http://bit.ly/EPOC-USC-CI

perfSONAR

Monitoring end-to-end systems

Jason Zurawski
zurawski@es.net
ESnet / Lawrence Berkeley National Laboratory

Training Workshop for Network Engineers and Educators on Tools and Protocols for High-Speed Networks
University of South Carolina
July 22-23, 2019

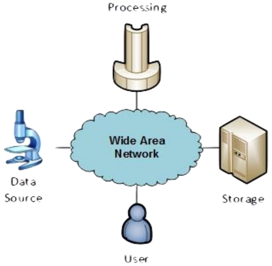
ESnet ENERGY SCIENCES NETWORK   GÉANT   INDIANA UNIVERSITY   INTERNET2   UNIVERSITY OF MICHIGAN

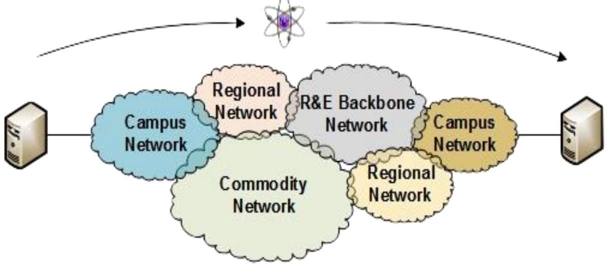
Outline

- **Introduction**
- Hardware & Software
- Tool Use
- Regular Testing
- Use Cases

The R&E Community

- The global Research & Education network ecosystem is comprised of hundreds of international, national, regional and local-scale resources – each independently owned and operated.
- This complex, heterogeneous set of networks ***must*** operate seamlessly from “end to end” to support science and research collaborations that are distributed globally.





- Data mobility is required; there is no liquid market for HPC resources (people use what they can get – DOE, XSEDE, NOAA, etc. etc.)
  - To stay competitive, we must learn the use patterns, and support them
  - This may mean making sure your network, and the networks of others, are functional

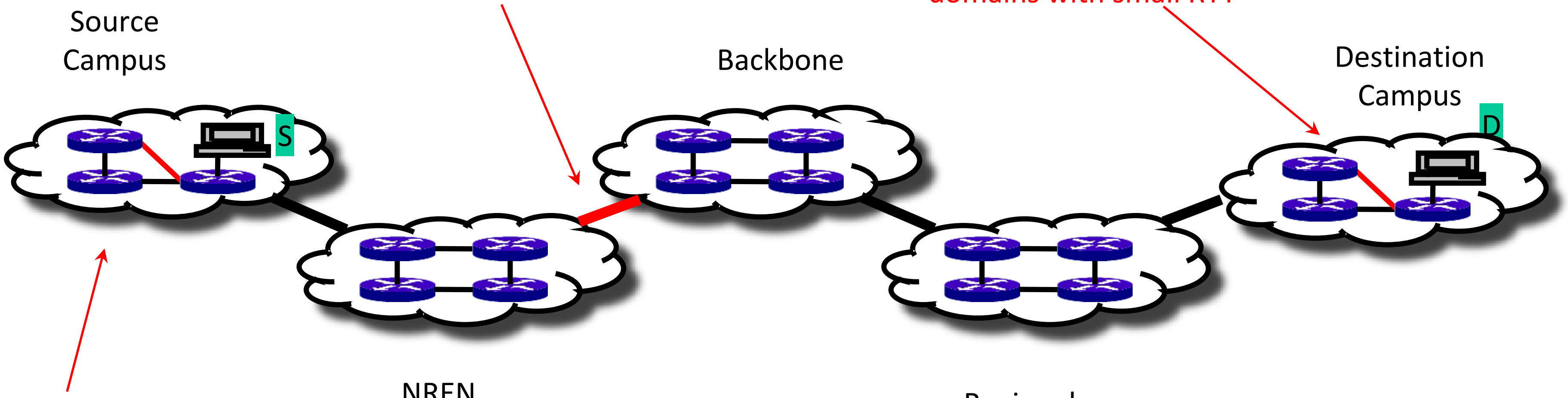
Lets Talk Performance …

"In any large system, there is always something broken."
- ***Jon Postel***

- Modern networks are occasionally designed to be *one-size-fits-most*
  - e.g. if you have ever heard the phrase "converged network", the design is to facilitate CIA (Confidentiality, Integrity, Availability)

- It's all TCP
  - Bulk data movement is a common thread (move the data from the microscope, to the storage, to the processing, to the people – and they are all sitting in different facilities)
  - This fails when TCP suffers due to path problems (***<u>ANYWHERE</u>*** in the path)
  - It's easier to work with TCP than to fix it (20+ years of trying…)
- TCP suffers the most from unpredictability; Packet loss/delays are the enemy
  - Small buffers on the network gear and hosts
  - Incorrect application choice
  - Packet disruption caused by overzealous security
  - Congestion from herds of mice
- It all starts with knowing your users, and knowing your network

Where Are The Problems?



Congested or faulty links between domains

Latency dependant problems inside domains with small RTT

Source Campus

Backbone

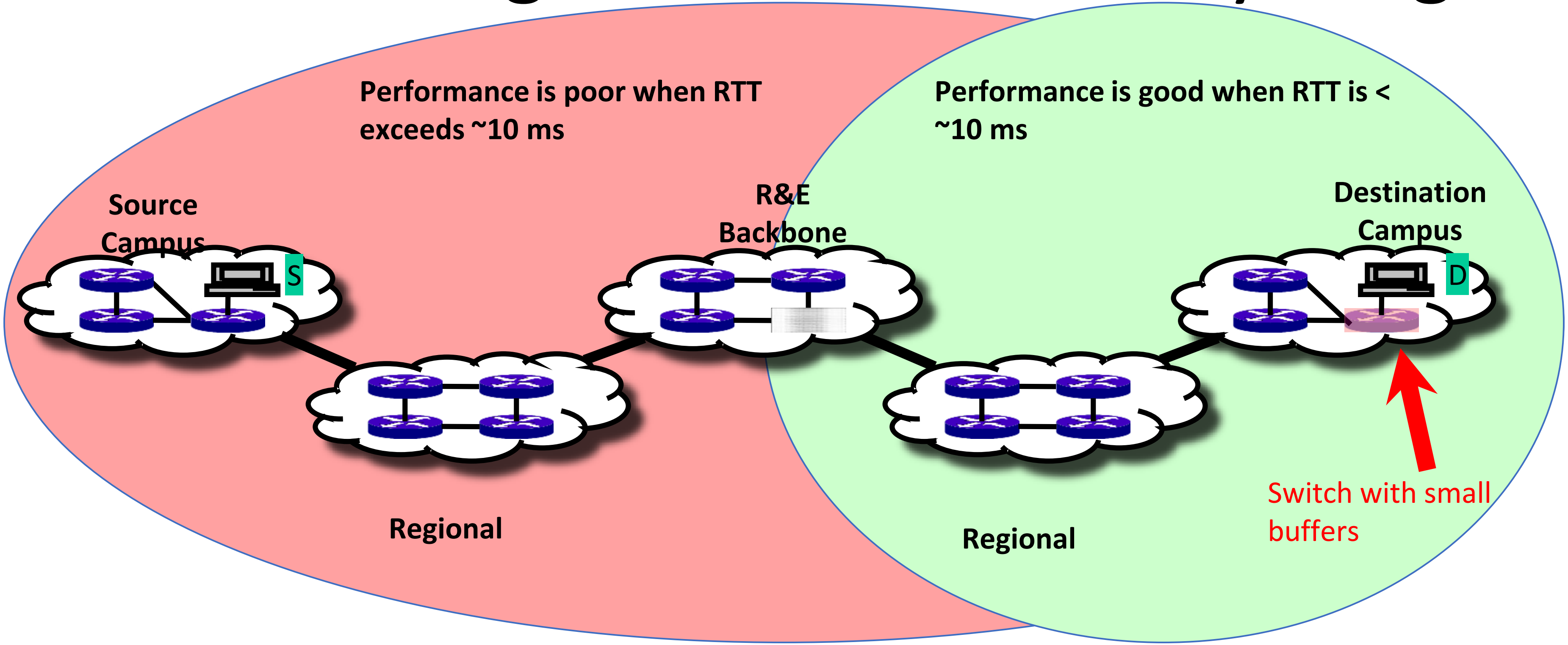
Destination Campus

NREN

Regional

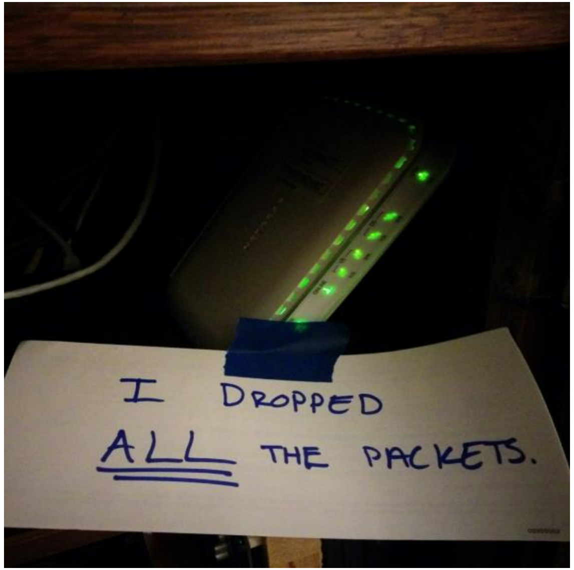
Congested intra- campus links

Local Testing Will Not Find Everything


Performance is poor when RTT exceeds ~10 ms
Performance is good when RTT is < ~10 ms
Source Campus
S
R&E Backbone
Destination Campus
D
Regional
Regional
Switch with small buffers

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



Problem Statement: Hard vs. Soft Failures

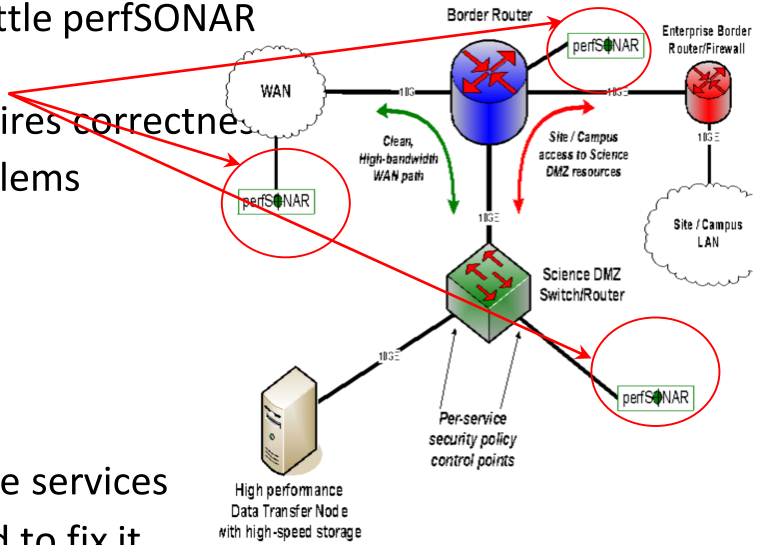
- **“Hard failures” are the kind of problems every organization understands**
  - **Fiber cut**
  - **Power failure takes down routers**
  - **Hardware ceases to function**
- **Classic monitoring systems are good at alerting hard failures**
  - **i.e., NOC sees something turn red on their screen**
  - **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?**

Network Monitoring

- **All networks do some form monitoring.**
  - Addresses needs of local staff for understanding state of the network
    - Would this information be useful to external users?
    - Can these tools function on a multi-domain basis?
- **Beyond passive methods, there are active tools.**
  - E.g. often we want a ‘throughput’ number. Can we automate that idea?
  - Wouldn’t it be nice to get some sort of plot of performance over the course of a day? Week? Year? Multiple endpoints?
- **perfSONAR = Measurement Middleware**

perfSONAR

- All the previous Science DMZ network diagrams have little perfSONAR boxes everywhere
  - The reason for this is that consistent behavior requires correctness
  - Correctness requires the ability to find and fix problems
    - ***You can't fix what you can't find***
    - ***You can't find what you can't see***
    - ***perfSONAR lets you see***
- Especially important when deploying high performance services
  - If there is a problem with the infrastructure, need to fix it
  - If the problem is not with your stuff, need to prove it
    - Many players in an end to end path
    - Ability to show correct behavior aids in problem localization


Border Router
perfSONAR
Enterprise Border Router/Firewall
WAN
10GE
10GE
10GE
10GE
10GE
Clean, High-bandwidth WAN path
Site / Campus access to Science DMZ resources
perfSONAR
Site / Campus LAN
Science DMZ Switch/Router
perfSONAR
Per-service security policy control points
High performance Data Transfer Node with high-speed storage

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
  - This data is interesting to network researchers as well as network operators

Simulating Performance

- It’s infeasible to perform at-scale data movement all the time – as we see in other forms of science, we need to rely on simulations
- Network performance comes down to a couple of key metrics:
  - Throughput (e.g. “how much can I get out of the network”)
  - Latency (time it takes to get to/from a destination)
  - Packet loss/duplication/ordering (for some sampling of packets, do they all make it to the other side without serious abnormalities occurring?)
  - Network utilization (the opposite of “throughput” for a moment in time)
- We can get many of these from a selection of active and passive measurement tools – enter the perfSONAR Toolkit

