Delay Limitations When Extending Metro Ethernet into the Wide Area

Joseph F. Lutz
LightRiver Technologies, 3732 Mt. Diablo Boulevard, Suite 156, Lafayette, California 94549
Lutz@LightRiver.com

Advances in layer 2 ethernet transport now make it possible to extend an ethernet network natively into the metro, regional, and even national spaces. Next generation SONET MSPPs are now being deployed using GFP, VCAT, RPR to implement these networks. Transport bandwidth on these networks typically starts at the VT1.5 or STS-1 rate and climbs the SONET hierarchy from there. The TCP/IP is now basically THE default network and transport layer transmission protocols. It is widely (almost universally) deployed and very flexible. And Ethernet has become essentially the default layer 2 transport protocol upon which this layer 3-4 tcp/ip protocol is transported.

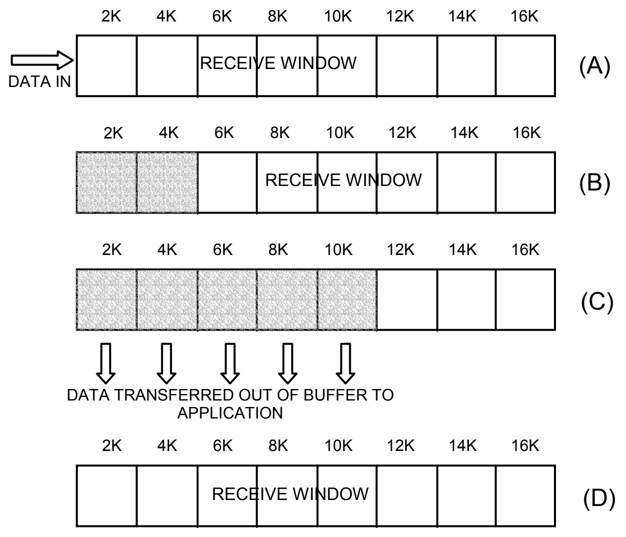
Users of these networks not surprisingly expect that if they put in a big pipe, or long fat pipe, as they are sometimes called, the performance of their network applications, i.e. ftp, web access, etc. will track the increase in bandwidth. TCP however has some properties that limit through-put when used over long fat pipes. As the bandwidth and delay of a Long Fat Pipe increases the through-put of a tcp/ip session can actually decrease, [1,2] This often provides a nasty surprise to an end user when only a fraction (maybe 10% or less) of a new OC-3 circuit translates into actual throughput. In fact it is not just the latency that is of concern but it is the bandwidth delay product that is actually the gating factor. A brief explanation of how tcp works will help explain this phenomenon.

TCP Flow Control

When a tcp session is established it is the tcp receiver that determines the incoming data flow. When a tcp session is being set up each end of the link allocates an amount of receive buffer and makes this known to the sending end. What the receiver in essence tells the sender is: “here is the max amount of data you can send me at one time, when you receive an acknowledgement from me of the data you just sent, then you can send me some more data”. This receive buffer is also referred to as the receive window or sliding window because as data is received the process can be viewed as the window sliding to the right. Figure1 shows this process graphically. In figure1A the receiver has just allocated 16KB of buffer space. The window is usually an integer multiple of the MSS (maximum segment size), a full discussion of TCP is not necessary for the purposes of this paper however the reader is encouraged to read [1] for a very lucid description of a TCP session.

During a set up handshake with the transmitter the receiver sends the window (buffer) size to the transmitter. The transmitter now knows that it can not send more than 16KB at one time. In figure1B the transmitter send 4KB and the receiver takes this 4KB from the link into its receive buffer leaving 12KB available for new data. The transmitter maintains a send buffer keeping track of the data that has been sent, upon sending the 4KB bytes the transmitter knows it is allowed to send no more than 12KB. The transmitter now sends an additional 6KB of data which is clocked into the receiver buffer from the link as shown in figure1C. The receive window has shrunk to 6KB. The receiver then decides it is time to send the data up to the application. It also decides that now would be a good time to send an acknowledgement (ACK) to the transmitter letting it know that the 10K bytes arrived ok. From the ACK sent by the receiver, the transmitter will know the state of the receive window. By sending an updated status in an ACK packet the receiver controls the flow of data that is allowed to be transmitted over the link.

