl=58 time=31809.864 ms
64 bytes from 74.125.230.112: icmp_seq=192 ttl=58 time=30809.192 ms
64 bytes from 74.125.230.112: icmp_seq=193 ttl=58 time=29824.379 ms
64 bytes from 74.125.230.112: icmp_seq=194 ttl=58 time=28848.364 ms
64 bytes from 74.125.230.112: icmp_seq=195 ttl=58 time=27962.353 ms
64 bytes from 74.125.230.112: icmp_seq=196 ttl=58 time=26982.090 ms
64 bytes from 74.125.230.112: icmp_seq=197 ttl=58 time=26000.297 ms
64 bytes from 74.125.230.112: icmp_seq=198 ttl=58 time=25024.054 ms
64 bytes from 74.125.230.112: icmp_seq=199 ttl=58 time=24038.550 ms
64 bytes from 74.125.230.112: icmp_seq=201 ttl=58 time=22056.121 ms
64 bytes from 74.125.230.112: icmp_seq=200 ttl=58 time=23057.466 ms
64 bytes from 74.125.230.112: icmp_seq=202 ttl=58 time=21094.977 ms
64 bytes from 74.125.230.112: icmp_seq=203 ttl=58 time=20114.617 ms
4.125.230.112: icmp_seq=204 ttl=58 time=19153.674 ms
64 bytes from 74.125.230.112: icmp_seq=205 ttl=58 time=18197.613 ms
Etherealmind.com


Last 3 Hours
Seconds
2.2
2.0
1.8
1.6
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
11:40 12:00 12:20 12:40 13:00 13:20 13:40 14:00 14:20
Median Ping RTT (855.5 ms avg) 0 1/20 2/20 3/20 4/20 10/20 19/20
Packet Loss: 33.34 % average 83.91 % maximum 62.05 % current
Probe: 20 ICMP Echo Pings (1024 Bytes) every 300 seconds created on Fri Jul 16 14:34:50 2010
RRDTOOL / TOBI OETIKER


Upload Bandwidth
16Mb/s
4Mb/s
1Mb/s
256Kb/s
64Kb/s
16Kb/s
1KB
4KB
16KB
64KB
256KB
1MB
4MB
.5s
1s
2s
4s
cable
dsl
fiber

But... where did all that latency come from?

Why was the modern 'high speed' Internet so 'slow'?

- Bufferbloat project: started January, 2011
- Mission:
  - Gather together experts to tackle networking queue management and system problem(s), particularly those that effect wireless networks, home gateways, and edge routers
  - Spread the word to correct basic assumptions regarding goodput and good buffering on the laptop, home gateway, core routers and servers.
  - Produce tools to demonstrate and diagnose the problem
  - Do experiments in advanced congestion management

    Produce patches to popular operating systems at the device driver, queuing, and TCP/ip layers to fix the problems.

The CeroWrt Project

Cerowrt sub-project, started March, 2011

- Why were newer home routers performing so badly?
- Why wasn't AQM deployed?
- What was holding up IPv6?
- What was wrong with wireless-n?

- OpenWrt derived – not a fork!
- 680Mhz CPU, simple hardware, (no network offloads), no binary blobs, ag71xx network driver....

