**Introduction**

The goal of this report is to offer guidance on the implementation of networking as part of integrated, high-performance cyberinfrastructure. Though it includes findings drawn from optimizing SLASH2 operation between Pittsburgh Supercomputing Center (PSC) and collaborating sites, the applicability is broad, extending well beyond DXC and SLASH2. Some of the experiences resulted from debugging new network infrastructure along with the installation and integration of SLASH2 itself. The focus is on metro and WAN connectivity issues rather than data center LAN. The targeted audience includes members of the research and education community who are planning, deploying, debugging, and monitoring high performance cyberinfrastructure and networking.

PSC’s Data Exacell and other high performance computing resources are located at 4350 Northern Pike. Monroeville, PA. Collaborating site are located within the Pittsburgh region, West Virginia, and Minnesota.

**Executive summary**

**General network design recommendations:**

* Networking for DXC follows the current best-practices as described by the Energy Sciences Network (ESnet) ScienceDMZ <https://fasterdata.es.net/science-dmz/> architecture. Particular factors to be considered:
  + WAN and metro area networking equipment needs to have large enough packet buffers available **end-to-end** to sustain multi-gigabit per second data rates. Only having externally facing large packet buffers at a site’s border is not sufficient. Small packet buffers anywhere along the path can not hold enough packets to smooth occasionally bursty packet arrival behavior and packets will be dropped. Even a small amount of packet loss due to congestion or other factors can seriously decrease throughput.
  + The data center network connection should be on, or very near, the site’s border. Providing clean, high performance connectivity gets increasingly more difficult the further an endpoint is embedded within a site.
* Deploy perfSONAR network test nodes (see <http://www.perfsonar.net/>) at the sites’ edges and close to the data transfer endpoints if these the edge and transfer endpoints are separated by multiple network hops.

**Operational recommendations:**

* The initial test team will ideally comprise a domain scientist and a site network engineer who is familiar with the network topology and the test tools. There may be a couple of iterations of the team before all necessary participants are identified and engaged.
* The team needs to agree on the set of tests to be run. Also helpful is to develop an estimate in advance of what the expected results will be.
* Up to date diagrams of network infrastructure need to be shared among the team members. Discuss the level of detail and security/access policy in advance. Detailed network architecture diagrams are rarely public information due to security concerns.

**Test tools**

The most commonly used network test tools include:

* ping - reports reachability and measures the round trip time between endpoints
* traceroute - reports the network Layer3 path hops between endpoints (RTT with intermediate routers are not indicative of problems
* iperf3, iperf, and nuttcp - network throughput tests run between a sender/receiver
* perfSONAR network test platform and measurement archive:
  + Offers ping, traceroute, iperf3/iperf/nuttcp bandwidth, and latency test tools in an integrated package
  + Supports scheduling of multiple ongoing network tests between the nodes. The results of these scheduled tests are stored in a web accessible measurement archive. The advantage offered by the archive are performance graphs that readily show anomalous behavior and trends over time.
  + Allows manual testing to further investigate unexpected test results.
  + Supports a “third party testing mode” between nodes so an engineer can run tests between sites without requiring a login.

Each of the above tools is straightforward to run and offers results that are meaningful to researchers and engineers having an intermediate level of networking knowledge.

Additional tools that offer a detailed view into network transfers include:

* tcpdump - collects headers from each packet
* tcptrace - analyzes the network packet headers and can prepare data for plotting
* xplot - graphing tool use to display the network traffic captured by tcpdump and processed by tcptrace
* Web10G - Linux kernel patch that reports the TCP statistics for each end host connection

These tools are more accessible to network engineers with specialized knowledge to interpret the results. We have not needed to rely on these for DXC or SLASH2 network debugging, however we may decide to use them in the future for more fine-grained study of performance.

The DXC team also monitors throughput as reported by the SLASH2 application itself.