Outline

- Introduction
- **Hardware & Software**
- Tool Use
- Regular Testing
- Use Cases

perfSONAR Toolkit

- The “perfSONAR Toolkit” is an open source implementation and packaging of the perfSONAR measurement infrastructure and protocols
  - http://docs.perfsonar.net/install_getting.html
- All components are available as RPMs, DEBs, and bundled as CentOS 7, Debian 7,8,9 or Ubuntu 14 and 16 -based packages (as for perfSONAR v. 4.0.1)
  - perfSONAR tools are much more accurate if run on a dedicated perfSONAR host
- Very easy to install and configure
  - Usually takes less than 30 minutes

Hardware Considerations

- http://docs.perfsonar.net/install_hardware.html
- Dedicated perfSONAR hardware is best
  - Server class is a good choice
  - Desktop/Laptop/Mini (Mac, Shuttle, ARM) can be problematic, but work in a diagnostic capacity
- Other applications running may perturb results (and measurement could hurt essential services)
- Running Latency and Throughput on the Same Server
  - If you can devote 2 interfaces – version 3.4 and above of the toolkit will support this.
  - If you can't, note that Throughput tests can cause increased latency and loss (latency tests on a throughput host are still useful however)

perfSONAR

Hardware Considerations

- http://docs.perfsonar.net/install_hardware.html
- 1Gbps vs 10Gbps testers
  - There are a number of problem that only show up at speeds above 1Gbps – both are still super useful
- Virtual Machines do not always work well as perfSONAR hosts (use specific)
  - Clock sync issues are a bit of a factor
  - throughput is reduced significantly for 10G hosts
  - VM technology and motherboard technology has come a long way, YMMV
  - NDT/NAGIOS/SNMP/1G BWCTL are good choices for a VM, OWAMP/10G Throughput are not
  - Docker containers being tested for performance as well; TBD

ESnet ENERGY SCIENCES NETWORK GÉANT INDIANA UNIVERSITY INTERNET2 UNIVERSITY OF MICHIGAN

Preparing The Software

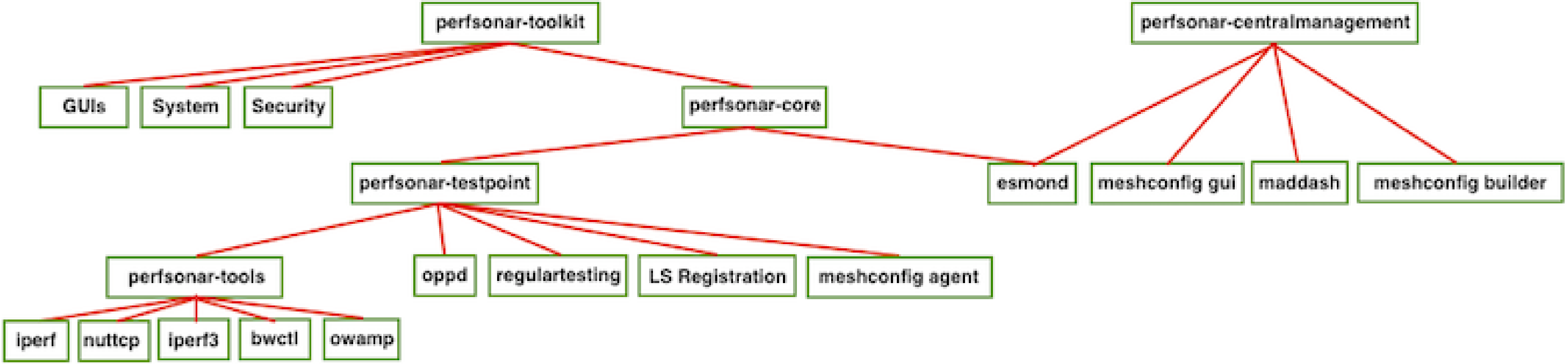
- The best source of information is here:
  - http://docs.perfsonar.net

- The two viewpoints of the perfSONAR Owner:
  - Cattle, not pets: it’s an expendable server that is not tightly integrated (e.g. if it is owned or dies, remove the carcass and move on)
  - Treasured members of the family: each is integrated into configuration and user management (e.g. secured and watched like a child)
- Either viewpoint can be supported, know the tools and what you want (e.g. are willing to put into the task)

Install Options: Classic or Advanced

- CentOS 7 ISO image
  - Full toolkit install
  - Easy, all contained
- Want more control? Bundle of packages
  - perfsonar-tools
  - perfsonar-testpoint
  - perfsonar-core
  - perfsonar-toolkit
  - perfsonar-centralmanagement (pSConfig, MaDDash, Measurement Archive)
  - + optional packages
  - CentOS 7, Debian 8 – 9, Ubuntu 14 – 16

Package bundles structure



Outline

- Introduction
- Hardware & Software
- **Tool Use**
- Regular Testing
- Use Cases

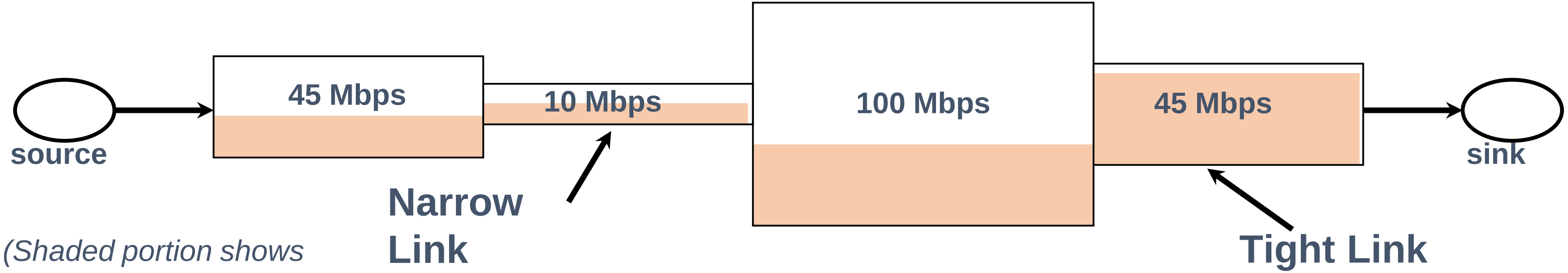
Let’s Talk about Throughput

- 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 could bulk data movement perform)
- Ways to game the system
  - Parallel streams
  - Manual window size adjustments
  - ‘memory to memory’ testing – no spinning disk

What Throughput Tells Us

- Let’s start by describing throughput, which is vague.
  - Capacity: link speed
    - Narrow Link: link with the lowest capacity along a path
    - Capacity of the end-to-end path = capacity of the narrow link
  - Utilized bandwidth: current traffic load
  - Available bandwidth: capacity – utilized bandwidth
    - Tight Link: link with the least available bandwidth in a path
  - Achievable bandwidth: includes protocol and host issues (e.g. BDP!)

All of this is “memory to memory”, e.g. we are not involving a spinning disk (more later)



(Shaded portion shows background traffic)

Let’s Talk about Throughput

- Few of the tools that pScheduler (the control/policy wrapper) knows how to talk with:
  - Iperf2
    - Default for the command line (e.g. `pscheduler task throughput --dest HOST` will invoke this)
    - Some known behavioral problems (Older versions were CPU bound, hard to get UDP testing to be correct)
  - Iperf3
    - Default for the perfSONAR regular testing framework, can invoke via command line switch (`pscheduler task –tool iperf3 throughput --dest HOST`)
    - New brew, has features iperf2 is missing (retransmissions, JSON output, daemon mode, etc.)
    - Note: Single threaded, so performance is gated on clock speed. Parallel stream testing is hard as a result (e.g. performance is bound to one core)
  - Nuttcp
    - Different code base, can invoke via command line switch (`pscheduler task –tool nuttcp throughput –-dest HOST`)
    - More control over how the tool behaves on the host (bind to CPU/core, etc.)
    - Similar feature set to iperf3

Meet pScheduler (the pS 4.0 replacement for BWCTL)

- New in the perfSONAR 4.0 release is a replacement for BWCTL as the control wrapper used to perform tests. To find out more about the usage and terminology of pScheduler, read up at:

  http://docs.perfsonar.net/pscheduler_intro.html

- Information on converting what you remember from BWCTL to the new pScheduler format can be found at:

  https://fasterdata.es.net/performance-testing/network-troubleshooting-tools/pscheduler/

Front End

- pScheduler is operated using a single command-line program:

```
pscheduler
```

- Autocompletes easily on most systems:

`psc` *Tab*

Command Format

- All commands follow the same format:

```
pscheduler command [ arg … ]
```

Getting Help

- The `--help` switch can be used at any point along the command line for assistance:

```
pscheduler --help
pscheduler command --help
```

Task Commands

- `task` – Give pScheduler a task that consists of making one or more measurements (*runs*).
- `result`– Fetch and display the results of a single, previously-concluded run by its URL.
- `watch` – Attach to a task identified by URL and show run results as they become available.
- `cancel` – Stop any future runs of a task.

Diagnostics and Administrivia

- `ping` – Determine if pScheduler is running on a host.
- `clock` – Check and compare the clock(s) on pScheduler host(s).
- `debug` – Enable debugging on pScheduler's internal parts.
  - Only needed for debugging pScheduler itself.
- `diags` – Produce a diagnostic dump for the perfSONAR team to use in resolving problems.
- `internal` – Do special things with pScheduler's internals.
  - Rarely needed; usually at the direction of the development team.

The **`task`** Command

- Asks pScheduler to do some work

- Replaces the **`bwctl`** family of commands used in earlier versions of perfSONAR

Synopsis

pscheduler task [*task-opts*] *test* [*test-opts*]

- ***task-opts*** – Switches related to everything but the test itself
  - Scheduling
  - Other behaviors (output format, etc.)
- ***test*** – What test the task is to perform (e.g., `throughput` or `trace`)
- ***test-opts*** – Test-specific switches and parameters

Starting Simple

`pscheduler` *Front-end command*

`    task` *pScheduler command*

`    rtt` *Test type (round-trip time)*

`    --dest localhost` *Where the pings go*

`        --length 512` *Packet size in bytes*

Line breaks and indentation added for clarity.

The Output Part I

```
% pscheduler task rtt --dest localhost --length 512
Submitting task...
Task URL:
https://ps.example.net/pscheduler/tasks/87e29f38-5b46…
Fetching first run...

Next run:
https://ps.example.net/pscheduler/tasks/87e29f38-5b46…
Starts 2016-12-07T07:57:30-05:00 (~7 seconds)
Ends   2016-12-07T07:57:41-05:00 (~10 seconds)
```

The Output Part II

```
Waiting for result...
1      127.0.0.1  520 Bytes  TTL 64  RTT   0.0430 ms
2      127.0.0.1  520 Bytes  TTL 64  RTT   0.0590 ms
3      127.0.0.1  520 Bytes  TTL 64  RTT   0.0640 ms
4      127.0.0.1  520 Bytes  TTL 64  RTT   0.0540 ms
5      127.0.0.1  520 Bytes  TTL 64  RTT   0.0620 ms
0% Packet Loss  RTT Min/Mean/Max/StdDev =
0.043000/0.056000/0.064000/0.010000 ms


No further runs scheduled.
```

Specifying Durations

- Subset of ISO 8601 Duration:

`PT19S`	*19 seconds*
`PT3M`	*3 minutes*
`PT2H5M`	*2 hours, 5 minutes*
`P1D`	*1 day*
`P3DT2H46M`	*3 days, 2 hours, 46 minutes*
`P2W`	*2 weeks*

- Inexact units (months, years) are not supported.

Specifying Dates and Times

- ISO 8601 timestamp:
  - Absolute `2016-03-19T12:05:19`

- Coming in a future release:
  - Relative to Now `PT10M` *ISO 8601*
  - Even Boundary `@PT1H` *@ + ISO 8601 Duration*

Task Options: Start Time

- **`--start` *`t`*** – Start at time `t`.
- **`--slip` *`d`*** – Allow the start time of run(s) to slip by duration `d`.
- **`--sliprand` *`f`*** – Randomize slip time as fraction `f` of available. (Range `[0.0, 1.0]`)

Task Options: Start Time

```
pscheduler task rtt


    --start 2017-05-01T12:00
```

Start May 1, 2017 at noon

```
        --slip PT8M
```

Slip start up to 8 minutes

```
        --sliprand 0.5
```

Randomly slip up to 4 minutes

```
        --dest www.example.com
```

Line breaks and indentation added for clarity.

Task Options: Repetition

- **`--repeat`** ***`d`*** – Repeat runs every duration *d*.
  - Other forms (notably CRON-like specification) to be added later.
- **`--until`** ***`t`*** – Continue repeating until time *t*.
  - Default is forever.
- **`--max-runs`** ***`n`*** – Allow the task to run up to *n* times.
  - Default is no upper limit.

Task Options: Behavior

- **`--import` *`f`*** – Import JSON for the task from file *`f`* (use `-` for standard input)
- **`--export`** – Dump the task specification as JSON to standard output but don't run it.
- **`--url`** – If the task is created, dump its URL to standard output and exit.
- **`--format` *`f`*** – If results are to be displayed, use format ***`f`***, which is one of `text` (the default), `html` or `json`.
- **`--assist` *`s`*** – Ask server *`s`* for assistance in setting up the task
  - Use this when the pScheduler server is not available on the local host.
  - `PSCHEDULER_ASSIST` from the environment

Task Options: Selecting a Tool

- **`--tool t`** – Add tool *t* to the list of tools which can be used to run the test.
  - Can be specified multiple times for multiple tools.

