Fixing Wifi Latency… Finally!

**THIS!**
(Sub 40-msec latency
2 station test, lowest mcs(0)
wifi rate (1mbit))

**NOT THIS!**
(1 to 2 second latency)

Dave Taht
director, make-wifi-fast project
dave at taht.net
Overview

Grokking Bufferbloat
What is wrong with Wi-Fi
Fixes and new software stack
Problems & Futures
What is Bufferbloat?

- Undesirable latency and jitter that comes from excessive buffering. See wikipedia, etc
- Quick test is to use DSLReports.com/speedtest
- Best test is Flent (www.flent.org)
Linux Bufferbloat fixes: 2011-2016

- Linux 3.3: Byte Queue Limits
- Linux 3.4 RED bug fixes & IW10 added & SFQRED
- Linux 3.5 Fair/Flow Queuing packet scheduling (fq_codel, codel)
- Linux 3.7 TCP small queues (TSQ)
- Linux 3.12 TSO/GSO improvements
- Linux 3.13 Host FQ + Pacing (sch_fq)
- Linux 3.15 Change to microseconds from milliseconds throughout networking kernel
- Linux 3.17 Network Batching API
- The Linux stack is now mostly “pull through”, where it used to be “push”, and looks nothing like it did 6 years ago.
- At least a dozen other improvements I forget
- Linux 4.8 – TCP BBR
  ... (and BSD just got fq_codel!)
Basically – everything – except WiFi (and lte) can be debloated now.
- And we just made a big dent in WiFi
Grokking Bufferbloat fixes in 20 Minutes

Or 20 hours... (or 6 years)
Quick n’ Dirty Page Load Time (PLT)
Fully-loaded network
Linux 4.4 FIFO txqueue 1000
(10ms base RTT + 1 sec delay)

# flent -l 300 -H server --streams=12 tcp_n dow n &

How long will it take to download slashdot's main page?
10 seconds? 20? 100?
(It takes 8 seconds w/no delay)
Slashdot takes 4 minutes with 1 second delay

FINISHED --2016-10-22 13:43:35--
Total wall clock time: 232.8s
Downloaded: 62 files, 1.8M in 0.9s (77KB/s)

Connect time accounted for 99.6% of the runtime, and the real throughput was: 77KB/sec.

Bufferbloat latency matters.
Bufferbloat still with us at high rates

Fix ISP bufferbloat FIRST!

Courtesy dslreports:
http://www.dslreports.com/speedtest/5408767
http://www.dslreports.com/speedtest/5448121
Wi-Fi Fallacies

- “Everybody knows” Wi-Fi performance is bad...
- But... a lot of the bad performance that has been written off as “wifi interference” and “wifi is just like that”, was actually queuing delay - bufferbloat.
  - At the ISP link
  - In the wifi environment itself
- We demonstrate that this can be fixed largely through software – not “more stuff”.
Eliminating bufferbloat from WiFi

- Doing Better Benchmarks
- Designing the stack better for user experience
- Eliminating queueing delay everywhere.

- Keep in your head: 1 single 1514 byte packet at 1MBIT (mcs0)... = 13ms to transmit
We Developed Better Benchmarks

- DSLReports test for bufferbloat
- Flent has extensive plotting facilities & wraps netperf/ping/d-itg for
  - rtt_fair*: Lets you test multiple numbers of stations
  - rrul, rrul_be: Exercises up/down and hw queues simultaneous
  - tcp_nup, tcp_ndown: Test variable numbers of flows
  - http
  - rrul_voip
  - About 90 other tests
- TSDE
- Tcptrace -G & Xplot.org
- Wireshark
Flent – our most useful tool

- Robust, repeatable, accurate measurements
- Visually represents detail lost in single number benchmarks
- Shows RRUL (Realtime Response Under Load)
- Shows CDF (Cumulative Distribution Function)
- Shows box plots, for easy comparison of multiple test runs
- Open Source! https://flent.org
What Tools and Practice Helped?

- Repeatable tests with plots (not just single numbers)
  - Multiple stations doing different things
  - Long duration tests
  - Varying RTTs
  - Packet Captures
  - Tracing
  - Sampling

- Field Testing

- NOT. JUST. BANDWIDTH.
Other Key Benchmarks

- Looked at other benchmarks - at Quality of Experience (QOE)
  - VoIP Mean Opinion Scores (MOS)
  - Web Page Load Time (PLT)
  - Latency under load (vs various mcs wifi rates)
  - Long Term stability and “smoothness”
  - Flow completion time (FCT)
  - Loss, jitter and queuing

- Many benchmarks are basically broken because they don't test under load
- Flent compares well to other tests with problems:
  - Iperf3 “millabursts”
  - Tmix
  - PPS
Improve design parameters of stack

