Jon's Performance Musings: The Myth of Gigabit

Back to Basics

As we become jaded by ever-increasing speeds and feeds, it's good to go back to the basics and review what's really going on under the covers. Unfortunately, the speeds of hardware and modern communications allow us to hide a lot of sloppyness. Poor coding and mis-configuration may not be noticed, but think of how much better things could be if we did our work intelligently and cleanly.

One Bit at a Time

I chuckle when I see a diagram of an Ethernet network which uses a "fat" pipe as a graphic for the backbone. The fat pipe implies that lots of data can go through the pipe at the same time, as in a flow of water. The truth is that the fat pipe, in one instant, carries just one bit. So the analog would be a very thin pipe, figuratively one drop wide, under lots of pressure. Yes, cumulatively, the throughput is there. Lots of data traverses the pipe in a very short time. But the analogy falls apart when you consider that an Ethernet backbone has multiple sources and sinks connected to it. Kinda like a network of pipes, each trying to add or take a bucket of water, sometimes at the same time.

Buckets, Packets and Train Cars

One bit at a time, yes. But we work with data. Data may be web pages, application traffic, financial messages, or disk I/O traffic. It could be any size. The data doesn't go onto the network as one contiguous stream of bits. You are welcome to look up the TCP/IP model, or Internet Protocol Suite, to see the function of each layer of the model. What you do need to know is that your message will be broken up into packets, typically 1,460 bytes each (see MTU or maximum transmission unit), wrapped with addressing and data integrity information, and queued onto that single-bit pipe.

It's like breaking the message up and loading it onto individual train cars, and then sending those train cars out. But the analogy breaks down again, because the Internet Layer of the protocol doesn't guarantee that the train cars ever get to their destination. It doesn't even guarantee that the cars will take the same route, nor that they will arrive in the order that they were sent.

The Transport Layer of the protocol has the responsibility of waiting for the packets, re-assembling them into the proper order, and then telling the sender that they have arrived properly, or that they didn't arrive properly.

Collisions and Drops

There are lots of things that can happen to packets. When a particular route or backbone gets busy, senders may try to send at the same instant, causing a collision and retries. As the packets traverse the network, they move from router to router. Each router has to receive the

packet and then figure out where to send it next. If a router is too busy or has some other problem, it may simply drop the packet. There may be transmission errors which corrupt the data. And networks tend to have "trees" of routers, where the top level routers are consolidating traffic from the lower levels.

Packet routes may shift as a result of changing network conditions. It's not unusual for a ping or a traceroute to report wildly differing numbers if one is using the Internet. When they do one can assume that there is congestion somewhere along the path, or that the path is shifting.

Round Trips, Windows, and Throughput

Legacy protocols were designed such that each transmission required an acknowledgement before the next packet could be sent. In the Internet world this is simply not practical. With round-trip times ranging from a couple of milliseconds to 200 milliseconds and beyond, throughput would be abysmal. Some protocols and TCP/IP in particular work with a window, where a certain number of packets are sent without an ack, and the receiver acks when all the packets in the window are received.

It's easy to calculate the throughput of a single TCP/IP conversation. One needs to know the window size and the round trip time (RTT). The formula is:

throughput = windowsize / RTT

Note that the formula doesn't use the speed of the link directly. It is difficult to impossible to determine the minimum speed of each hop in the trip. That number is implied in the round-trip time, but it is more dependent on the number of hops and how busy each hop is than on the raw speed of the links involved. Let's look at some numbers.

Window KB	RTT ms	Throughput MB/second	% of 1Gb trunk
1.5	2	0.75	0.8%
8.0	10	0.80	0.8%
8.0	50	0.16	0.2%
8.0	100	0.08	0.1%
8.0	50	0.16	0.2%
16.0	50	0.32	0.3%
88.0	50	1.76	1.8%
128.0	50	2.56	2.6%
256.0	80	3.20	3.2%

The numbers show that it's difficult for a single conversation to use the full bandwidth available to it. They also show that the bigger the window, the better the utilization of the bandwidth. A bigger window means more packets sent out before the sender has to wait for an ack.

Tuning

How can we maximize the throughput and utilization of the link, and minimize utilization of other system resources such as CPU? What is tuneable?

Physical

The packet size is tuneable, but most architectures already default to the proper value. For NonStop, the default is the appropriate MTU size of 1,460 bytes. There are other options such as the obsolete 512 byte MTU. Yes, there are such things as "jumbo packets" (approx 9KB) and such, but these are limited to specialized networks since their support is not generally standard.