- If not provided, a tool is automatically selected from those available.

Test Options

- Parameters for the test
  - Dependent on which test is being carried out.
  - See guide documents for each test for specifics.

- Example:

```
psc task … trace --dest host.example.org
```

Putting the Parts Together

`psc task`

`--start 2016-05-04T19:20` *Start at the specified time*

`--repeat PT15M`
Repeat every 15 minutes

`--max-runs 100`
Stop after 100 successful runs

`trace --dest ps.example.org` *Trace to* *`ps.example.org`*

`--length 384`
Send 384-byte packets

`--hops 42`
Max. 42 hops to the destination

Throughput task Example (iperf2)

```
[ps-iniu@pS40-n1-c7-7 ~]$ pscheduler task throughput --source wash-pt1.es.net --dest sunn-pt1.es.net
Submitting task...
Task URL:
https://wash-pt1.es.net/pscheduler/tasks/11f74cc2-4d49-4170-b9c4-19ad1d5cc563
Running with tool 'iperf3'
Fetching first run...

Next scheduled run:
https://wash-pt1.es.net/pscheduler/tasks/11f74cc2-4d49-4170-b9c4-19ad1d5cc563/runs/4819e120-3140-4d71-a766-bc21adef1f66
Starts 2017-07-21T12:30:25-07:00 (~7 seconds)
Ends   2017-07-21T12:30:44-07:00 (~18 seconds)
Waiting for result...

* Stream ID 5
Interval       Throughput     Retransmits    Current Window
0.0 - 1.0      37.79 Mbps     0              903.75 KBytes
1.0 - 2.0      581.12 Mbps    0              8.21 MBytes
2.0 - 3.0      1.89 Gbps      0              24.11 MBytes
3.0 - 4.0      5.91 Gbps      0              67.00 MBytes
4.0 - 5.0      9.59 Gbps      0              79.86 MBytes
5.0 - 6.0      9.89 Gbps      0              79.88 MBytes
6.0 - 7.0      9.90 Gbps      0              80.19 MBytes
7.0 - 8.0      9.90 Gbps      0              80.24 MBytes
8.0 - 9.0      9.90 Gbps      0              80.26 MBytes
9.0 - 10.0     9.89 Gbps      0              80.26 MBytes

Summary
Interval       Throughput     Retransmits
0.0 - 10.0     6.75 Gbps      0
```

N.B. This is what perfSONAR Graphs – the average of the complete test

Throughput task Example (iperf3)

```
[ps-iniu@pS40-n1-c7-7 ~]$ pscheduler task --tool iperf3 throughput --source wash-pt1.es.net --dest sunn-pt1.es.net --
interval PT2S
Submitting task...
Task URL:
https://wash-pt1.es.net/pscheduler/tasks/5c1f457f-e5aa-463f-b475-7226dcc74dc7
Running with tool 'iperf3'
Fetching first run...

Next scheduled run:
https://wash-pt1.es.net/pscheduler/tasks/5c1f457f-e5aa-463f-b475-7226dcc74dc7/runs/3561e7c0-8471-4fb7-8c60-16c9d7fe151a
Starts 2017-07-21T12:48:56-07:00 (~6 seconds)
Ends   2017-07-21T12:49:15-07:00 (~18 seconds)
Waiting for result...

* Stream ID 5
Interval       Throughput     Retransmits    Current Window
0.0 - 2.0      365.48 Mbps    0              9.49 MBytes
2.0 - 4.0      5.26 Gbps      0              79.88 MBytes
4.0 - 6.0      9.89 Gbps      0              80.16 MBytes
6.0 - 8.0      9.89 Gbps      0              80.27 MBytes
8.0 - 10.0     9.89 Gbps      0              80.31 MBytes

Summary
Interval       Throughput     Retransmits
0.0 - 10.0     7.06 Gbps      0

No further runs scheduled.
[ps-iniu@pS40-n1-c7-7 ~]$
```

N.B. This is what perfSONAR Graphs – the average of the complete test

Throughput task Example (nuttcp)

```
[ps-iniu@pS40-n1-c7-7 ~]$ pscheduler task --tool nuttcp throughput --source wash-pt1.es.net --dest sunn-pt1.es.net --
interval PT2S
Submitting task...
Task URL:
https://wash-pt1.es.net/pscheduler/tasks/40aef448-2ba4-48db-8242-cf27c64853bb
Running with tool 'nuttcp'
Fetching first run...

Next scheduled run:
https://wash-pt1.es.net/pscheduler/tasks/40aef448-2ba4-48db-8242-cf27c64853bb/runs/36b18c33-45d6-4ea8-9523-0e12d352e222
Starts 2017-07-21T12:53:26-07:00 (~5 seconds)
Ends   2017-07-21T12:53:42-07:00 (~15 seconds)
Waiting for result...

* Stream ID 1
Interval       Throughput     Retransmits    Current Window
0.0 - 2.0      829.94 Mbps    0              26.16 MBytes
2.0 - 4.0      7.77 Gbps      0              78.02 MBytes
4.0 - 6.0      9.90 Gbps      0              78.10 MBytes
6.0 - 8.0      9.90 Gbps      0              78.14 MBytes
8.0 - 10.0     9.90 Gbps      0              78.44 MBytes

Summary
Interval       Throughput     Retransmits
0.0 - 10.0     7.62 Gbps      0

No further runs scheduled.
[ps-iniu@pS40-n1-c7-7 ~]$
```

N.B. This is what perfSONAR Graphs – the average of the complete test

Outline

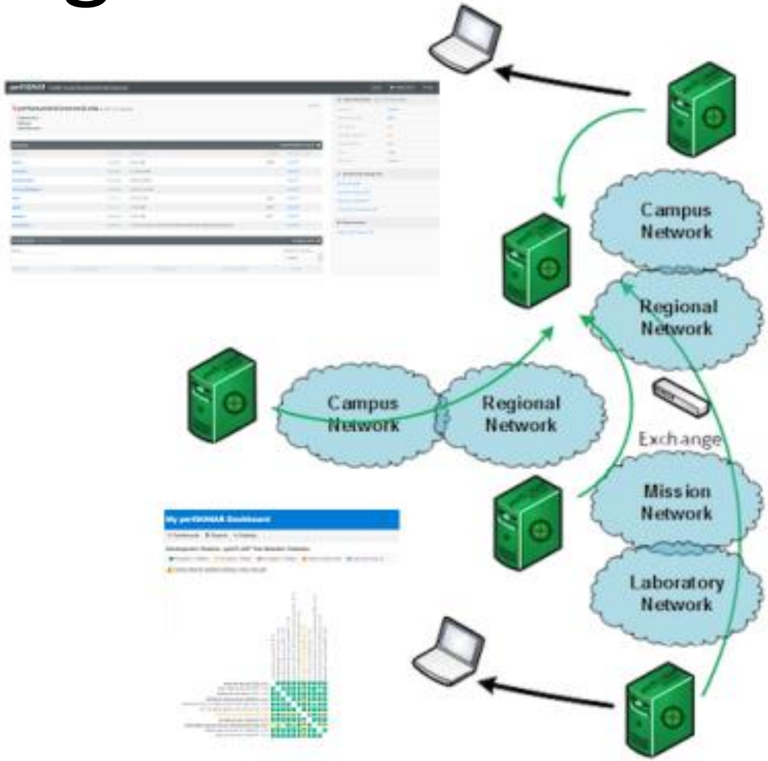
- Introduction
- Hardware & Software
- Tool Use
- **Regular Testing**
- Use Cases

Regular Testing

- There are a couple of ways to do this.
  - Beacon: Let others test to you (e.g. no regular 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 (e.g. consume a testing configuration that includes tests to everyone, and incorporate into a visualization)

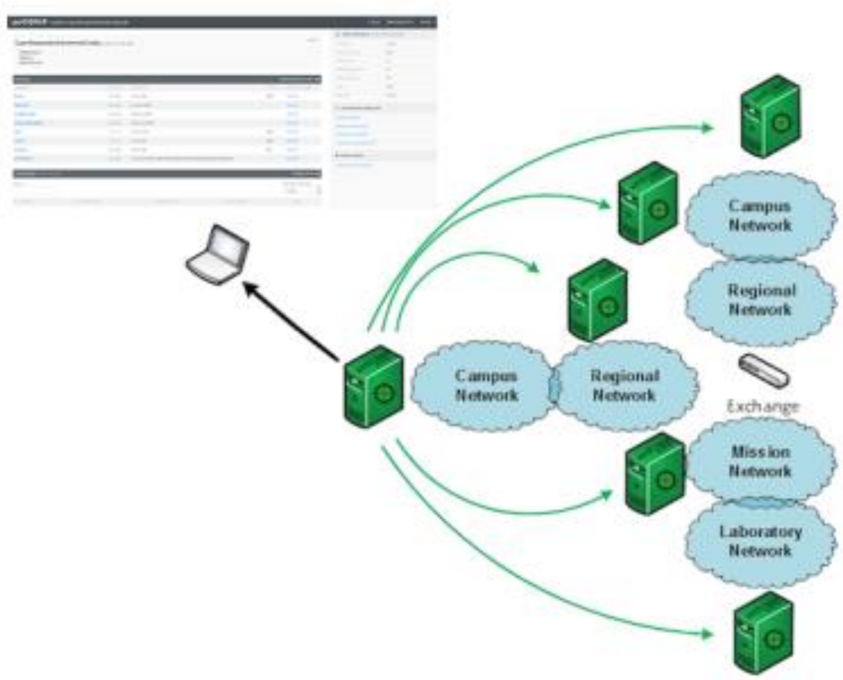
Regular Testing - Beacon

- 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)



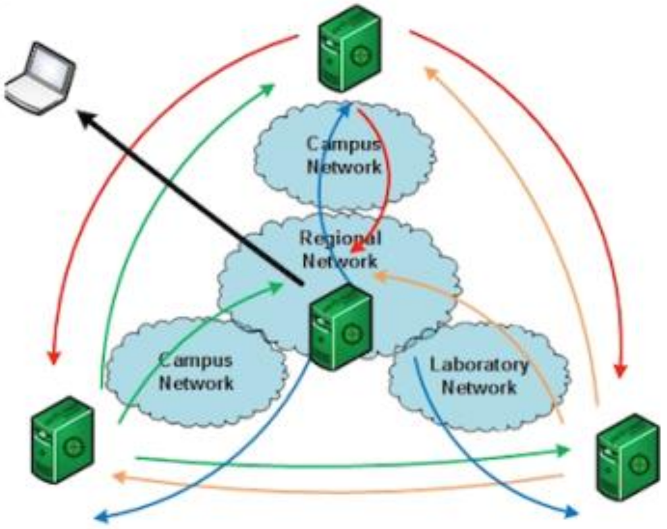
Regular Testing - Island

- The island setup allows a site to test against any number of the 1200+ 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.
  - Throughput will not be as valuable when the latency is small



Regular Testing - Mesh

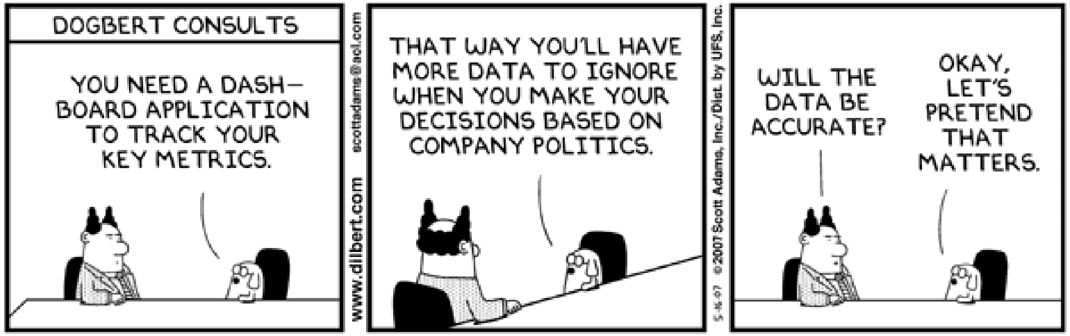
- A full mesh requires more coordination:
  - A full mesh means all hosts involved are running the same test configuration
  - A partial mesh could mean only a small number of related hosts are running a testing configuration
- In either case – bandwidth and latency will be valuable test cases



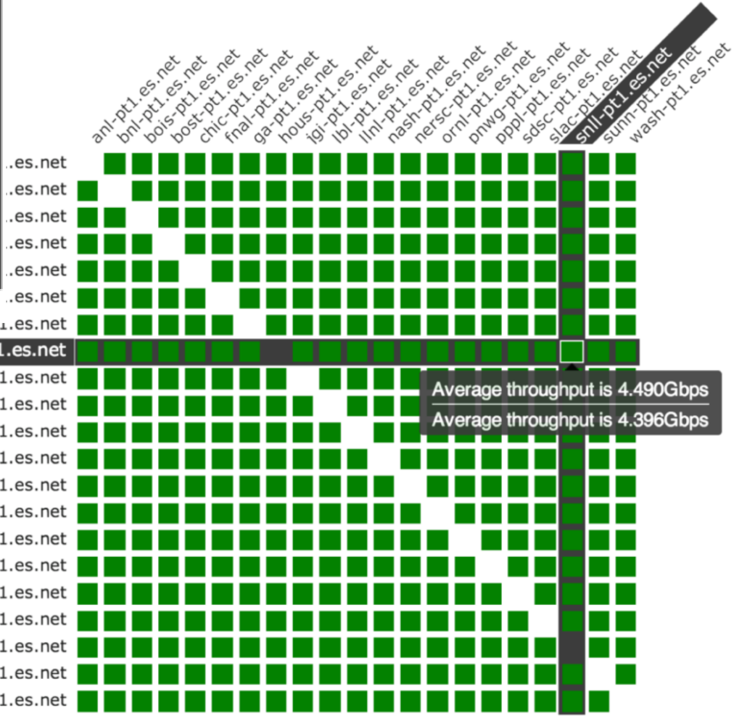
Importance of Regular Testing

- 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

MaDDash: http://ps-dashboard.es.net


DOGBERT CONSULTS
YOU NEED A DASHBOARD APPLICATION TO TRACK YOUR KEY METRICS.
scottadams@aol.com
www.dilbert.com
THAT WAY YOU'LL HAVE MORE DATA TO IGNORE WHEN YOU MAKE YOUR DECISIONS BASED ON COMPANY POLITICS.
©2007 Scott Adams, Inc./Dist. by UFS, Inc.
WILL THE DATA BE ACCURATE?
OKAY, LET'S PRETEND THAT MATTERS.