Fortunately in this simple example the receiver buffer never became full and could basically flow continuously. Had the receive window become full the transmitter would have had to stop and wait for an ACK before it could transmit again. If this were to happen the result would be a decrease in throughput, both sides of the connection are just sitting and waiting, and no data is being transmitted. This could happen in a couple of ways. Firstly if you were connected to a very busy web or ftp server the server may have to process many I/O requests and may not be able to transfer data from the buffer in a timely fashion. The other possibility more pertinent to this paper is that the latency of the link maybe such that the transmitter has sent all the data it is allowed to and then just has to wait until an ACK works it’s way across the network before it can send more data. Note that this decrease in throughput could be happening over an error free link. If errors do occur this would only decrease the through-put more as data packets would also have to be re-transmitted.



(A) The receive buffer is empty and awaiting data.

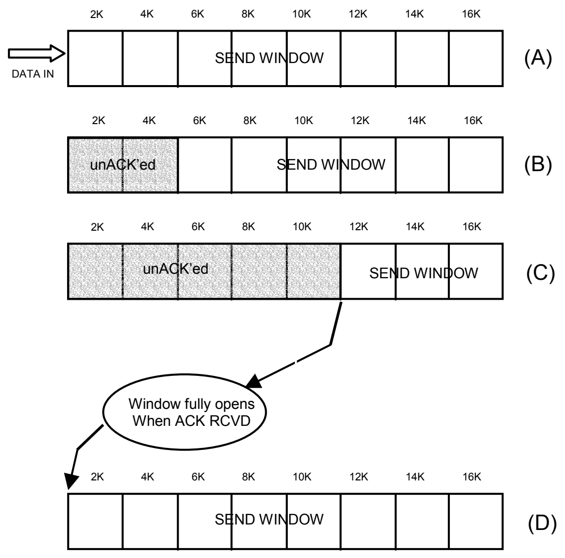
(B) 4K of data has entered the buffer. The window slides to the right. Available buffer space is now 12K.

(C) An additional 6K of data has entered the buffer. Available buffer space is now 6K. Data is transferred out of the buffer and an ACK is sent to the transmitter.

(D) Once the data is transferred to the application the receive window opens fully.

The Receive Window Process

Fig. 1.



(A) The transmitter has not sent any data yet.

(B) 4K of data is sent. The transmitter keeps a copy of this data in the send buffer.

(C) An additional 6K of data is sent. The send window has shrunk to 6K before the arrival of the ACK. After the ACK is received there is no reason to keep the unACK'ed data as the receiver has received it OK.

(D) The send window is now wide open.

Fig. 2.

At the transmitter the send window would track as shown in figure2. Notice that the transmitter keeps a copy of the data as a way of monitoring how much data has been sent in relation to how much it is allowed to send. Keeping a copy also allows for retransmission in case of lost packets or CRC errored packets. The receiver can use several mechanisms as defined by IETF RFCs to obtain a retransmission of data. An ACK may also be lost in transmission in which case the transmitter would have to resend the data because it would have no way of knowing that the receiver got it ok, this also would greatly reduce the throughput.

A critical point to remember here is that all of this is happening at layer 4, the TCP layer. We have not mentioned SONET, Ethernet or IP up to this point.