**Principles of testing**

* Agreeing to allow ping and traceroute between the endpoints is helpful.
* Get traceroutes in both directions and make sure they match the expected path(s)
* Segmenting a path and diagnosing the pieces is extremely helpful when problems exist.
* Some people are apprehensive about a 10G perfSONAR “filling the pipe”. Using the TCP bandwidth test is the same as a standard TCP transfer starting up on any appropriately-tuned, modern host, going through slow-start, and reaching steady-state congestion avoidance with fair sharing of available bandwidth
* Ideally, perfSONARs are deployed at multiple key locations within the campus infrastructure. An ongoing set of scheduled tests is configured to run between the perfSONARs to establish typical performance levels
* A perfSONAR “Maddash” performance reporting dashboard can be set up as a quick-view interface to the results of scheduled testing (see <http://software.es.net/maddash/> )

**Lessons learned from working with partners:**

This section lists the causes, and describes some of their symptoms, of slow throughput discovered during work with collaborating sites. None of these is specific to DXC or SLASH2. The three most frequently encountered causes of problems were firewalls, MTU mismatches, and inadequate TCP packet buffering.

**Firewalls**

Firewalls were either misconfigured or inappropriately configured for the application. In some cases we had to revisit firewall rules several times to correct multiple issues. These included:

* Inserting basic firewall rules to permit SLASH2 data transfer and control traffic.
* Re-inserting basic firewall rules when we discovered they had not been made to persist across reboot
* Correcting firewall rules that were more appropriate to interactive sessions, terminating connections after 30 minutes of idle time. SLASH2 operation was destabilized in that situation

If a data transfer endpoint is not located at the campus edge, but is farther within campus infrastructure, the site team may not be aware that a firewall, or firewalls, exist in the path. perfSONAR testing can be used to indicate the presence of firewalls. Example output from a blocked throughput test:

bwctl -T iperf3 -c perfsonar.site.edu

iperf3: error - unable to connect to server: Connection refused

perfSONAR may require a tweak to align range of ports on the firewall AND the perfSONAR. The relevant configuration file to check the test port pool on the perfSONAR is:

pS v3.5.1 /etc/bwctl-server/bwctl-server.limits

Make sure the test port on the perfSONAR and the open ports on the firewall do in fact match!

**MTU**

MTU (“maximum transmission unit”, the maximum network layer packet size) mismatches, in tandem with blocked ping/ICMP messages, are a frequent cause of blocked connections, particularly in newly upgraded infrastructure. The most commonly used MTU values for an end host are either 1500 Bytes or 9000 Bytes (also referred to as “jumbo” packets). MTU settings on infrastructure devices such as routers and switches is larger to accommodate additional information in each packet such as VLAN tags.

Modern TCP implementations try to “discover” the maximum packet size by sending “probing” packets of increasing sizes into the network. The end host then waits for a returned ICMP “Fragmentation needed” packet to indicate that the maximum MTU of the path has been exceeded, thereby setting the upper boundary for packet size.. Sites that routinely block ICMP, perhaps thinking that they are blocking a security threat, break this Path MTU Discovery (PMTUD) mechanism. Ways to detect this problem:

* An application connection or an iperf3 test may simply block totally and not connect
* An iperf3 test connects then transfers a small amount of data, e.g, 80Mb/s in the first 2 seconds, 5 Mb/s sec. 2-4, then 0.0 Mb/s for the rest of the test. The probing traffic got through, the ICMP message requiring packet fragmentation was blocked, the sending host thought it was fine to send, the infrastructure or end host dropped the packets
* Try pinging with various MTUs, particularly around 1472/1473 and 8972/8973 and force the test to prohibit fragmentation (-M do). Example:

[benninge@pscuxa ~]$ ping -c 3 -s 1472 -M do 10.32.5.188

PING 10.32.5.188 (10.32.5.188) 1472(1500) bytes of data.

1480 bytes from 10.32.5.188: icmp\_seq=1 ttl=64 time=0.577 ms

1480 bytes from 10.32.5.188: icmp\_seq=2 ttl=64 time=0.339 ms

1480 bytes from 10.32.5.188: icmp\_seq=3 ttl=64 time=0.306 ms

--- 10.32.5.188 ping statistics ---

3 packets transmitted, 3 received, 0% packet loss, time 2001ms

rtt min/avg/max/mdev = 0.306/0.407/0.577/0.121 ms

You have mail in /var/spool/mail/benninge

[benninge@pscuxa ~]$ ping -c 3 -s 1473 -M do 10.32.5.188

PING 10.32.5.188 (10.32.5.188) 1473(1501) bytes of data.