© Scott Adams, Inc./Dist. by UFS, Inc.



Average throughput is 4.490Gbps

Average throughput is 4.396Gbps

ESnet ENERGY SCIENCES NETWORK GÉANT INDIANA UNIVERSITY INTERNET2 UNIVERSITY OF MICHIGAN

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
  - MaDDash display
  - Integration with NAGIOS
- This infrastructure is deployed now – perfSONAR hosts at facilities can take advantage of it
- At-a-glance health check for data infrastructure

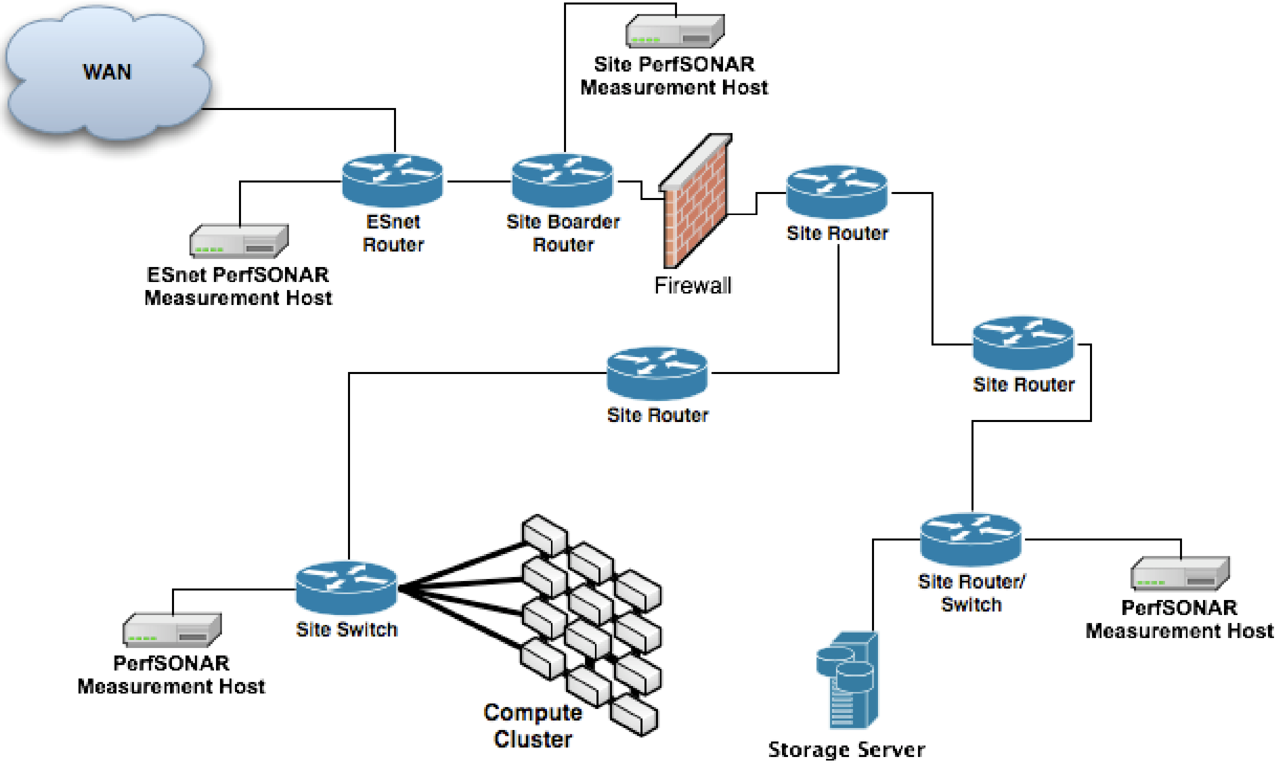
Develop a Test Plan

- What are you going to measure?
  - Achievable bandwidth
    - ***2-3 regional destinations***
    - 4-8 important collaborators
    - 4-8 times per day to each destination
    - 20 second tests within a region, 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
  - MadDash

perfSONAR Deployment Locations

- Critical to deploy near key resources such as DTNs
- 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
- Effective test methodology derived from protocol behavior
  - TCP suffers much more from packet loss as latency increases
  - TCP is more likely to cause loss as latency increases
  - Testing should leverage this in two ways
    - Design tests so that they are likely to fail if there is a problem
    - Mimic the behavior of production traffic as much as possible
  - Note: don’t design your tests to succeed
    - The point is not to “be green” even if there are problems
    - The point is to find problems when they come up so that the problems are fixed quickly

Sample Site Deployment

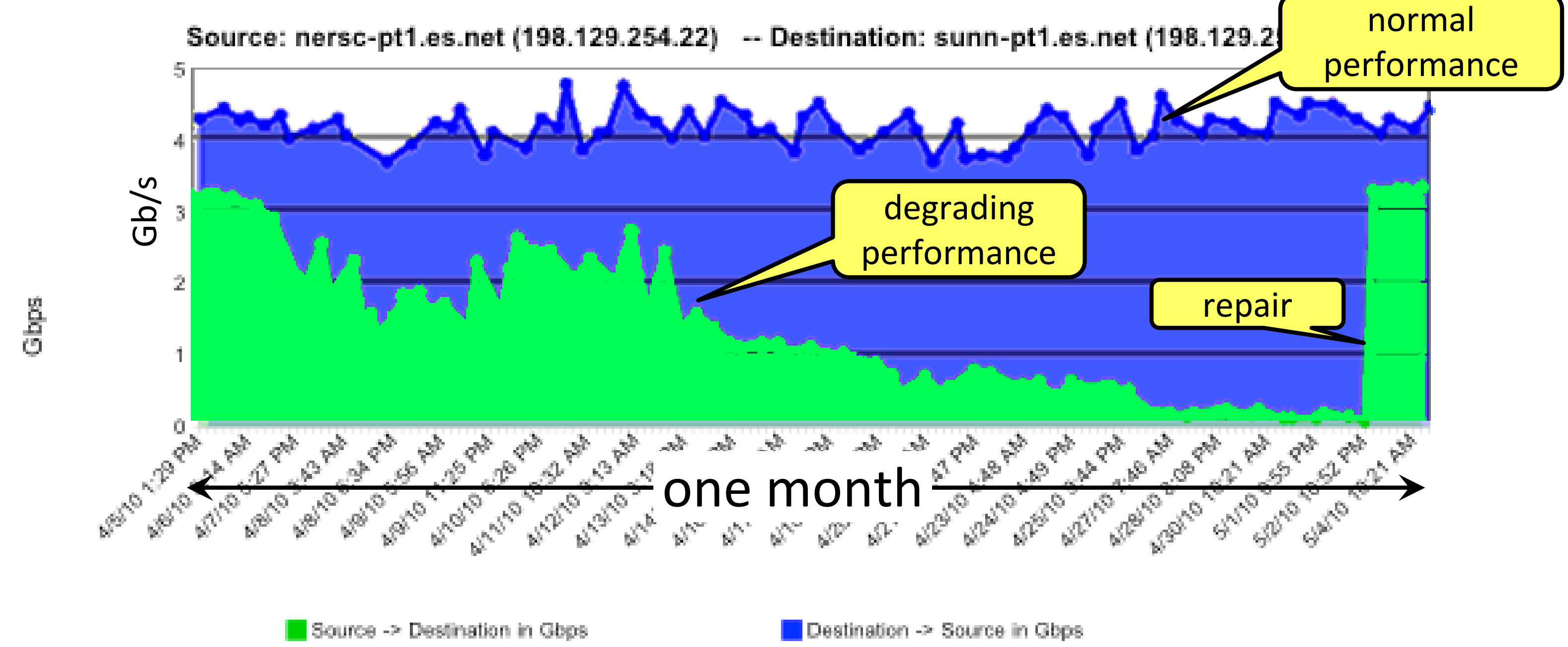


Outline

- Introduction
- Hardware & Software
- Tool Use
- Regular Testing
- **Use Cases**

Success Stories - #1 Failing Optic(s)

- First example –featuring a backbone network
  - Similar to frog boiling, hard alarms don’t notice gradual failure