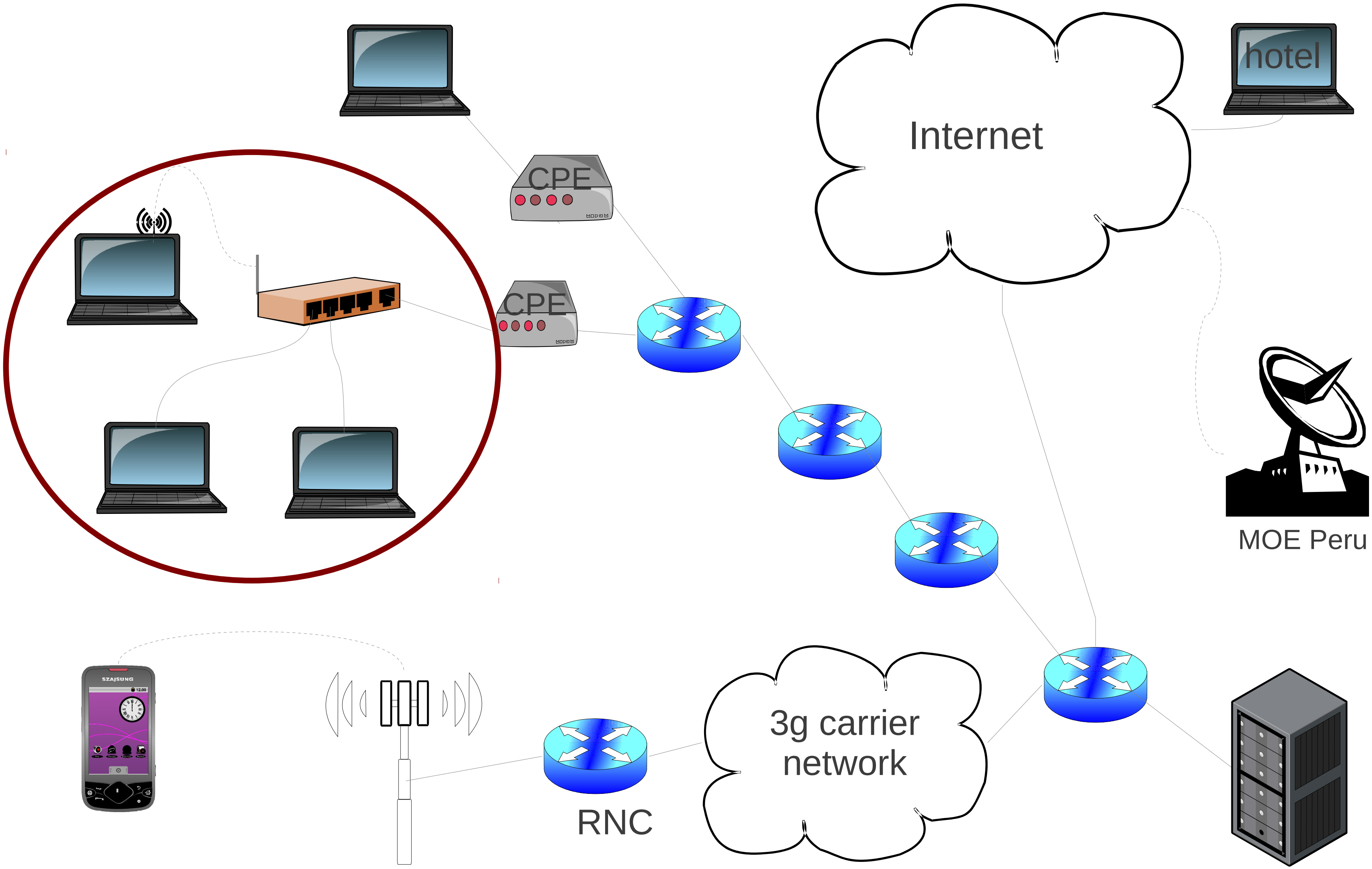
The CeroWrt Concept

- Thoroughly analyze **one** fully open source routing platform
- Create a testbed (bloatlab #1 at isc.org)
- Do research into the big networking problems...
- Rapidly feed results and code back into OpenWrt and into mainline Linux and (thus) out across the industry
- Wash, rinse, repeat

While we were at it... what else could modern home gateways do better?

- AQM, Traffic shaping, Fair Queuing
- IPv6
- Reliability, Security
- Split DNS and DNSSEC
- Mesh Routing
- File Storage
- Unattended setup/configuration/operation
- But first...

Cerowrt tackles bufferbloat in CPE



CeroWrt's progress so far

ECN fixes

IPv6 routing fixes

Wireless infinite retry fix...

30% speedup on ethernet

Wireless IPv6 DSCP classification...

… by august, 2011 we'd got to where two CeroWrt routers behaved, almost, sort of, like what ns2 models would predict, but not... quite... RED was misbehaving... servers and hosts configured nearly identically still exhibited huge latencies...

Then we put on a BOF...

