The Internet is Broken, and How to Fix It

Jim Gettys

Bell Labs

jg@freedesktop.org

Many real time applications such as VOIP, gaming, teleconferencing, and performing music together, require low latency. These are increasingly unusable in today's Internet, and not because there is insufficient bandwidth, but that we've failed to look at the Internet as a end to end system. The edge of the Internet runs congested most often today.

Where once a home user's Internet connection consisted of a single computer, it now consists of a dozen or more devices – smart phones, TV's, Apple TV's/Roku devices, tablet devices, home security equipment, and one or more computer per household member. More Internet connected devices are arriving every year, which often perform background activities without user's intervention, inducing transients on the network. These devices need to effectively share the edge connection, in order to make each user happy. All can induce congestion and bufferbloat that baffle most Internet users.

The CoDel ("coddle") AQM algorithm provides the "missing link" necessary for good TCP behavior and solving bufferbloat. But CoDel by itself is insufficient to provide reliable, predictable low latency performance in today's Internet.

Bottlenecks are most common at the "edge" of the Internet and there you must be very careful to avoid queuing delays of all sorts. Your share of a busy 802.11 conference network (or a marginal WiFi connection, or one in a congested location) might be 1Mb/second, at which speed a single packet represents 13 milliseconds. Your share of a DSL connection in the developing world may similarly limited. Small business often supports many people on limited bandwidth. Budget motels commonly use single broadband connections among all guests.

Only a few packets can ruin your whole day! A single IW10 TCP open has immediately blown any telephony jitter budget at 1Mbps (which is about 16x the bandwidth of conventional POTS telephony).

Ongoing technology changes makes the problem more challenging. These include:

- Changes to TCP, including the IW10 initial window changes and window scaling.
- NIC Offload engines generate bursts of line rate packet streams at multi-gigabit rates. These features are now "on" by default even in cheap consumer hardware including home routers, and certainly in data centers. Whether this is advisable (it's not…) is orthogonal to the reality of deployed hardware and current device drivers and default settings.
- Deployment of "abusive" applications (e.g. HTTP/1.1 using many > 2 TCP connections, sharded web sites, BitTorrent). As systems designers, we need to remove the incentives for such abusive application behavior, while protecting the user's experience. Network engineers must presume software engineers will optimize their application performance, even to the detriment of other uses of the Internet, as the abuse of HTTP by web browsers and servers demonstrates.
- The rapidly increasing number of devices sharing home and small office links.

All of these factors contribute to large line rate bursts of packets crossing the Internet to arrive at a user's edge network, whether in his broadband connection, or more commonly, in their home router.

Requirements

Not all requirements apply everywhere. For example, regulating different user's total bandwidth isn't necessary in an Internet core router, as it is already regulated at the edge of the network. Home and small businesses routers have different requirements than core routers. These requirements include:

1. Handle changing bandwidth quickly and robustly, since both wireless and broadband system's bandwidth is variable

2. Preserve good utilization of your bandwidth, while retaining real time performance for latency sensitive traffic, such as VOIP, gaming, etc.

3. Buffers should really "work", and not be perpetually full. If they run full, bad things happen when bursts occur.

4. "Fair" division of bandwidth among users, and "fairness" between different applications of that user.

5. Solving the BitTorrent problem, redux. But BitTorrent is as just an example of what other applications may want and need to do; our systems still need to protect themselves from this behavior.

6. Trying to deal with VPN's, as best we can.

7. Good behavior for "ant" protocols such as DNS, DHCP, RA, etc, so that the network operates well even under extreme load.

To achieve these requirements, we need to simultaneously solve a number of problems, not necessarily (or even desirably) in one algorithm.

- Bufferbloat itself, only soluble by a suitable adaptive AQM algorithm, ensures buffers kept generally empty so they (and TCP itself) can function properly. TCP's responsiveness to sharing bandwidth between competing flows depends on the square of the delay: 10 times to much buffering induces 100 times the delay. An AQM algorithm that can adapt to wireless (or variable broadband links) successfully has not been available; existing algorithms are unsuitable. Bandwidth utilization argues for an AQM which is reasonably efficient and properly adaptive to available bandwidth. Some AQM's may manage latency well, but not necessarily allow for good utilization of available bandwidth. Even fewer adjust to variable bandwidth. To the extent possible, we'd like to have our cake and eat it too. CoDel (pronounced "coddle") has the needed characteristics and is showing excellent results.

