Utilizing Linux for OC Distance Emulation for SAN to SAN Testing

Clea Zolotow, Senior Technical Staff Member, IBM
Lisa Martinez, Advanced Technical Sales Specialist, IBM
Andre Koster, XIV Development Engineer, IBM
Anthony Hunt, Distinguished Engineer, IBM

ABSTRACT

In order to accomplish long distance emulation for the SAN to SAN appliance testing, a Linux box (the Distance Emulator, DE) needed to be set up and configured between the source and destination fabrics, here the Brocade (IBM 2498)..This Linux appliance utilizes the NetEm emulation which "reproduces network dynamics by delaying, dropping, duplicating or corrupting packets."[i] Utilizing the Linux box resulted in a successful test of latency (distance) delays which testing SAN to SAN migration effectiveness. We explain here how the test was set up and the Linux parameters used to obtain the latency.

SAN to SAN Testing Architecture

The Distance Emulator (DE) was part of the testing methodology. Below is the test architecture, where the ESX host is attached to the Brocade Fabric (2498) via the SAN to SAN device. Note the secondary SAN to SAN device acts as a proxy to the destination fabric, as the SAN to SAN is not connected to that fabric. The proxy SAN to SAN then will "present" the destination LUNS to the migration SAN to SAN. Here, we show the test methodology for the specific Vicom device, code-named Piper.