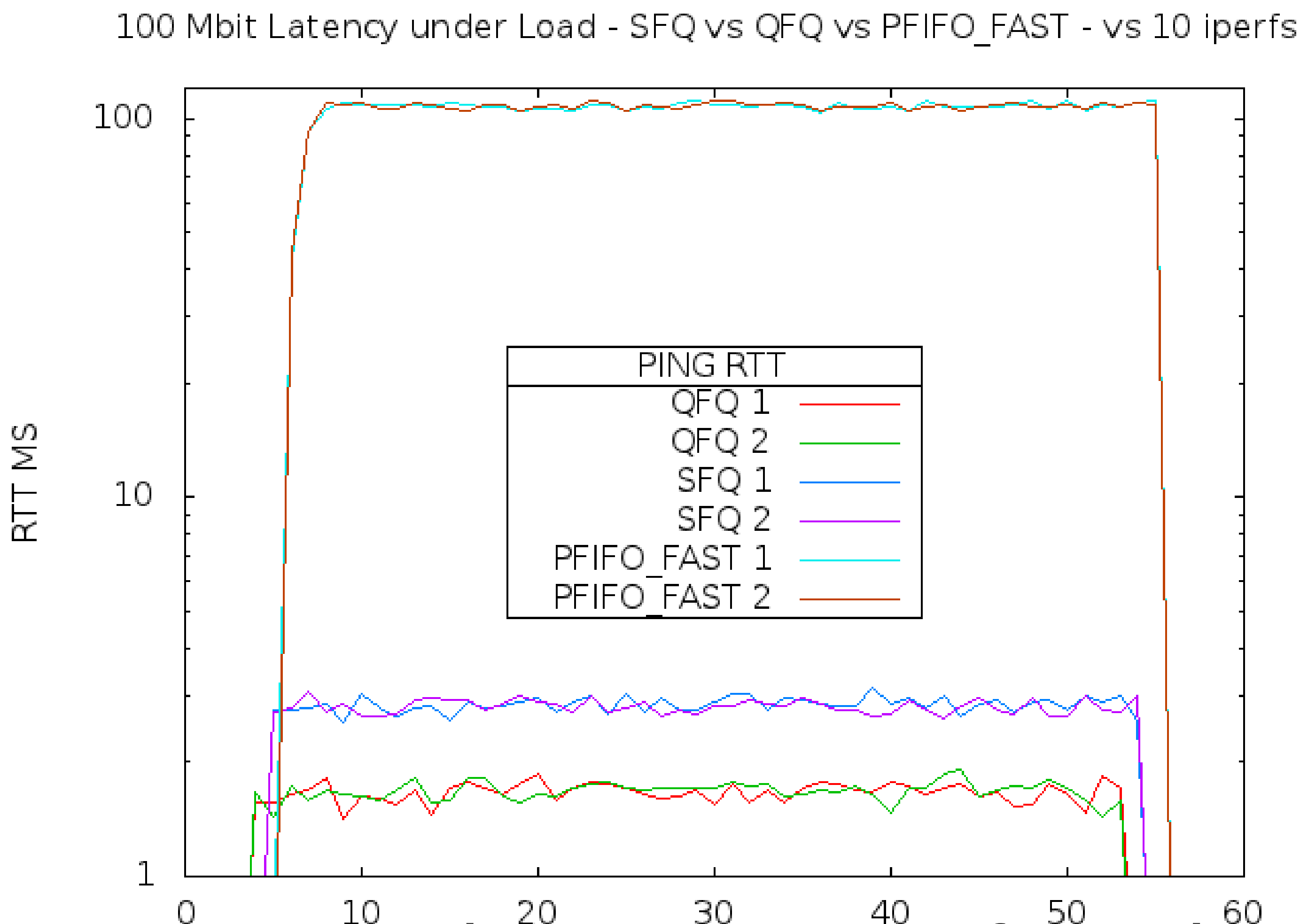
Where does the base latency in hosts/servers come from?

- Enormous device driver TX rings!
  - Typically set 256 to 4096 entries
  - Up to 100s of megabytes in the driver with TSO enabled
  - OK, maybe that's useful for 10GigE maybe, but does not scale down to 10 or 100Mbit, or even GigE when cpus can saturate the wire... which in the multicore age, they can easily do – except that they first saturate the wire, then fill all the buffers.
  - TSO/GSO is bad for the net at sub-gige speeds.
- Excessive (unmanaged) buffering in the OS above the tx-ring
- No Active Queue Management on the network – not just routers, but servers and clients.

Byte Queue Limits in Linux 3.3

- Developed by Tom Herbert at Google
- Establishes a byte limit for queue depth
  - Manages the tx ring
  - 5+ orders of magnitude of network buffering eliminated
  - Now part of Linux 3.3
- Support for many of the common ethernet cards
- Works well at 1-10GigE
- Below GigE, can be tuned further with one configuration change (2-3 big packets is good)

Latency under load (Linux 3.2 + BQL + SFQ or QFQ



(in 3.3 SFQ and QFQ now perform the same (e.g – REALLY well, with 1ms latency under load)

Improvements to Linux 3.3 AQMs

(Eric Dumazet has been very busy...)

- RED implementation bug introduced in 2009 fixed...
- Sally Floyd's Adaptive RED implemented
- QFQ (quick fair queuing) debugged
- SFQ (stochastic fair queuing) vastly improved
  - Hash permutation no longer affects streams
  - Head of Line problem fixed
  - Queues and flows configurable
- QFQ and SFQ Queues & Depths are now manageable to enormous numbers of flows.... and there's now... REDSFQ...

