Troubleshooting Network Performance Issues with Active Monitoring

NANOG 64
Brian Tierney, ESnet, bltierney@es.net
June 2, 2015

This document is a result of work by the perfSONAR Project (http://www.perfsonar.net) and is licensed under CC BY-SA 4.0 (https://creativecommons.org/licenses/by-sa/4.0/).
Who Am I?

• My Background
  – Lawrence Berkeley National Lab
  – ESnet
  – Big Science Data
Energy Sciences Network

- Connects Department of Energy National Laboratories to universities and research institutions around the world
- Many sites with 100G connections to ESnet today - Berkeley, Livermore, Stanford, Fermi, Brookhaven, Oakridge, Argonne
ESnet / DOE National Lab Network Profile

- Small-ish numbers of very large flows over very long distances:
  - Between California, Illinois, New York, Tennessee, Switzerland
- High-speed “Access” links - 100G sites connected to 100G core
- Nx10G hosts, future Nx40G hosts, dedicated to Data Transfer
- GridFTP / Globus Online / Parallel FTP
- LHC detectors to data centers around the world (future 180Gbps)
- Electron microscopes to supercomputers (20k – 100k FPS per camera)
Introduction & Motivation

Brian Tierney, ESnet
bltierney@es.net
June 2, 2015

This document is a result of work by the perfSONAR Project (http://www.perfsonar.net) and is licensed under CC BY-SA 4.0 (https://creativecommons.org/licenses/by-sa/4.0/).
perfSONAR Goals

• Originally developed to help troubleshoot large science data transfers
  – E.g.: LHC at CERN; Large genomic data sets, etc

• Useful for any network doing large file transfers. E.g.:
  – Into Amazon EC2
  – Large HD movies (Hollywood post production)
  – Many new “big data” applications
What is perfSONAR?

• perfSONAR is a tool to:
  – Set (hopefully raise) network performance expectations
  – Find network problems (“soft failures”)
  – Help fix these problems
• All in multi-domain environments
• These problems are all harder when multiple networks are involved

• perfSONAR is provides a standard way to publish active and passive monitoring data
  – This data is interesting to network researchers as well as network operators
Problem Statement

• In practice, performance issues are prevalent and distributed.
• When a network is underperforming or errors occur, it is difficult to identify the source, as problems can happen anywhere, in any domain.
• Local-area network testing is not sufficient, as errors can occur between networks.
Where Are The Problems?

Congested or faulty links between domains

Latency dependant problems inside domains with small RTT

Congested intra-campus links
Local Testing Will Not Find Everything

Performance is poor when RTT exceeds ~10 ms

Performance is good when RTT is < ~10 ms

Switch with small buffers
A small amount of packet loss makes a huge difference in TCP performance

With loss, high performance beyond metro distances is essentially impossible
Soft Network Failures

- Soft failures are where basic connectivity functions, but high performance is not possible.
- TCP was intentionally designed to hide all transmission errors from the user:
  - “As long as the TCPs continue to function properly and the internet system does not become completely partitioned, no transmission errors will affect the users.” (From IEN 129, RFC 716)
- Some soft failures only affect high bandwidth long RTT flows.
- Hard failures are easy to detect & fix
  - soft failures can lie hidden for years!
- One network problem can often mask others
- Hardware counters can lie!
Hard vs. Soft Failures

• “Hard failures” are the kind of problems every organization understands
  – Fiber cut
  – Power failure takes down routers
  – Hardware ceases to function

• Classic monitoring systems are good at alerting hard failures
  – i.e., NOC sees something turn red on their screen
  – Engineers paged by monitoring systems

• “Soft failures” are different and often go undetected
  – Basic connectivity (ping, traceroute, web pages, email) works
  – Performance is just poor

• How much should we care about soft failures?
Causes of Packet Loss

- Network Congestion
  - Easy to confirm via SNMP, easy to fix with $$
  - This is not a ‘soft failure’, but just a network capacity issue
  - Often people assume congestion is the issue when it fact it is not.
- Under-buffered switch dropping packets
  - Hard to confirm
- Under-powered firewall dropping packets
  - Hard to confirm
- Dirty fibers or connectors, failing optics/light levels
  - Sometimes easy to confirm by looking at error counters in the routers
- Overloaded or slow receive host dropping packets
  - Easy to confirm by looking at CPU load on the host
Sample Soft Failure: failing optics

- **failing optics**: Gb/s performance degrading one month before repair.
Sample Soft Failure: Under-powered Firewall

Inside the firewall
- One direction severely impacted by firewall
- Not useful for science data

Outside the firewall
- Good performance in both directions
Sample Soft Failure: Host Tuning

MTU Changed to 9000
TCP Window settings changed

MTU = 1500 on 10G Host

Graph Key
- Src-Dst throughput
- Dst-Src throughput

perfSONAR BWCTL Graph

Throughput test between Source: nersc-pty1.es.net(198.129.254.22) -> Destination:
198.125.240.244(198.125.240.244)

10s BWCTL TCP Testing
Timezone: GMT-0400 (EDT)
Soft Failure: Under-buffered Switches

Average TCP results, various switches

• Buffers per 10G egress port, 2x parallel TCP streams,
• 50ms simulated RTT, 2Gbps UDP background traffic

<table>
<thead>
<tr>
<th>Buffer Size</th>
<th>Switch 1</th>
<th>Switch 2</th>
<th>Switch 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>1MB</td>
<td>Brocade MLXe¹</td>
<td>9MB Arista 7150</td>
<td>16MB Cisco 6704</td>
</tr>
<tr>
<td>9MB</td>
<td>16MB Cisco 6704</td>
<td>64MB Brocade MLXe¹</td>
<td>90MB Cisco 6716²</td>
</tr>
<tr>
<td>16MB</td>
<td>64MB Brocade MLXe¹</td>
<td>90MB Cisco 6716²</td>
<td>200MB Cisco 6716³</td>
</tr>
<tr>
<td>20MB</td>
<td>90MB Cisco 6716²</td>
<td>200MB Cisco 6716³</td>
<td>VOQ Arista 7504</td>
</tr>
</tbody>
</table>

6/2/15, 20
What is perfSONAR?