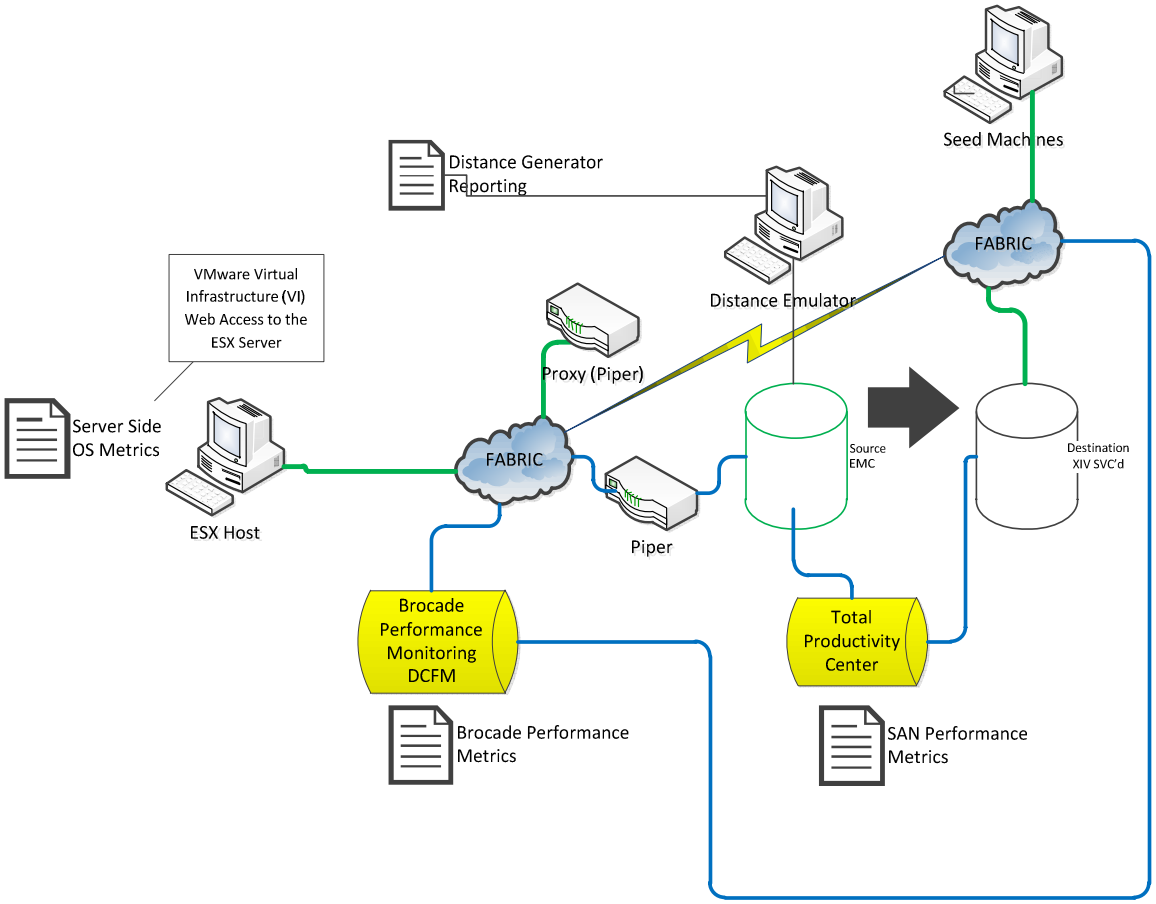

Figure 1: SAN to SAN Test Scenario

Currently, there are two 1 Gbps Ethernet lines attaching the fabrics to the Distance Emulator (DE). However, only one of these is active.

The Relationship Between Bandwidth and Delay

There are different types of migrations, depending on the end-result. These include technology refresh (migration to another vendor would fall under this category) as well as the purpose we are testing here: datacenter migration. In order to size the migration, we need to understand the relationship between bandwidth and delay.

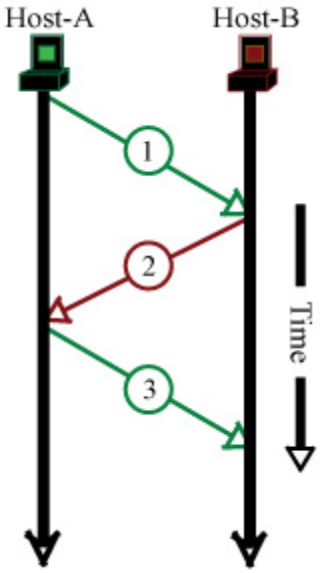

Figure 2: Bandwidth and Delay

The Bandwidth-Delay-Product (BDP) is the quantity of data that can be in transit on a network, at a given time. The BDP is the product of the network bandwidth and the Round Trip Time (RTT).

BDP = B/W x RTT

Where:
B/W is the peak bandwidth of the link (based upon the slowest link in the path)
RTT is the round-trip delay, which can be measured with the ping command.

Without getting into TCPIP window analysis, note that when the TCP window size, based on the transmitting end-system buffers, is set at 64 kilobytes (kb) and the RTT is 50 milliseconds, the Bandwidth Delay Product is 3.2 kilobyte-seconds. Based on the transmitting host configuration, the maximum data transfer rate would be 10.4 Mbps, based on using the formula below:

Window Size / RTT = Effective Bandwidth

Calculating the optimal window size is located here: http://www.speedguide.net/bdp.php. An example configuration is below. The bandwidth here is 149 Mbps for an OC-3 and an assumed 30 ms delay time. This would give is a good estimation of what the TCP window size should be, approximately. Note that the TCP window size should be a multiple of the Maximum Segment Size (MSS).[ii]

The TCP Maximum Segment Size (MSS) defines the maximum amount of data that a host is willing to accept in a single TCP/IP datagram.[iii]

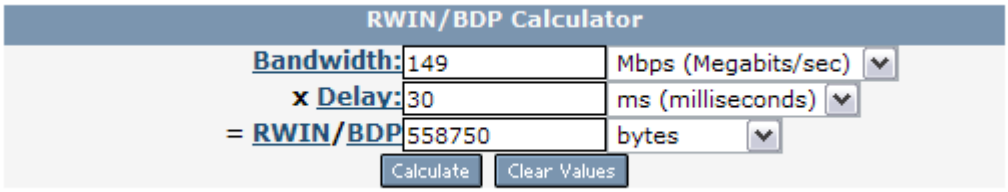

Figure 3: SG Bandwidth*Delay Product Calculator

However, we can look at this another way: TCP Window Size / Round Trip Latency (ping time) = MAX Bandwidth. Since we know what the TCP Window Size is (from the NetEm parms) we can calculate the max bandwidth and therefore not invoke the rate parms in the OC-emulator.

Remember that "NetEm is a network emulator in the Linux kernel 2.6.7 and higher that reproduces network dynamics by delaying, dropping, duplicating or corrupting packets."[iv]

Therefore, with a max bandwidth of an OC-3 = 149 Mbps and the assumed latency of 30 ms, we can do the following:

Max Bandwidth / Round Trip Latency = TCP Window size

Therefore:

149 / 30 = 5.13 is to be input into the OC-emulator parameters, and thereby limit via the TCP window size, rather than the bandwidth. This is important as we want to measure the bandwidth with the latency included and not set an a priori bandwidth specification (which would then override the latency affect).