ping: local error: Message too long, mtu=1500

ping: local error: Message too long, mtu=1500

ping: local error: Message too long, mtu=1500

--- 10.32.5.188 ping statistics ---

3 packets transmitted, 0 received, +3 errors, 100% packet loss, time 1999m

* A switch MTU bug uncovered by running iperf tests with differing MTU sizes

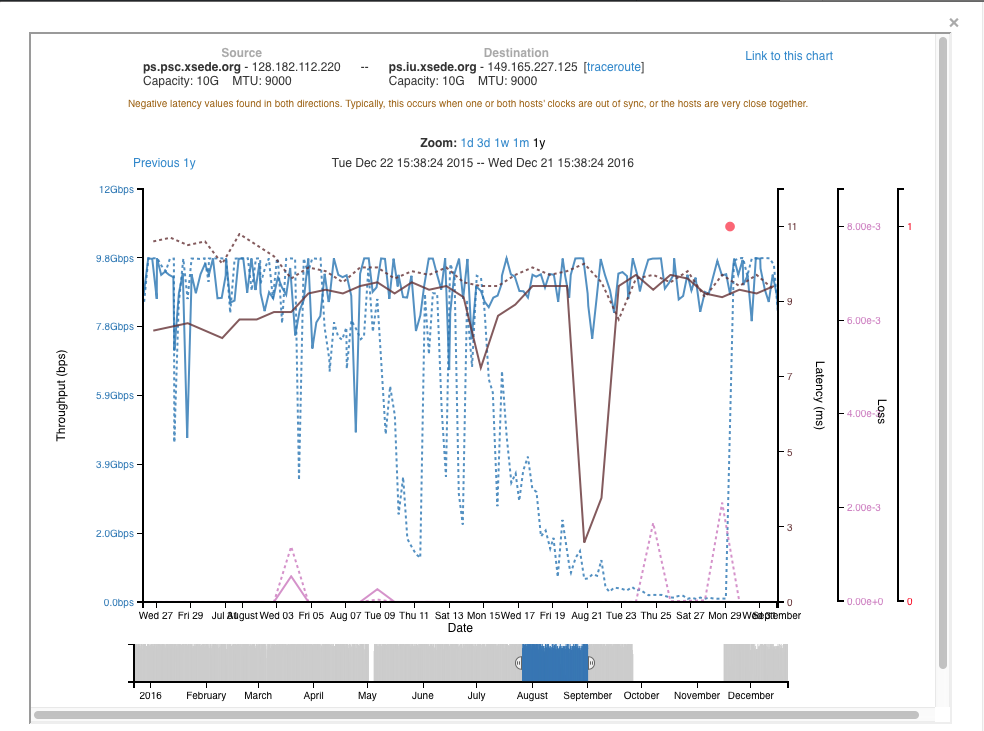
**Under-sized packet buffers**

As described in the *General network design recommendations* section above, appropriate sizing of buffers is critical to supporting high performance network throughput.

* Evidence of small network infrastructure buffers can be seen in decreasing sustained throughput as the distance (hence, the network Round Trip Time) between endpoints decreases. Buffer sizes that can sustain greater than 9Gb/s between sites within the same metropolitan areas will struggle to sustain greater than 1Gb/s even half-way across the country.
* Informative references for packet buffer sizing are:
  + <https://fasterdata.es.net/network-tuning/router-switch-buffer-size-issues/>
  + [http://people.ucsc.edu/~warner/buffer.html](http://people.ucsc.edu/%7Ewarner/buffer.html)
  + <http://www.es.net/assets/pubs_presos/NANOG64mnsmitasinbltierneybuffersize.pptx.pdf>
* 10s of MBytes per port will probably be sufficient (within the United States). 20MB shared among 24 or 48 ports will not.
* Small maximum TCP buffer limits in end hosts will also limit throughput. The Linux default settings are less than 15% what they need to be to support high throughput on a 10GbE link. For more information, see the ESnet page at <https://fasterdata.es.net/host-tuning/>

**Other issues encountered:**

* Failing optic module with throughput gradually declining:



* perfSONARs could not be permanently deployed in all desired locations. To work around this issues, the network engineers configured special paths for testing through new infrastructure. This configuration was not persistent and needed to be reapplied after a site power outage.
* The engineers at each site have varying priorities so the investigation can be on hold for days or weeks at a time.
* There was intermittent routing instability for one of the participating sites that goes through a regional network (i.e., another administrative domain).