- Good real time performance for latency sensitive traffic argues for classification, since even a single packet on a low bandwidth link (or a heavily loaded link for which your share of bandwidth is low) is significant.

- "Fairness" between applications is also essential. We should reduce/eliminate the current perverse incentives for applications to abuse the network, as HTTP does today. We've had an arms race conspiracy for the last decade between web browsers and web sites to minimize latency that is destructive to other traffic we may care about (such as telephony, teleconferencing and gaming). Sometimes this is best addressed by fixing protocols to be both more efficient and more friendly to the network, as HTTP/1.1 pipelining and now SPDY are intended to do. But the "web site sharding" problem is impossible for clients to avoid.

- "Fairness" across users. To an ISP, fairness is/should be between paying customers and not individual users: inside the house in a home router, fairness is between users (or other policies the home user wishes to enforce; e.g. guest traffic might only be allowed to use 10% of my network if my network is busy). It might also mean (since devices are often associated with users), “fairness” between devices.

- “Fairness” is also between different flows of different RTT’s.TCP itself is not “fair”. TCP makes no guarantees of “fairly” dividing bandwidth between flows of different RTT’s, nor can it solve the fairness problem between users and wishing it would do so is futile and counter productive. BitTorrent may have a hundred flows simultaneously, and it isn’t “fair” for a background protocol to compete with other traffic of others in my house, or even interactive or real time traffic of my own on my own computer. The basic observations are:

1. One size does not fit all
2. Exactly what “fairness” means depends on location
3. “Fairness” may also mean that heavy users won’t compete with my traffic at busy times of day, or if they have already used too much bandwidth, or…
4. “Fairness” is in the eye of the beholder.

Since “fairness” cannot be guaranteed by TCP, AQM, necessary as it is, cannot be the entire solution for reliable low latency applications to flourish.

The “Edge” of the Internet

Systems/devices in the edge of the Internet have a fundamental advantage over an Internet core router: the ratio of CPU cycles available/packet is much, much more favorable, and computation often comes for “free” hidden behind cache misses. Techniques in that in previous decades wereprohibitive, such asfair queuing, can be usedeven on very fast links. So, for example, Dave Taht’s and Eric Dumazet in the bufferbloat project’s experiments with thefq_codel queue discipline is that it is comparable in speed to Linux’s current default pfifo_fast queue discipline, consuming only 2% of a modern CPU at 10GigE speeds. On current home router hardware with GigE Ethernet, fq_codelprofiles similarly well. Fair queuing as a default queue discipline is therefore now feasible in all edge hosts and devices.

Today’s Internet violates the principle of “least surprise.” Inexpert users, (particularly in remote locations such as New Zealand) are baffled when transfers have vastly different results when competing transfers have very different RTT’s. We can solve this “surprising” behavior that most Internet users don’t understand (and today pester their ISP’s about) using fair queuing.

Fair queuing has many other good features: it naturally prioritizes short lived flows and flows which are not elephants (which may be DNS lookups, DHCP requests, TCP opens, etc.) without requiring explicit classification rules, nor does it require knowledge of the insides of encrypted packets. The early results for fq_codel look wonderful, even without other classification rules or diffserv support. Andrew McGregor reports “phenomenal” results in New Zealand using the fq_codel queue discipline and port based QoS classification rules.

Since fairness is in the eye of the beholder, that “fair” queuing will be different at different locations in the edge of the network. The queues may very well be keyed differently on a per application, per user, per machine/device, or per customer basis depending on exactly how close to the “edge” of the network you are located. Home routers have a particularly complex problem: “fair” should probably be measured by “air time” to individual stations, rather than bytes, and may also need to enforce bandwidth allocation as well (e.g. for guest networks). AQM is also “interesting”: to keep total latency low when there are multiple active stations, AQM will need to be run across multiple queues to these active stations.

Diffserv & Classification

Even with both AQM and “fair” queuing, Diffserv (and specific classification tricks) are still necessary. Using diffserv immediately solves the BitTorrent problem covered in another blog post: AQM, by ensuring latency is kept reasonably low, defeats Ledbat’s attempt to stay out of the way of TCP traffic.