- perfSONAR is a tool to:
  - Set network performance expectations
  - Find network problems ("soft failures")
  - Help fix these problems
  - All in multi-domain environments

- These problems are all harder when multiple networks are involved

- perfSONAR is provides a standard way to publish active and passive monitoring data

- perfSONAR = Measurement Middleware
  - *You can’t fix what you can’t find*
  - *You can’t find what you can’t see*
  - *perfSONAR lets you see*
perfSONAR History

- perfSONAR can trace its origin to the Internet2 “End 2 End performance Initiative” from the year 2000.

- What has changed since 2000?
  - The Good News:
    - TCP is much less fragile; Cubic is the default CC alg, autotuning is and larger TCP buffers are everywhere
    - Reliable parallel transfers via tools like Globus Online
    - High-performance UDP-based commercial tools like Aspera
    - *more good news in latest Linux kernel, but it will take 3-4 years before this is widely deployed*
  - The Bad News:
    - The wizard gap is still large
    - Jumbo frame use is still small
    - Under-buffered and switches and routers are still common
    - Under-powered/misconfigured firewalls are common
    - Soft failures still go undetected for months
    - User performance expectations are still too low
The perfSONAR collaboration

- The perfSONAR collaboration is a Open Source project lead by ESnet, Internet2, Indiana University, and GEANT.
  - Each organization has committed 1.5 FTE effort to the project
  - Plus additional help from many others in the community (OSG, RNP, SLAC, and more)

- The perfSONAR Roadmap is influence by
  - requests on the project issue tracker
  - annual user surveys sent to everyone on the user list
  - regular meetings with VO using perfSONAR such as the WLCG and OSG
  - discussions at various perfSONAR related workshops

- Based on the above, every 6-12 months the perfSONAR governance group meets to prioritize features based on:
  - impact to the community
  - level of effort required to implement and support
  - availability of someone with the right skill set for the task
perfSONAR Toolkit

• The “perfSONAR Toolkit” is an open source implementation and packaging of the perfSONAR measurement infrastructure and protocols
  – [http://www.perfsonar.net](http://www.perfsonar.net)

• All components are available as RPMs, and bundled into a CentOS 6-based “netinstall” (Debian support in next release)

• perfSONAR tools are much more accurate if run on a dedicated perfSONAR host, not on the data server

• Easy to install and configure
  • Usually takes less than 30 minutes
perfSONAR Toolkit Components

• BWCTL for scheduling periodic throughout (iperf, iperf3), ping and traceroute tests
• OWAMP for measuring one-way latency and packet loss (RFC4856)
• “esmond” for storing measurements, summarizing results and making available via REST API
• Lookup service for discovering other testers
• On-demand test tools such as NDT (Network Diagnostic Test)
• Configuration GUIs to assist with managing all of the above
• Graphs to display measurement results
Who is running perfSONAR?

- Currently over 1400 deployments world-wide

http://stats.es.net/ServicesDirectory/
perfSONAR Deployment Growth
Active vs. Passive Monitoring

Passive monitoring systems have limitations

- Performance problems are often only visible at the ends
- Individual network components (e.g. routers) have no knowledge of end-to-end state
- perfSONAR tests the network in ways that passive monitoring systems do not

More perfSONAR hosts = better network visibility

- There are now enough perfSONAR hosts in the world to be quite useful
perfSONAR Dashboard: Raising Expectations and improving network visibility

Status at-a-glance
• Packet loss
• Throughput
• Correctness

Current live instances at:
• [http://ps-dashboard.es.net/](http://ps-dashboard.es.net/)
• And many more

Drill-down capabilities:
• Test history between hosts
• Ability to correlate with other events
• Very valuable for fault localization and isolation
perfSONAR Hardware

• These days you can get a good 1U host capable of pushing 10Gbps TCP for around $500 (+10G NIC cost).
  – See perfSONAR user list

• And you can get a host capable of 1G for around $100!
  – Intel Celeron-based (ARM is not fast enough)
  – e.g.: http://www.newegg.com/Product/Product.aspx?Item=N82E16856501007

• VMs are not recommended
  – Tools work better if can guarantee NIC isolation
Active and Growing perfSONAR Community

• Active email lists and forums provide:
  – Instant access to advice and expertise from the community.
  – Ability to share metrics, experience and findings with others to help debug issues on a global scale.

• Joining the community automatically increases the reach and power of perfSONAR
  – The more endpoints means exponentially more ways to test and discover issues, compare metrics
perfSONAR Community

• The perfSONAR collaboration is working to build a strong user community to support the use and development of the software.

• perfSONAR Mailing Lists
  – Announcement Lists:
    • https://mail.internet2.edu/wws/subrequest/perfsonar-announce
  – Users List:
    • https://mail.internet2.edu/wws/subrequest/perfsonar-users
Use Cases & Success Stories

Brian Tierney, ESnet, bltierney@es.net
June 2, 2015

This document is a result of work by the perfSONAR Project (http://www.perfsonar.net) and is licensed under CC BY-SA 4.0 (https://creativecommons.org/licenses/by-sa/4.0/).
Success Stories - Failing Optic(s)

- normal performance
- degrading performance
- repair

one month

Source: nersc-pt1.es.net (198.129.254.22) -- Destination: sunn-pt1.es.net

Gb/s

6/2/15
Success Stories - Failing Optic(s)

- Adding an Attenuator to a Noisy (discovered via OWAMP) Link
Success Stories – Firewall trouble
Success Stories – Firewall trouble

• Results to host behind the firewall:

![Graph showing throughput test results between specific sources and destinations.](image-url)
Success Stories – Firewall trouble

• In front of the firewall:

![Graph of Throughput test between Source and Destination](image)

- Throughput test between Source: `ntg-perfsonar.services.brown.edu` (128.148.230.33) and Destination: `perfig.colorado.edu` (198.59.55.26)
Success Stories – Firewall trouble

• Want more proof – let’s look at a measurement tool through the firewall.
  — Measurement tools emulate a well behaved application
• ‘Outbound’, not filtered:

```
nuttcp -T 10 -i 1 -p 10200 bwctl.newy.net.internet2.edu
  92.3750 MB / 1.00 sec = 774.3069 Mbps  0 retrans
111.8750 MB / 1.00 sec = 938.2879 Mbps  0 retrans
111.8750 MB / 1.00 sec = 938.3019 Mbps  0 retrans
111.7500 MB / 1.00 sec = 938.1606 Mbps  0 retrans
111.8750 MB / 1.00 sec = 938.3198 Mbps  0 retrans
111.8750 MB / 1.00 sec = 938.2653 Mbps  0 retrans
111.8750 MB / 1.00 sec = 938.1931 Mbps  0 retrans
111.9375 MB / 1.00 sec = 938.4808 Mbps  0 retrans
111.6875 MB / 1.00 sec = 937.6941 Mbps  0 retrans
111.8750 MB / 1.00 sec = 938.3610 Mbps  0 retrans
```

1107.9867 MB / 10.13 sec = 917.2914 Mbps 13 %TX 11 %RX 0 retrans 8.38 msRTT
Success Stories – Firewall trouble

• ‘Inbound’, filtered:

```
nuttcp -r -T 10 -i 1 -p 10200 bwctl.newy.net.internet2.edu
4.5625 MB / 1.00 sec = 38.1995 Mbps 13 retrans
4.8750 MB / 1.00 sec = 40.8956 Mbps 4 retrans
4.8750 MB / 1.00 sec = 40.8954 Mbps 6 retrans
6.4375 MB / 1.00 sec = 54.0024 Mbps 9 retrans
5.7500 MB / 1.00 sec = 48.2310 Mbps 8 retrans
5.8750 MB / 1.00 sec = 49.2880 Mbps 5 retrans
6.3125 MB / 1.00 sec = 52.9006 Mbps 3 retrans
5.3125 MB / 1.00 sec = 44.5653 Mbps 7 retrans
4.3125 MB / 1.00 sec = 36.2108 Mbps 7 retrans
5.1875 MB / 1.00 sec = 43.5186 Mbps 8 retrans

53.7519 MB / 10.07 sec = 44.7577 Mbps 0 %TX 1 %RX 70 retrans 8.29 msRTT
```
Success Stories - Host Tuning

- long path (~70ms), single stream TCP, 10G cards, tuned hosts
- Why the nearly 2x uptick? Adjusted `net.ipv4.tcp_rmem/wmem` maximums (used in auto tuning) to 64M instead of 16M.

perfSONAR BWCTL Graph

Graph Image

Throughput test between Source: pnwg-pt1.es.net -- Destination: wash-pt1.es.net

perfSONAR
Success Stories - Fiber Cut

- perfSONAR can't fix fiber cuts, but you can see the loss event and the latency change due to traffic re-routing
Success Stories - Monitoring TA Links

The image shows a graph from MaDDash monitoring system. The graph illustrates a packet loss event, route change to prior link, latency improvement, throughput back to normal, corresponding low throughput, loss event, and throughput events. The data covers dates from Oct 14 to Oct 16, 2014.
Success Stories - MTU Changes


Graph Key
- Blue: Src-Dst throughput
- Green: Dst-Src throughput

MTU Settings changed from 1500 to 9000

Timezone: GMT-0600 (MDT)
Success Stories - Host NIC Speed Mismatch

Sending from a 10G host to a 1G host leads to unstable performance

http://fasterdata.es.net/performance-testing/troubleshooting/interface-speed-mismatch/
Another Failing Line Card Example
Success Stories Summary

• **Key Message(s):**
• This type of active monitoring find problems in the network and on the end hosts.

• Ability to view long term trends in latency, loss, and throughput is very useful
  • Ability to show someone a plot, and say “what did you do on Day X? It broke something.”
  • Ability to show impact of an upgrade (or lack thereof).
Deployment & Advanced Regular Testing Strategies

Brian Tierney, ESnet, bltierney@es.net
June 2, 2015

This document is a result of work by the perfSONAR Project (http://www.perfsonar.net) and is licensed under CC BY-SA 4.0 (https://creativecommons.org/licenses/by-sa/4.0/).
Importance of Regular Testing

• We can’t wait for users to report problems and then fix them (soft failures can go unreported for years!)
• Things just break sometimes
  – Failing optics
  – Somebody messed around in a patch panel and kinked a fiber
  – Hardware goes bad
• Problems that get fixed have a way of coming back
  – System defaults come back after hardware/software upgrades
  – New employees may not know why the previous employee set things up a certain way and back out fixes
• Important to continually collect, archive, and alert on active throughput test results
A perfSONAR Dashboard

http://ps-dashboard.es.net

(google “perfSONAR Maddash” for other public dashboards)
perfSONAR Deployment Locations

• perfSONAR hosts are most useful if they are deployed near key resources such as data servers

• More perfSONAR hosts allow segments of the path to be tested separately
  – Reduced visibility for devices between perfSONAR hosts
  – Must rely on counters or other means where perfSONAR can’t go
Example perfSONAR Host Placement
Regular perfSONAR Tests

• We run regular tests to check for three things
  – TCP throughput
  – One way packet loss and delay
  – traceroute
• perfSONAR has mechanisms for managing regular testing between perfSONAR hosts
  – Statistics collection and archiving
  – Graphs
  – Dashboard display
  – Integration with NAGIOS
• Many public perfSONAR hosts around the world that you may be able to test against
Regular Testing