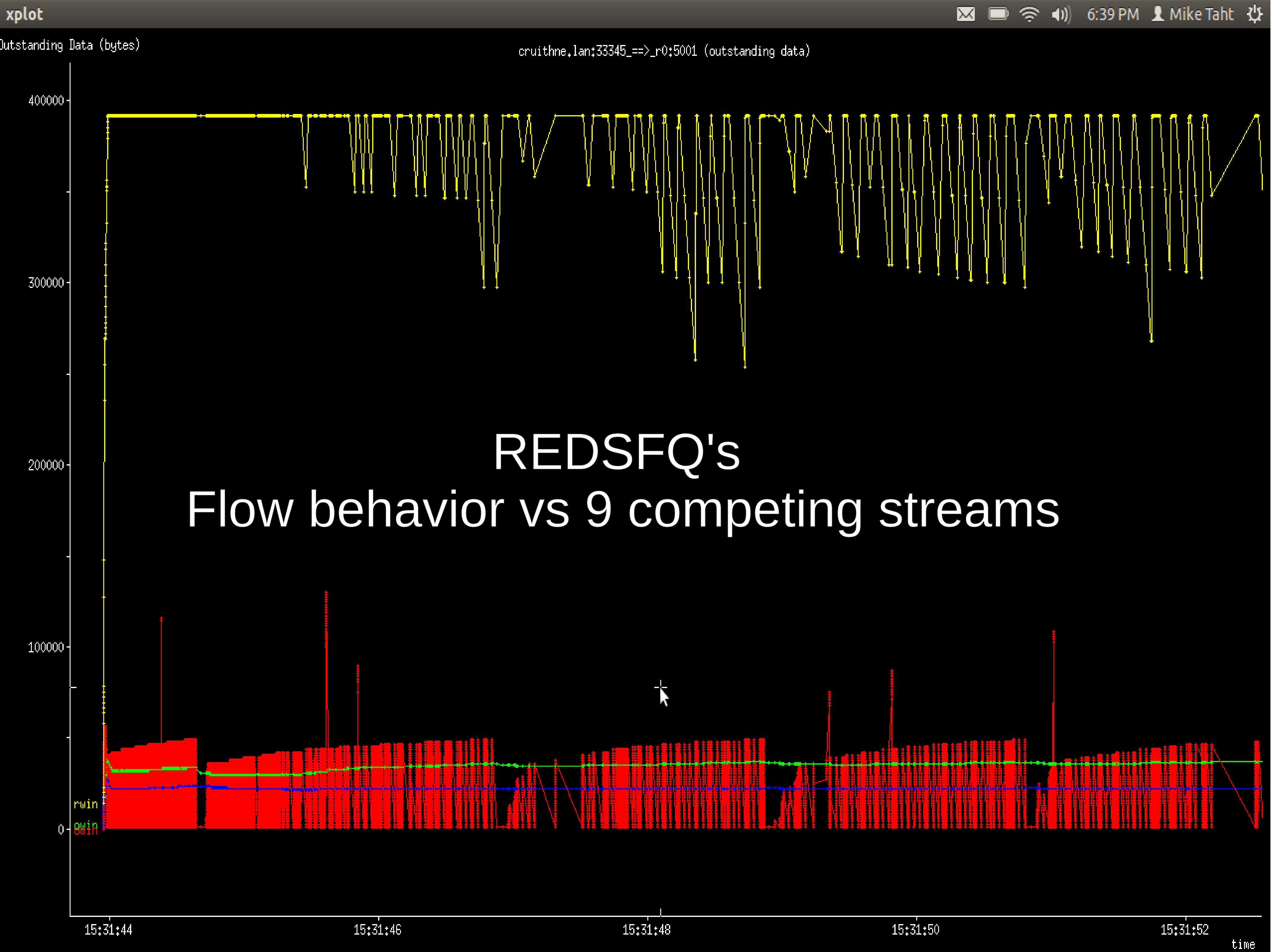
REDSFQ

- Hybrid AQM of the new SFQ and RED implementations that takes the best (we think) from both concepts and puts them into one easy to use AQM.
  - Fair Queuing via SFQ, Queues independent...
  - RED for the individual queue management
  - TSO/GSO packet scheduling WORKS for hosts!
  - HEAD DROP support
  - ECN support
  - 1 line configuration
  -

REDSFQ Configuration

“Our biggest mistake, was in making queue management optional... and then, making it scary.” – Dennis Gentry

- For hosts, servers, routers running at line rate, REDSFQ is one line of configuration...
- With bandwidth shaping on CPE... 3 lines of configuration...
- More testing and analysis is required to create a good, universal default.
- All the code is now in Linux 3.3!


xplot
6:39 PM
Mike Taht
Outstanding Data (bytes)
cruithne.lan:33345_==>_r0:5001 (outstanding data)
REDSFQ's
Flow behavior vs 9 competing streams
400000
300000
200000
100000
0
rwin
awin
owin
15:31:44
15:31:46
15:31:48
15:31:50
15:31:52
time

The new AQMs are cool, but...

- Far more testing, **at scale**, is needed
- LEDBAT is a problem, wireless still has problems
- Some network attacks are maybe easier
- Best effects require some tuning below 100Mbit
- Other vendors and Oses still have problems
- Deployment will take years
- But on the whole, things are looking pretty good
- And new AQMs are being developed, by Van Jacobson, Kathie Nichols and others, which thanks to BQL, can actually work just like the ns2 models...

