Alcatel·Lucent

Bufferbloat

Views of the Elephant

Jim Gettys

Bell Labs

October 6, 2012

james.gettys@alcatel-lucent.com, jg@freedesktop.org

Outline

Demonstration

Essential Information

Path Bufferbloat

Aggregate Bufferbloat

Transient Bufferbloat

Solutions

Common Laments of the 2012 Era Internet....

"Daddy, the Internet is Slow Today!"

"Junior, stop what your are doing on your computer so I can make a phone call!!!"

"Boy, this conference's wireless network was working fine before everyone sat down, but now it's horrible!"

"My cell phone's 3g network is incredibly slow here, though my signal strength is good!"

"Sorry, I didn't understand what you said, Skype/Vonage is having problems right now and you dropped out for an instant!"

"My home network is useless whenever I backup my computer!"

"This motel's network is unusable; even Google Search is glacial!"

What do all of these laments have in common?

Bufferbloat!

Demonstration video

http://www.youtube.com/watch?v=npiG7EBzHOU

Buffers are everywhere, and will fill!

Buffers are necessary to smooth bursts of packets.

A **single** TCP connection will fill any size buffer given time; the buffer your TCP will fill is the buffer just before the bottleneck link.

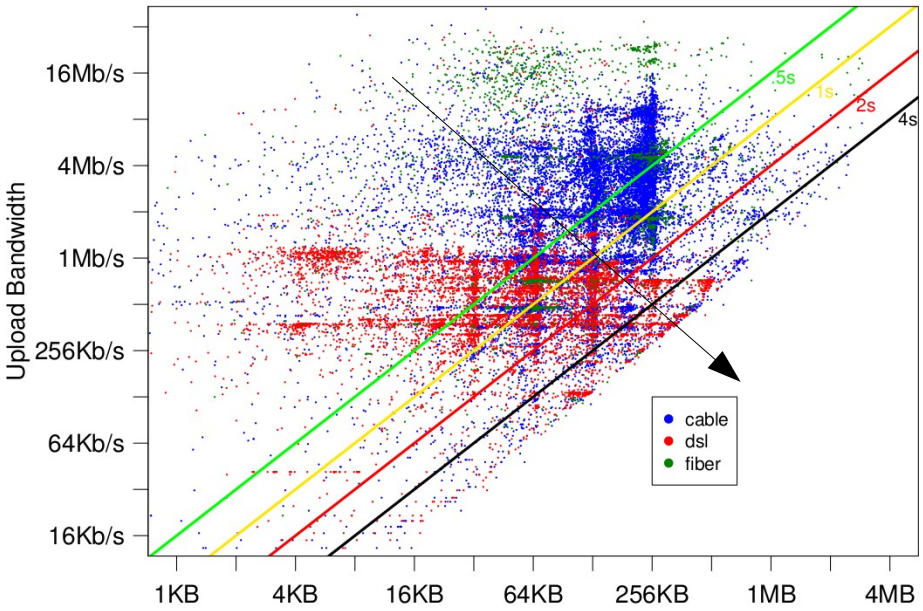
TCP's design presumption, that there will be timely packet loss to signal congestion, has been violated by these buffers.

UDP applications (e.g. RTCWeb) have no such limitations (and do not yet even deal with congestion control: see IETF rmcat working group.

These buffers, no longer serving their original purpose, only add delay.