Applications like BitTorrent (or, for that matter, sharing links with services that you may provide from home), may use many (even hundreds) of flows. For some applications (not BitTorrent!), these may even be short flows, and therefore compete strongly with interactive applications, such as web surfing. Without a “hint” that these particular services should be queued at higher, or lower priority than your interactive use, you cannot prioritize your link’s usage properly. Whether this “hint” is via Diffserv, or via port numbers is not the issue here; one way or the other we need to both have the intention of the traffic (e.g. scavenger, interactive, real-time sensitive), and properly handle the situation. Fair queuing by itself, while very helpful, does not solve this problem, particularly for real time traffic.

Some traffic is really, really time sensitive: but may not have an assigned port number: diffserv marking handles this case nicely. To meet real time application performance (e.g. VOIP) on a busy home or conference wireless network we need to be very careful, and use facilities such as 802.11e QOS queues to minimize latency. Other applications (e.g. backup) may be very happy to just scavenge bandwidth. Such applications need to be able to be deployed easily without expecting everyone to update their network environment, nor users to visit their home routers and set up explicit port rules. Again, diffserv marking handles this case very nicely.

Some will say diffserv is not deployed: but this belief is incorrect. The gaming industry noticed that Linux’s PFIFO_FAST queue discipline (which is the Linux default, and therefore the default in most of today’s home routers) honored diffserv marking and are using it today to improve real time performance. Some SIP ATA adapters also implement diffserv marking.

Diffserv has an Achilles heel: if the user’s do not have control over whether the diffserv marking is honored in the home router (“diffserv domain” in its terminology), vendors and software may “game” diffserv to the point of uselessness. Home routers must have facilities to detect diffserv marking both so users can control it’s use, and to enable push-back on software and hardware vendors who abuse diffserv’s intent, that a network owner be able to control their own network.

Flies in the Ointment

Unless bufferbloat is fixed (by deploying of CoDel), to achieve even mediocre latency today you must severely bandwidth shape broadband service, which also defeats features like Comcast’s Powerboost. Good utilization of bandwidth that you’ve already paid good money for is impossible until those links are debloated. But AQM by itself can’t solve transient bufferbloat at all at a bottleneck.

The line rate bursts of packets arrive at the broadband head-end, and are typically dumped into a single queue (which today suffers badly from bufferbloat). SP’s provision their telephony and possibly other services onto separate queues, to which you, as a customer, have no access. As discussed in another blog article, ISP’s (unintentionally though I believe it was) currently therefore have a fundamental advantage over others in providing many new Internet services. If you care about innovation in the Internet, you must care about this problem deeply.

Diffserv was designed before current broadband systems deployed. Broadband effectively “split” the responsibility for the network between the ISP and the user; you do not have full control over your “diffserv domain”. You may have control of what order packets are provided upstream, but downstream, you don’t (since that occurs in the ISP). And since the broadband link is also often the bottleneck link, combined with the technology changes I noted in the introduction, we have a fundamental issue. How does the user regain control of incoming traffic and therefore their own network, and prioritize the traffic properly, whether by port number or otherwise?

There are (at least!) two possible solutions to this problem.

1. build some protocol by which a home router can communicate to the broadband head end it's intentions. This seems like a lot of work; it is related to the work the IETF Port Control Protocol working group has underway.

2. Andrew McGregor suggest that the broadband head-end, by observing how upstream traffic is marked with diffserv marking, could invert the process in the broadband head-end going downstream. Monitoring flows is no longer the issue it once was, and this idea should be explored by those expert in that area.

Other ideas are welcome!

That there is a single queue on most broadband links available is clearly broken. You would like to be able to guarantee other traffic cannot interfere with VOIP, gaming or music playing, for example. A home router can work around this problem upstream to a good extent, but downstream, as outlined above, not so much.

Whether diffserv marking should have any affect inside the ISP's network (the ISP's diffserv domain) is outside this discussion, which is entirely about using diffserv and multiple queues in the broadband devices to help fix the queuing problems at the broadband edge, where the queuing problems are today most acute.

That broadband technology often already has support for these queues (e.g. DOCSIS flows) that ISP's can use exclusively for their services, is galling. That you have no way having these queues, even for pay, without buying the ISP's service is a fundamental network neutrality issue, in my view dangerous to the Internet's long term innovation and health.

Summary and Conclusions

CoDel, fair queuing, and and diffserv and conventional classification comprise the fundamental building blocks for a reliable low latency Internet.

A huge amount of engineering and deployment work remains. It is vital to understand that there is no single "magic bullet" to drain our current swamp. To achieve the goal of a low latency, predictable, reliable behavior, high performance Internet, however, all of the techniques above all need to be brought to bear.

This is a "systems integration" problem of first magnitude.