

Will cheap RAM break the Internet?

A Fair Bit

An abundance of buffers hides the Internet's dirty little secret.

By Kurt Seifried

In this article, I'm not going to talk about an emerging technology (don't get me wrong, I love new technology) but about something even more interesting: An emergent behavior that was never expected: bufferbloat.

Bufferbloat is not a recent phenomenon; however, it has only recently been uncovered and understood, and developers will likely be grappling with it for some time. Additionally, this problem, if left unchecked, will make the Internet painfully slow to use, greatly reducing the availability of services. Remember, availability is one of the three legs of the AIC triad (along with integrity and confidentiality).

A Brief History

People started making networking equipment back when memory was expensive, so most networking gear didn't have much in the way of memory or buffers. Data came in, and it was promptly sent out. If it couldn't be sent out, it was usually dropped, and the device might send a congestion notification back to the original system.

Then memory got cheap. In an effort to make networks faster and more reliable, networking equipment acquired memory and buffers. This was great for a while; if a link was temporarily congested, data could be queued and sent once the congestion cleared. In fact, it worked so well that folks started adding memory and buffers at multiple levels of the network.

Your operating system has network buffers; the device driver moving packets through your network card has a buffer, and the physical network card itself might have a buffer. If you add buffers at every system along the way, you end up

with a whole lot of buffers, all behaving differently (some will drop data if they get full, some are so large that they almost never get full, etc.).

The result of all this bufferbloat is that suddenly the network no longer runs smoothly. In fact, it seems to be moving in fits and starts, working properly for several seconds and then suddenly slowing down.

The weird thing is, data – lots of data, in fact – is still coming through, but everything feels slow. Like a lot of emergent behavior, the basic rules governing the problem are simple, but the behavior that comes out of these “simple” rules is often horribly complex and difficult to understand.

Why Bufferbloat Affects Latency

Even when data still seems to be getting through, bufferbloat is affecting latency. The buckets are full, but you don't know that, so you keep sending data as

fast as you can, and the data starts getting dropped because the buckets are full. But, you don't know that, especially if you have multiple systems with buckets inline; it could take a while for the buckets in the systems near you to fill up.

This behavior has only recently become a significant problem, even though network equipment has had large buffers for some time. So what gives? One reason appears to be that people are using networks more heavily (e.g., using BitTorrent and streaming video online), so buffers are filling up.



KURT SEIFRIED

Kurt Seifried is an Information Security Consultant specializing in Linux and networks since 1996. He often wonders how it is that technology works on a large scale but often fails on a small scale.

The second reason is that vendors and service providers have deployed equipment with lots of buffers, especially on wireless networks where the cost of re-transmitting data is high (both in network terms and in battery usage on client devices).

Why Latency Matters

A lot of services require latency to be predictable (e.g., not be “jittery”), and many more services rely latency to be low. VoIP and video conferencing, for example, are pretty much useless with latency jitter – if even a few packets are delayed, it can mess things up. Multi-player games also rely heavily on low latency; if you’re trying to run around and shoot at people, but you’re 100 milliseconds slower than everyone else, it’s going to make the game a whole lot less fun.

If you are one second slower, it will make the game impossible. In the future, these types of interactive services will only become more common. If the latency between you and remote sites is slow or jittery, a lot of things simply will not work (like VoIP).

Measuring Network Latency

Although you have several ways to measure network latency, for long-term analysis, your best bet is SmokePing [1], which comes standard on most distributions. I won’t cover installation or configuration other than to say the default polling time is 300 seconds (five minutes), which is way too slow. If you want lots of data, set the polling time to 15 seconds with 10 pings (pings are sent at one-

BANDWIDTH VS. LATENCY

Bandwidth is the total capacity of the pipe, which determines how many kilobits, megabits, gigabits, or whatever you can shove through it per second, on average. The most critical word being “average.” Latency is how long it takes one bit of information to go from point a to point b. The result is that a network link might have a great-sounding “speed” (i.e., 975 Mbps), but it might have terrible latency (2 hours and 20 minutes), which is what you

get if it takes an hour to write a terabyte of data to a hard drive, 20 minutes to drive it somewhere, and another hour to read the data off of it.

Unfortunately, network providers only tell us what the bandwidth of a link is, and even then, they often hedge their bets with phrases like “up to” or simply lie. Few, if any, providers tell you the actual latency, and I suspect most don’t know with much certainty.

second intervals). The SmokePing program runs in the background as a daemon and has a nice web interface that makes exploring the data easy.

Recreating Bufferbloat Problems

Admittedly, I was a little skeptical at first that performance on saturated networks would be made worse by bufferbloat. In the case of my 50Mb down and 3Mb up cable connection, I reliably get ping times of 50ms (to seifried.org), and I was able to slow it to one second with a single high-speed upload (saturating my down link also worked, but not as well). The main tool I used was iperf [2], a network tool that comes on almost every Linux distribution as a package and measures network performance by sending as much data as fast as it can. With iperf, I was able to saturate my 3Mb up-link easily. What is especially interesting is that the default iperf setting of sending data for 10 seconds

```
iperf -c server
```

(Figure 1A) affected my network link, but not as badly as I’d feared. However running iperf for 120 seconds

```
iperf -c server -t 120
```

(Figure 1B) basically killed everything, with ping latency climbing from 50ms (to seifried.org) to more than 1,000ms in the span of about 15 seconds.

Latency stayed there until the test stopped. Just to make sure the results were legitimate, I repeated it a few times, and I got exactly the same results every time.

How To Fix Latency

Unfortunately, the solution for latency depends heavily on equipment manufacturers and operators, especially network providers.

As the consumers, we need to start measuring network latency and complaining when it gets bad. However, the providers need to start using things like active queue management [3], and they need to tune the buffers in their networks, which can be tricky because there are so many.

The best answer I have is to visit the Bufferbloat [4] website, which has a number of projects going on to measure and address this issue. A talk with slides explains the issue in depth, and I highly recommend you view it [5]. ■■■

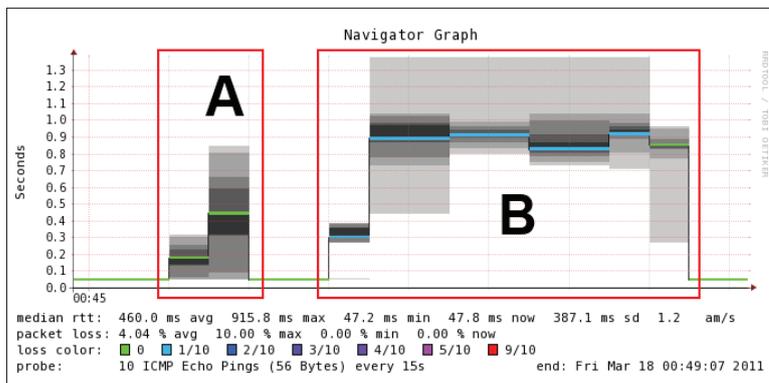


Figure 1: SmokePing test results when sending data for 10 seconds (A) and 120 seconds (B).

INFO

- [1] SmokePing: <http://oss.oetiker.ch/smokeping/>
- [2] iperf: <http://sourceforge.net/projects/iperf/>
- [3] Active queue management: http://en.wikipedia.org/wiki/Active_Queue_Management
- [4] Bufferbloat: <http://www.bufferbloat.net/>
- [5] Bufferbloat – Dark Buffers in the Internet: <http://mirrors.bufferbloat.net/Talks/BellLabs01192011/>