**Future network plans**

As SLASH2 matures, development efforts will become increasingly focused on performance, both file system and network. Further study of SLASH2 network performance characteristics may be desirable to understand where improvements can be made. The integration of advanced networking capabilities, such as those offered by Software Defined Networking (SDN), are being investigated to see what benefits these could bring to SLASH2 operation. One example is the bandwidth control (QoS) that can be readily imposed through SDN metering and queuing. The DXC team is already working on integrating this QoS with SLASH2 replication.

========================================================================

**Some examples of iperf3 results:**

**End host with TCP buffer-limited throughput**

An indicator is minimal variation in reported “Bandwidth” and “Cwnd”.

[username@tsunami ~]$ bwctl -T iperf3 -t 20 -s host1 -c host2  
bwctl: Using tool: iperf3  
bwctl: 136 seconds until test results available  
  
SENDER START  
Connecting to host host2, port 5244  
[ 14] local host1 port 42471 connected to host2 port 5244  
[ ID] Interval Transfer Bandwidth Retr Cwnd  
[ 14] 0.00-1.00 sec 521 MBytes 4370 Mbits/sec 93 928 KBytes (omitted)  
[ 14] 1.00-2.00 sec 531 MBytes 4457 Mbits/sec 17 1.24 MBytes (omitted)  
[ 14] 2.00-3.00 sec 526 MBytes 4414 Mbits/sec 25 1.45 MBytes (omitted)  
[ 14] 3.00-4.00 sec 525 MBytes 4404 Mbits/sec 9 980 KBytes (omitted)  
[ 14] 0.00-1.00 sec 541 MBytes 4540 Mbits/sec 22 1.14 MBytes   
[ 14] 1.00-2.00 sec 549 MBytes 4604 Mbits/sec 22 1.24 MBytes   
[ 14] 2.00-3.00 sec 526 MBytes 4414 Mbits/sec 43 1.27 MBytes   
[ 14] 3.00-4.00 sec 530 MBytes 4446 Mbits/sec 18 1.31 MBytes   
[ 14] 4.00-5.00 sec 521 MBytes 4373 Mbits/sec 30 1.37 MBytes   
[ 14] 5.00-6.00 sec 531 MBytes 4456 Mbits/sec 17 1.41 MBytes   
[ 14] 6.00-7.00 sec 514 MBytes 4310 Mbits/sec 22 1.25 MBytes   
[ 14] 7.00-8.00 sec 535 MBytes 4488 Mbits/sec 18 1.31 MBytes   
[ 14] 8.00-9.00 sec 526 MBytes 4415 Mbits/sec 34 1.23 MBytes   
[ 14] 9.00-10.00 sec 524 MBytes 4394 Mbits/sec 11 1.34 MBytes   
[ 14] 10.00-11.00 sec 524 MBytes 4394 Mbits/sec 23 1.38 MBytes   
[ 14] 11.00-12.00 sec 528 MBytes 4425 Mbits/sec 11 1.44 MBytes   
[ 14] 12.00-13.00 sec 520 MBytes 4362 Mbits/sec 27 779 KBytes   
[ 14] 13.00-14.00 sec 524 MBytes 4394 Mbits/sec 21 980 KBytes   
[ 14] 14.00-15.00 sec 529 MBytes 4435 Mbits/sec 13 1.48 MBytes   
[ 14] 15.00-16.00 sec 530 MBytes 4446 Mbits/sec 23 858 KBytes   
[ 14] 16.00-17.00 sec 525 MBytes 4404 Mbits/sec 15 1015 KBytes   
[ 14] 17.00-18.00 sec 521 MBytes 4373 Mbits/sec 17 998 KBytes   
[ 14] 18.00-19.00 sec 524 MBytes 4394 Mbits/sec 9 1.09 MBytes   
[ 14] 19.00-20.00 sec 531 MBytes 4456 Mbits/sec 30 1.20 MBytes   
- - - - - - - - - - - - - - - - - - - - - - - - -  
[ ID] Interval Transfer Bandwidth Retr  
[ 14] 0.00-20.00 sec 10.3 GBytes 4426 Mbits/sec 426 sender  
[ 14] 0.00-20.00 sec 10.3 GBytes 4434 Mbits/sec receiver  
  
iperf Done.  
SENDER END

**Local test with small buffered infrastructure**

Although “Bandwidth” remains high, there are many retransmitted packets per reporting interval. The packets are being dropped by the infrastructure. This amount of packet loss when supporting a wide area connection will substantially decrease throughput.