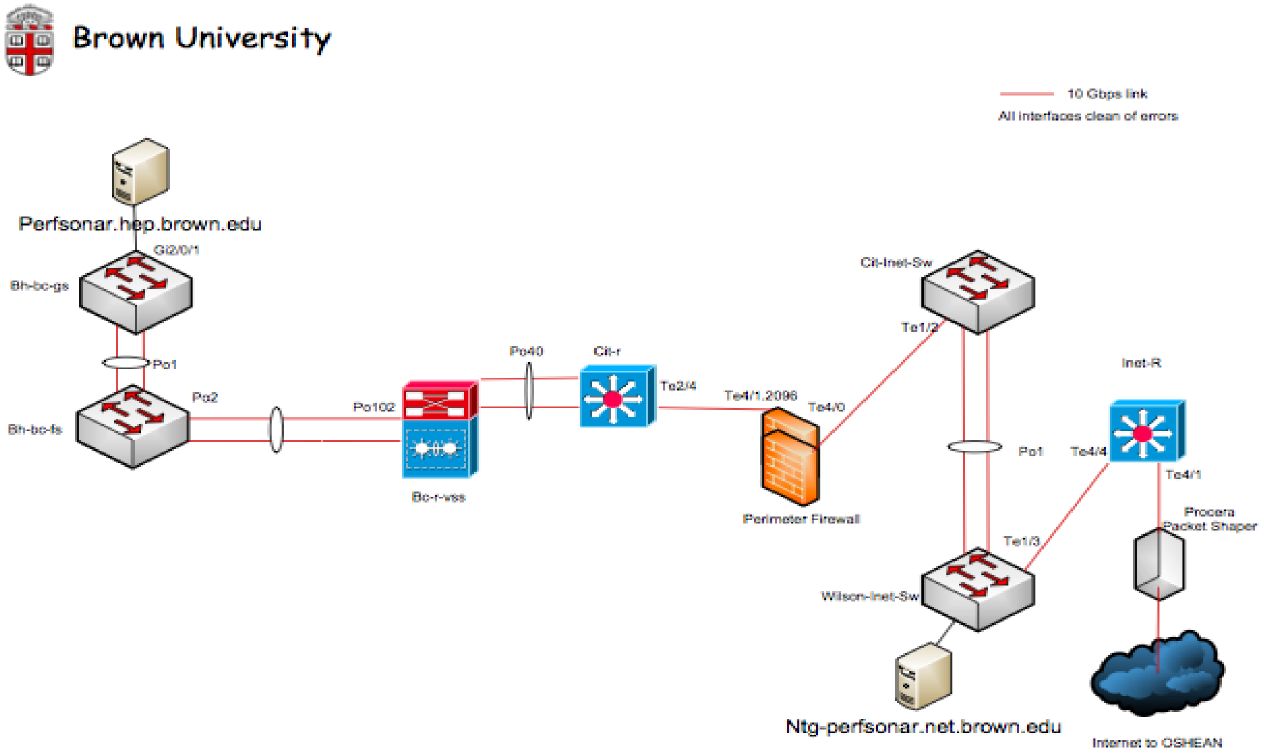
normal performance

degrading performance

repair

one month

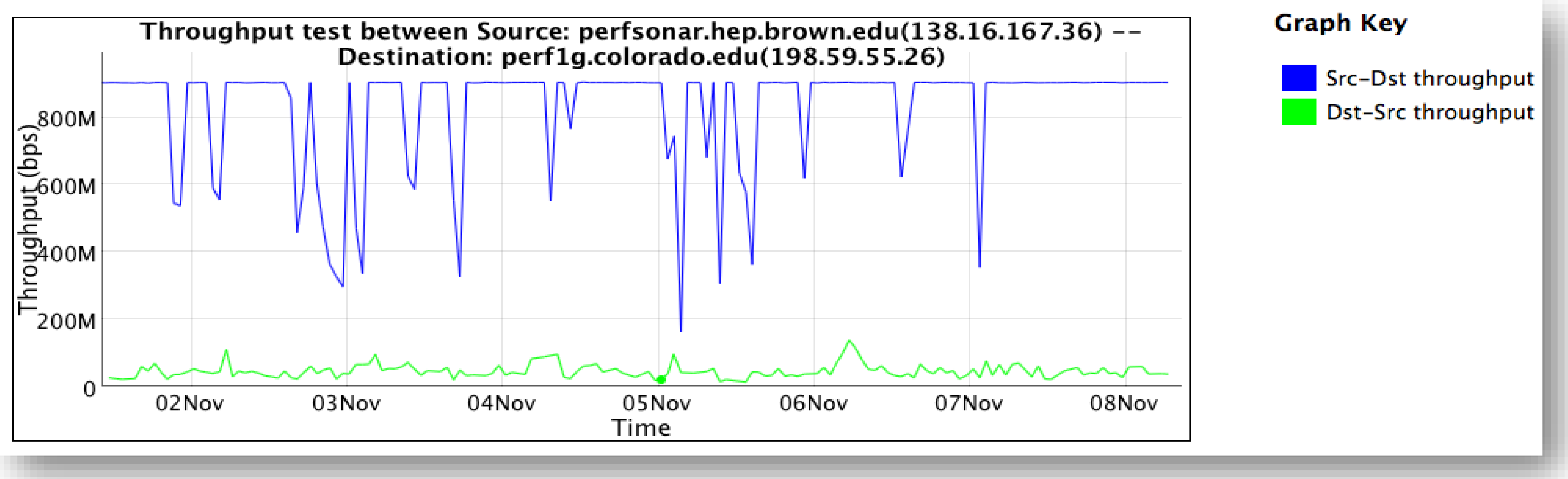
Success Stories - #2 Brown University



ESnet ENERGY SCIENCES NETWORK GÉANT INDIANA UNIVERSITY INTERNET2 UNIVERSITY OF MICHIGAN

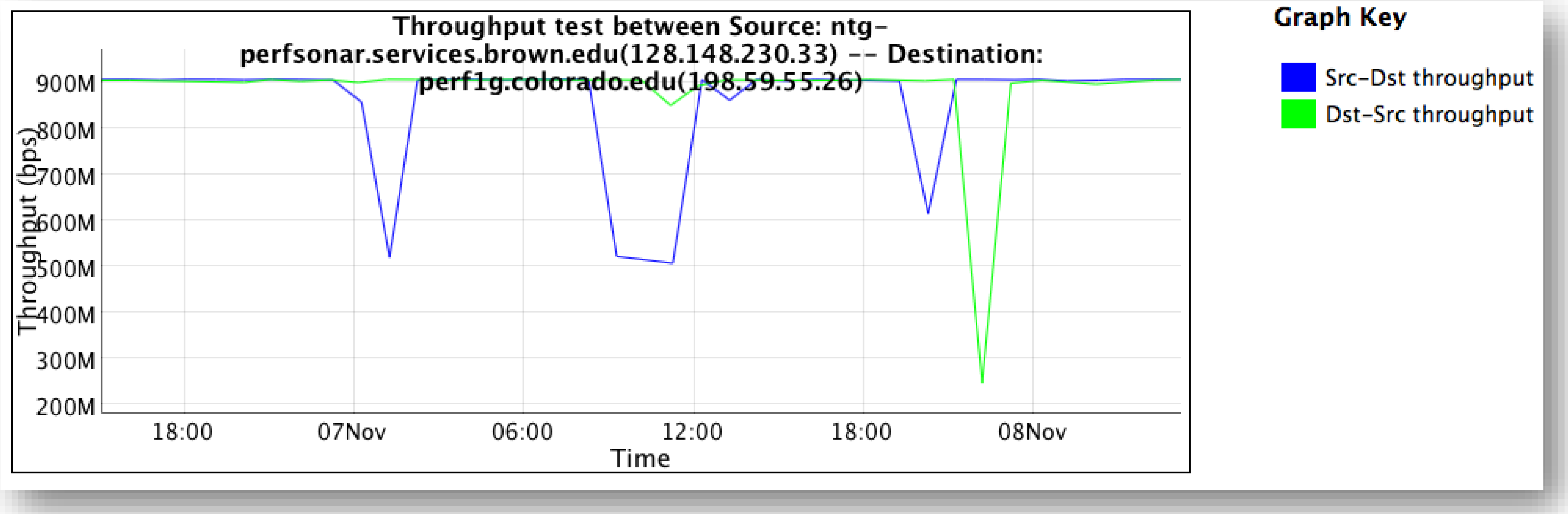
Success Stories - #2 Brown University Example

- Results to host behind the firewall:



Success Stories - #2 Brown University Example

- In front of the firewall:



Success Stories - #2 TCP Dynamics

- Want more proof – lets 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
```

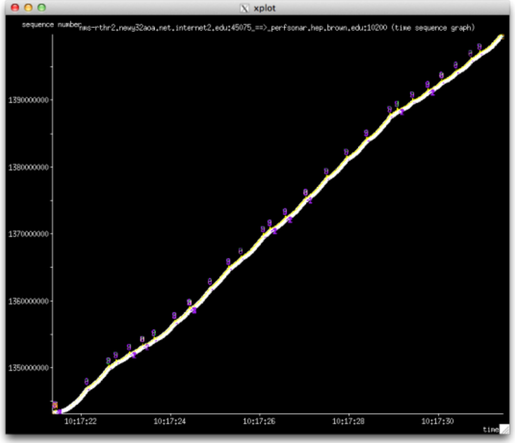
Success Stories - #2 TCP Dynamics Through Firewall

- ‘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
```

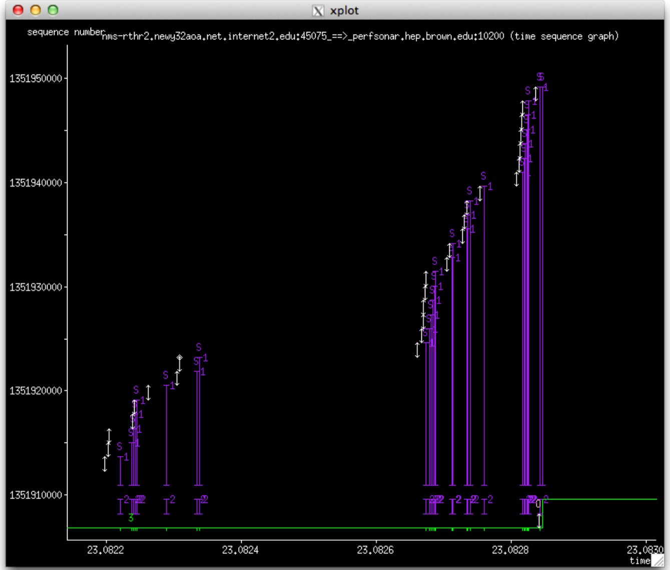
Success Stories - #2 tcptrace output: with and without a firewall



xplot

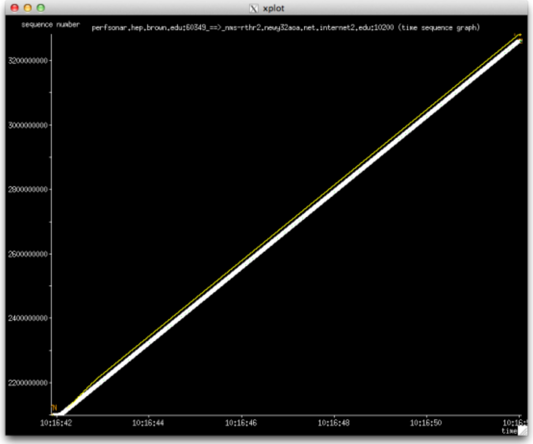
sequence number nms-rthr2.newy32aoa.net.internet2.edu:45075_==>_perfsonar.hep.brown.edu:10200 (time sequence graph)

firewall



xplot

sequence number nms-rthr2.newy32aoa.net.internet2.edu:45075_==>_perfsonar.hep.brown.edu:10200 (time sequence graph)



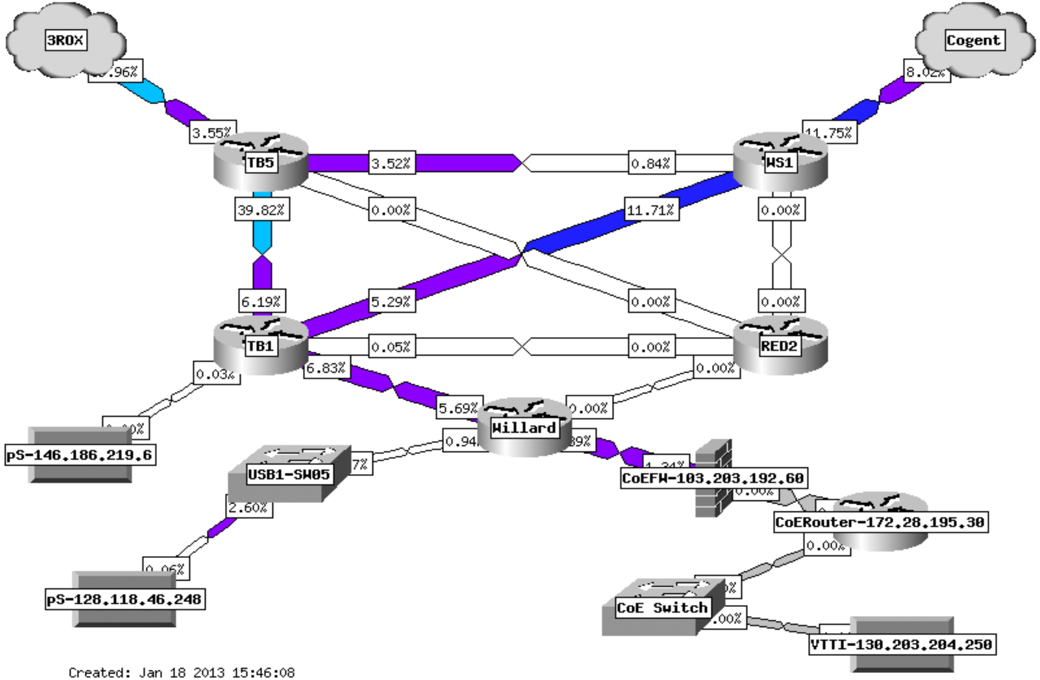
xplot

sequence number perfsonar.hep.brown.edu:60349_==>_nms-rthr2.newy32aoa.net.internet2.edu:10200 (time sequence graph)

No firewall

Success Stories - #3 PSU

- PSU = Firewalls for some. The college of engineering has one, central IT does not



Created: Jan 18 2013 15:46:08

Success Stories - #3 PSU

- Initial Report from network users: performance poor both directions
  - Outbound and inbound (normal issue is inbound through protection mechanisms)
- From previous diagram – CoE firewalll was tested
  - Machine outside/inside of firewall. Test to point 10ms away (Internet2 Washington)
- Low, but no retransmissions?

```
j@ssstatecollege:~> nuttcp -T 30 -i 1 -p 5679 -P 5678 64.57.16.22
    5.8125 MB /   1.00 sec =   48.7565 Mbps     0 retrans
    6.1875 MB /   1.00 sec =   51.8886 Mbps     0 retrans
    6.1250 MB /   1.00 sec =   51.3957 Mbps     0 retrans
    6.1250 MB /   1.00 sec =   51.3927 Mbps     0 retrans

  184.3515 MB /  30.17 sec =   51.2573 Mbps 0 %TX 1 %RX 0 retrans 9.85 msRTT
```

Success Stories - #3 PSU

- Observation: `net.ipv4.tcp_window_scaling` did not seem to be working
    - 64K of buffer is default. Over a 10ms path, this means we can hope to see only 50Mbps of throughput:
    - **BDP (50 Mbit/sec, 10.0 ms) = 0.06 Mbyte**
- Implication: something in the path was not respecting the specification in RFC 1323, and was not allowing TCP window to grow
    - TCP window of 64 KByte and RTT of **1.0 ms** <= **500.00 Mbit/sec.**
    - TCP window of 64 KByte and RTT of **5.0 ms** <= **100.00 Mbit/**sec.
    - TCP window of 64 KByte and RTT of **10.0 ms** <= **50.00 Mbit/sec.**
    - TCP window of 64 KByte and RTT of **50.0 ms** <= **10.00 Mbit/sec.**
- Reading documentation for firewall:
    - **TCP flow sequence checking** was enabled
    - What would happen if this was turn off (both directions?

