The State of the Art in (Linux) Congestion Control

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

March 10, 2014
Hindmost, Bufferbloat.net

dave.taht@bufferbloat.net
Overview

- Background -
  - Signal Processing verses Packet Theory
  - The case for reducing RTTs
  - Congestion in the Queues
  - Problem: Bufferbloat at all the layers

- Reducing RTTs end to end with New Protocols
- Packet Schedulers and AQMs
- New problems
A “signal” encodes data. It wiggles and jiggles and delivers a $X$ amount of data to the other side depending on the encoding...

Interference is bad! Data mixed up with other data results in no data.

Forward error correction and other encodings are used to not lose data.

Losing data is bad! Every Bit is Sacred!

Recently retries and retransmits added to the EE's toolbox – “Get the data through! No matter how long it takes!”
- Especially on wireless technologies
- And I wish that acks, retries and retransmits had stayed at layer 3

Data “are signals” “that must be preserved”

You connect applications together via dedicated circuits
- If you run out of circuits you run out of capacity
VERSES PACKET THEORY

• Data can be broken up into tiny packets with their own self integrity and delivered in any order to be reassembled at the other side.
• Once broken up into packets, data can AND SHOULD be mixed up with other data.
• Losing a packet is a signal of congestion – and it is normal and desirable to run with some loss because otherwise you aren't fully utilizing the network.
  – But you need timely notification of a loss
  – Packets are nearly always paired in a control loop
• Not every packet is sacred!
  – https://www.bufferbloat.net/projects/bloat/wiki/Humor
Figure 1: Window Flow Control ‘Self-clocking’
BELLHEADS and NETHEADS have always been in conflict

- Layer 2 and down
- EEs
- IEEE 802 Group
- Verilog, VHDL

- Layer 3 and up
- CS and Networking
- IETF
- C, Python

In the 90s it was ATM vs Ethernet


- Now its GPON/wifi/3g... nearly every new technology...
Some Packet History

• 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."

• 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...

• Are these giant flows still packets? Average packet size today is still ~300 bytes...
  VOIP/gaming/signalling packets generally still less than 126 bytes
• Donald Davies, Leonard Kleinrock and Paul Baran are all good reads

• [http://www.rand.org/content/dam/rand/pubs/research_memoranda/2006/RM3420.pdf](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](http://dspace.mit.edu/bitstream/handle/1721.1/11562/33840535.pdf)

• Are Donald Davies 11 volumes on packet switching not online??

Other Good Reads:


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/~ecn5032/handouts/94LelandSelfSim.pdf
- Google for lots of papers on “Packet Pairing”.
- Van Jacobson's Queue Rant: http://www.pollere.net/Pdfdocs/QrantJul06.pdf
- Pfabric:
- On the Co-existence of AQM and Low Priority Congestion Control:
- See this past IETF's ICCRG proceedings
TCP's behavior (TCP 101)

- TCP will always fill the biggest buffer on the path
- As the delays get larger – congestion avoidance mode geometrically gets slower
- With CUBIC, the sawtooth looks more like an S-curve
- This is a ns2 model from:
  [http://staff.science.uva.nl/~delaat/netbuf/bufferbloat_BG-DD.pdf](http://staff.science.uva.nl/~delaat/netbuf/bufferbloat_BG-DD.pdf)
  (.5Mbit uplink)
Random sample of 5000
from 104839 crossing this range
out of 6870022 original responses
(116 of which cover this range)
Web Browsing is dependent on **RTT**

**Page Load Time vs. RTT**

- Page Load Time is sensitive to *round-trip* latency
  - Google data shows 14x multiplier
    - +200ms RTT = +2.8 seconds PLT
- Diminishing returns from increased data rate
  - Page Load Time at 10 Mbps almost indistinguishable from 6 Mbps

Source: SPDYEssentials, Roberto Peon & William Chan, Google Tech Talk, 12/8/11

**Gaming, DNS, and VOIP traffic are even more sensitive to RTT!**
Figure 1: Window Flow Control ‘Self-clocking’
Latency with Load Today

Most of this latency comes from Queue Delay (bufferbloat)
Bufferbloat

- **Wikipedia**: “a phenomenon in a packet-switched computer network whereby excess buffering of packets inside the network causes high latency and jitter, as well as reducing the overall network throughput.”