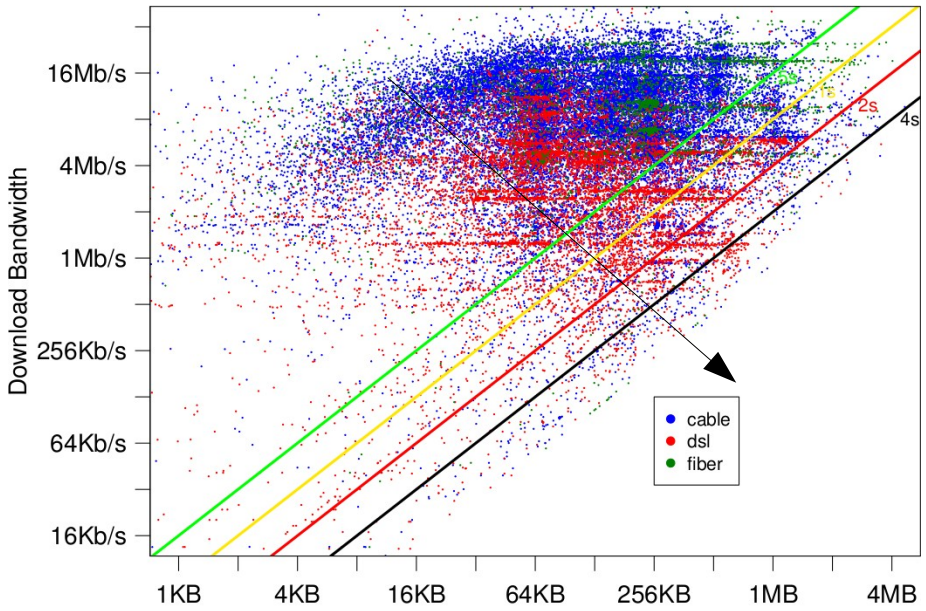
AQM (Automatic Queue Management) is essential to avoid elephant flows and queues filling: and TCP's responsiveness is *quadratic* in the delay, the elephant herds really don't like to share

"Netalyzr: Illuminating Edge Network Neutrality, Security, and Performance" C. Kreibich, N. Weaver, B. Nechaev, and V. Paxson



Green diagonal line == .5 second latency

Uplink



black diagonal line == 4 second latency

Downlink

Arrow direction is increasing latency

Note: telephony standards for latency are maximum of 150ms!!!

This data is a *lower* bound on the severity of the broadband bufferbloat problem.

What happens when a network is slow due to bufferbloat? - protocols fail due to both packet loss & high latency timeouts

Once a network/link exhibits high latency and bad packet loss, other critical, statistically insignificant but mission critical packets can't do their jobs

- DNS – adding 100's ms or seconds of latency to lookups kills web browser performance and losses cause lookup failures
- ARP - relies on timely resolution to find other devices on your network
- DHCP - if these packets are lost or excessively delayed, machines can't get on the network
- RA and ND - essential for IPv6 functioning
- VOIP/teleconferencing- needs about a single packet per 10ms flow in order to be good, and less than 30ms jitter.
- Gamers - will get fragged more often with latencies above their twitch factor
- Responsiveness of all network applications, web or otherwise, suffers

Protocols can fail entirely with timeouts & excessive packet loss

Buffers only fill when they are **next** to a saturated bottleneck – at all other times they are “dark”

Your hosts, in your applications, and in socket buffers and network layers

- Your MAC itself may have packet buffers internally;
- Network device drivers themselves
- Your network interface's ring buffer potentially buffers **thousands** of packets
- And the VM system your OS may be running on top of may add yet more layers

Your wireless access, in **both** directions

- Cellular Wireless Systems have major problems: it's why your cell phone may be very slow
- 802.11 has similar issues: long packet delays destroy timely notification

Your switch fabric (8 ms/switch at 1GBPS): how many hops, how congested?

Your home router – potentially megabytes

Your CPE/cable modem/FIOS box – potentially megabytes

The head-ends of those connections (e.g. DSLAM, CMTS, etc.)

Each and every router and switch in your path, and the line cards in those routers

Bufferbloat Situation

Buffers only fill before a bottleneck. But those bottlenecks are now routinely next to any wireless device

Hypothesis: most (but not all) bufferbloat locations we experience are in the edge: e.g. home & cellular networks

But beware of back-haul networks too!