Getting back to Cerowrt... All this... is fully usable on home routers and CPE!

- Extra CPU load is almost immeasurable compared to wireless, crypto, web server, etc
- Memory use is minimal – and as you have smaller buffer sizes in the first place, it's a net win on memory, actually.
- Most of the new stuff has been tested on CeroWrt (the rest requires waiting for the 3.3 mainline to settle down)
- Much better network behavior is very possible in the home using these techniques

Cerowrt's Future Directions

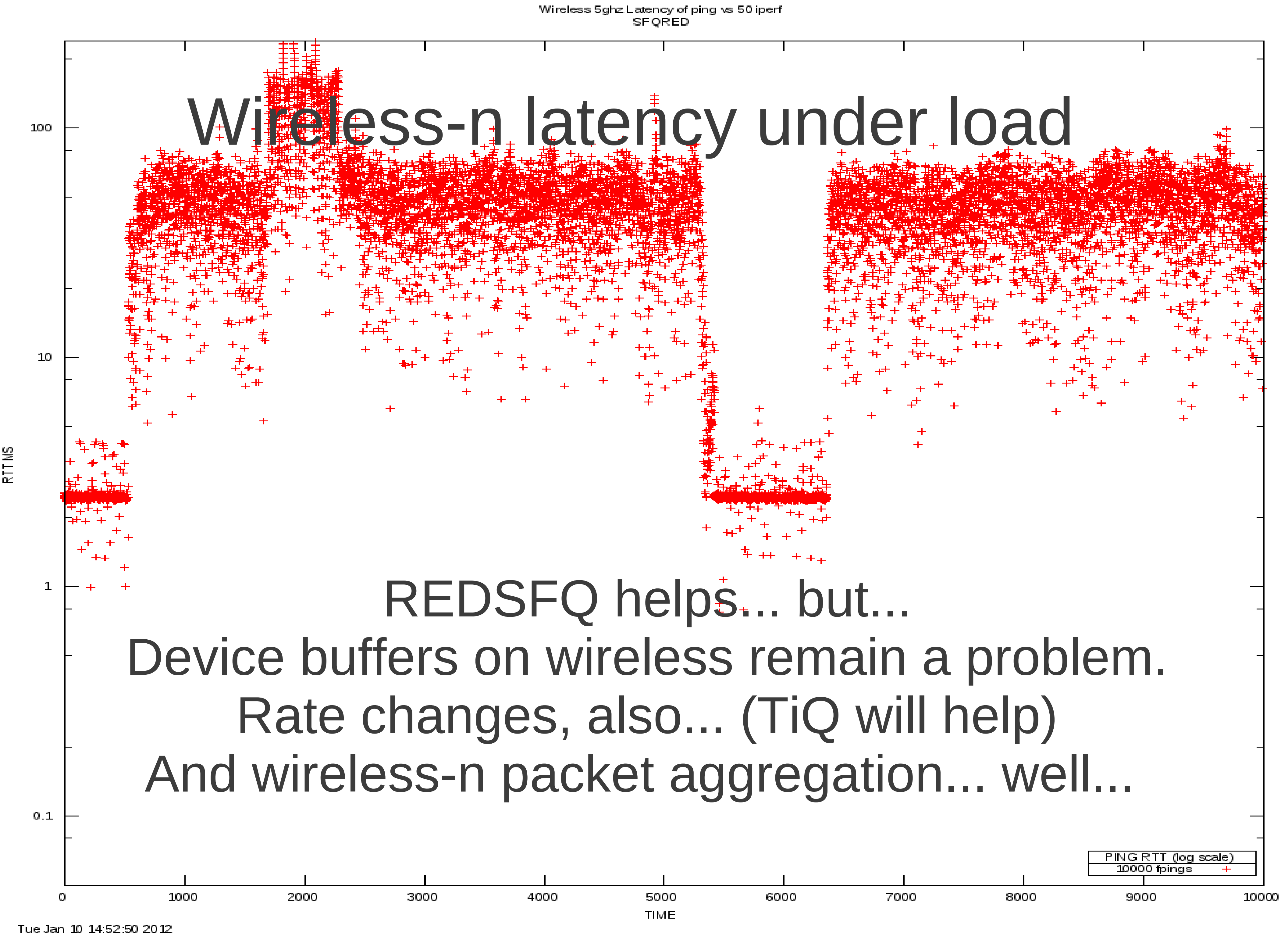
- Presently on hold waiting for linux 3.3 to stabilize
- Testing these new technologies at scale on x86/hgw
- Shaper Probing (automagic CPE AQM configuration)
- New AQMS (revised RED replacement from VJ, Kathie, packet schedulers for wireless, etc)
- IPv6 – note, we have treated ipv6 as first class for the AQM development...
- Split DNS and DNSSEC
- Mesh (babel) and other routing protocols (ospf, olsr)
- http://www.bufferbloat.net/projects/cerowrt/roadmap
- Wireless-n is the biggest bufferbloat-related problem...