Below is the configuration of the Linux box as the Distance Emulator running SUSE Linux Enterprise Server (SLES). Note the bridge ports interfacing to the Brocade Fabric as source and destination.

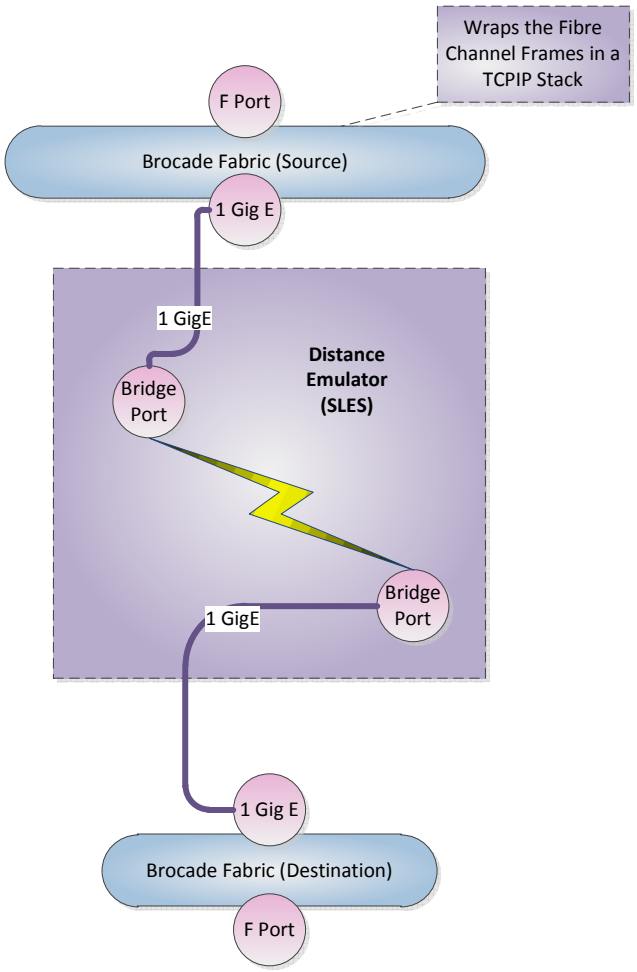

Figure 4: Migration Flow from Host Fabric through Distance Emulator to Destination Fabric

Note that this configuration will limit traffic through the Distance Emulator. Each 1 GigE wire supports (1024*0.7) = 716.8 effective thruput. Therefore, given the following test cases note that utilizing only 1 GigE line, we can only support the OC12 use case, not the OC48. We will have to install the full complement of 4 GigE lines in order to cover the thruput by the OC48.

Linkages and Latency Use Cases
Effective Thruput Rate (not actuals)

	Link	Mbp/s	MBp/s	MBp/m	GBp/h
Telecom	DS3	45	6	336	20
	OC3	149	19	1,115	65
	OC12	601	75	4,510	264
	OC48	2,405	301	18,040	1,057
Ethernet	*1 GigE Line*	717	90	5,376	315
	2 GigE Lines	1,434	179	10,752	630
	3 GigE Lines	2,150	269	16,128	945
	4 GigE Lines	2,867	358	21,504	1,260

To emulate the OC-3, we configured it thusly:

tc qdisc add dev eth8 parent 1:1 handle 1:0 tbf rate 152576kbit buffer 1600 limit 3000

Where:

tc - show / manipulate traffic control settings

qdisk - a disk-based quorum daemon for CMAN / Linux-Cluster - qdisc is short for 'queuing discipline' and it is elementary to understanding traffic control. Whenever the kernel needs to send a packet to an interface, it is enqueued to the qdisc configured for that interface. Immediately afterwards, the kernel tries to get as many packets as possible from the qdisc, for giving them to the network adaptor driver.

tbf – token bucket filter where, limit or latency is the "size - in bytes - of the packets queue." Limit is the number of bytes that can be queued waiting for tokens to become available. You can also specify this the other way around by setting the latency parameter, which specifies the maximum amount of time a packet can sit in the TBF. The latter calculation takes into account the size of the bucket, the rate and possibly the peakrate (if set).