- Bufferbloat is really two things:
  - Excessive buffering at the device, device driver, network queue and tcp/udp queue layers in network stacks on all modern operating systems (windows, mac, Linux, etc)
  - Lack of good packet scheduling and active queue management at ANY layer in all Oses and in common edge devices such as home network gateways, dslams, and cable head ends.

- Without fixing the first, you can't reason about the second.
- You only see the latency spikes when under load.
- Queues are usually either empty, or full.
- All sorts of loads exist, from constant, to transient. Transient spikes exist, but are hard to see. Easy to feel or hear, however. It's easier to create constant loads and measure against those... but not necessarily an accurate representation of reality.
Smokeping

RTT of this path is less than 10ms!
- rsync of X Consortium archives from my house to expo.x.org.
- “good behavior” are when I suspended the copy to get work done.
In 2009 Bufferbloat was at all layers of the network stack

- Applications
- TCP
- CPU scheduler
- FIFO Queuing systems (qdiscs)
- The device drivers (tx rings & buffers)
- The devices themselves
- The mac layer (on the wire)
- Switches, routers, etc.
Network “Constants”

- Speed of light in the medium
- Media Acquisition time (Cable Request/Grants, Wifi EDCA)
- Min/max Packet size (MTU)
- RTT
- 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 GigE.
- If you start fixing the endpoints, your choices change
- Most of these constants are actually variables
TCP Improvements 2008-2014

- Proportional Rate Reduction (3.0)
- Byte Queue Limits (3.3) (more of a device driver fix)
- IW10
- TSO/GSO improvements (3.12)
- Fair/Flow Queuing packet scheduling (3.6)
- Pacing (linux 3.13)
- The Linux stack is now mostly “pull through”, where it used to be “push”, and looks nothing like it did 3 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.

There are a whole bunch of new things slated for 3.14-3.16 under test from google's make-tcp-fast team.
LEDBAT

• Low Priority Congestion Control algorithm
  - Tries to assure some low rate of goodput relative to other loads on the link
  - Bitorrent is similar but different enough to not be directly comparable
  - Usually tries to hold latencies below 100ms

• Does not work as intended in a packet scheduled/AQM age:
  “On the Co-existence of AQM and Low Priority Congestion Control”
  - Aqm or packet scheduling turn delay based tcps back into loss based ones.

• But does it matter if you nearly eliminate queue bloat?
WEBRTC

- Upcoming videoconferencing standard for inter browser video/audio communication
- P2P, encrypted, does not hole punching
- Early congestion control attempts were not very good...
- Code is Open source, available in firefox/chrome go try it!
- Or play with it at places like http://appear.in
- Multiple W3C, IETF working groups on the case
QUIC
“Quick UDP Internet Connections”

- Developed and deployed by google in 2013
- Replacement for TCP for web transactions
- Aims for highly secure, 0 RTT setup
- Does forward error correction, packet pacing, speculative retransmission – incorporating nearly every idea that can't fit into TCP, baked or otherwise
- Now semi-deployed now in chrome. Now in it's 13th protocol revision.
- What else can it be used for besides web transactions? DNS? File transfer? Video?
- Is it really secure?
Other interesting “new” protocols

- Multipath TCP
- Data Center TCP
- Mosh
- Trickle

- But in all cases the endpoint protocols have to compete with the control loops of legacy TCP.
- There are 10+ billion endpoints to fix.
- Can we fix the Queue models in the routers, instead?
Can we fix the routers with more intelligent Queue Management?