• Optimize for “time to service” wifi stations
  – Pack Aggregates Better
  – Serve Stations better

• Control Queue Length with codel

• Be fair to flows

• Be fair to stations

• Optimize for TXOPs
  – ~1300/sec
  – Into which you pour bandwidth sanely
Typical Wifi Testbench

Over the air

Over a wire

Photos courtesy: Candelatech
Real World WiFi uses
Problems with WiFi Lab Bench Testing

- Distance
- Multipath
- (many) Legacy Devices
- (many) Competing devices
- Reasoning from non Internet scale RTTs
- Treating UDP and TCP results with equal weight
- Rate over range without looking at latency or rate control
- Tests that don’t run long enough
- Summary statistics that don’t look at inter-station behaviors
- Not measuring latency under load
10s of seconds of firmware buffering

Seconds of driver buffering

10s of seconds of wifi buffering

Current Wi-Fi Stack
100 stations ath10k
Feb, 2016

RTT Fair - variable number of hosts (upload only)
Download, upload, ping (scaled versions)
nopatch
What we did to fix WiFi Latency

- No changes to WiFi!
  - Backoff function stays the same
  - Only modified the queuing and aggregation functions

- We changed
  - Created “Mac80211 intermediate queues”
  - Added per station queueing
  - Generalized the fq_codel implementation (fq_impl.h)
  - Removed the qdisc layer entirely
  - fq_codel'd per station
  - Put in RR Fair Queuing between stations (currently)
  - DRR Airtime Fairness between stations (pending)
Overall Philosophy

- One aggregate TXOP in the hardware
- One aggregate queued on top, ready to go
- One being prepared
- The rest of the packets being fq’d per station
- Total delay = 2-12ms in the layers
  - This is plenty of time (BQL not needed)
  - Currently max sized txops (we can cut this)
- Codel moderates the whole thing
Intermediate Queues Benefits

- Per device max queuing, not per SSID
- Minimal buffering at driver – 2 TXOPs max!
- fq_codel per station
  - Mixes flows up with fair queueing
  - Controls queue length for big flows
  - Has a maximum amount of bytes AND packets
  - Also enables lossless congestion control (ECN)
- RR switching between stations in AP or meshy modes
Using 802.11 Intermediate Queues

- Add a callback for ops->wake_tx_queue to activate.
- Packets then are no longer pushed down by the mac80211 layer (i.e. mac80211 will no longer call drv_tx() for data packets).
- Instead, packets get queued to the intermediate queues, and mac80211 will call drv_wake_tx_queue() to notify the driver of which TXQ has new packets pending. It is then the responsibility of the driver to pull the packets it needs.
- drv_tx is still called for non-data packets.
WiFi Queue Rework

- Qdisc disabled
- Buffering moves into the MAC80211 “intermediate queue” layer, managed by fq_codel
- Keep max of 2 aggregates pending (1.2-10ms) in the drivers
- When one is completed, another is formed.
- That's it.
Our Results – How did we do?

- Decreased Wi-Fi latency to < 40 msec (from peaks of 1-2 seconds) across all mcs rates + ~4ms per active station
- Added “air time fairness” so that slow stations don't hog all the airtime
- Right Sized the buffers for all rates – even as they change!
- Vastly improved ability to handle > 1 stations at full data rate, with fuller and fair sharing of bandwidth
- Showed that fq+AQMs such as fq_codel eliminate the need for most QoS settings.
Decreasing Latency on Wi-Fi

1+sec latency
Linux 4.4
@mcs0

Sub 40-ms latency
Linux 4.9?
@mcs0
100 stations ath10k
Feb, 2016
100 Stations transmitting full rate

- Test was designed to have 100 stations transmitting at full rate simultaneously.
- First chart shows linux stock 4.4 Wi-Fi stack. Only five stations were able to start up at the beginning, the remaining 95 are blocked by too much buffering causing timeouts for TCP. When those five complete, the next set begins. All the while, over 15+ sec latency. BAD.
- Second chart shows the same test case, with the ath10k POC Wi-Fi stack. All 100 stations start immediately, using full bandwidth, equal sharing, 150-300 msec latency. GOOD.
100 stations
Ath10k Airtime Fair (wip)
Low latency/Good bandwidth
At all mcs rates (ath9k HT20)
VOIP MOS scores & Bandwidth
3 stations contending (ath9k)

Best Effort – scheduled sanely on the AP –
Better than the HW VO queue!
Aggregation improvements (Ath9k)
Web PLT improvements under competing load (ath9k)

FIFO load time was 35 sec on the large page!
Nearly flat latency across all WiFi mcs rates
Throughput Improved
2 Fast Stations, 1 Slow