• Options:
  – Beacon: Let others test to you
    • No regular testing configuration is needed
  – Island: Pick some hosts to test to – you store the data locally.
    • No coordination with others is needed
  – Mesh: full coordination between you and others
    • Use a testing configuration that includes tests to everyone, and incorporate into a dashboard
Regular Testing - Beacon

• The beacon setup is typically employed by a network provider (regional, backbone, exchange point)
  – A service to the users (allows people to test into the network)
  – Can be configured with Layer 2 connectivity if needed
  – If no regular tests are scheduled, minimum requirements for local storage.
  – Makes the most sense to enable all services (bandwidth and latency)
Regular Testing - Island

- The island setup allows a site to test against any number of the perfSONAR nodes around the world, and store the data locally.
  - No coordination required with other sites
  - Allows a view of near horizon testing (e.g. short latency – campus, regional) and far horizon (backbone network, remote collaborators).
  - OWAMP is particularly useful for determining packet loss in the previous cases.
Regular Testing - Mesh

- A full mesh requires more coordination:
  - A full mesh means all hosts involved are running the same test configuration
  - A partial mesh could mean only a small number of related hosts are running a testing configuration
- In either case – bandwidth and latency will be valuable test cases
Develop a Test Plan

• What are you going to measure?
  – Achievable bandwidth
    • 2-3 regional destinations
    • 4-8 important collaborators
    • 4-6 times per day to each destination
    • 10-20 second tests within a region, maybe longer across oceans and continents
  – Loss/Availability/Latency
    • OWAMP: ~10-20 collaborators over diverse paths
  – Interface Utilization & Errors (via SNMP)

• What are you going to do with the results?
  – NAGIOS Alerts
  – Reports to user community?
  – Internal and/or external Dashboard
http://stats.es.net/ServicesDirectory/
Common Questions

• Q: Do most perfSONAR hosts accept tests from anyone?
  – A: Not most, but some do. ESnet allows owamp and iperf TCP tests from all R&E address space, but not the commercial internet.

• Q: Will perfSONAR test traffic step on my production traffic?
  – A: TCP is designed to be friendly; ESnet tags all our internal tests as ‘scavenger’ to ensure tests traffic is dropped first. But too much testing can impact production traffic. Need to be careful of this.

• Q: How can I control who can run tests to my host?
  – A: router ACLs, host ACLs, bwctld.limits
  – Future version of bwctl will include even more options to limit testing
    • only allow iperf between 12am and 6am
    • only allow 2 tests per day from all but the following subnets
Use of Measurement Tools

Brian Tierney, ESnet, bltierney@es.net
June 2, 2015

This document is a result of work by the perfSONAR Project (http://www.perfsonar.net) and is licensed under CC BY-SA 4.0 (https://creativecommons.org/licenses/by-sa/4.0/).
Tool Usage

• All of the previous examples were discovered, debugged, and corrected through the aide of the tools that are on the perfSONAR toolkit
  • perfSONAR specific tools
    • bwctl
    • owamp
  • Standard Unix tools
    • ping, traceroute, tracepath
  • Standard Unix add-on tools
    • Iperf, iperf3, nuttcp
Default perfSONAR Throughput Tool: iperf3

• iperf3 (http://software.es.net/iperf/) is a new implementation if iperf from scratch, with the goal of a smaller, simpler code base, and a library version of the functionality that can be used in other programs.

• Some new features in iperf3 include:
  – reports the number of TCP packets that were retransmitted and CWND
  – reports the average CPU utilization of the client and server (-V flag)
  – support for zero copy TCP (-Z flag)
  – JSON output format (-J flag)
  – “omit” flag: ignore the first N seconds in the results

• On RHEL-based hosts, just type ‘yum install iperf3’

Sample iperf3 output on lossy network

- Performance is < 1Mbps due to heavy packet loss

<table>
<thead>
<tr>
<th>ID</th>
<th>Interval</th>
<th>Transfer</th>
<th>Bandwidth</th>
<th>Retr</th>
<th>Cwnd</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>0.00-1.00</td>
<td>sec 139 MBytes</td>
<td>1.16 Gbits/sec</td>
<td>257</td>
<td>33.9 KBytes</td>
</tr>
<tr>
<td>15</td>
<td>1.00-2.00</td>
<td>sec 106 MBytes</td>
<td>891 Mbits/sec</td>
<td>138</td>
<td>26.9 KBytes</td>
</tr>
<tr>
<td>15</td>
<td>2.00-3.00</td>
<td>sec 105 MBytes</td>
<td>881 Mbits/sec</td>
<td>132</td>
<td>26.9 KBytes</td>
</tr>
<tr>
<td>15</td>
<td>3.00-4.00</td>
<td>sec 71.2 MBytes</td>
<td>598 Mbits/sec</td>
<td>161</td>
<td>15.6 KBytes</td>
</tr>
<tr>
<td>15</td>
<td>4.00-5.00</td>
<td>sec 110 MBytes</td>
<td>923 Mbits/sec</td>
<td>123</td>
<td>43.8 KBytes</td>
</tr>
<tr>
<td>15</td>
<td>5.00-6.00</td>
<td>sec 136 MBytes</td>
<td>1.14 Gbits/sec</td>
<td>122</td>
<td>58.0 KBytes</td>
</tr>
<tr>
<td>15</td>
<td>6.00-7.00</td>
<td>sec 88.8 MBytes</td>
<td>744 Mbits/sec</td>
<td>140</td>
<td>31.1 KBytes</td>
</tr>
<tr>
<td>15</td>
<td>7.00-8.00</td>
<td>sec 112 MBytes</td>
<td>944 Mbits/sec</td>
<td>143</td>
<td>45.2 KBytes</td>
</tr>
<tr>
<td>15</td>
<td>8.00-9.00</td>
<td>sec 119 MBytes</td>
<td>996 Mbits/sec</td>
<td>131</td>
<td>32.5 KBytes</td>
</tr>
<tr>
<td>15</td>
<td>9.00-10.00</td>
<td>sec 110 MBytes</td>
<td>923 Mbits/sec</td>
<td>182</td>
<td>46.7 KBytes</td>
</tr>
</tbody>
</table>
• BWCTL is the wrapper around all the perfSONAR tools
• Policy specification can do things like prevent tests to subnets, or allow longer tests to others. See the man pages for more details
• Some general notes:
  – Use ‘-c’ to specify a ‘catcher’ (receiver)
  – Use ‘-s’ to specify a ‘sender’
  – Will default to IPv6 if available (use -4 to force IPv4 as needed, or specify things in terms of an address if your host names are dual homed)
bwctl features