Home routers and hosts, tablets and cell phones are at least as bad as broadband

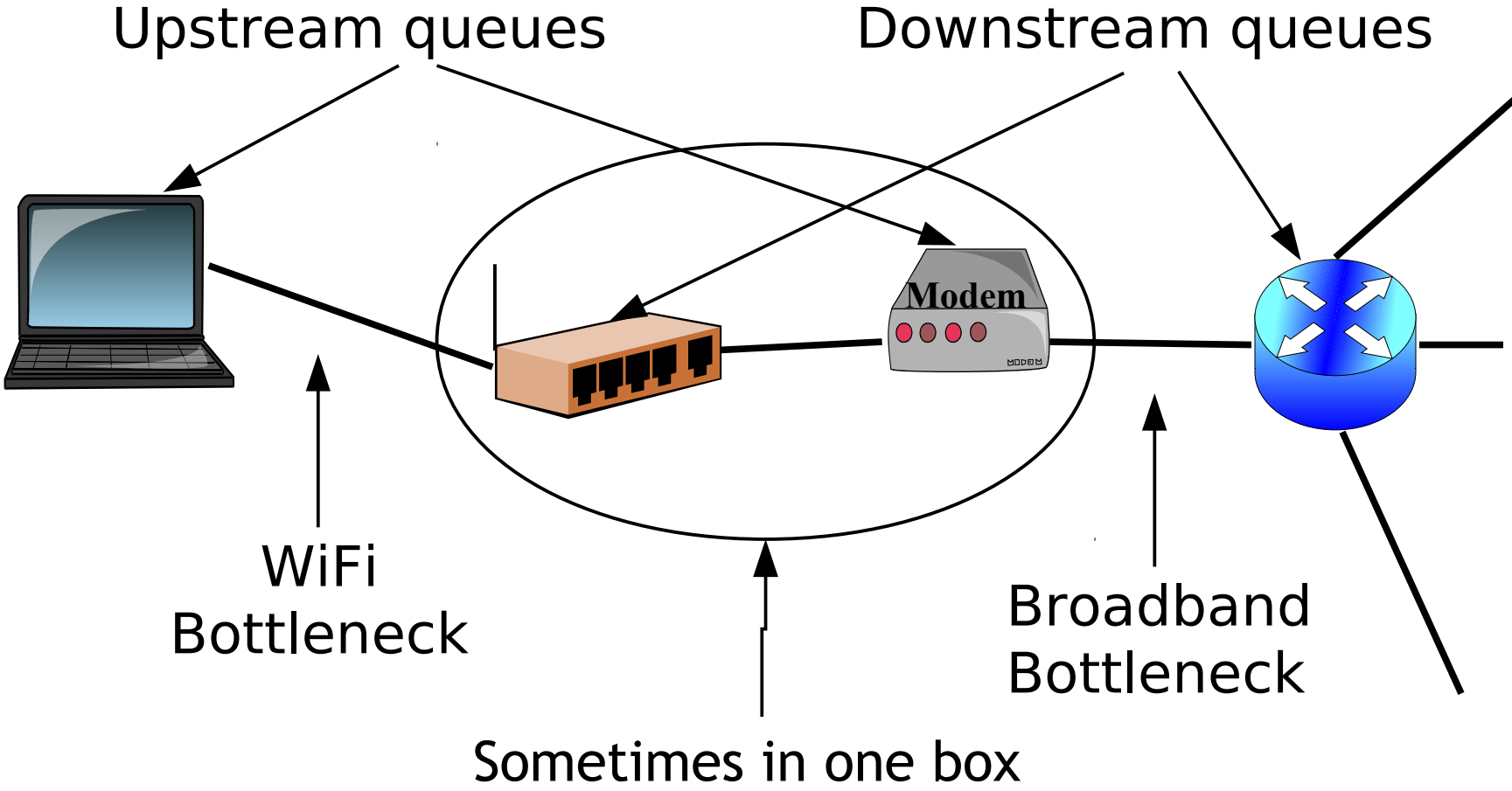
Problems are all over the Internet: the edge is likely the most severe, though it is endemic in hosts, home routers, broadband gear, 3g, some switches, overloaded routers....

Reminder: there are *two* bottlenecks are in play in the home

Broadband hop (single bloated queue!)

Wireless hop (potentially four HW queues in 802.11)

Home bufferbloat: there are usually two bottlenecks! And it moves!



Transient Bufferbloat

Web browsers and Web sites are doing ***Evil*** together

- Web browsers no longer limit the number of TCP connections
- Web sites are now often “sharded”: split across a number of different names
- Has destroyed any congestion avoidance when Web surfing

Extreme example

CNN.com-> induces 60 simultaneous TCP connections!

- @ IW 4, that is 240 packets, or 360K bytes (3Mb) in flight to your customer's broadband connection, *which has a single grossly overbuffered queue so from the server side you see little packet loss!*
- What do you think happens to your customer's VOIP/Skype conversation?

Is this what you want to do to your “customer”?