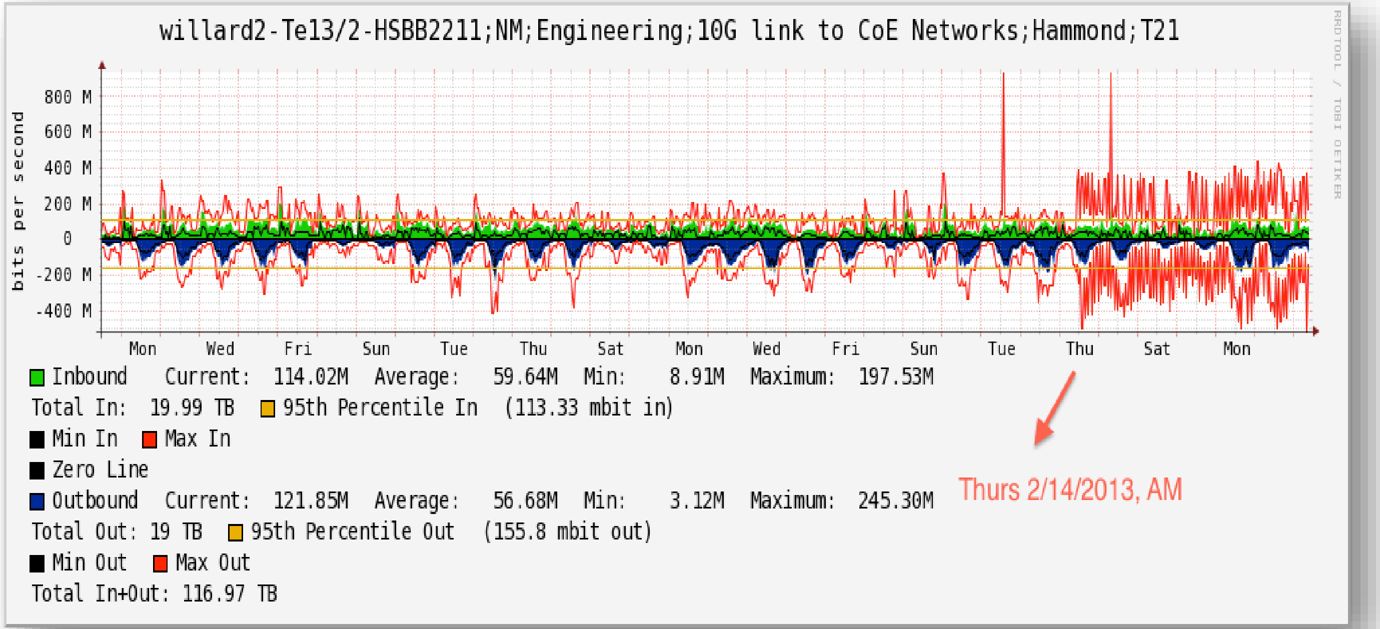
Success Stories - #3 PSU

```
j@ssstatecollege:~> nuttcp -T 30 -i 1 -p 5679 -P 5678 64.57.16.22
   55.6875 MB /   1.00 sec =  467.0481 Mbps     0 retrans
   74.3750 MB /   1.00 sec =  623.5704 Mbps     0 retrans
   87.4375 MB /   1.00 sec =  733.4004 Mbps     0 retrans
   91.7500 MB /   1.00 sec =  770.0544 Mbps     0 retrans
   88.6875 MB /   1.00 sec =  743.5676 Mbps    28 retrans
   69.0625 MB /   1.00 sec =  578.9509 Mbps     0 retrans

 2300.8495 MB /  30.17 sec =  639.7338 Mbps 4 %TX 17 %RX 730
retrans 9.88 msRTT
```

Success Stories - #3 PSU

- Was this impacting people? Oh yes it was:



🟩 Inbound   Current: 114.02M   Average: 59.64M   Min: 8.91M   Maximum: 197.53M

Total In: 19.99 TB 🟨 95th Percentile In (113.33 mbit in)

⬛ Min In 🟥 Max In

⬛ Zero Line

🟦 Outbound   Current: 121.85M   Average: 56.68M   Min: 3.12M   Maximum: 245.30M

Total Out: 19 TB 🟨 95th Percentile Out (155.8 mbit out)

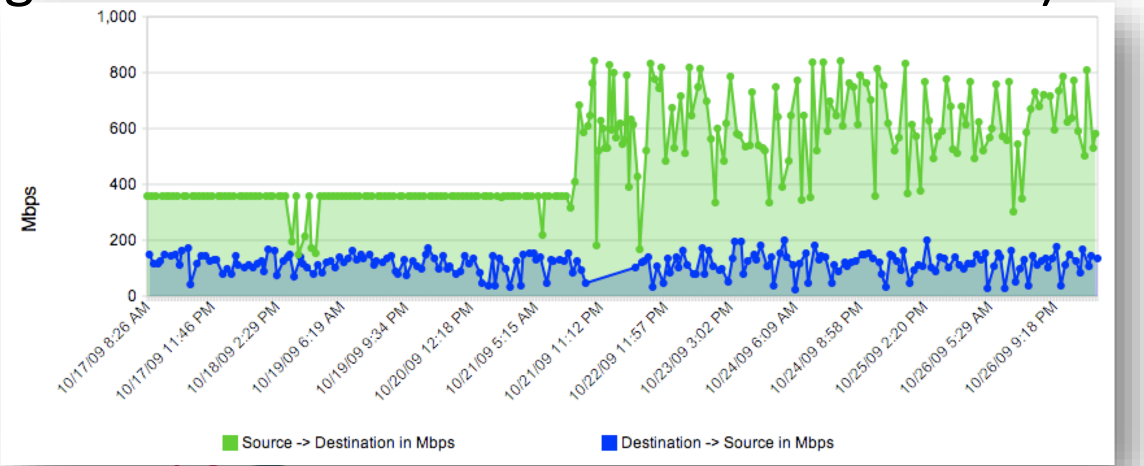
⬛ Min Out 🟥 Max Out

Total In+Out: 116.97 TB

Thurs 2/14/2013, AM

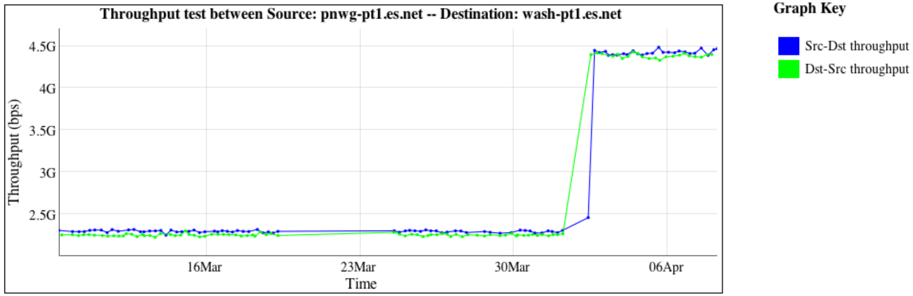
Success Stories - #4 Host Tuning

- Simple example – play with the settings in `/etc/sysctl.conf` when running some BWCTL tests.
- See if you can pick out when we raised the memory for the TCP window (ignore the blue – this is a known firewall)



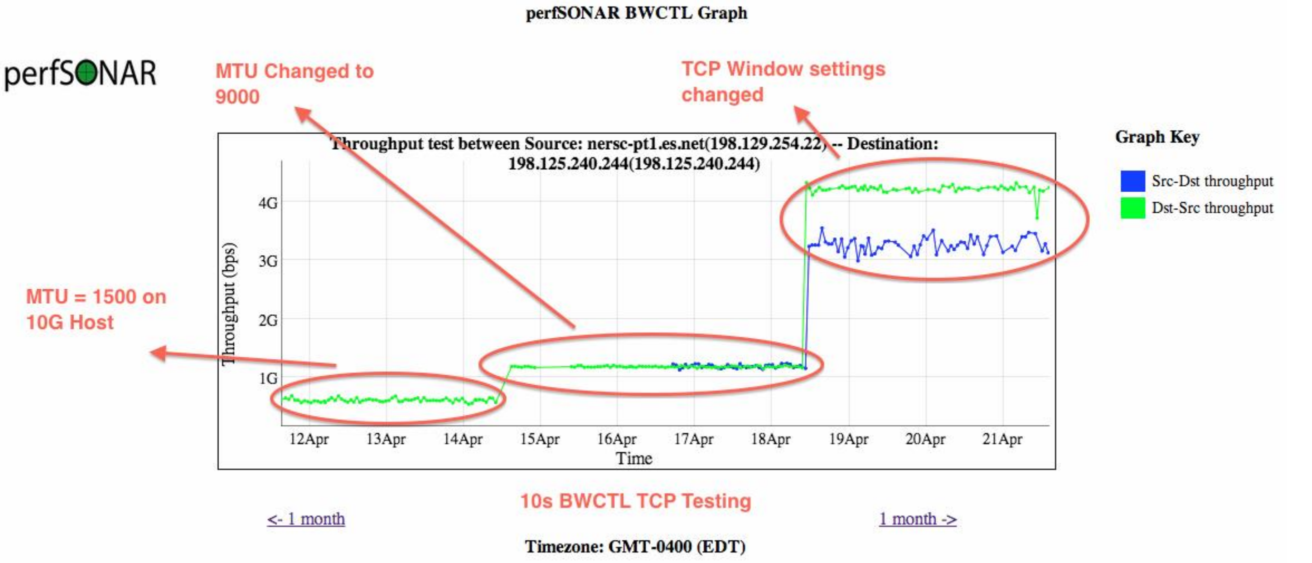
Success Stories - #4 Host Tuning

- Another example – 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.
- As the path length/throughput expectation increases, this is a good idea. There are limits (e.g. beware of buffer bloat on short RTTs)



Success Stories - #4 Host Tuning

- A more complete view – showing the role of MTUs and host tuning (e.g. ‘its all related’):



perfSONAR BWCTL Graph

MTU Changed to 9000

TCP Window settings changed

MTU = 1500 on 10G Host

10s BWCTL TCP Testing

<- 1 month

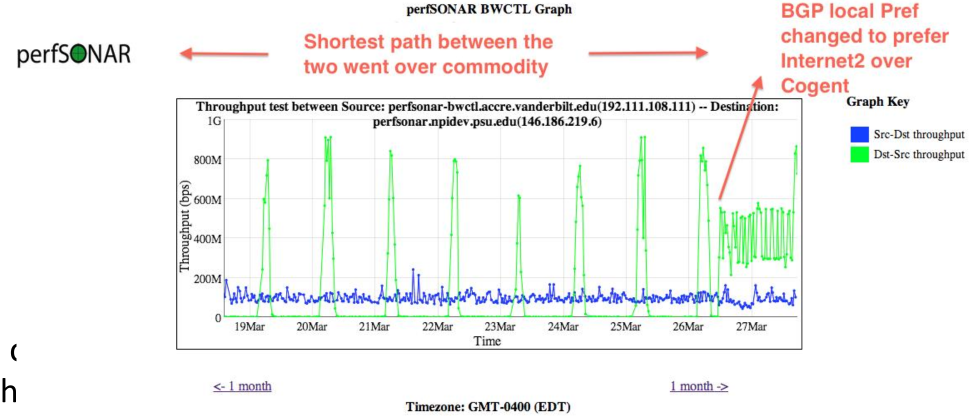
1 month ->

Timezone: GMT-0400 (EDT)

Success Stories - #6 R&E vs. Commodity Routing

- Some campuses don’t need to be told that the R&E path is ‘better’, others need to figure it out on their own.
- BWCTL results between PSU and Vanderbilt (science driver was genomics)
  - Normally low results over the course of the day. ‘spikes’ at night.
  - Traceoutes:
    - PSU -> Cogent -> Century Link -> Vanderbilt
    - Vanderbilt -> SOX -> NLR (dated) -> 3ROX -> PSU
  - Asymmetry is not bad by itself, unless …

Success Stories - #6 R&E vs. Commodity Routing



perfSONAR BWCTL Graph

Shortest path between the two went over commodity

BGP local Pref changed to prefer Internet2 over Cogent

Throughput test between Source: perfsonar-bwctl.accre.vanderbilt.edu(192.111.108.111) -- Destination: perfsonar.npidev.psu.edu(146.186.219.6)

Graph Key

🟦	Src-Dst throughput
🟩	Dst-Src throughput

<- 1 month

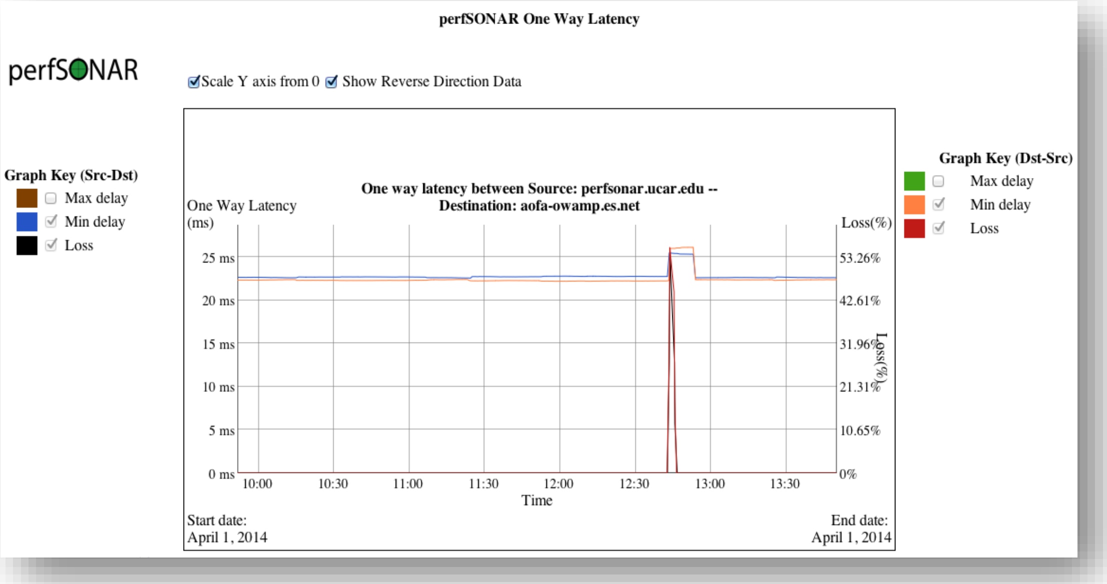
1 month ->

Timezone: GMT-0400 (EDT)

