acmqueue

Case Study

BufferBloat: What's Wrong with the Internet?

A discussion with Vint Cerf, Van Jacobson, Nick Weaver, and Jim Gettys

Internet delays are now as common as they are maddening. That means they end up affecting system engineers just like all the rest of us. And when system engineers get irritated, they often go looking for what's at the root of the problem. Take Jim Gettys, for example. His slow home network had repeatedly proved to be the source of considerable frustration, so he set out to determine what was wrong, and he even coined a term for what he found: bufferbloat.

Bufferbloat refers to excess buffering inside a network, resulting in high latency and reduced throughput. Some buffering is needed; it provides space to queue packets waiting for transmission, thus minimizing data loss. In the past, the high cost of memory kept buffers fairly small, so they filled quickly and packets began to drop shortly after the link became saturated, signaling to the communications protocol the presence of congestion and thus the need for compensating adjustments.

Because memory now is significantly cheaper than it used to be, buffering has been overdone in all manner of network devices, without consideration for the consequences. Manufacturers have reflexively acted to prevent any and all packet loss and, by doing so, have inadvertently defeated a critical TCP congestion-detection mechanism, with the result being worsened congestion and increased latency.

Now that the problem has been diagnosed, people are working feverishly to fix it. This case study considers the extent of the bufferbloat problem and its potential implications. Working to steer the discussion is Vint Cerf, popularly known as one of the "fathers of the Internet." As the co-designer of the TCP/IP protocols, Cerf did indeed play a key role in developing the Internet and related packet data and security technologies while at Stanford University from 1972-1976 and with DARPA (the U.S. Department of Defense's Advanced Research Projects Agency) from 1976-1982. He currently serves as Google's chief Internet evangelist.

Van Jacobson, presently a research fellow at PARC where he leads the networking research program, is also central to this discussion. Considered one of the world's leading authorities on TCP, he helped develop the RED (random early detection) queue management algorithm that has been widely credited with allowing the Internet to grow and meet ever-increasing throughput demands over the years. Prior to joining PARC, Jacobson was a chief scientist at Cisco Systems and later at Packet Design Networks.

Also participating is Nick Weaver, a researcher at ICSI (International Computer Science Institute in Berkeley where he was part of the team that developed Netalyzr, a tool that analyzes network connections and has been instrumental in detecting bufferbloat and measuring its impact across the Internet.

Rounding out the discussion is Gettys, who edited the HTTP/1.1 specification and was a co-designer of the X Window System. He now is a member of the technical staff at Alcatel-Lucent Bell Labs, where he focuses on systems design and engineering, protocol design, and free software development.

VINT CERF What caused you to do the analysis that led you to conclude you had problems with your home network related to buffers in intermediate devices?

JIM GETTYS I was running some bandwidth tests on an old IPsec (Internet Protocol Security)-like device that belongs to Bell Labs and observed latencies of as much as 1.2 seconds whenever the device was running as fast it could. That didn't entirely surprise me, but then I happened to run the

same test without the IPsec box in the way, and I ended up with the same result. With 1.2-second latency accompanied by horrible jitter, my home network obviously needed some help. The rule of thumb for good telephony is 150-millisecond latency at most, and my network had nearly 10 times that much.

My first thought was that the problem might relate to a feature called PowerBoost that comes as part of my home service from Comcast. That led me to drop a note to Rich Woundy at Comcast since his name appears on the Internet draft for that feature. He lives in the next town over from me, so we arranged to get together for lunch. During that lunch, Rich provided me with several pieces to the puzzle. To begin with, he suggested my problem might have to do with the excessive buffering in a device in my path rather than with the PowerBoost feature. He also pointed out that ICSI has a great tool called Netalyzr that helps you figure out what your buffering is. Also, much to my surprise, he said a number of ISPs had told him they were running without any queue management whatsoever—that is, they weren't running RED on any of their routers or edge devices.

The very next day I managed to get a wonderful trace. I had been having trouble reproducing the problem I'd experienced earlier, but since I was using a more recent cable modem this time around, I had a trivial one-line command for reproducing the problem. The resulting SmokePing plot clearly showed the severity of the problem, and that motivated me to take a packet-capture so I could see just what in the world was going on. About a week later, I saw basically the same signature on a Verizon FiOS [a bundled home communications service operating over a fiber network], and that surprised me. Anyway, it became clear that what I'd been experiencing on my home network wasn't unique to cable modems.