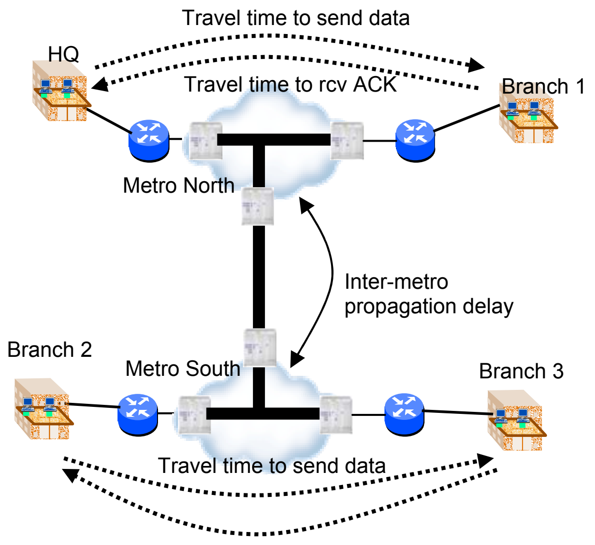


Fig. 3.

For maximum through-put the transmit window needs to remain open for a complete round trip propagation delay interval. Stated mathematically:

$$RW \geq BW \times RTT \quad \text{or} \quad BW \leq (RW/RTT) \qquad (1)$$

RW is the receive window (receive buffer)
BW is the transmission speed (throughput)
RTT is the round trip delay.

If the RW is less than the delay bandwidth product the transmitter will spend some time sitting idle, i.e. not transmitting, while waiting for an ACK packet. A safety margin of up to a factor of two is also required, [3]. See table 1 for a comparison of maximum theoretical throughputs, buffer sizes, distances, and delays.

For intra-metro networks bandwidth delay product will most likely not be a concern.

Table 1. Maximum Theoretical Throughput

One Way Distance	RTT	Receive Window - Bytes			
		8760	17520	65536	250 KByte
64 miles	1mS	70 Mb/s	140 Mb/s	524.3 Mb/s	2 Gb/s
320	5 mS	14	28	104.9	400 Mb/s
640	10 mS	7	14	52.4	200 Mb/s
1282	20 mS	3.5	7	26.2	100 Mb/s
3205	50 mS	1.4	2.8	10.5	40 Mb/s

The distance delay is based upon using SMF-28 with a propagation delay of 7.8uS/mile. In addition on long amplified non regenerated spans, dispersion compensating modules (DCM) will be used. On a span of approximately 850 miles of SMF-28, DCMs can add about 2mS of round trip delay, [6].

The first column of receive window, 8760 Bytes is the default receive buffer in windows 98, 17520 and 65536 Bytes are default buffers in Windows 2000, depending upon which service pack you have. In general operating systems limit the amount of memory that can be used by an application for buffering data. The host system must be configured to support large enough buffers for reading and writing data to the network. The IETF has tackled this issued several RFCs that address this problem, [4]. Typical Unix systems have initial default buffer sizes ranging between 8kB – 32kB and expand to a default maximum value for the socket buffer between 128kB and 1MB, [5]. If the default buffer size is not large enough, the application must set its send and receive socket buffer sizes (at both ends) to at least the delay bandwidth product of the link. Some network applications support options for the user to set the socket buffer size (for example, Cray UNICOS FTP); many do not, [5]. This ends up becoming a trade off for the system administrator, he can set the system wide default window to a *"large"* value but this is very inefficient as many applications then end up consuming system resources that they do not use.

CLEC Implementation into the Wide Area

LightRiver Technologies has a client that implemented a metro-ethernet network as depicted by the Metro North network of figure3. The MSPP interfaces were 10/100 fast Ethernet and network throughput was measured, by the CLEC, at around 88Mb/s. The CLEC measured throughput using ftp and some low cost shareware products. This is typical of many CLECs and ICOs. In these environments test equipment other than a BER tester is hard to come by due to budget constraints. Next the CLEC deployed Metro South as a stand alone metro-ethernet network. The performance of Metro South being the same as in the North. Eventually a customer in Metro North wanted to connect a branch office in Metro South to his network in Metro North. The CLEC leased an OC-3 that traversed two IXCs to link Metro North and Metro South. All services were Ethernet private line. A 100 Mb/s service was sold to the customer. The distance between the two metros was approximately 400 miles.