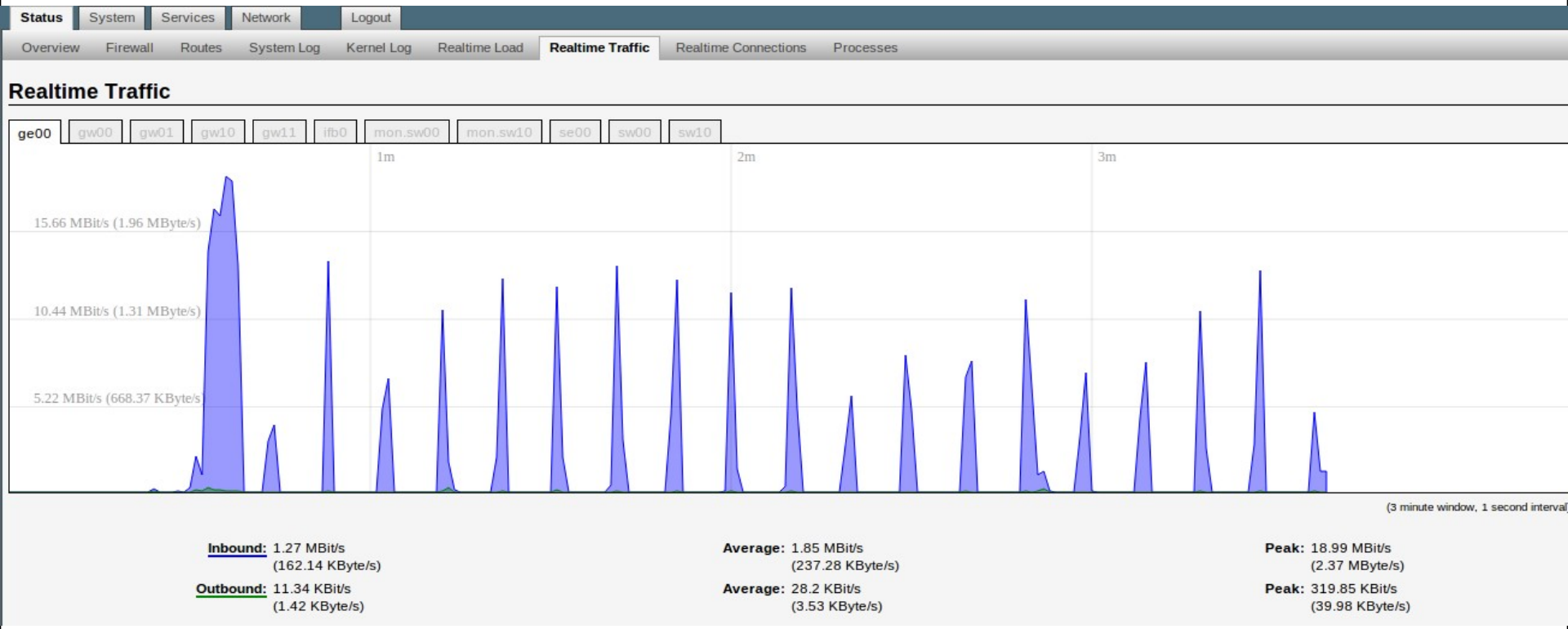
Streaming Video Isn't Streaming at all

"Streaming" video is really TCP transfers of video "chunks"

Think of it as periodic moderate sized file transfer

Each "chunk" is inducing temporary latency to your broadband connection to the Internet, by running TCP as fast as it can to transfer the "chunk"

Netflix “streaming” Behaviour



Host bufferbloat, and your home router

Since today's home routers are usually using general purpose operating systems (usually Linux), the problem is on both sides of your wireless link

Buffers hide in multiple places in modern OS's + hardware
(*Linux, Macintosh, Windows alike*)

Let's do a simple calculation, presuming 10Mbps actual data transfer rate:

- 256 packets is of order 3,000,000 bits == 1/3 of a second (one way)

