The hidden cause of slow Internet and how to fix it

•

By experimenting with pings and various levels of load on his Internet connection, he discovered that latencies were often four to 10 times larger than what should have been expected. He termed the phenomenon, “bufferbloat.” His conclusion was that critical data packets were trapped in buffers that were excessively large.

From the time Gettys made his observation and began to publicize it, researchers from companies such as Cisco and Google, standards groups like the IETF and major research universities have been investigating, testing, and writing about bufferbloat. We also conducted our own simple tests. Bufferbloat is real. What is not fully understood is the extent of its impact on the normal flow of Internet traffic.

So, who is most affected by this phenomenon?
Anyone who is actively browsing or using search engines. Also, anyone who is using real-time applications like voice or video. An example would be employees working from home, on the road in hotels or at Wi-Fi hot spots. Our research showed that hotels and Wi-Fi cafes are prone to very bad bufferbloat issues.

What kind of traffic is affected?
Traffic flowing on links which have high-bandwidth utilization in the opposite direction will deteriorate. Applications using small packets such as VoIP, DNS, and ARP can also suffer. The impact on VoIP will be increased latency and jitter. DNS queries may be returned in two to eight times the normal response time.

How could a problem affecting the operation of the Internet hide for such a long time?
There are three primary reasons. First, the issue is closely tied to how the TCP protocol operates and how network buffers are managed. Neither of these is broadly understood. Second, there is a widespread belief

that dropping packets in the Internet is always a bad thing. The truth is that dropping packets is absolutely essential to the proper operation of TCP. Third, there is a wide conviction that the way to eliminate nearly any deterioration in performance is to add bandwidth.

So, what exactly is bufferbloat?

In an attempt to reduce packet loss in the internet, network operators, developers, and engineers have increased the size of network buffers many times over. This increases latency but has little effect on throughput. Consequently, critical small packets such as those in VoIP, DNS, and TCP 'acks' can become trapped in the buffers behind much larger packets from file transfers and other bulk transfers, such as adaptive bit rate video.

There is a perception problem related to buffer management. Tests, white papers and even instructors often describe buffers as small chunks of memory. More often than not, buffers can hold hundreds and even thousands of packets at any instant.

And, they aren't just in the network devices. They are also in the protocol stack of the end station, the network card driver and every gateway in the path between the end stations.

What is bufferbloat's impact on TCP operation?

The vast majority of our network traffic uses TCP as the transport protocol. Understanding how TCP operates reveals why bufferbloat is a problem. When a TCP connection is established, there is a three-way handshake in which the sending and receiving TCP entities negotiate the parameters for the exchange, including initial sequence numbers.

Let's say an FTP server has been asked to transfer a large file. TCP typically begins its transfer by sending four segments and awaiting acknowledgement of their delivery. The usual acknowledgement policy is to send an `ack' after every other received segment.

When, the four segments are `acked', the receiver increases the send rate by sending eight segments and awaits acknowledgements. After acknowledgement of those segments, the send rate is increased to 16 and so forth.

This phase of delivery is referred to as slow start. The idea is to saturate the link with packets. However, at a level called the slow start threshold, the sender increases the rate more slowly by adding one segment at a time in each round, rather than doubling the rate.

Nevertheless, there will be a critical point at which the connection will be overloaded because a buffer will overfill. One or more packets will be dropped.

When the sender detects that this has occurred, it generally cuts its send rate in half and re-initiates slow start. Eventually, the TCP rate will adapt to the capacity of the circuit that is being used. This combined set of steps is known as the TCP congestion control algorithm.

So, how does bufferbloat interfere?

Let's consider a connection between a high-speed link and a low-speed link. This is a situation where buffers are considered critical. For example, suppose we have a 1Gbps to a residential gateway like a cable or DSL modem. Also, suppose the modem is connected to an ISP connection that provides 10Mbps down and 2Mbps up.

The FTP server will fill the buffer going into the fast connection more quickly than the egress rate into the slower link. It is the rate at which the acks return that ultimately determines the rate at which the sender can transmit.

However, if that buffer is large, two things can occur. First, if the buffer fills, the last packet to arrive is dropped. This is called tail drop. The ack that informs the sender of this drop will not be sent until the next packet (after the discard one) arrives and is declared out-of-order.

It could take considerable time for it to get through the large buffer. Some experiments we did with adaptive bit rate video showed that nearly 200 segments could be delivered before the sending station would retransmit the dropped segment.

Also, if there are multiple flows coming into it, the queue may evolve into a standing queue. That is, it may reach a steady-state in which there is a fixed or nearly fixed number of packets in the queue. If this amount is not enough to overfill the buffer, no packets are dropped and TCP congestion control is defeated. However, latency for all users of the buffer has increased.