• BWCTL lets you run any of the following between any 2 perfSONAR nodes:
  – iperf3, nuttcp, ping, owping, traceroute, and tracepath

• Sample Commands:
  • bwctl -c psmsu02.aglt2.org -s elpa-pts1.es.net -T iperf3
  • bwping -s atla-pts1.es.net -c ga-pts1.es.net
  • bwping -E -c www.google.com
  • bwtraceroute -T tracepath -c lbl-pts1.es.net -l 8192 -s atla-pts1.es.net
  • bwping -T owamp -s atla-pts1.es.net -c ga-pts1.es.net -N 1000 -i .01
Host Issues: Throughput is dependent on TCP Buffers

• A prequel to using BWCTL throughput tests – The Bandwidth Delay Product
  – The amount of “in flight” data allowed for a TCP connection (BDP = bandwidth * round trip time)
  – Example: 1Gb/s cross country, ~100ms
    • 1,000,000,000 b/s * .1 s = 100,000,000 bits
    • 100,000,000 / 8 = 12,500,000 bytes
    • 12,500,000 bytes / (1024*1024) ~ 12MB
  – Most OSs have a default TCP buffer size of 4MB per flow.
    • This is too small for long paths
    • More details at https://fasterdata.es.net/host-tuning/

• Host admins need to make sure there is enough TCP buffer space to keep the sender from waiting for acknowledgements from the receiver
Network Throughput

• Start with a definition:
  – **network throughput** is the rate of successful message delivery over a communication channel
  – Easier terms: how much data can I shovel into the network for some given amount of time

• What does this tell us?
  – Opposite of utilization (e.g. its how much we can get at a given point in time, minus what is utilized)
  – Utilization and throughput added together are capacity

• Tools that measure throughput are a simulation of a real work use case (e.g. how well might bulk data movement perform)

• Ways to game the system
  – Parallel streams
  – Manual window size adjustments
  – ‘memory to memory’ testing – no disk involved
Throughput Test Tools

- Varieties of throughput tools that BWCTL knows how to manage:
  - Iperf (v2)
    - Default for the command line (e.g. bwctl –c HOST will invoke this)
    - Some known behavioral problems (CPU hog, hard to get UDP testing to be correct)
  - Iperf3
    - Default for the perfSONAR regular testing framework, can invoke via command line switch (bwctl –T iperf3 –c HOST)
  - Nuttcp
    - Different code base, can invoke via command line switch (bwctl –T nuttcp –c HOST)
    - More control over how the tool behaves on the host (bind to CPU/core, etc.)
    - Similar feature set to iperf3
BWCTL Example (iperf)

> bwctl -T iperf -f m -t 10 -i 2 -c sunn-pt1.es.net
bwctl: 83 seconds until test results available
bwctl: exec_line: /usr/bin/iperf -B 198.129.254.58 -s -f m -m -p 5136 -t 10 -i 2.000000
bwctl: run_tool: tester: iperf
bwctl: run_tool: receiver: 198.129.254.58
bwctl: run_tool: sender: 198.124.238.34
bwctl: start_tool: 3598657357.738868

Server listening on TCP port 5136
Binding to local address 198.129.254.58
TCP window size: 0.08 MByte (default)

[ 16] local 198.129.254.58 port 5136 connected with 198.124.238.34 port 5136
[ ID] Interval         Transfer    Bandwidth
[ 16]  0.0- 2.0 sec     90.4 MBytes     379 Mbits/sec
[ 16]  2.0- 4.0 sec     689 MBytes     2891 Mbits/sec
[ 16]  4.0- 6.0 sec     684 MBytes     2867 Mbits/sec
[ 16]  6.0- 8.0 sec     691 MBytes     2897 Mbits/sec
[ 16]  8.0-10.0 sec     691 MBytes     2898 Mbits/sec
[ 16]  0.0-10.0 sec     2853 MBytes     2386 Mbits/sec
[ 16] MSS size 8948 bytes (MTU 8988 bytes, unknown interface)

This is what perfSONAR Graphs – the average of the complete test
BWCTL Example (iperf3)

> bwctl -T iperf3 -t 10 -i 2 -c sunn-pt1.es.net

bwctl: run_tool: tester: iperf3
bwctl: run_tool: receiver: 198.129.254.58
bwctl: run_tool: sender: 198.124.238.34
bwctl: start_tool: 3598657653.219168
Test initialized
Running client
Connecting to host 198.129.254.58, port 5001
[ 17] local 198.124.238.34 port 34277 connected to 198.129.254.58 port 5001
[ ID] Interval            Transfer   Bandwidth       Retransmits
[ 17]   0.00-2.00   sec   430 MBytes 1.80 Gbits/sec  2
[ 17]   2.00-4.00   sec   680 MBytes 2.85 Gbits/sec  0
[ 17]   4.00-6.00   sec   669 MBytes 2.80 Gbits/sec  0
[ 17]   6.00-8.00   sec   670 MBytes 2.81 Gbits/sec  0
[ 17]   8.00-10.00  sec   680 MBytes 2.85 Gbits/sec  0
[ ID] Interval            Transfer   Bandwidth       Retransmits
Sent
[ 17]   0.00-10.00  sec  3.06 GBytes 2.62 Gbits/sec  2
Received
[ 17]   0.00-10.00  sec  3.06 GBytes 2.63 Gbits/sec
iperf Done.
bwctl: stop_tool: 3598657664.995604