VC I assume you weren't the only one making noises about these sorts of problems?

JG I'd been hearing similar complaints all along. In fact, Dave Reed [Internet network architect, now with SAP Labs] about a year earlier had reported problems in 3G networks that also appeared to be caused by excessive buffering. He was ultimately ignored when he publicized his concerns, but I've since been able to confirm that Dave was right. In his case, he would see daily high latency without much packet loss during the day, and then the latency would fall back down again at night as flow on the overall network dropped.

Dave Clark [Internet network architect, currently senior research scientist at MIT] had noticed that the DSLAM (Digital Subscriber Line Access Multiplexer) his micro-ISP runs had way too much buffering—leading to as much as six seconds of latency. And this is something he'd observed six years earlier, which is what had led him to warn Rich Woundy of the possible problem.

VC Perhaps there's an important life lesson here suggesting you may not want to simply throw away outliers on the grounds they're probably just flukes. When outliers show up, it might be a good idea to find out why.

NICK WEAVER But when testing for this particular problem, the outliers actually prove to be the *good* networks.

JG Without Netalyzr, I never would have known for sure whether what I'd been observing was anything more than just a couple of flukes. After seeing the Netalyzr data, however, I could see how widespread the problem really was. I can still remember the day when I first saw the data for the Internet as a whole plotted out. That was rather horrifying.

NW It's actually a pretty straightforward test that allowed us to capture all that data. In putting together Netalyzr at ICSI, we started out with a design philosophy that one anonymous commenter

later captured very nicely: "This brings new meaning to the phrase, 'Bang it with a wrench.'" Basically, we just set out to hammer on everything—except we weren't interested in doing a bandwidth test since there were plenty of good ones out there already.

I remembered, however, that Nick McKeown and others had ranted about how amazingly over-buffered home networks often proved to be, so buffering seemed like a natural thing to test for. It turns out that would also give us a bandwidth test as a side consequence. Thus we developed a pretty simple test. Over just a 10-second period, it sends a packet and then waits for a packet to return. Then each time it receives a packet back, it sends two more. It either sends large packets and receives small ones in return, or it sends small packets and receives large ones. During the last five seconds of that 10-second period, it just measures the latency under load in comparison to the latency without load. It's essentially just a simple way to stress out the network.

We didn't get around to analyzing all that data until a few months after releasing the tool. Then what we saw were these very pretty graphs that gave us reasonable confidence that a huge fraction of the networks we had just tested could not possibly exhibit good behavior under load. That was a very scary discovery.

JG Horrifying, I think.

NW It wasn't quite so horrifying for me because I'd already effectively taken steps to mitigate the problem on my own network—namely, I'd paid for a higher class of service on my home network specifically to get better behavior under load. You can do that because the buffers are all sized in bytes. So if you pay for the 4x bandwidth service, your buffer will be 4x smaller in terms of delay, and that ends up acting as a boundary on how bad things can get under load. And I've taken steps to reduce other potential problems — by installing multiple access points in my home, for example.

JG The problem is that the next generation of equipment will come out with even larger buffers. That's part of why I was having trouble initially reproducing this problem with DOCSIS (Data over Cable Service Interface Specification) 3.0 modems. That is, because I had even more extreme buffering than I'd had before, it took even longer to fill up the buffer and get it to start misbehaving.

VC What I think you've just outlined is a measure of goodness that later proved to be exactly the wrong thing to do. At first, the equipment manufacturers believed that adding more buffers would be a good thing, primarily to handle increased traffic volumes and provide for fair access to capacity. Of course, it has also become increasingly difficult to buy a chip that doesn't have a lot of memory in it.

NW Also, to the degree that people have been testing at all, they've been testing for latency or bandwidth. The problem we're discussing is one of latency under load, so if you test only quiescent latency, you won't notice it; and if you test only bandwidth, you'll never notice it. Unless you're testing specifically for behavior under load, you won't even be aware this is happening.