Managing buffers

For some time, there has been an awareness that network queues should be managed. To add priority to certain traffic, the IP layer diffserv bits can be set to implement a policy that gives preference to certain types of traffic, such as network control or VoIP. They accomplish this by separating those priority traffic types into separate queues.

But, this does not eliminate bufferbloat. Some queues containing the non-prioritized traffic continue to have the problem of being too large. These often contain many large TCP segments. So, we still have the problem of the negative impact on the TCP congestion mechanism.

Several active queue management (AQM) techniques that have been introduced include RED (Random Early Discard) and WRED (Weighted RED). These were designed to discard packets when the buffer reached a critical level, but was not necessarily full. But these techniques were flawed and configuring RED proved to be difficult. Consequently RED and WRED are not widely implemented. What was needed was an automatic, never adjust method.

In 2012, Kathie Nichols and Van Jacobsen began to promote a technique called CoDel or Controlling Queue Delay. This method manages a queue by tracking the time a packet is in the queue, since the time of occupancy in the queue is really the crucial issue.

There are two critical parameters, interval and threshold. If an interval worth of packets have delays longer than target, packets are randomly dropped. Note that this technique does not depend on the size of the queue. Nor is it tail-drop.

Testing the procedure showed general better latency behavior than RED and far better results. This was especially true with wireless access links. Also, the technique promised to be easy to embed in hardware.

The next recommendation for mitigation of bufferbloat came from Dave Taht, Eric Dumazet, Jim Gettys and a few others. Called fq-codel, it is intended to provide a more uniform impact on the various flows through the queue. Even Kathie Nichols and Van Jacobson are advocating the use of fq-codel.

This method separates the queue into 1024 sub-queues by default. Then it randomly assigns each new flow to a separate queue. Within each sub queue, Codel is applied to help with TCP congestion control. The de-queue policy is based on DRR (Deficit Round Robin).

What do Codel and fq-codel do?
First, they make sure that TCP congestion control functions as designed. Second, by mixing the packets in the queues, small critical packets such as DNS responses and TCP acks don't get trapped in large queues. In other words, it makes the treatment of large packets and small packets more equitable. Considerable research has demonstrated the benefits of using fq-codel. In fact, it's in the latest distributions of Linux.

Where do we go from here?
Unfortunately, awareness of bufferbloat is still not widespread. We need more network operators and users to run readily available tests like ICSI Netylyzr and the test at www.DSLReports/speedtest/.

Then, if you detect a significant bufferbloat issue, you have several alternatives:

1. Change your access hardware to devices using a new distribution of Linux containing fq-codel. Make sure the feature is turned on.

2. Place a device between your computer and the gateway/router that has the fq-codel capability turned on. That will limit the use of the router's large buffers.

3. If all else fails, apply rate limiting to uplinks and download links to something just under their rated capacity. This will help to eliminate large standing buffers. It will cost you a small decrease in throughput under light load. However, it should dramatically improve critical flows such as DNS, ARP, and TCP acknowledgements.

There are several vendors keenly interested in mitigating bufferbloat. Cisco, in partnership with Comcast, has embraced a queue management technique call PIE (Proportional Integral controller Enhanced) principally developed by Cisco Distinguished Research Engineer Rong Pan.

Time-Warner Cable seems well versed on the topic and is prepared to take steps towards alleviating bufferbloat. Actiontec, a major supplier of residential gateways to Verizon and Centurylink, has studied bufferbloat. They say they are taking steps to mitigate its effects. Ruckess Wireless, a partner of Juniper, is committed to continued improvement of the access link buffering issue.

But some vendors we talked to seemed unaware of bufferbloat. Others, like Cox Cable, said the issue depended on the manufacturers of hardware and silicon. Unfortunately, most major network test equipment manufacturers we contacted seemed unaware of the issue.

This situation needs to change. It is critical to understand that overall throughput is not the most significant detrimental factor, especially with activities such as browsing. The most significant factor is delay.

Responses with HTTP GET commands are often short bursty file transfers, where the slow start process barely begins until it is terminated. So, delay in session establishment and termination becomes a significant influence on session duration. Also, a typical visit to a popular web site can frequently have 10 to 25 DNS query/response exchanges that precede the get commands. If these are slowed by a factor of three due to bufferbloat, you will certainly notice it.

We strongly recommend that network operators study the vast amount of research already available on the topic of bufferbloat. Then, at critical network connections such as wireless and mobile access points, we need to test for bufferbloat. You will probably want to have the data from these tests to talk with your service provider or wireless access point vendor.

Hippensteel is a professor, consultant and writer with over 40 years' experience in higher education. He can be reached at pjhippen@centurylink.net.