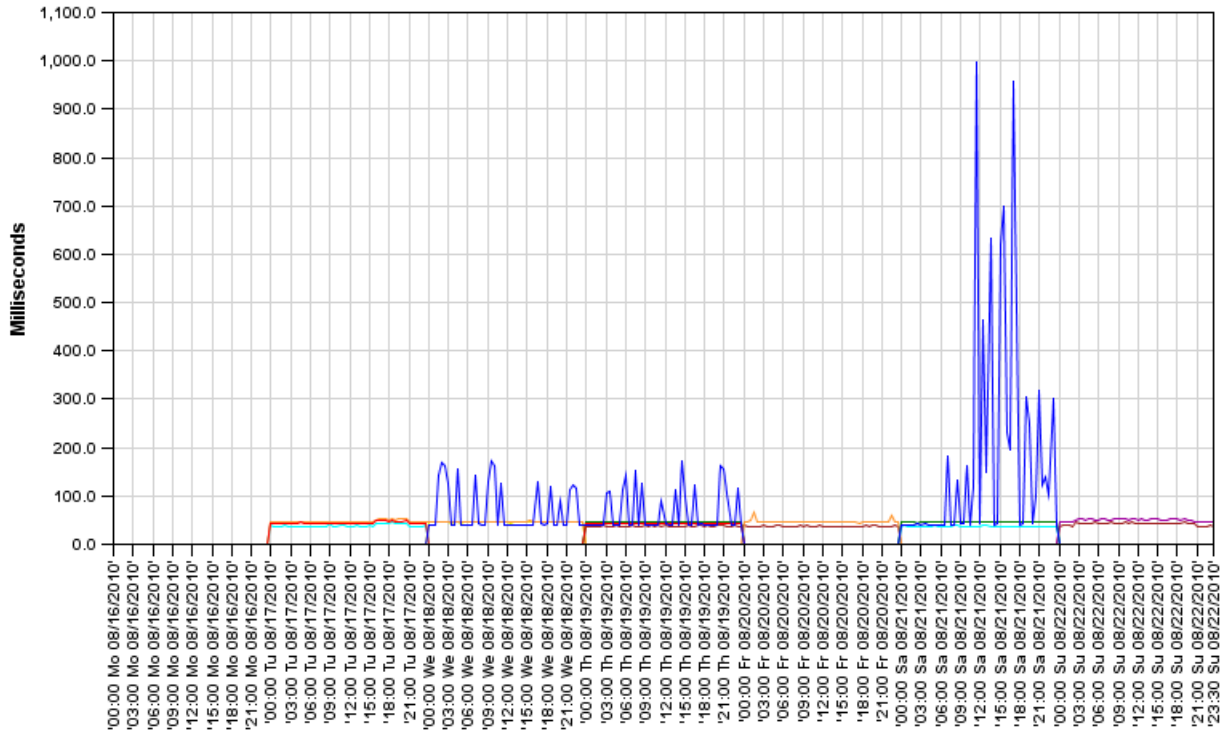
Window

Setting the window size is the best way to tune your connection. The default for NonStop varies from 8KB to 88KB, depending on the level of software you are using. 8KB is certainly too small, and 88KB may still be too small. Look for the TCPSENDSPACE and TCPRECVSPACE parameters and set them to a value that makes sense for the application message size. These may also be set using the socket options SO_SNDBUF and SO_RCVBUF. NonStop allows values from 512 bytes through 262,144 bytes (1MB for the CLIM system). While memory is usually not an issue for modern NonStop, we don't recommend setting things to the maximum value unless the connection is very clean (no lost packets) and the link is used for transmitting files. We recommend setting these parameters to something slightly larger than the application requires, either the largest application message size or page size if a browser-based application.

The Myth of Gigabit

Setting the appropriate MTU and window sizes will help performance. The fewer packets that need to be sent means less load on the CPU and interrupt system. The fewer times that we need to wait for an ack, especially when the public Internet network is involved, will certainly speed up throughput.

On the other hand, we saw how difficult it is for a single conversation to fully utilize the bandwidth that's available. Of course, the higher the bandwidth, the less chance of a collision or other contention, and packets will transit the link faster. High-bandwidth networks are wonderful, but the issue of throughput is complex.



Jon E. Schmidt
Transaction Design, Inc.
San Rafael, CA, 94901, USA
1.415.256.8369
inform@banbottlenecks.com

Jon is the founder of Transaction Design, Inc. (TDI), a consulting firm located in the San Francisco Area which specializes in capacity/performance studies with clients worldwide. He is the creator of the Ban Bottlenecks® service and has an extensive background in the implementation, testing, and tuning of high-availability systems.