VAN JACOBSON I think there's a deeper problem. We know the cause of these big queues is data piling up wherever there's a fast-to-slow transition in the network. That generally happens either going from the Internet core out to a subscriber (as with YouTube videos) or from the subscriber back into the core, where a fast home network such as a 54-megabit wireless hits a slow 1- to 2-megabit Internet connection.

In Jim's case, a Linux machine defaulted to about a megabyte of data in transit, which amounts to several seconds' worth of delay over a 2-megabit line. That's the way Linux ships, and that's the way the YouTube servers come configured out of the box. Unless those things get reconfigured, they send a megabyte.

All that data flows through the network until it piles up at the subscriber link. If some manufacturer were to produce a home router with a very small buffer, data would get thrown away, and that manufacturer would very quickly get a reputation for making crummy routers. Then a competitor would be sure to say, "That router has a 90 percent loss rate, so you want my box instead since it's got enough memory to avoid losing anything at all." That's how we end up with marketplace pressure on manufacturers to put bigger and bigger buffers into their equipment.

JG This is not just a phenomenon at the broadband edge. In the course of pulling on a string to figure out why my home router was misbehaving so badly, I discovered that all our operating systems—Linux, Windows, and Macintosh alike—also are guilty of resorting to excessive buffering. This phenomenon pervades the whole path. You definitely can't fix it at any single location.

VJ Think of this as the network version of "Mutually Assured Destruction." Unless you can get everybody to cooperate on reducing the buffers for every piece of equipment, the right strategy for any provider is to increase its buffer. You can make your Wi-Fi work better if you put more buffer into it. That's also the way to make your router work better and to make your end system work better.

JG I would argue that point. You can make things "better" in some very limited ways. But if all you're measuring is bandwidth, then you don't see all the havoc you're creating by contributing to the failure of even routine network services such as lookups of IP addresses. Those kinds of failures are now commonplace in part because these big buffers and the long delays they're causing are working at odds with the underlying protocols.

VJ You're preaching to the choir, Jim.

■■

Excess buffering largely defeats TCP's congestion-avoidance mechanisms, which rely on a low level of timely packet drops to detect congestion. With buffering, once packets reach a chokepoint, they start to queue. If more packets come in than can be transmitted, the queue lengthens. The more packets in the queue, the higher the latency. Eventually packets are dropped, notifying the communications protocol of the congestion.

Bufferbloat allows these queues to grow too long before any packets are dropped. As a result, the buffers become flooded with packets and then take time to drain before they can allow in any additional packets. All the end user sees of this is slowed response. Services that require low latency—such as network gaming, VoIP, or chat programs—can slow to the point of becoming unusable.

Also, because of the widespread nature of the problem, there are growing concerns that the stability of the Internet itself could be compromised.

VC The key issue we've been talking about is that all this excessive buffering ends up breaking many of the timeout mechanisms built into our network protocols. That gets us to the question of just how bad the problem really is and how much worse it's likely to get as the speeds out at the edge of the net continue to increase. I'd assume the problem is only going to get worse as the buffers start to fill up even faster.

NW No, actually, that will make things better. While it's true that the larger problem has to do with buffers that are too big, if you look at your own situation in terms of time rather than capacity and you can manage to double the performance on your link, then you ought to be able to cut your delay by half.

VC Good point. The issue here seems to revolve around how long it takes for the buffer to fill up. If

it's on a slow line, it's going to take forever, and then the buffer is going to hold all that stuff and permanently impair the round-trip time.

JG The fundamental observation, however, is a bit subtler in that there actually is no single right size for a buffer. Think about wireless, for example, where you can easily have two or three orders of magnitude variance in bandwidth. We have broadband systems that can run as much as 100 megabits or more, but at the low end, they provision only 10 megabits or less. That's an order-of-magnitude swing right there. No single static size makes sense all the time.

NW What's even worse is that if you think about it in terms of time—such that you size the buffer for X milliseconds rather than X kilobytes (which you can do with just a little more logic)—that still doesn't entirely do the job. It gives you a 90 percent solution because that will still induce some latency under load. It will be bounded just to the point where it's about the maximum latency—or about the minimum latency under load for a simple queue if you're still looking to achieve full-TCP throughput.

VJ It won't even work with Jim's Wi-Fi example. Move your Wi-Fi receiver two inches and your throughput can go from 100 megabits down to a megabit, which means your queue delay will go up by a factor of 100. There's just no static buffer-management strategy—and no static buffer size—that will work in that situation.