- RED developed 1992 – still the gold standard
- Too hard to configure, not deployed
- RED, ARED, LRED, *RED adherents – active queue length management - “AQM”
- SFQ, SQF, DRR adherents - “Packet Scheduling”
- SFB = SFQ + the Blue AQM
- New work: Codel, FQ_Codel and PIE
- Very promising reductions (below 20ms) in queue length while maintaining good put and improved fairness
- FQ_Codel: a hybrid of DRR, SQF, and Codel

- Answer: Yes

- CeroWrt
  - Test tool for new bufferbloat fighting technologies
  - Has all the AQM/Packet scheduling technologies developed through 2013

- All available in openwrt, dd-wrt, linux mainline, etc. Today.

- Please try them!
Codel

- Breakthrough paper by Kathleen Nichols and Van Jacobson in ACM Queue
  - Suggests seeking to a minimum as the goal of an AQM
  - Treats latency as the core metric
  - Uses a very smooth control law (invsqrt) to control drop scheduler
  - Best reference (update to ACM Queue) now released as an Internet Draft at:
  - Highly influential on the redesign of several TCP mechanisms as well

- Open Issues
  - Can we come up with a better control law?
  - Leveraging the existing timestamp facility, can we come up with a better rate limiter/pacer?
  - How do codel and fq_codel behave differently?
  - How does the target parameter interact with HTB/HFSC?
  - Can the core ideas be applied to layer 2 technogies like GPON, wireless, 3g, etc.?
  - Can ECN handling be improved? (no love for random drop)
  - Google Summer of Code: Volunteer wanted for updating the NS2 and NS3 models
CoDel Drop Scheduler Behavior

Trying to find the RTT

Quiescence

Sojourn time (time above target)

Target

Interval

Time between drops
PIE

- 3-8 bands of PI controller, for low/medium and high congestion handling
- While in the Linux kernel now, what's there is very different from the original code & paper
- Good operating range
- NS2 simulations of the DOCSIS 3.1 stack are part of Google Summer of Code
More FQ + AQM-like variants

- SFB
- SFQRED
- SFQARED (Linux only)
- SFQ with a better hash (ARRIS)
- QFQ + Codel or RED (Linux or BSD)
- WFQ + Codel or RED (BSD)
- LRED (ARRIS paper)
- FQ_PIE (in ns2)
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 (but increasingly out of date)
Web traffic vs various qdiscs

CDF of Web Page Load Time under Tested Conditions

Video at: http://www.youtube.com/watch?v=NuHYOu4aAqg
Current cable modem performance
20Mbit Down/8Mbit up

Realtime Response Under Load
Download, upload, ping (scaled versions) - Campground PFIFO_FAST

TCP download
- BE
- BK
- CS5
- EF
- Avg

TCP upload
- BE
- BK
- CS5
- EF
- Avg

Ping (ms)
- UDP EF
- UDP BK
- UDP BE
- ICMP
- Avg

Local/remote: lupin-gw/snapon.lab.bufferblob.net - Time: 2013-01-11 12:50:00.893620 - Length/step: 60s/0.20s
TCP's behavior (TCP 101)

- TCP will always fill the biggest buffer on the path
- As the delays get larger – congestion avoidance mode geometrically gets slower
- With CUBIC, the sawtooth looks more like an S-curve
- This is a ns2 model from: http://staff.science.uva.nl/~delaat/netbuf/bufferbloat_BG-DD.pdf (.5Mbit uplink)
Cable modem performance w htb + 3 tier fq_codel

Realtime Response Under Load
Download, upload, ping (scaled versions)

TCP download
- BE
- BK
- CS5
- EF
- Avg

TCP upload
- BE
- BK
- CS5
- EF
- Avg

Ping (ms)
- UDP EF
- UDP BK
- UDP BE
- ICMP
- Avg

Low rate flows retain low latency
Full download rate achieved
Handles Priority & Background traffic
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?
- 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?
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 routers
  - 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?