- Letting BGP ‘figure it
    - Yes, shortest path
    - R&E should always be preferred to commodity when available
- Managing local prefs can be a ‘pain’ for those that are not used to it. The end result is hard to argue with
- The ‘blue’ line? Over NLR in the dying days – and the Cisco 650x in that region was known to have a bad card/optic that was never replaced (e.g. packet loss all over the place)

Success Stories - #8 Fiber Cut

- Not that perfSONAR could help fix this (that’s up to your local DOT and provider …), but it does have an interesting signature in terms of loss and latency:



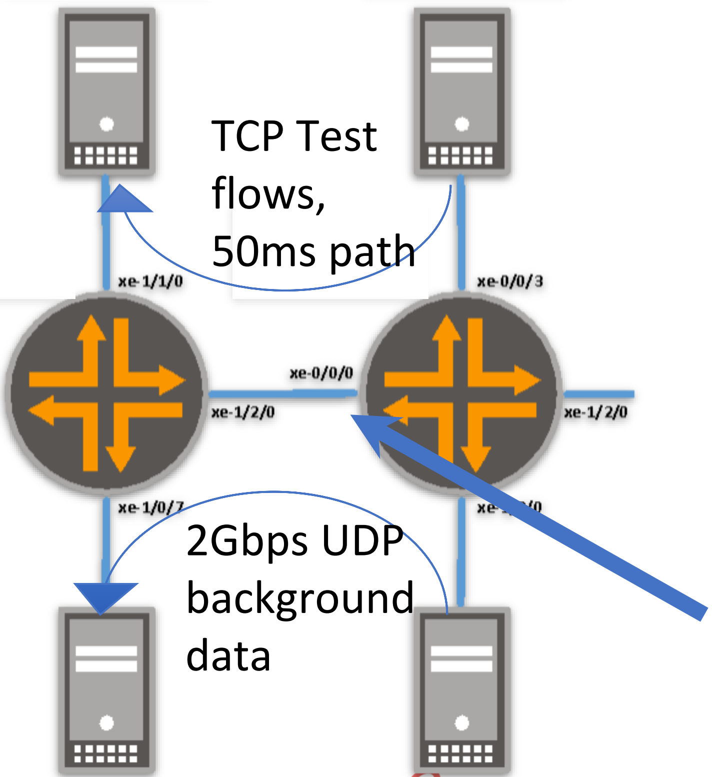
perfSONAR One Way Latency

☑Scale Y axis from 0 ☑ Show Reverse Direction Data

#9 Buffer Tuning Experiment

30 Second test, 2 TCP streams

Buffer Size	Packets Dropped	TCP Throughput
120 MB	0	8Gbps
60 MB	0	8Gbps
36 MB	200	2Gbps
24 MB	205	2Gbps
12 MB	204	2Gbps
6 MB	207	2Gbps

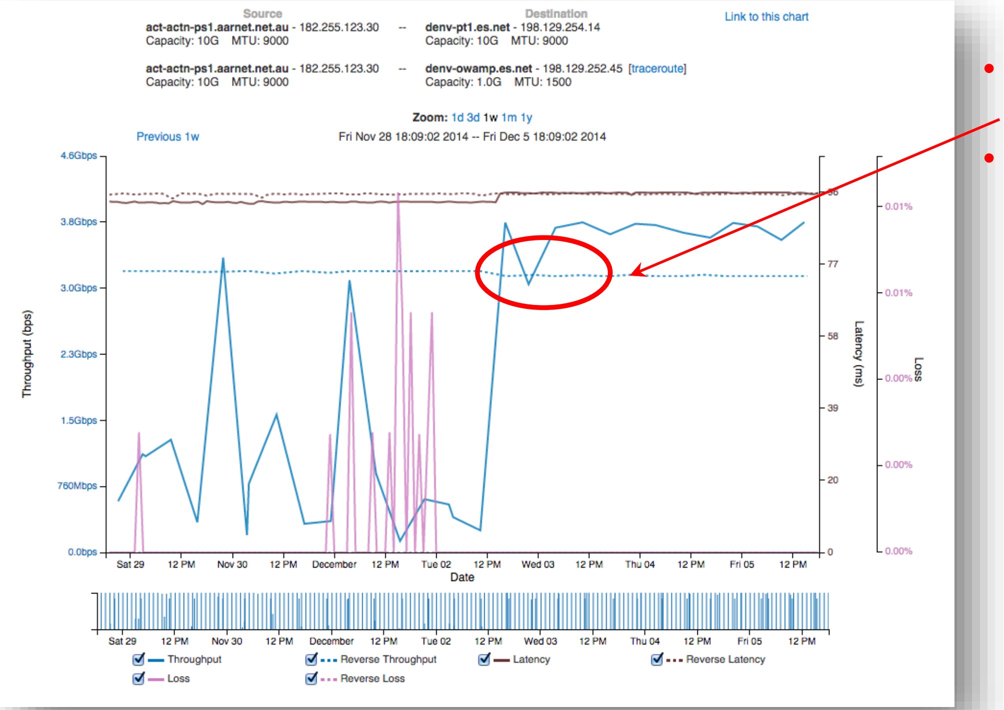


TCP Test flows, 50ms path

2Gbps UDP background data

Modify this egress buffer size

#10 BGP Peering Migration



Source — Destination — Link to this chart

act-actn-ps1.aarnet.net.au - 182.255.123.30 -- **denv-pt1.es.net** - 198.129.254.14
Capacity: 10G MTU: 9000 — Capacity: 10G MTU: 9000

act-actn-ps1.aarnet.net.au - 182.255.123.30 -- **denv-owamp.es.net** - 198.129.252.45 [traceroute]
Capacity: 10G MTU: 9000 — Capacity: 1.0G MTU: 1500

Zoom: 1d 3d 1w 1m 1y

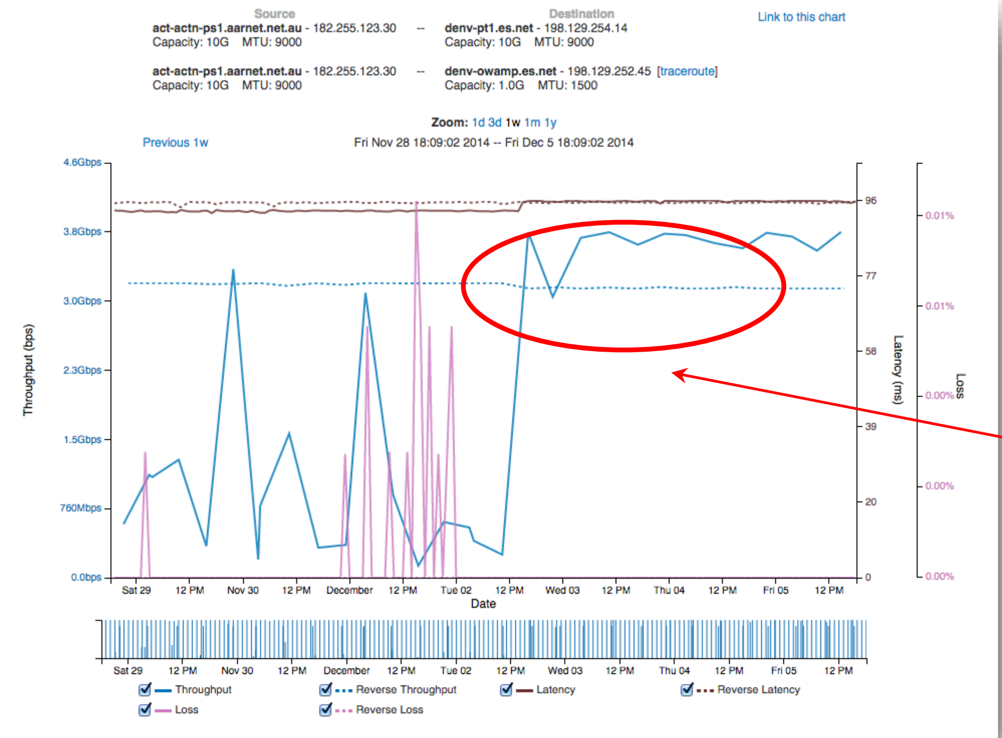
Previous 1w — Fri Nov 28 18:09:02 2014 -- Fri Dec 5 18:09:02 2014

☑ Throughput ☑ Reverse Throughput ☑ Latency ☑ Reverse Latency
☑ Loss ☑ Reverse Loss

- Peering moved from 10G link to 100G link
- Latency change shows path change

ESnet ENERGY SCIENCES NETWORK GÉANT INDIANA UNIVERSITY INTERNET2 UNIVERSITY OF MICHIGAN

#10 BGP Peering Migration



Source — Destination — Link to this chart

act-actn-ps1.aarnet.net.au - 182.255.123.30 -- denv-pt1.es.net - 198.129.254.14
Capacity: 10G MTU: 9000 — Capacity: 10G MTU: 9000

act-actn-ps1.aarnet.net.au - 182.255.123.30 -- denv-owamp.es.net - 198.129.252.45 [traceroute]
Capacity: 10G MTU: 9000 — Capacity: 1.0G MTU: 1500

Zoom: 1d 3d 1w 1m 1y

Previous 1w

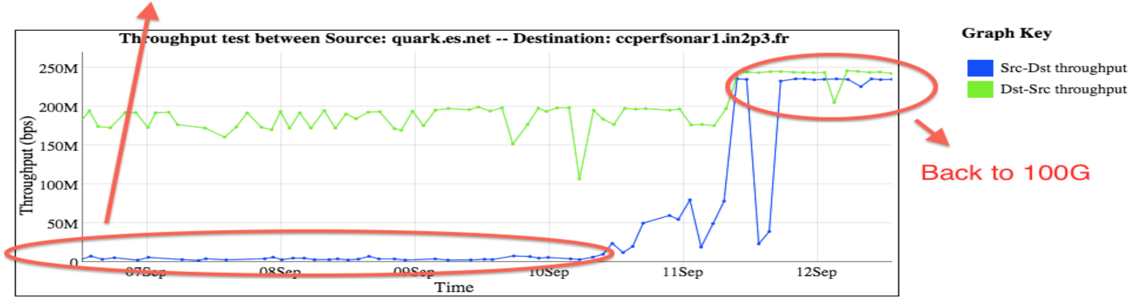
Fri Nov 28 18:09:02 2014 -- Fri Dec 5 18:09:02 2014

- Performance increases
- Performance stabilizes

#11 Monitoring TA Links

perfSONAR BWCTL Graph

Really poor performance over 10G infrastructure?



Graph Key

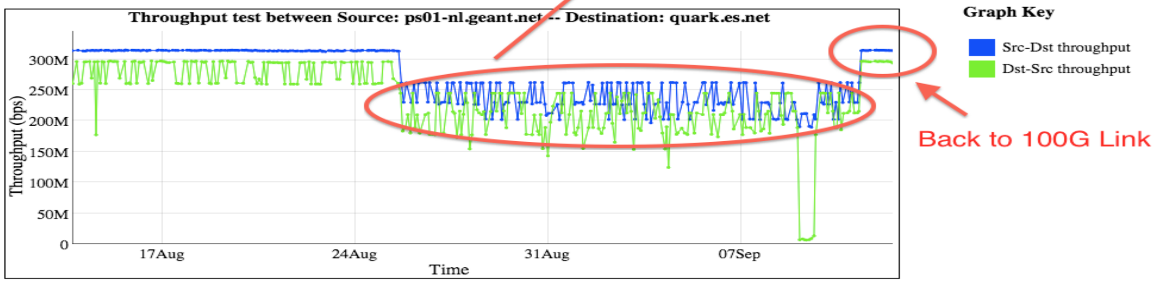
🟦 Src-Dst throughput
🟩 Dst-Src throughput