Wireless Bufferbloat is an unsolved problem!!!!

- The distance around the earth, on wires, is around 330 ms
- The “distance” between your laptop and AP can exceed 8000 ms and bandwidth 20Mbit/ tops
- Telephony standard is < 250ms!!!
- A Classic BDP calculation does not work at all
- No AQM we know of can reduce the queue-sizes as the bandwidth and delay vary considerably, even under good conditions...

There rarely are good conditions!

Wireless-n latency under load



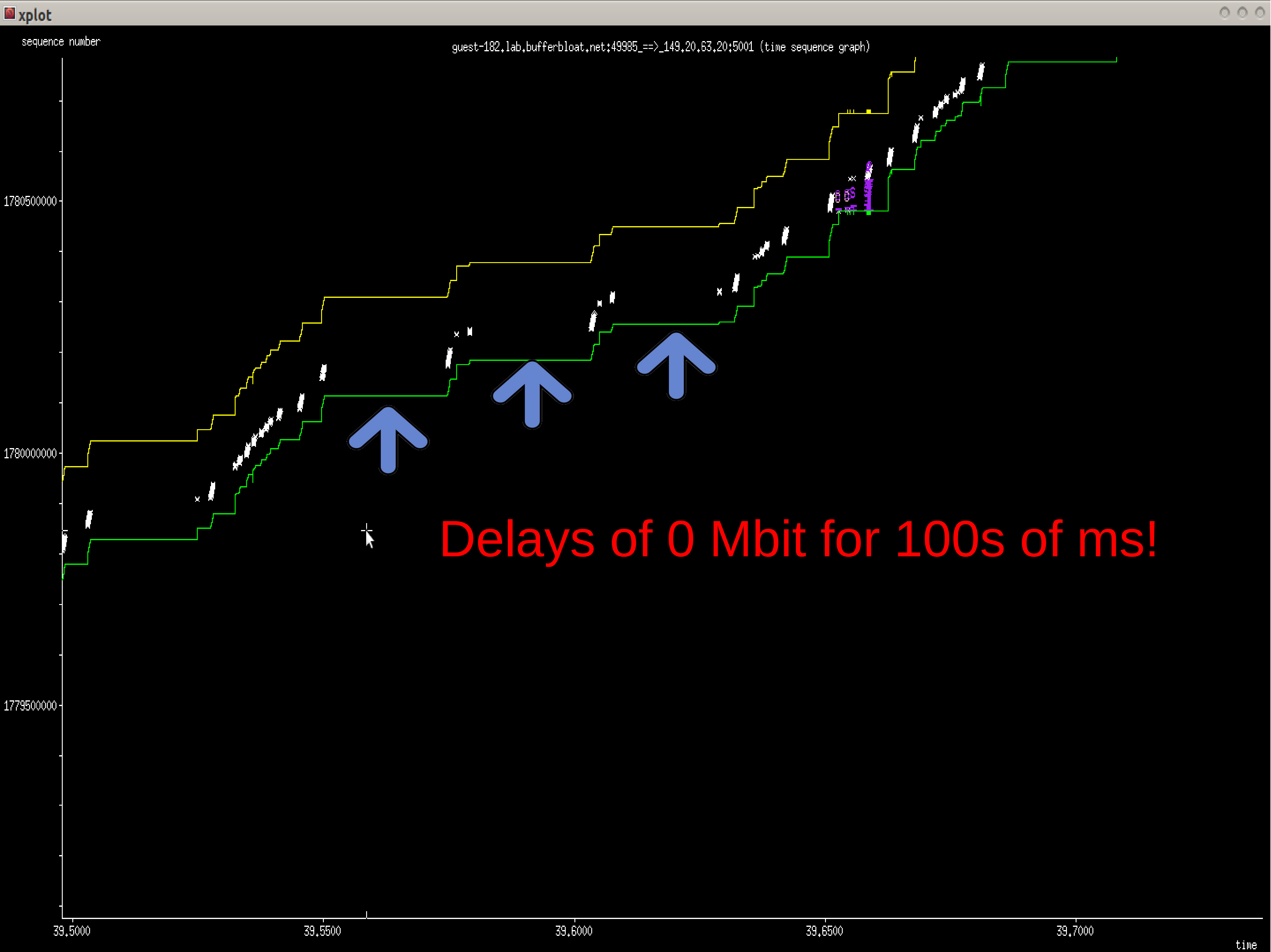
REDSFQ helps... but...

Device buffers on wireless remain a problem.