VC That suggests anything that's going to work will have to be dynamic and adaptable. But it also raises the odd problem that your device at the edge of the net might end up communicating with devices at various locations in the network with different path dynamics. That gets to a question about how you go about observing flows and distinguishing between them. Is that something that would become necessary if we were to move at all toward dynamically adapting the buffer *delay* (if I might be permitted to use that term rather than buffer *size*) for all the various flows you might be trying to engage with at any given time?

VJ No. What you've actually got moving around on the Internet are packets, and it's hard to tell which packets constitute a "flow." What we do know, however, is that the data piles up wherever there's a fast-to-slow transition, and nowhere else. The router on the upstream end of that transition knows there's a big honking queue piling up. If you can somehow mitigate the queue at the point where you can actually see it forming, or at least signal the endpoints about the queue they're creating and tell them they need to deal with that, you don't need to do any sort of complicated flow analysis.

VC Could this queue show up anywhere on the net, and not just at the edges?

VJ Theoretically, yes. But I think that theoretical conclusion has impeded a lot of work in this area because it leads people to think about the problem as potentially being anywhere. Yet the economics of the Internet tends to ensure a very high-bandwidth core where we aggregate traffic from a lot of low-bandwidth links. Remember also that the individual subscribers from whom you're aggregating aren't correlated, so as you add together their different traffic streams, the traffic tends to get smoother. This is the point Nick McKeown made a few years ago when he was talking about excessive buffering in the core. There's really no need for humongous buffers there since you're dealing with these huge aggregates, resulting in traffic that's pretty smooth.

NW The other issue regarding the Internet core is that the buffer gets to be really expensive once you start talking about 10-gigabit-plus links. Interestingly, if you talk to the Internet2 folks, their complaints about core routers have to do with there not being enough buffering, since their test

involves single-stream TCP or few-stream TCP throughput on 10-gigabit links. For that, you actually do need a good amount of buffering, but that's just not economically feasible right now for a lot of those core routers.

VJ But that is something else that has impeded progress here—all that R&D aimed at maximizing the bandwidth between supercomputer centers. Remember that most of the network research in the U.S. has been funded so far by the NSF (National Science Foundation) with the goal of demonstrating very high bandwidth between academic institutions. This is great for the .01 percent of the Internet community that has those links available to them, but it doesn't do much good for the billion other people in the world who have 2-megabit or less.

We've put a lot of effort into protocol, router, and algorithm development that makes it possible for a single TCP to saturate a 40-gigabit link, but we haven't put anything even remotely like that effort into producing usable cellphone data links, DSL links, or home-cable links. That's where the really hard problem is because it requires that you start to think about how the buffering actually works and how it turns into latency.

VC We actually are circling around the fact that this is a hard problem because of the environment. We should talk about whether we have a crisis on our hands right now and, if so, what's to be done about it.

JG We don't even know whether the current state of things is going to remain stable. I get very nervous when I consider the spasmodic behavior of just a single TCP flow, with all these wild excursions going on much of the time. It certainly seems to me that our current situation is not at all static.

We're seeing many new applications—such as streaming video and off-site system backup, for example—that are far more likely to saturate the links. All these applications are starting to become much more routine at the same time that Windows XP is retiring. That's actually a significant milestone because Windows XP didn't implement window scaling, so it tended not to saturate links in quite the same way as anything released more recently. Maybe I'm being paranoid, but I really don't like what I'm seeing right now.

NW Does any major TCP stack exist now that doesn't do proper window scaling and so doesn't saturate everything up to a near-gigabit link?

JG I think they all do window scaling now—at least anything more recent than Windows XP. Windows, Mac OS X, and Linux are all very happy to run up to multigigabit speeds.

VJ But the packet output isn't driven by window scaling. Window scaling just gives you room to expand the window, but it expands only if the system defaults to a large window. For years, the default window was sized to fit the roughly 100-millisecond/1-mbps TCP paths you typically found back then. At some point over the past five or six years, people have decided that the common path ought to be considered a gigabit or more. As a consequence, the buffer size has been bumped up to more than a megabyte.