Burst AKA buffer AKA maxburst: Size of the bucket, in bytes. This is the maximum amount of bytes that tokens can be available for instantaneously. In general, larger shaping rates require a larger buffer. For 10mbit/s on Intel, you need at least 10kbyte buffer if you want to reach your configured rate!

If your buffer is too small, packets may be dropped because more tokens arrive per timer tick than fit in your bucket. **The minimum buffer size can be calculated by dividing the rate by HZ.**

Token usage calculations are performed using a table which by default has a resolution of 8 packets. This resolution can be changed by specifying the cell size with the burst. For example, to specify a 6000 byte buffer with a 16 byte cell size, set a burst of 6000/16. You will probably never have to set this. **Burst must be an integral power of 2.**

Therefore, the command adds the device eth8 to the parent device, where 1:1 denotes the first iteration of the device. The handle is what we call the device in reference to the command. TBF is the token bucket filter that takes the parameters for the emulation requested. Here, the rate is equal to the effective (or actual, not theoretical) thruput rate. The buffer is 1600

"NetEm consists of two portions, a small kernel module for a queuing discipline and a command line utility to configure it. The kernel module has been integrated in 2.6.8 (2.4.28) or later, and the command is part of the iproute2 package." [v]

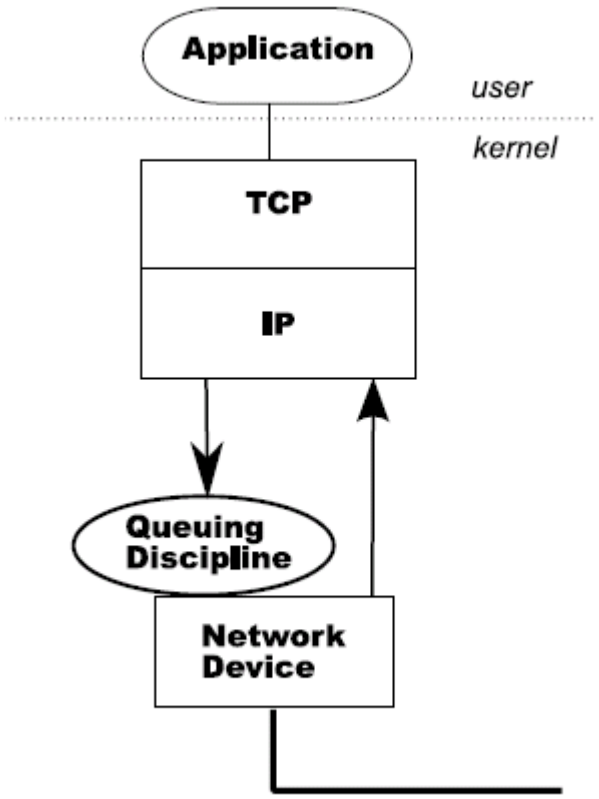

Figure 5: NetEm Queuing Discipline

Above in Figure 4 is the NetEm Queuing methodology. The discipline is between the protocol and the output device. Default queuing is FIFO. The interfaces used for the SAN to SAN device are:

1. Incoming from the destination fabric.
2. Exiting to the host fabric.

Here, we have added in the ability to take the incoming packets on the enqueue and place them with a time-stamp. The timer then takes the packets and moves them from the holding queue to the nested discipline, where the dequeue interface then gets the packets.[vi]

tc qdisc add dev eth8 root handle 1:0 netem delay 30ms

Where:

netem = NetEm provides Network Emulation functionality for testing protocols by emulating the properties of wide area networks. The current version emulates variable delay, loss, duplication and re-ordering.[vii]

This simple example shows how to create a delay of 30 milliseconds going out of the local Ethernet. A ping test on the local network should show an additional 30 ms of response time.

Remember that bridges need to be created. The WAN-Bridge from sysco on device called ‘br0’ and will add to it all found network interfaces. The system must have at least two (2) network interface cards (NICs).

Applying these delay times via the Distance Emulator resulted in a successful test of the various SAN to SAN migration devices.