[username@tsunami ~]$ bwctl -T iperf3 -t 20 -c host 3 -s host4  
bwctl: Using tool: iperf3  
bwctl: 29 seconds until test results available  
  
SENDER START  
Connecting to host host3, port 5579  
[ 14] local host4 port 39149 connected to host3 port 5579  
[ ID] Interval Transfer Bandwidth Retr Cwnd  
[ 14] 0.00-1.00 sec 1.02 GBytes 8768 Mbits/sec 9 1.60 MBytes (omitted)  
[ 14] 1.00-2.00 sec 1.15 GBytes 9914 Mbits/sec 0 2.43 MBytes (omitted)  
[ 14] 2.00-3.00 sec 1.15 GBytes 9887 Mbits/sec 209 2.06 MBytes (omitted)  
[ 14] 3.00-4.00 sec 1.15 GBytes 9883 Mbits/sec 230 1.13 MBytes (omitted)  
[ 14] 0.00-1.00 sec 1.15 GBytes 9898 Mbits/sec 83 1.05 MBytes   
[ 14] 1.00-2.00 sec 1.15 GBytes 9895 Mbits/sec 65 1.31 MBytes   
[ 14] 2.00-3.00 sec 1.15 GBytes 9901 Mbits/sec 0 2.23 MBytes   
[ 14] 3.00-4.00 sec 1.15 GBytes 9909 Mbits/sec 0 2.87 MBytes   
[ 14] 4.00-5.00 sec 1.15 GBytes 9867 Mbits/sec 502 1.77 MBytes   
[ 14] 5.00-6.00 sec 1.15 GBytes 9899 Mbits/sec 104 1.53 MBytes   
[ 14] 6.00-7.00 sec 1.15 GBytes 9885 Mbits/sec 167 1.26 MBytes   
[ 14] 7.00-8.00 sec 1.15 GBytes 9901 Mbits/sec 0 2.21 MBytes   
[ 14] 8.00-9.00 sec 1.15 GBytes 9896 Mbits/sec 120 2.08 MBytes   
[ 14] 9.00-10.00 sec 1.15 GBytes 9877 Mbits/sec 310 1.36 MBytes   
[ 14] 10.00-11.00 sec 1.15 GBytes 9890 Mbits/sec 90 1.32 MBytes   
[ 14] 11.00-12.00 sec 1.15 GBytes 9889 Mbits/sec 123 1.54 MBytes   
[ 14] 12.00-13.00 sec 1.15 GBytes 9898 Mbits/sec 72 2.05 MBytes   
[ 14] 13.00-14.00 sec 1.15 GBytes 9896 Mbits/sec 86 1.49 MBytes   
[ 14] 14.00-15.00 sec 1.15 GBytes 9901 Mbits/sec 96 1.16 MBytes   
[ 14] 15.00-16.00 sec 1.15 GBytes 9878 Mbits/sec 126 1.48 MBytes   
[ 14] 16.00-17.00 sec 1.15 GBytes 9896 Mbits/sec 132 1.55 MBytes   
[ 14] 17.00-18.00 sec 1.15 GBytes 9901 Mbits/sec 10 1.84 MBytes   
[ 14] 18.00-19.00 sec 1.15 GBytes 9906 Mbits/sec 0 2.58 MBytes   
[ 14] 19.00-20.00 sec 1.15 GBytes 9865 Mbits/sec 395 1.17 MBytes   
- - - - - - - - - - - - - - - - - - - - - - - - -  
[ ID] Interval Transfer Bandwidth Retr  
[ 14] 0.00-20.00 sec 23.0 GBytes 9893 Mbits/sec 2481 sender  
[ 14] 0.00-20.00 sec 23.1 GBytes 9910 Mbits/sec receiver  
  
iperf Done.

SENDER END