NW Unfortunately, that's almost exactly the case. If you look at my house, in fact, going from my computer to the file server, I've got a gig link at 10-plus milliseconds.

VJ Have you done the experiment of varying the window size and then looking at throughput to your file server as a function of the window size to see whether or not you actually need the megabyte?

NW No, I have not.

VJ Well, I have, and the bandwidth flattops at an eight-packet window. I'm talking about a Linux file server running a very current kernel. Typically, the situation is that you're in a home environment with large bandwidth and very short delays, or you're going out over the Internet where you're going to encounter small bandwidth and much longer delays—often 10 times or more longer than what you have at home. You could pick a window size that was appropriate for the Internet—something like 100 kilobytes, which is 10 kilobits into 100 milliseconds—and that would work more than adequately for your home environment as well.

But that just isn't what we see anymore. Instead, we ship several megabytes, and that's when you hear the complaint: "Oh my God, I'm seeing a second of delay." Well, yes, of course. That's how your system is configured.

JG What's more, you find that sort of thing in several different places. The operating system itself may have problems. If you look at the device drivers, you'll find they've also grown some very large buffers of their own. Basically, whenever you look inside any of our current operating systems, you'll find a recurring theme of bufferbloat at multiple layers.

That's actually part of the problem: people are thinking of this as a system of layers, where they don't have to worry about how everything actually works, either above them or below them. They aren't really thinking through all the consequences of what they're doing—and it shows!

■■

If we do indeed face an impending crisis, what's to be done? Bufferbloat presents a hard problem, with no single right solution. And making headway to mitigate the current situation will require a multipronged effort that involves ISPs, operating system implementers, application vendors, and equipment manufacturers.

Since network hardware is clearly a primary culprit, it would seem that reducing buffer size is all that ought to really be necessary. But buffer size cannot be configured on most routers and switches. Nor can buffer size be static. It must be sized dynamically, meaning that both hardware and software modifications will be required.

There also are some new queue-management algorithms that are currently being studied. Although the classic RED queue-management approach serves as a starting point for some of this research, it is not itself equal to the present challenge. Still, efforts to create an improved version of RED are already under way.

VC It certainly sounds as though equipment with smaller buffers would help keep a lot of the worst cases from occurring simply because there wouldn't be enough buffer space to create as much of a problem. The other possibility, I guess, would be to provide a way to discard backed-up packets and send a summary report back to the source of congestion whenever you see queues building up. But that gets to this awkward problem of not knowing what the source of the congestion might be unless you're monitoring source/destination pair traffic or something along those lines. Given that we now seem to know something about the nature of the problem, can you speculate a little on what, if anything, might be done to attack it?

VJ The approach you just described was the basic idea behind RED: whenever you've got a big queue, you should notify one or more of the endpoints that there's a problem so they will reduce the window, which in turn will reduce the queue. That provides an alternative to tracking flows, which at best is a very state-intensive problem. And with IPv6, it can turn into an impossible problem because someone who doesn't want to be policed will just spread traffic over a zillion addresses, and there's no way to detect whether those are separate flows or all coming from a single user.

Instead of trying to infer the flows, you might just pick a uniformly distributed random number and, when the packet with that number comes up, mark or drop it. So if, say, 90 percent of the packets come from one source, you would have a 90 percent probability that the dropped packet is from that source.

NW There's also a cool little idea out of Case Western Reserve University for handling queue management remotely. Basically, their observation is that if you are on the path that all the traffic passes through—for example, the NAT (network address translation) that's separate from the cable modem—you can track delay increase. Once the delay increase gets above a threshold of, say, 100 milliseconds, you can start using the RED approach for queue management or whatever it is you need to do to get the traffic to back off. The problem with this, however, is that it provides only an 80 percent solution for the telephony crowd.

VJ If I understand what is being proposed there, you're putting in an appliance to measure delay, but if you're looking at aggregated traffic, you don't know how it de-aggregates.

NW That's partially why this proposal focused on the home gateway. Basically, there aren't all that many large flows going through your NAT box at any point in time.

VJ But if you're on the gateway, you've already got the queue, so you don't need any state at all. The queue is its own state.