Rate changes, also... (TiQ will help)

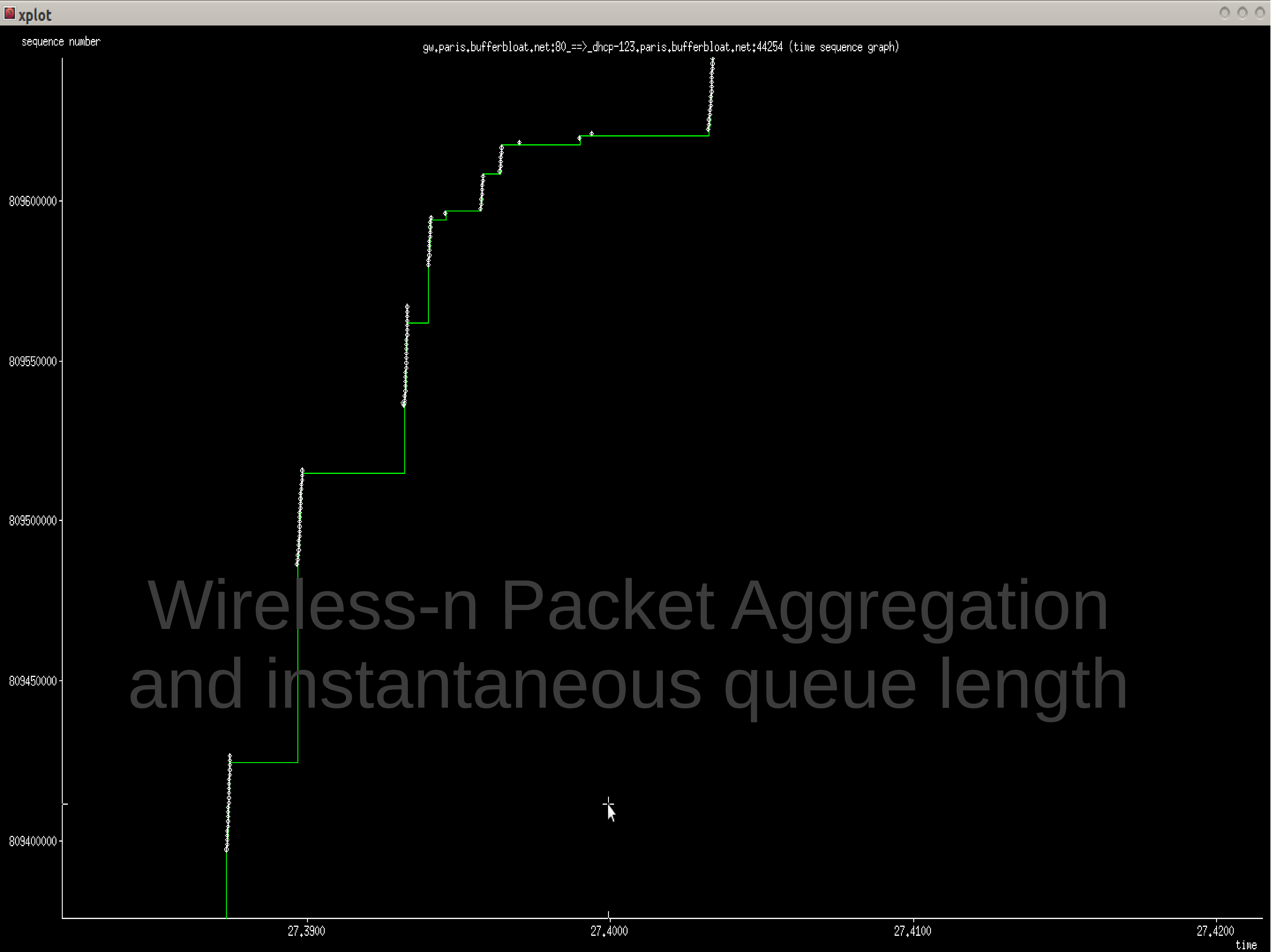
And wireless-n packet aggregation... well...


xplot
sequence number
guest-182.lab.bufferbloat.net:49985_==>_149.20.63.20:5001 (time sequence graph)
1780500000
1780000000
1779500000
39.5000
39.5500
39.6000
39.6500
39.7000
time
Bandwidth is Not a Constant
It might be 100 Mbit
… for several ms