Figure 7: Throughput for TCP download traffic (to clients).
Open Problems

• What stats/knobs to export?
• TSQ interaction issue
• Adding more drivers/devices
  – Some devices don’t tell you how much they are aggregating
  – Some don’t have a tight callback loop
  – Some expose insufficient rate control information
  – All have excessive internal buffering
  – Some have massively “hidden buffers” in the firmware
  – Some have all of these issues!
  – All of them have other bugs!
Debug Knobs and Stats

- Aggh! We eliminated the qdisc layer!
  - tc -s qdisc show dev wlp3s0
    qdisc noqueue 0: dev wlp3s0 root refcnt 2
    Sent 0 bytes 0 pkt (dropped 0, overlimits 0 requeues 0)
    backlog 0b 0p requeues 0
- Ifconfig wlp3s0  # still works!
  - Link encap:Ethernet  HWaddr 30:10:b3:77:bf:7b
    inet addr:172.22.224.1 Bcast:172.22.224.255 Mask:255.255.255.0
    inet6 addr: fe80::3210:b3ff:fe77:bf7b/64 Scope:Link
    UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
    RX packets:2868196 errors:0 dropped:0 overruns:0 frame:0
    TX packets:2850840 errors:0 dropped:0 overruns:0 carrier:0
    collisions:0 txqueuelen:1000
    RX bytes:2331543636 (2.3 GB)  TX bytes:1237631655 (1.2 GB)
Top Level Statistics

- root@nemesis:/sys/kernel/debug/ieee80211/phy0# cat aqm

  access name value
  R fq_flows_cnt 4096
  R fq_backlog 0
  R fq_overlimit 0
  R fq_overmemory 0
  R fq_collisions 14
  R fq_memory_usage 0
  RW fq_memory_limit 4194304 # 4MB on 802.11n per device not SSID
  RW fq_limit 2048 # packet limit IMHO too high by default
  RW fq_quantum 1514 # 300 currently by default, no...
Per SSID stats

- /sys/kernel/debug/ieee80211/phy0/netdev:wlp3s0/aqm

<table>
<thead>
<tr>
<th>AC</th>
<th>Backlog-bytes</th>
<th>Backlog-packets</th>
<th>New-flows</th>
<th>drops</th>
<th>marks</th>
<th>overlimit</th>
<th>collisions</th>
<th>tx-bytes</th>
<th>tx-packets</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>0</td>
<td>0</td>
<td>617594</td>
<td>1</td>
<td>18</td>
<td>0</td>
<td>14</td>
<td>1940667</td>
<td>617602</td>
</tr>
</tbody>
</table>

- Even more detail via:
  /sys/kernel/debug/ieee80211/phy0/netdev:wlp3s0/stations/80:2a:a8:18:1b:1d
AP's and meshes are fine
Hosts...TSQ Issue?
Great Host Latency
suboptimal throughput

Let's not fix this in the driver!
Futures

- Add Airtime fairness
- Further reduce driver latency
- Improve rate control
- Minimize Multicast
- Remove reorder buffers
- Reduce excessive retries – OK to lose some packets!
- Add more drivers/devices
  - Figure out APIs
Airtime Fairness (ATF)

- Ath9k driver only
  - Switch to choosing stations based on sharing the air fairly
  - Sum the rx and tx time to any given station, figure out (via DRR) which stations should be serviced next.
  - Huge benefits with a mixture of slow (or legacy) and fast stations. The fast ones pack WAY more data into their slot, the slow ones slow down slightly.
  - Patches available now!
  - Needs **accurate** rx and tx statistics
Explicit bandwidth/latency tradeoff

- We currently do an implicit reduction in TXOP size (from the codel AQM)
- It would be better to explicitly use shorter TXOPs under contention from multiple stations
- This will cost “bandwidth” - but improve latency.
QoS TID rework?

- VI queue has been broken for forever
- We've shown better scheduling with good aggregation can work better than explicit QoS
- Current airtime fairness code fails with multiple levels of QoS
- Can't we get rid of the per-tid stuff except when absolutely necessary?
Questions?

1+sec latency
Linux 4.4
@mcs0

Sub 40-ms latency
Linux 4.9?
@mcs0
Bufferbloat Resources

• Make-wifi-fast project plan:
  https://docs.google.com/document/d/1Se36svYE1Uzpppe1HWNeyat_sAGghB3kE285LEiJBW4

• Bufferbloat Web Site: www.bufferbloat.net
  – Mailing lists, blogs, web resources, irc channel

• A HUGE thanks to all the volunteers, donors, testers, helpers, and to Karlstadt University, the Shuttleworth Foundation, Nlnet Foundation, ICEI, Google Fiber, and Comcast Research for their support!