JG One of the reasons queue management is even being considered for home networks is because so many of the broadband networks don't provide any. I recently read a paper that indicated as much as 30 percent of residential broadband networks run without any queue management whatsoever. That leaves it to the home routers to handle the problem. Then we've got the additional complication of the huge dynamic range presented by wireless. Van has told me that classic RED cannot be used to address that particular problem.

VC I'm trying to think about how we might start moving toward a solution. It seems there are at least three different pieces to consider. First, there's the party who's suffering the effects of bufferbloat in, say, a residential setting. Then there's the service provider that, I believe, really wants to provide a better quality of service for its customers. Finally, there are all those equipment and software vendors that might be persuaded to address the problem if they thought it would make the ISPs and end users happier.

The question is: under what conditions might we get all those parties to cooperate? Or is this going to remain largely a research problem?

NW Well, there have been a couple of cases where the application vendors concluded they were causing too much damage and therefore started making changes. BitTorrent is the classic example. It is shifting to delay-based congestion control specifically to: (a) be friendlier to TCP because most of the data carried by BitTorrent really is lower-priority stuff; and (b) mitigate the "you can't run BitTorrent and Warcraft at the same time" problem. So, there's some hope.

JG Right. There are also a number of queue-management algorithms other than classic RED that we're ready to start experimenting with. Van is working with Kathie Nichols [founder and CEO of Pollere Inc., specializing in network architecture and performance] on something they're calling RED Lite, and we hope to start playing around with that soon. There's also a queue-management algorithm called SFB (Stochastic Fair Blue) that's been implemented in Linux and we're now getting ready to try it out.

A number of us are trying now to set up OpenWrt (a Linux-based program for embedded devices)

so we can build, at least as a proof of principle, a home router that's well behaved and might even function reasonably well.

VJ There's also a separate piece that's important to understand. Because there is no static answer to this problem, there's no answer that's right for all environments, and any answer needs to evolve over time. People need some no-brainer, easy-to-use measurement tools that say, "Your home router's delay is getting really bad."

For example, one group that's massively affected by this bufferbloat problem is YouTube. It's in a position to observe the congestion at the upstream end of the residential links. It would be really simple to add instrumentation to their servers to measure the buffering delay on each video sent out and then associate that with the destination IP address. That could be viewed at the individual level (via the player), the prefix level, or the network level to provide some understanding of where the delays are occurring.

JG The only place where the problem really matters is where you hit that point of bandwidth transition. Anywhere else, the excessive buffering might not matter at all, apart from being a waste of memory. But out at the edge of the network, we're certainly getting hurt pretty badly right now. That applies to the operating systems, the home routers, the broadband gear you find in people's homes, and the broadband gear that's installed at the ISP end. So, among those four places, we certainly ought to focus on producing some simple tools that can help people identify where their problems are coming from.

VJ I have four family members in my household. It used to be that none of us watched video over the net, but now everybody watches video or plays games over the net, or tries to do both at the same time. It only takes one person watching YouTube while somebody else is watching Netflix to completely saturate the Internet-to-home link.

That's not the fault of YouTube or Netflix or anybody in the household. It's just the way people are using the Internet now. This is a problem we need to fix so that people can continue using the Internet the ways they've grown accustomed to.

■■

The bufferbloat problem is likely to worsen as the demand grows for Internet-intensive activities such as streaming video and remote data backup and storage, with the online user experience bound to deteriorate as a consequence. The problem is most visible at the edge of the network, but evidence of it can also be found in the Internet core, in corporate networks, and in the boundaries between ISPs. Besides calling for simple tools that people can use to find and measure bufferbloat, active queue management for all devices is also in order. The best algorithm for the job remains to be determined, however.

VC A comment came up earlier about using carrier-grade NATs to track delay increase. It sounds to me, though, that even if they were situated where congestion could be detected, there might still be a problem. That is, if you're downstream from the congestion and your equipment has too much buffer in it, the problem is going to persist. Is that a correct observation?

VJ No, because you hit the problem wherever there's a fast-to-slow transition, which is going to occur wherever traffic moves from the high-bandwidth Internet to the subscriber link.

VC So a carrier-grade NAT isn't necessarily going to help unless that transition is up ahead of you in the stream?

VJ Exactly right. There are some really complicated things you can do to figure that out, but that's really difficult and nowhere near 100 percent reliable. What it really comes down to is that you want to solve the problem wherever the big queue is.