This is what perfSONAR Graphs – the average of the complete test
BWCTL Example (nuttcp)

```bash
> bwctl -T nuttcp -f m -t 10 -i 2 -c sunn-plt1.es.net
nuttcp-t: buflen=65536, nstream=1, port=5001 tcp -> 198.129.254.58
nuttcp-t: connect to 198.129.254.58 with mss=8948, RTT=62.418 ms
nuttcp-t: send window size = 98720, receive window size = 87380
nuttcp-t: available send window = 74040, available receive window = 65535
nuttcp-r: buflen=65536, nstream=1, port=5001 tcp
nuttcp-r: send window size = 98720, receive window size = 87380
nuttcp-r: available send window = 74040, available receive window = 65535
```

<table>
<thead>
<tr>
<th>Buffer Size</th>
<th>Throughput</th>
<th>Retransmissions</th>
</tr>
</thead>
<tbody>
<tr>
<td>131.0625 MB</td>
<td>549.7033 Mbps</td>
<td>1 retrans</td>
</tr>
<tr>
<td>725.6250 MB</td>
<td>3043.4964 Mbps</td>
<td>0 retrans</td>
</tr>
<tr>
<td>715.0000 MB</td>
<td>2998.8284 Mbps</td>
<td>0 retrans</td>
</tr>
<tr>
<td>714.3750 MB</td>
<td>2996.4168 Mbps</td>
<td>0 retrans</td>
</tr>
<tr>
<td>707.1250 MB</td>
<td>2965.8349 Mbps</td>
<td>0 retrans</td>
</tr>
</tbody>
</table>

nuttcp-t: 2998.1379 MB in 10.00 real seconds = 307005.08 KB/sec = 2514.9856 Mbps
nuttcp-t: 2998.1379 MB in 2.32 CPU seconds = 1325802.48 KB/cpu sec
nuttcp-t: retrans = 1
nuttcp-t: 47971 I/O calls, msec/call = 0.21, calls/sec = 4797.03
nuttcp-t: 0.0user 2.3sys 0:10real 23% 0i+0d 768maxrss 0+2pf 156+28csw

nuttcp-r: 2998.1379 MB in 10.07 real seconds = 304959.96 KB/sec = 2498.2320 Mbps
nuttcp-r: 2998.1379 MB in 2.36 CPU seconds = 1301084.31 KB/cpu sec
nuttcp-r: 57808 I/O calls, msec/call = 0.18, calls/sec = 5742.21
nuttcp-r: 0.0user 2.3sys 0:10real 23% 0i+0d 770maxrss 0+4pf 9146+24csw
BWCTL Example (nuttcp, [1%] loss)

> bwctl -T nuttcp -f m -t 10 -i 2 -c sunn-pt1.es.net
bwctl: exec_line: /usr/bin/nuttcp -vv -p 5004 -i 2.000000 -T 10 -t 198.129.254.58
nuttcp-t: buflen=65536, nstream=1, port=5004 tcp -> 198.129.254.58
nuttcp-t: connect to 198.129.254.58 with mss=8948, RTT=62.440 ms
nuttcp-t: send window size = 98720, receive window size = 87380
nuttcp-t: available send window = 74040, available receive window = 65535
nuttcp-r: send window size = 98720, receive window size = 87380
nuttcp-r: available send window = 74040, available receive window = 65535

<table>
<thead>
<tr>
<th>Transfer Size</th>
<th>Transfer Time</th>
<th>Bandwidth (Mbps)</th>
<th>Retransmissions</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.3125 MB</td>
<td>2.00 sec</td>
<td>26.4759 Mbps</td>
<td>27 retrans</td>
</tr>
<tr>
<td>3.5625 MB</td>
<td>2.00 sec</td>
<td>14.9423 Mbps</td>
<td>4 retrans</td>
</tr>
<tr>
<td>3.8125 MB</td>
<td>2.00 sec</td>
<td>15.9906 Mbps</td>
<td>7 retrans</td>
</tr>
<tr>
<td>4.8125 MB</td>
<td>2.00 sec</td>
<td>20.1853 Mbps</td>
<td>13 retrans</td>
</tr>
<tr>
<td>6.0000 MB</td>
<td>2.00 sec</td>
<td>25.1659 Mbps</td>
<td>7 retrans</td>
</tr>
</tbody>
</table>

nuttcp-t: 25.5066 MB in 10.00 real seconds = 2611.85 KB/sec = 21.3963 Mbps
nuttcp-t: 25.5066 MB in 0.01 CPU seconds = 1741480.37 KB/cpu sec
nuttcp-t: retrans = 58
nuttcp-t: 409 I/O calls, msec/call = 25.04, calls/sec = 40.90
nuttcp-t: 0.0user 0.0sys 0:10real 0% 0i+0d 768maxrss 0+2pf 51+3csw

nuttcp-r: 25.5066 MB in 10.30 real seconds = 2537.03 KB/sec = 20.7833 Mbps
nuttcp-r: 25.5066 MB in 0.02 CPU seconds = 1044874.29 KB/cpu sec
nuttcp-r: 787 I/O calls, msec/call = 13.40, calls/sec = 76.44
nuttcp-r: 0.0user 0.0sys 0:10real 0% 0i+0d 770maxrss 0+4pf 382+0csw

6/2/15
Throughput Expectations

Q: What iperf through should you expect to see on a uncongested 10Gbps network?
A: 3 - 9.9 Gbps, depending on