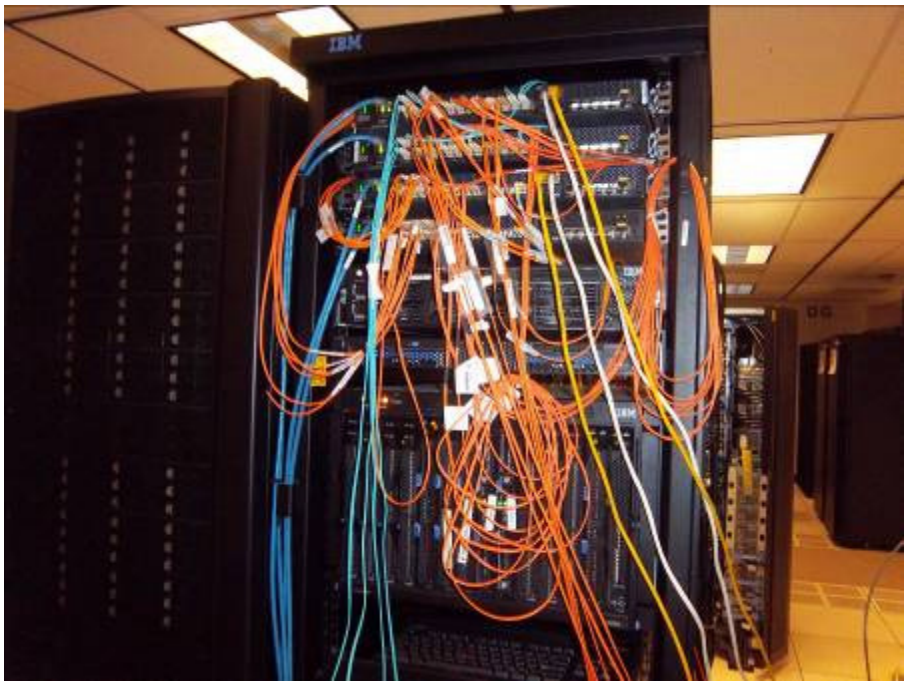

Figure 6: Brocade Fabric (Source and Destination)

Figure 7: EMC Clarion CX 700 Nameplate

Figure 8: Back of the Vicom SAN to SAN and SAN to SAN Proxy

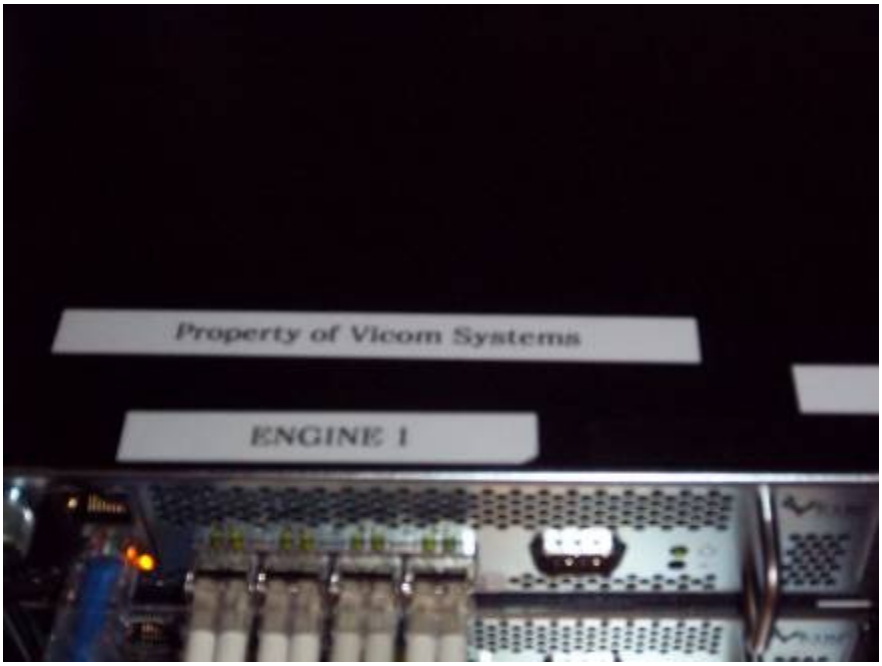

Figure 9: Top of the SAN to SAN Device, 1 Engine of 2

Figure 10: Front View of the Distance Emulator (SLES)