VC This might be a good time to raise the more general question of why a control system doesn't work well unless it happens to have just exactly the right information available to it—that is, whatever information aligns best with that system's core algorithm. The sense I'm getting from this conversation is that it's proving difficult to get all the information that would be helpful in figuring out what the right buffer size ought to be at any moment.

VJ That's true, and I was a major contributor to the misinformation. I helped put people on the wrong track with RED, which attempts to extract some information from the average queue size, but it turns out there's nothing that can be learned from the average queue size.

There was some later work on algorithms such as BLUE from the University of Michigan and a successor that's coming from the Hamilton Institute in Ireland, both of which look at the time you've had a queue above threshold, which is a much more reliable indicator than anything you can obtain from looking at the queue size.

But there's a problem. You need to have about 100 milliseconds of queue for your connection to get started. You deliver a window's worth of packets to the bottleneck, which temporarily creates a queue, the queue drains into the wire, and everything runs smoothly with no queue. If you didn't have that queue, you would get pathetic throughput and massive packet loss, so this is good queue. But when you dump a second's worth of data into a 100-millisecond path and end up with a second's worth of queue, that sits there forever. That's bad queue: it's not doing you any good; it's just generating delay.

What's needed is an algorithm that can tell the difference between good queue and bad queue. That requires a dynamic analysis that turns out to be fairly easy. The time behavior of the queue distinguishes between good and bad since the good queue goes away, but the bad queue doesn't. If you just track the minimum over time, everything below that minimum can be considered bad queue, while everything above it qualifies as good queue.

VC It's heartening to hear you say that you can apply fairly straightforward dynamic analysis techniques to distinguish between good queue and bad queue at that point in the network where the data-rate transition occurs. There might be hope for designing a more adaptable edge device, at least for residential settings.

VJ I'm fairly optimistic about that.

VC Let us suppose for the sake of argument that the equipment makers come to recognize there is a problem and even go so far as to indicate a willingness to cooperate, and that the ISPs, moreover, say they're prepared to acquire the appropriate equipment. Would it then be possible or appropriate to incorporate in those devices a measurement capability to detect any subsequent problems, or at least to provide some sense of how well the queue-management algorithms are doing once they've been deployed?

VJ We definitely want to have a self-measurement capability, but it probably shouldn't reside in the devices. It's the end users who suffer the pain of those problems and they need to be able to pin down the cause. It's pretty straightforward to give tools to end users. TCP's normal round-trip time measurements show the delay your traffic is experiencing. If that delay is excessive, you can use Netalyzr or a tool such as Pathchar to determine where it's occurring.

JG Steve Bauer [of MIT] and I had a conversation with Doug Suttles at Ookla, which is the parent company for Speedtest.net, regarding this very point. We all agreed we need end-user tools that point out where in the path the bloated buffers are. Doug is very interested, so hopefully we'll be able to do something about that.

One other point I want to make is that while the worst of the problems appear to be out at the edge, we know there are bufferbloat problems in other places as well, such as the boundaries between various ISPs, where you sometimes find congestion. I also see problems wherever IPsec tunnels appear to terminate, and I think many corporate networks might be running right now without any queue management whatsoever. We're going to find this becoming much more of a problem as more and more of the machines online become capable of trivially saturating the corporate network links.

Some things are being done to provide relief over the very short term by allowing for dynamic adjustments in the buffering for certain devices in response to latency or bandwidth criteria instead of just giving people a static-sized buffer and leaving it at that.

The cable industry, for example, is making some changes along these lines even now to mitigate the severity of the problem. Still, the proper solution that's really called for is to have real queue management everywhere.

VC I wonder whether it's possible to persuade the equipment manufacturers that it's in their best interests to change the way buffers are used—specifically to allow more dynamics in the scaling of the end-user buffer size.

What you've done so far is to illustrate the nature of the problem and the conditions under which it occurs, and that ought to at least serve as an important starting point. This discussion makes it pretty clear just how hard a problem this is. It's also an important one to solve because the experiences people have with the network are unlikely to end up being happy ones unless we find a better solution than what we have right now.

LOVE IT, HATE IT? LET US KNOW

feedback@queue.acm.org

© 2011 ACM 1542-7730/11/1000 $10.00