- RTT
- TCP tuning
- CPU core speed, and ratio of sender speed to receiver speed
• **OWAMP** = One Way Active Measurement Protocol
  – E.g. ‘one way ping’

• **Some differences from traditional ping:**
  – Measure each direction independently (recall that we often see things like congestion occur in one direction and not the other)
  – Uses small evenly spaced groupings of UDP (not ICMP) packets
  – Ability to ramp up the interval of the stream, size of the packets, number of packets

• **OWAMP is most useful for detecting packet train abnormalities on an end to end basis**
  – Loss
  – Duplication
  – Out of order packets
  – Latency on the forward vs. reverse path
  – Number of Layer 3 hops

• Does require some accurate time via NTP – the perfSONAR toolkit does take care of this for you.
What OWAMP Tells Us

• OWAMP is very useful in regular testing
  – Congestion or queuing often occurs in a single direction
  – Packet loss information (and how often/how much occurs over time) is more valuable than throughput
    • This gives you a ‘why’ to go with an observation.
  – If your router is going to drop a 50B UDP packet, it is most certainly going to drop a 1500B/9000B TCP packet

• Overlaying data
  – Compare your throughput results against your OWAMP – do you see patterns?
  – Alarm on each, if you are alarming (and we hope you are alarming …)
OWAMP (initial)

> owping sunn-owamp.es.net
Approximately 12.6 seconds until results available
--- owping statistics from [wash-owamp.es.net]:8885 to [sunn-owamp.es.net]:8827 ---
SID: c681fe4ed67f1b3e5faeb249f078ec8a
100 sent, 0 lost (0.000%), 0 duplicates
one-way delay min/median/max = 31/31.1/31.7 ms, (err=0.00201 ms)
one-way jitter = 0 ms (P95-P50)
Hops = 7 (consistently)
no reordering

--- owping statistics from [sunn-owamp.es.net]:9027 to [wash-owamp.es.net]:8888 ---
SID: c67cfc7ed67f1b3eaab69b94f393bc46
100 sent, 0 lost (0.000%), 0 duplicates
one-way delay min/median/max = 31.4/31.5/32.6 ms, (err=0.00201 ms)
one-way jitter = 0 ms (P95-P50)
Hops = 7 (consistently)
no reordering

This is what perfSONAR Graphs – the average of the complete test
OWAMP (w/ loss)

> owping sunn-owamp.es.net
Approximately 12.6 seconds until results available

--- owping statistics from [wash-owamp.es.net]:8852 to [sunn-owamp.es.net]:8837 ---
SID: c681fe4ed67f1f0908224c341a2b83f3
100 sent, 12 lost (12.000%), 0 duplicates
one-way delay min/median/max = 31.1/31.1/31.3 ms, (err=0.00502 ms)
one-way jitter = nan ms (P95-P50)
Hops = 7 (consistently)
no reordering

--- owping statistics from [sunn-owamp.es.net]:9182 to [wash-owamp.es.net]:8893 ---
SID: c67cfc7ed67f1f09531c87cf38381bb6
100 sent, 0 lost (0.000%), 0 duplicates
one-way delay min/median/max = 31.4/31.5/31.5 ms, (err=0.00502 ms)
one-way jitter = 0 ms (P95-P50)
Hops = 7 (consistently)
no reordering

This is what perfSONAR Graphs – the average of the complete test
OWAMP (w/ re-ordering)

> owping sunn-owamp.es.net
Approximately 12.9 seconds until results available

--- owping statistics from [wash-owamp.es.net]:8814 to [sunn-owamp.es.net]:9062 ---
SID: c681fe4ed67f21d94991ea335b7a1830
100 sent, 0 lost (0.000%), 0 duplicates
one-way delay min/median/max = 31.1/106/106 ms, (err=0.00201 ms)
one-way jitter = 0.1 ms (P95-P50)
Hops = 7 (consistently)
1-reordering = 19.000000%
2-reordering = 1.000000%
no 3-reordering

--- owping statistics from [sunn-owamp.es.net]:8770 to [wash-owamp.es.net]:8939 ---
SID: c67cfc7ed67f21d994c1302dff644543
100 sent, 0 lost (0.000%), 0 duplicates
one-way delay min/median/max = 31.4/31.5/32 ms, (err=0.00201 ms)
one-way jitter = 0 ms (P95-P50)
Hops = 7 (consistently)
no reordering
BWCTL (Traceroute)

```bash
> bwtraceroute -T traceroute -s sacr-pt1.es.net -c wash-pt1.es.net
bwtraceroute: Using tool: traceroute
traceroute to 198.124.238.34 (198.124.238.34), 30 hops max, 60 byte packets
  1  sacrcr5-sacrpt1.es.net (198.129.254.37)  0.490 ms  0.788 ms  1.114 ms
  2  denvcr5-ip-a-sacrcr5.es.net (134.55.50.202)  21.304 ms  21.594 ms  21.924 ms
  3  kanscr5-ip-a-denvcr5.es.net (134.55.49.58)  31.944 ms  32.608 ms  32.838 ms
  4  chiccr5-ip-a-kanscr5.es.net (134.55.43.81)  42.904 ms  43.236 ms  43.566 ms
  5  washcr5-ip-a-chiccr5.es.net (134.55.36.46)  60.046 ms  60.339 ms  60.670 ms
  6  wash-pt1.es.net (198.124.238.34)  59.679 ms  59.693 ms  59.708 ms

> bwtraceroute -T traceroute -c sacr-pt1.es.net -s wash-pt1.es.net
bwtraceroute: Using tool: traceroute
traceroute to 198.129.254.38 (198.129.254.38), 30 hops max, 60 byte packets
  1  wash-te-perf-if1.es.net (198.124.238.33)  0.474 ms  0.816 ms  1.145 ms
  2  chiccr5-ip-a-washcr5.es.net (134.55.36.45)  19.133 ms  19.463 ms  19.786 ms
  3  kanscr5-ip-a-chiccr5.es.net (134.55.43.82)  28.515 ms  28.799 ms  29.083 ms
  4  denvcr5-ip-a-kanscr5.es.net (134.55.49.57)  39.077 ms  39.348 ms  39.628 ms
  5  sacrcr5-ip-a-denvcr5.es.net (134.55.50.201)  60.013 ms  60.299 ms  60.983 ms
  6  sacr-pt1.es.net (198.129.254.38)  59.679 ms  59.678 ms  59.668 ms
```