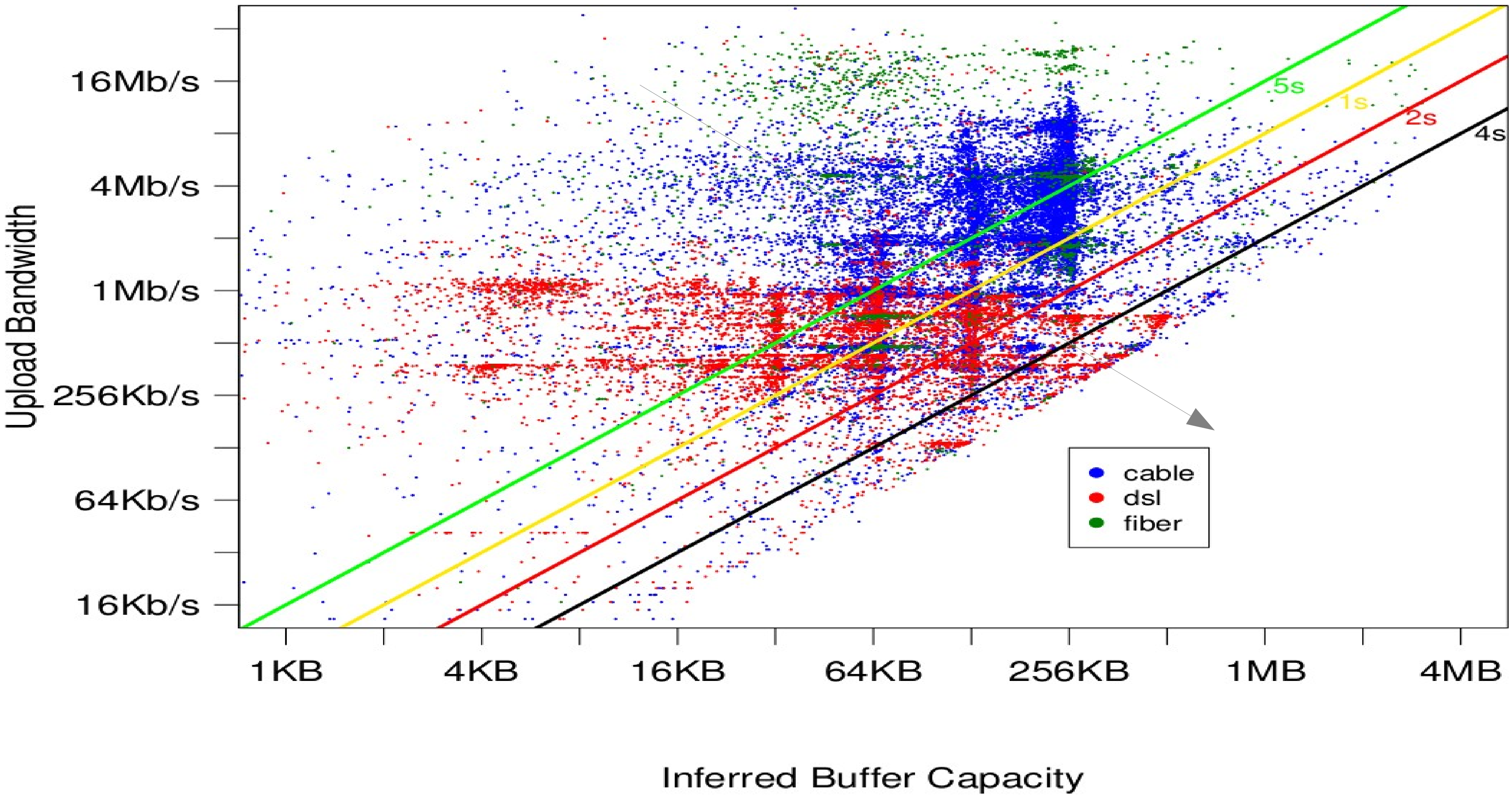

xplot
sequence number
guest-182.lab.bufferbloat.net:49985_==>_149.20.63.20:5001 (time sequence graph)
1780500000
1780000000
1779500000
39.5000
39.5500
39.6000
39.6500
39.7000
time
Delays of 0 Mbit for 100s of ms!


xplot
sequence number
guest-182.lab.bufferbloat.net:49985_==>_149.20.63.20:5001 (time sequence graph)
1780500000
1780000000
1779500000
39.5000
39.5500
39.6000
39.6500
39.7000
time
Or over gbit for an instant
– with wireless-N
aggregation, wedging as
many packets into the
backlog as possible...

Wireless-n Packet Aggregation and instantaneous queue length



In 2012-2015... Will this get better, or worse?



Questions?

Bufferbloat.net Resources

Bufferbloat.net: http://bufferbloat.net
Email Lists: http://lists.bufferbloat.net
IRC Channel: #bufferbloat on chat.freenode.net
CeroWrt: http://www.bufferbloat.net/projects/cerowrt
Other talks: http://mirrors.bufferbloat.net/Talks
Jim Gettys Blog – http://gettys.wordpress.com
Use Google scholar for bufferbloat.

Help out where you can! Resources, rackspace, developers, money, theorists, & awareness, are all needed... as...

We're all in this bloat together.