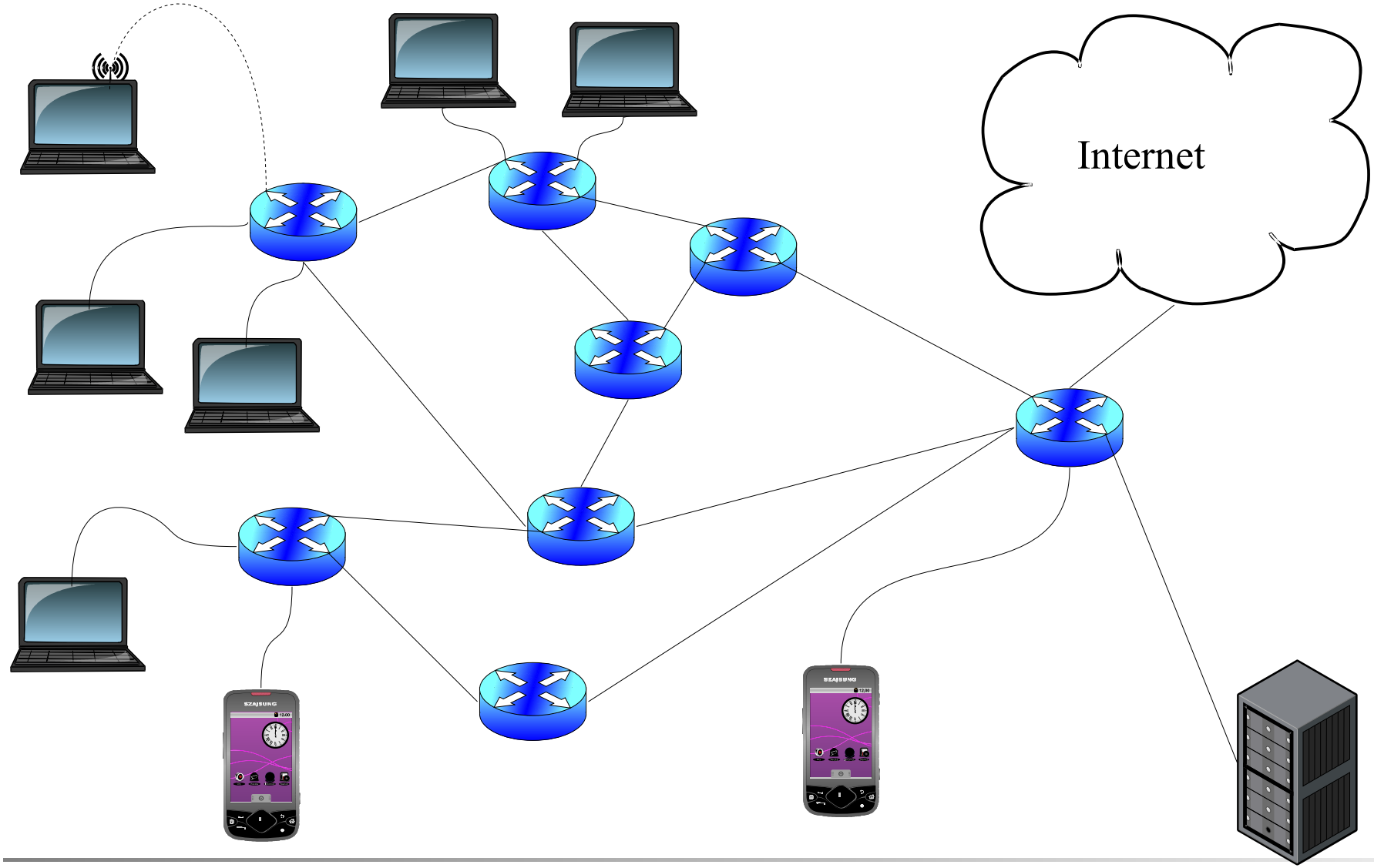
What happens at a busy conference, where your “fair share” might be 100Kbps? 30 seconds: applications (and people) timeout entirely....

Fixed size buffering is usually ***nonsense***

Buffering must always be dynamically managed!

Automatic Queue Management is a necessity, not a nice to have!

What happens in a complex network?



Aggregate Bufferbloat

What happens in a complex network?

Your packets can get stuck behind other people's queues, not just in your bottleneck, but in other bottleneck links.

So you can have **additive** bufferbloat; each saturated hop will add its delay.

We saw this in the 1980's and 1990's when the core of the Internet was always congested.

We thought (W)RED fixed this problem in the 1990's. But really, fiber and enough capacity fixed it instead.

Because (W)RED is fundamentally flawed... It requires tuning, and can hurt you if you tune it incorrectly. As a result, many network operators do not enable it even when they should.

But it's all you have today in most core routers....