When the circuit was brought up it was tested using both ftp and a shareware tool. The result was a throughput of around 4Mb/s. Not particularly good for a 100Mb/s regional Ethernet connection. As trouble shooting began the ping was measured to be 20mS and the receive window buffer was 16kB. A quick look at Table 1. shows the max throughtput that could be hoped for is less than 7 Mb/s, which is definitely the case here, by about a factor of two, [2,3]. However there is a problem here in that the distance between the two Metros is only 400 miles. But upon further investigation it was discovered that the fiber route miles was actually about 990 miles. Exact figures could not be gotten but the 990 miles was based upon admitted existing fiber routes. The original 400 miles given to us was based upon direct air miles, not the fiber route. A quick calculation shows that the one way fiber propagation delay is 7.8mS making the fiber RTT = 15.6mS. This leaves a little over4 mS unaccounted for. On a link of 990 miles it is possible that a DCM was in the link some which could have added on the order of 2mS of RTT, [6]. Then there is the SONET network elements (NEs) at the customer sites plus the SONET network elements of the two IXCs that provided transport. In this case there were two IXCs involved which means at a minimum there were at least two NEs each leg of the path for a total of 4 NEs. The delay across a SONET NE could possibly be somewhere in the 25uS to 75uS range depending upon a host of factors related to the NE itself and how the IXC was actually configuring it. Just picking a number, 50us, results in a one way delay of 250uS, or 500us of RTT due to the IXC SONET NEs.

Finally there is the delay of the MSPP that interfaces to the customer premises. The delay here will be composed of the Ethernet GFP VCAT mapper and the SONET cross connect. The GFP process is store and forward so the entire Ethernet frame is received before it is sent to the next stage. This makes the delay frame size dependent.

Typical delays are in the range of 1 to 2 SONET frames, 125uS – 250uS. If we assume we can get the Ethernet frames GFPed, VCATed, and unto the cross connect in 250uS this then gives us a 500uS RTT for each MSPP. Another delay factor to consider is that due to differential delay. One of the touted advantages of using VCAT is that the individual constituent STS-1s comprising the VCAT group can be treated independently by any intermediate network transport equipment. This means for example that if you have a VCAT group consisting of 2 STS-1s each STS-1 can take a different path through the network. At the egress point the MSPP buffers the first STS-1 to arrive until the second STS-1 arrives. Differential delay was probably not a factor in this case but it can not be ignored.

Delay Budget

Given the ramifications of delay on through-put performance it is desirable to account for sources of delay in any given link. Table 2 summaries the possible sources of delay in the network.

Table 2. Circuit Delay

One Way Delays			
Component	Delay	Instances	Total
Ingress MSPP	0.25 mS	1	0.25 mS
Egress MSPP	0.25 mS	1	0.25 mS
Transport NEs	0.05 mS	4	0.2 mS
Fiber	7.8 mS	1	7.8 mS
DCM	1.0 mS	1	1.0 mS
Diff. Delay	0 mS	1	0 mS
Total			9.5 mS

In this example the total one way delay is 9.5mS. Times two gives us 19 mS of RTT, pretty close to the 20 mS ping that was measured. In this example the calculated delay happened to work out pretty close to the measured delay. In reality however many of these numbers can not be known precisely but based upon reasonable estimates a safe "zone" of operation can be determined and decisions made accordingly.

References

1. Dr. Sidnie Feit, *TCP/IP* (McGraw-Hill, 1997), Chap. 10.
2. Brian Tierney, *TCP Tuning Guide For Distributed Application on Wide Area Networks*, http://www-didc.lbl.gov/tcp-wan-perf.pdf
3. Stanislav Shalunov, *TCP over WAN Performance Tuning and Troubleshooting*, http://www.internet2.edu/~shalunov/writing/tcp-perf.html
4. IETF RFC 1323: V. Jacbson, "TCP Extensions for High Performance"
5. Pittsburg Supercomputing Center, *Enabling High Performance Data Transfers*, http://www.psc.edu/networking/perf_tune.html#table
6. NFOEC: Vipin Patel, Peter C. Noutsios, "Latency on a Line Amplified, DWDM Backbone Network," in National Fiber Optics Engineering Conference (2003).