Figure 11: GigE Connections from Distance Emulator to Brocade Fabric

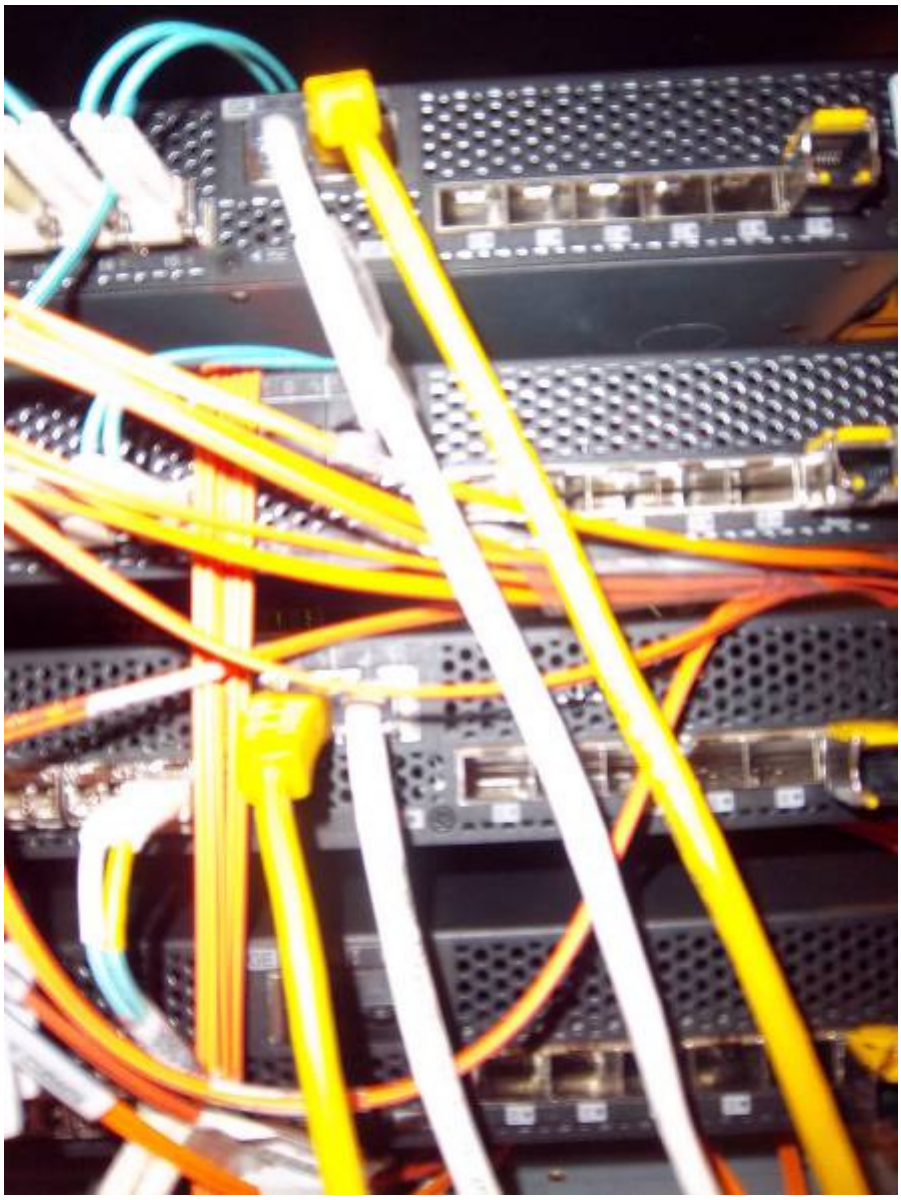

Figure 12: Brocade View of Distance Emulator (see source and destination attachment)

---

[i] http://tcn.hypert.net/tcmanual.pdf

[ii] http://www.speedguide.net/bdp.php

[iii] http://www.cisco.com/en/US/tech/tk827/tk369/technologies_white_paper09186a00800d6979.shtml

[iv] http://tcn.hypert.net/tcmanual.pdf

[v] IP routing utilities, http://developer.osdl.org/dev/iproute2

[vi] http://devresources.linuxfoundation.org/shemminger/LCA2005_paper.pdf

[vii] http://www.linuxfoundation.org/collaborate/workgroups/networking/netem