Mitigations

1) Twist knobs to reduce the problem; once you understand the problem, you can sometimes reduce bufferbloat by twisting knobs to less insane values.

2) Engineer to put the bottlenecks someplace where you can control the buffering: e.g. the bandwidth shaping “hack” I demonstrated.

The fundamental problem remains: the (wireless) hop next to the user's devices are highly variable and “correct” buffering is unknowable: you don't know either the delay, nor the bandwidth.

Drop tail queues always add unnecessary latency.

Fundamentally, all buffers before potential bottlenecks must be managed.

Solutions

Delay sensitive TCP's

- Doesn't solve the UDP problem, with WebRTC coming, this will soon become critical
- Requires fork-lift upgrade of everything
- We don't have a TCP algorithm that really “works” and does not lose in competition with existing TCP's

Automatic Queue Management

- Enable and configure (W)RED where appropriate, while waiting for something better
- “Something better” finally exists. And “Something better” must not require human intervention: it should “just work”, and “do no harm”.

But wasn't AQM solved in the 1990's?

The classic AQM algorithm is RED, (Floyd and Jacobson, 1993)

Over ten years ago, Kathie Nichols walked into Van's office one afternoon, and showed him that RED has two bugs

- Kathie Nichols & Van Jacobson twice tried to publish papers explaining RED's flaws
- A 1999 draft of *RED in a Different Light* draft did escape

(W)RED requires tuning, and the 100 or more papers about RED tuning in the last decade confirm this. Ergo, network operator's reluctance to enable RED is understandable, even if their fears are (usually, but not completely) excessive, since RED must be used carefully!

And RED can't work at all in the face of variable bandwidth, such as found in broadband, and wireless

We need something better: enter the CoDel (“Constant Delay”) algorithm

CoDel Status

See Van Jacobson's presentation at the Vancouver IETF of Kathie Nichols and Van Jacobson's new CoDel (Constant Delay) AQM algorithm: published in the CACM, July, 2012, available on the ACM Queue website: *Controlling Queue Delay*

Linux 3.5 has codel and fq_codel queue disciplines: fq_codel combines CoDel with SFQ: fq_codel by Eric Dumazet, SFQ by Paul McKenney

Work on refining the CoDel's design and implementation continues

Van Jacobson (and everyone who has worked on CoDel), recommend fq_codel over codel

We ***really***, ***really***, ***really*** like fq_codel

We cannot reach tolerable latencies due to transient bufferbloat without smarter queuing: head of line blocking kills low latency applications