81
BWCTL (Tracepath)

> bwtraceroute -T tracepath -s sacr-pt1.es.net -c wash-pt1.es.net

bwtraceroute: Using tool: tracepath
bwtraceroute: 36 seconds until test results available

SENDER START

1?: [LOCALHOST]    pmtu 9000
1:  sacrcr5-sacrpt1.es.net (198.129.254.37)            0.489ms
1:  sacrcr5-sacrpt1.es.net (198.129.254.37)            0.463ms
2:  denvcr5-ip-a-sacrcr5.es.net (134.55.50.202)         21.426ms
3:  kanscr5-ip-a-denvcr5.es.net (134.55.49.58)           31.957ms
4:  chiccr5-ip-a-kanscr5.es.net (134.55.43.81)           42.947ms
5:  washcr5-ip-a-chiccr5.es.net (134.55.36.46)           60.092ms
6:  wash-pt1.es.net (198.124.238.34)                    59.753ms reached

Resume: pmtu 9000 hops 6 back 59

SENDER END
BWCTL (Tracepath with MTU mismatch)

> bwtraceroute -T tracepath -c neJest.lbl.gov -s anl-rt1.es.net

bwtraceroute: Using tool: tracepath

1?: [LOCALHOST]  pm tu 9000

1: anl mr2-anlpt1.es.net (198.124.252.118)  0.249ms asymm 2
1: anl mr2-anlpt1.es.net (198.124.252.118)  0.197ms asymm 2
2: no reply
3: kanscr5-ip-a-chiccr5.es.net (134.55.43.82)  13.816ms
4: denvcr5-ip-a-kanscr5.es.net (134.55.49.57)  24.379ms
5: sacrcr5-ip-a-denvcr5.es.net (134.55.50.201)  45.298ms
6: sunncr5-ip-a-sacrcr5.es.net (134.55.40.6)  47.890ms
7: et-3-0-0-1411.er1-n1.lbl.gov (198.129.78.22)  50.093ms
8: t5-4.ir1-n1.lbl.gov (131.243.244.131)  50.772ms
9: t5-4.ir1-n1.lbl.gov (131.243.244.131)  52.669ms  pm tu 1500
9: neJest.lbl.gov (131.243.24.11)  49.239ms reached

Resume: pm tu 1500 hops 9 back 56
BWCTL (Ping)

> bwping -c nettest.lbl.gov -s anl-pt1.es.net

bwping: Using tool: ping
64 bytes from 131.243.24.11: icmp_seq=1 ttl=56 time=49.1 ms
64 bytes from 131.243.24.11: icmp_seq=2 ttl=56 time=49.1 ms
64 bytes from 131.243.24.11: icmp_seq=3 ttl=56 time=49.1 ms
64 bytes from 131.243.24.11: icmp_seq=4 ttl=56 time=49.1 ms
64 bytes from 131.243.24.11: icmp_seq=5 ttl=56 time=49.1 ms

To test to a host not running bwctl, use “-E”

> bwping -E -c www.google.com

bwping: Using tool: ping
PING 2607:f8b0:4010:800::1013(2607:f8b0:4010:800::1013) from 2001:400:2201:1190::3 : 56 data bytes
64 bytes from 2607:f8b0:4010:800::1013: icmp_seq=1 ttl=54 time=48.1 ms
64 bytes from 2607:f8b0:4010:800::1013: icmp_seq=2 ttl=54 time=48.2 ms
64 bytes from 2607:f8b0:4010:800::1013: icmp_seq=3 ttl=54 time=48.2 ms
64 bytes from 2607:f8b0:4010:800::1013: icmp_seq=4 ttl=54 time=48.2 ms
BWCTL (owamp)

> bwping -T owamp -4 -s sacr-pt1.es.net -c wash-pt1.es.net
bwping: Using tool: owamp
bwping: 42 seconds until test results available

--- owping statistics from [198.129.254.38]:5283 to [198.124.238.34]:5121 ---
SID: c67cee22d85fc3b2bbe23f83da5947b2
first: 2015-01-13T08:17:58.534
last: 2015-01-13T08:18:17.581
10 sent, 0 lost (0.000%), 0 duplicates
one-way delay min/median/max = 29.9/29.9/29.9 ms, (err=0.191 ms)
one-way jitter = 0.1 ms (P95-P50)
Hops = 5 (consistently)
no reordering
Testing Pitfalls

• Don’t trust a single result
  – you should run any test a few times to confirm the results
  – Hop-by-hop path characteristics may be continuously changing
• A poor carpenter blames his tools
  – The tools are only as good as the people using them, do it methodically
  – Trust the results – remember that they are giving you a number based on the entire environment
• If the site isn’t using perfSONAR – step 1 is to get them to do so
  – http://www.perfsonar.net
• Get some help from the community
  – perfsonar-user@internet2.edu
Resources

- perfSONAR website
  - http://www.perfsonar.net/
- perfSONAR Toolkit Manual
  - http://docs.perfsonar.net/
- perfSONAR mailing lists
  - http://www.perfsonar.net/about/getting-help/
- perfSONAR directory
  - http://stats.es.net/ServicesDirectory/
- FasterData Knowledgebase
  - http://fasterdata.es.net/
EXTRA SLIDES