![Figure 1: Window Flow Control ‘Self-clocking’](image-url)
Drop Tail FIFO queues are going to go away (in Linux) soon

- All the software infrastructure is in place in Linux to swap them out for something else.
- One line of configuration to change
- What that something else is remains to be decided.
- Fq_Codel is good for routers, sch_fq good for hosts
- Issues with interactions with multiple hardware queues
- Perhaps a more complex 3-band fq codel-ing qdisc is closer to what is needed
- Debate continues
Host Queue improvements: 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.
sch_hhf

- In linux 3.14
- From a 2002 paper
- Uses a bloom filter to deprioritize “heavy hitter” flows
- Can be used in combination with other schedulers/aqm
- That's all I know about it
Yet software rate limiting was the CPU intensive elephant in the room

- DOCSIS 3.1 model tightly intertwines the PIE AQM with access to the media and the rate limit set by the ISP
- Supports “Speedboost”

- HTB (used in cerowrt) peaks out at 50Mbit on 1990s hardware
- HFSC has interesting drop and latency behavior of its own
- TBF presently unusable
- Are they any other fast rate limiting algorithms out there???
Analysis Issues

• Most of the debloating work is being driven by web traffic analysis from the provider perspective
  – Traditional applications (file transfer, rsync, voip/videoconferencing) under-analyzed
  – Side effects on small/medium business underanalyzed
  – Tools for latency verses load still weak
  – Wifi and wireless work is lagging

• Help wanted!
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
Cambridge Cafe WIFI
Queue latency and goodput

Realtime Response Under Load - exclusively Best Effort
Download, upload, ping (scaled versions)
cambridge-wifi-no-fq_codel

TCP download
- BE
- BE2
- BE3
- BE4
- Avg

TCP upload
- BE
- BE2
- BE3
- BE4
- Avg

Ping (ms)
- UDP BE1
- UDP BE2
- UDP BE3
- ICMP
- Avg

Time
0 10 20 30 40 50 60 70

Mbits/s
0 0.2 0.4 0.6 0.8

ms
200 250 300 350 400
Cambridge Cafe Wifi
Latency and goodput with fq_codel

Realtime Response Under Load - exclusively Best Effort
Download, upload, ping (scaled versions)
cambridge-wifi-fq_codel

TCP download
- BE
- BE2
- BE3
- BE4
- Avg

TCP upload
- BE
- BE2
- BE3
- BE4
- Avg

Ping (ms)
- UDP BE1
- UDP BE2
- UDP BE3
- ICMP
- Avg

Too many buffers underneath on the AP
Guidelines going forward

- NEARLY EVERYTHING ON THE NETWORK IS IN A CONTROL LOOP THAT AFFECTS EVERYTHING ELSE
- CPU is not a problem
  - Using GPUs, and coprocessors might be interesting
- Memory is not a problem
- Low and variable bandwidth/RTT are always problems
- NEARLY EVERYTHING ON THE NETWORK IS IN A CONTROL LOOP THAT AFFECTS EVERYTHING ELSE
- Finding open hardware and software is a massive problem
- The best of what we've been able to do is “out there”, running, in most shipping versions of Linux...
  … but not OSX, iOS, android, or Windows...
  … Or cable head ends, dslams, cable modems or most places where we can do some good...
- NEARLY EVERYTHING ON THE NETWORK IS IN A CONTROL LOOP THAT AFFECTS EVERYTHING ELSE
Some pending Internet Drafts

- AQM working group
- Pending drafts:
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
RRUL Test-  
https://github.com/tohojo/netperf-wrapper

A big thanks to the bloat mailing list, Jim, Kathie, Van, and Eric, ISC, the ceroWrt contributors, OpenWrt, the Internet Civil Engineering Institute, and the Google Fiber for their interest and support in the work!