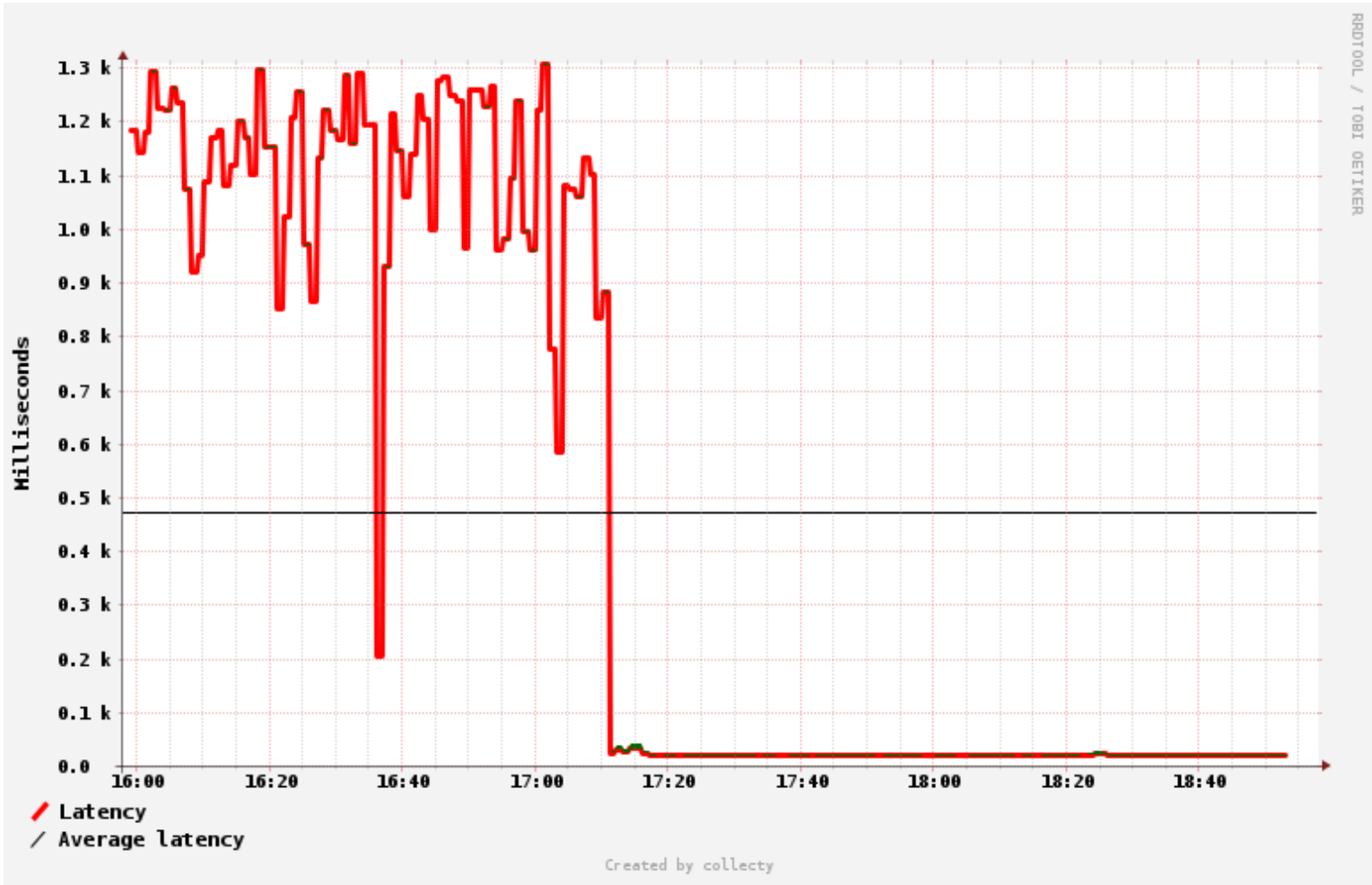
How fq_codel works:

- You put each flow in it's own queue; there are two sets of queues: those which do not build queues, and those which do build queues
- If a flow does not build a queue, it is scheduled before flows that builds a queue
- If a flow builds a queue, it gets put into a list of queues for lower priority scheduling. If the flow's queue empties, after a period, it is again put on the list of queues that gets priority treatment. fq_codel also avoids starvation of queues that do build a queue; they will make progress; just slowly

So without explicit classification, your DNS lookups, your TCP opens, your voip traffic, DHCP, RA, etc, fly through the router ahead of any bulk data

Fair queuing is only 2% of CPU on 10GigE: proof point that smart queuing is feasible on today's systems

IPFire DSL fq_codel results



1.2 s latency drops to 20ms latency while preserving utilization

But: differnet hardware may have additional buffering

Alcatel·Lucent

Current Experiences from Lantiq DSL driver

John Crispin (Linux Lantiq driver maintainer & key OpenWrt developer) has been working on confirming the IPfire results.

As in other technologies such as Ethernet, the “smart” hardware is getting in the way: minimum Lantiq ring buffer size is 16 packets

He's seeing 60ms latency under load on a .5Mbps uplink using fq_codel, but with 2 packets buffering (30 should be achievable).

Is CoDel universal? Jury is still out

CoDel is simpler than (W)RED to implement, and is very amenable to silicon. But CoDel may be difficult to retrofit into existing designs.

We're chasing several problems: we don't yet know if they are in our code, or the CoDel algorithm. We think we have at least one bug in the code outstanding, and have problems when running hundreds of simultaneous flows that may require changes to CoDel's control law

- Lack of funding for Kathy Nichols is slowing investigations. Sigh.

Unmodified, today's CoDel is not suitable if all your traffic is inside a data center where your RTT's are measured in microseconds; CoDel was designed for solving the severe problem we have at the edge of the network.

But the ideas in CoDel (e.g. Sojourn time) are new and allow for new attacks on the queue management problem

- e.g. Sojourn time allows use across multiple queues simultaneously: e.g. fq_codel.

There is No Single Bullet for a Low Latency Internet

Multiple queues are necessary in WiFi; but today we have only one bloated queue in broadband in each direction!

One 1500 byte packet @ 1Mbps == >13ms and for VOIP, we want low latency/jitter access to the medium.

AQM needed to avoid elephant flows and queues from filling: TCP's responsiveness is *quadratic* in the delay

Smart Queuing is also needed

- "Fair" depends on where you are: I don't mean simply TCP fair queuing but smart queuing among TCP flows, among devices, among customers, among policies; we must become much smarter than dumb FIFO queues at bottlenecks
- TCP fair queuing does help RTT fairness, ack compression, interactive versus non-interactive bulk transfers, etc. Having TCP fair queuing at the host reduces the surprising RTT behavior seen most dramatically in locations like New Zealand where RTT's are so dramatically different

Really Big Headaches

Current broadband has a single bloated queue

- The technologies admit to additional queues, but these are today only available to the ISP's telephony services

How to communication the customer's classification preferences?

Broadband splits the diffserv domain between the customer & the broadband head end; the customer only has control of the upstream, so how does the downstream know what the user needs?

- A explicit protocol
- Andrew McGregor's idea to infer incoming classification & marking from outgoing marking on flows
- Hypothesis: some bandwidth threshold, smarter queuing without classification and actual separate queues (e.g. fq_codel) is likely “good enough”.

Wireless is **hard**!

Remember, there can be (usually are) **multiple** buffers stacked in an system! This partition of queues makes proper queue management difficult.

Bandwidth is extremely variable: BQL (mostly) solves the driver bloat problem for Ethernet, but 802.11 can be fast enough we would like to use some of the smarter hardware at those bandwidths.

802.11n aggregation requires the driver to have access to many, many packets underneath Linux's queue disciplines.

Our operating system interfaces need rethought; how and where do we run (fq)CoDel across coupled queues?

But before CoDel/fq_codel, we had no hope for a solution.

Commercial Home Router Disaster

Commercial home routers are broken in 4 major ways

- Firmware is horribly antique and insecure; today's latest commercial home routers usually ships (at least) 5 year old software on new hardware, which seldom if ever is updated once “stable”, which then rots for years after that without update
- Decent IPv6 deployment is now gated by the home routers
- Extreme bufferbloat in all its forms
- Tragedy of the Commons: Funding model of the home router market is broken; there is next to no funding toward engineering to fix problems today: this means that little will happen without community participation

Time to roll up your sleeves and get your hands dirty...
OpenWrt is already years ahead of what you can buy at Best Buy.

In Disaster, There is Opportunity: CeroWrt

CeroWrt is an advanced build of OpenWrt, using WNDR 3700v2 and WNDR3800 routers for more flash, Atheros radios, and fast CPU

Every line of code is available to modify; changes that work go upstream to OpenWrt and Linux as fast as are validated

Today running Linux 3.3.8 release with CoDel, BQL. Running fq_codel on WiFi, which is today only partially effective due to buffering in the drivers due to 802.11n aggregation

DNSmasq & Current Bind & DNSsec in chroot jail available

Routes, not bridges; 6 networks in the box

Real web server, proxy, IPv6 support, mesh networking, extensive network test tools, etc.....

Come help test, develop, and improve

Demonstrate your heretical ideas with running code!

Next steps

Have to upend the dysfunctional home router ecology.

- Make upstream router distro viable to ODM's
- Bufferbloat in Wireless

Put heat into the economics

- Need consumer oriented tests to encourage changes in buying behavior
- Need to change the marketing discussion: bandwidth != speed. How to compare networks: Power?
- Need better tools for journeyman engineers

There is hope! But much work left to do...

You can suffer **much** less at home **immediately**, if you understand bufferbloat

Bufferbloat is now understood to be a serious problem in the technical community, but problems are all over the Internet, from end-to-end

DOCSIS (cable) will improve greatly very soon, with the deployment of a new DOCSIS buffer control amendment, starting market place pressure

We (now, with CoDel) have all the technologies required to build a low latency Internet, but it requires work in many places

Linux has made the most progress of any operating system to date with:

- BQL (Byte Queue Limits)
- TCP small queues
- Codel/FQ Codel

But wireless is **hard**, and there is much, much, much more to do.

Remember, we are all in this bloat together!

Please come help before we sink!

My Blog – http://gettys.wordpress.com

Other Information

http://www.bufferbloat.net/projects/bloat