Back to 100G

<- 1 month

perfSONAR BWCTL Graph

Use of other TA Connectivity

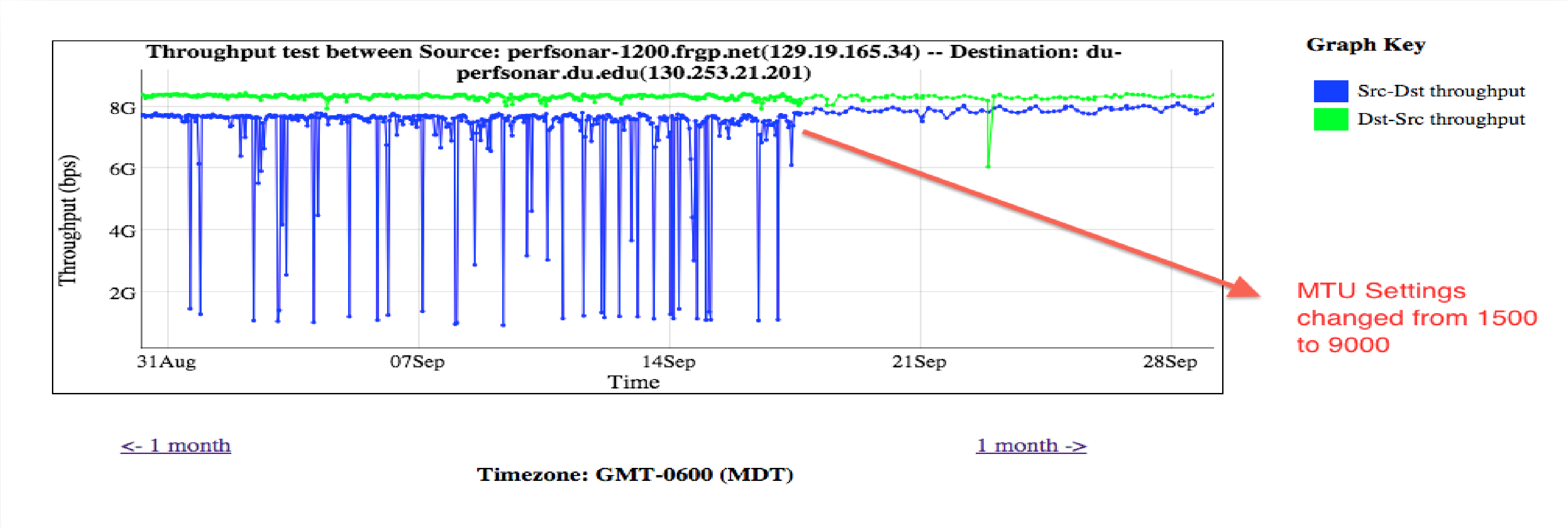


Graph Key

🟦 Src-Dst throughput
🟩 Dst-Src throughput

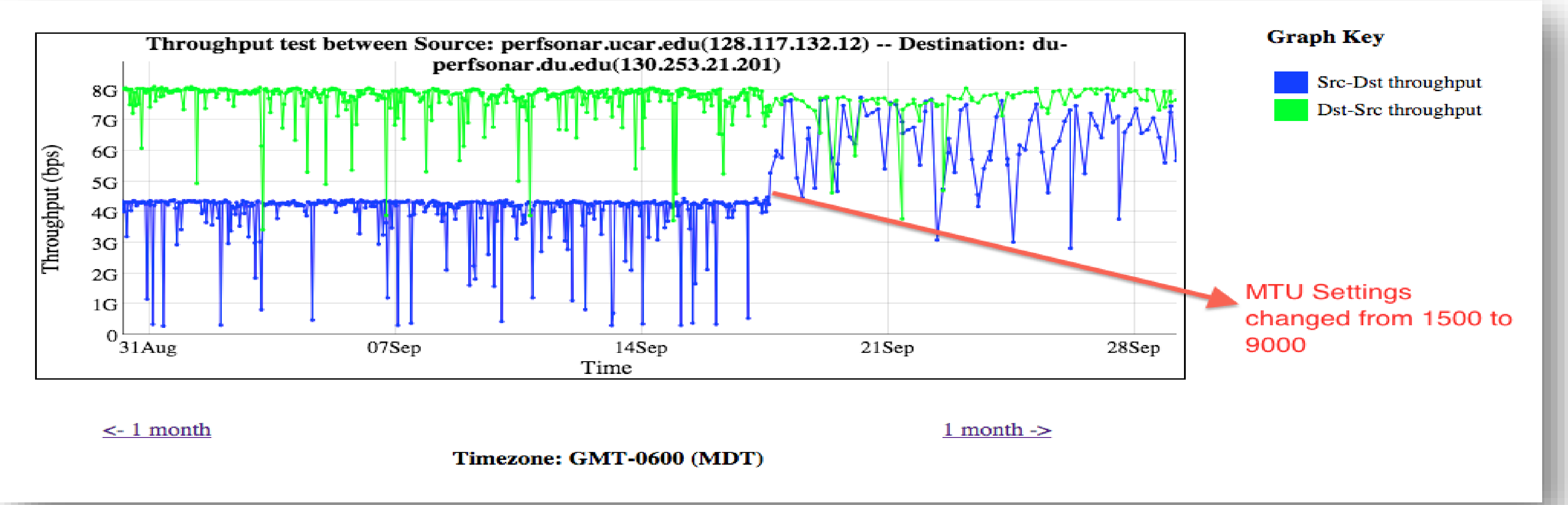
Back to 100G Link

#13 MTU Changes (Short RTT)



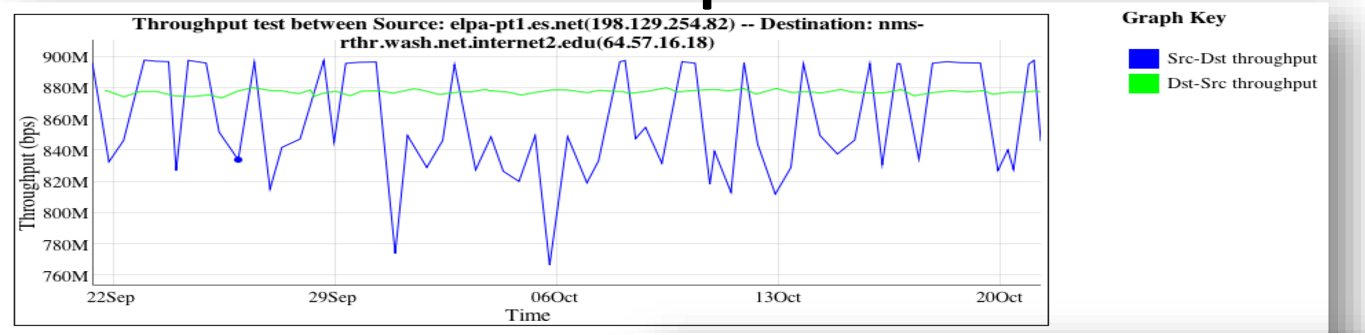
MTU Settings changed from 1500 to 9000

#13 MTU Changes (Longer RTT)



MTU Settings changed from 1500 to 9000

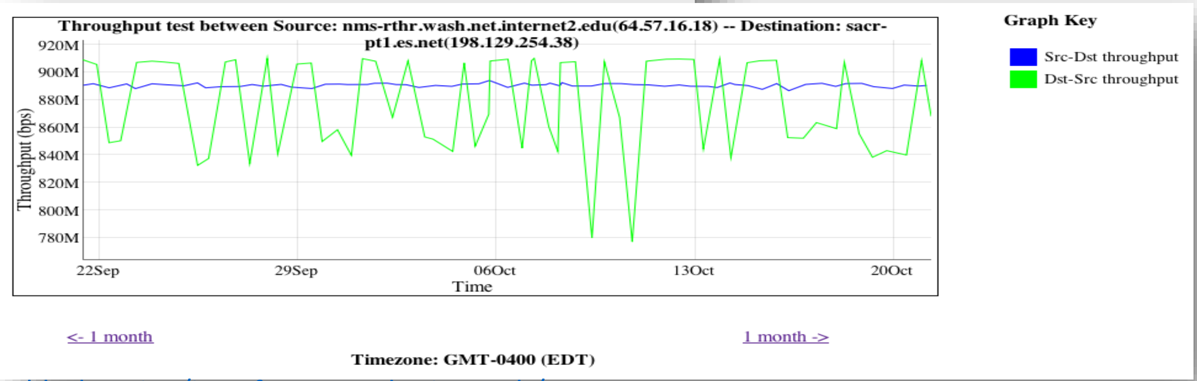
#14 Speed Mismatch



Throughput test between Source: elpa-pt1.es.net(198.129.254.82) -- Destination: nms-rthr.wash.net.internet2.edu(64.57.16.18)

<- 1 month

Timezone: GMT-0400 (EI



Throughput test between Source: nms-rthr.wash.net.internet2.edu(64.57.16.18) -- Destination: sacr-pt1.es.net(198.129.254.38)

<- 1 month

1 month ->

Timezone: GMT-0400 (EDT)

http://fasterdata.es.net/performance-testing/troubleshooting/interface-speed-mismatch/

http://fasterdata.es.net/performance-testing/evaluating-network-performance/impedence-mismatch/

http://bit.ly/EPOC-USC-CI

perfSONAR

Monitoring end-to-end systems

Jason Zurawski

zurawski@es.net

ESnet / Lawrence Berkeley National Laboratory

Training Workshop for Network Engineers and Educators on Tools and Protocols for High-Speed Networks
University of South Carolina
July 22-23, 2019

ESnet ENERGY SCIENCES NETWORK   GÉANT   INDIANA UNIVERSITY   INTERNET2   UNIVERSITY OF MICHIGAN

Outline

- Introduction
- Hardware & Software
- Tool Use
- Regular Testing
- Use Cases
- **Debugging**

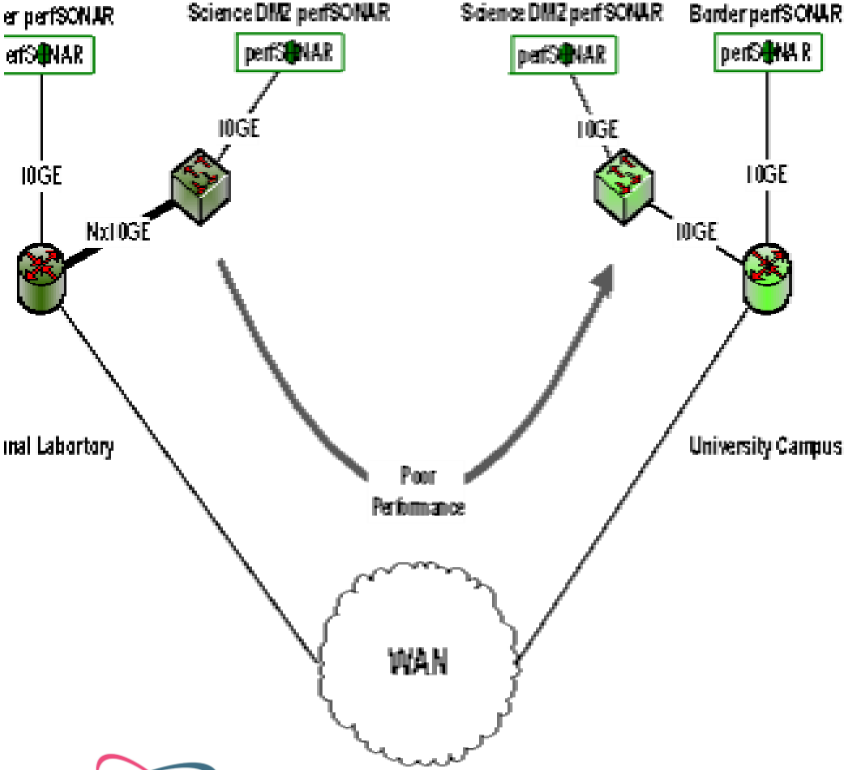
WAN Test Methodology – Problem Isolation

- We said it before, but it bears repeating: segment-to-segment testing is not helpful
  - TCP dynamics will be different, and in this case all the pieces do not equal the whole
    - E.g. high throughput on a 1ms path with high packet loss vs. the same segment in a longer 20ms path
  - Problem links can test clean over short distances
  - An exception to this is hops that go thru a firewall

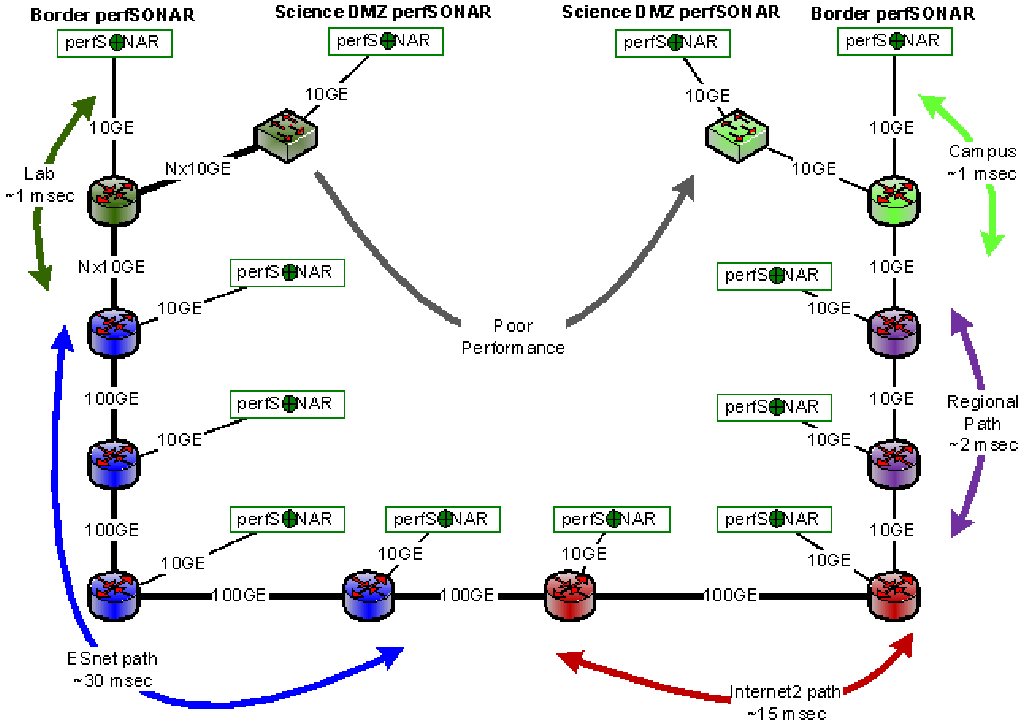
WAN Test Methodology – Problem Isolation

- Run long-distance tests
    - ***Run the longest clean test you can***, then look for the ***shortest dirty test*** that includes the path of the clean test
- In order for this to work, the testers need to be already deployed when you start troubleshooting
    - ESnet has at least one perfSONAR host at each hub location.
        - Many (most?) R&E providers in the world have deployed at least 1
    - If your provider does not have perfSONAR deployed ask them why, and then ask when they will have it done

Network Performance Troubleshooting Example

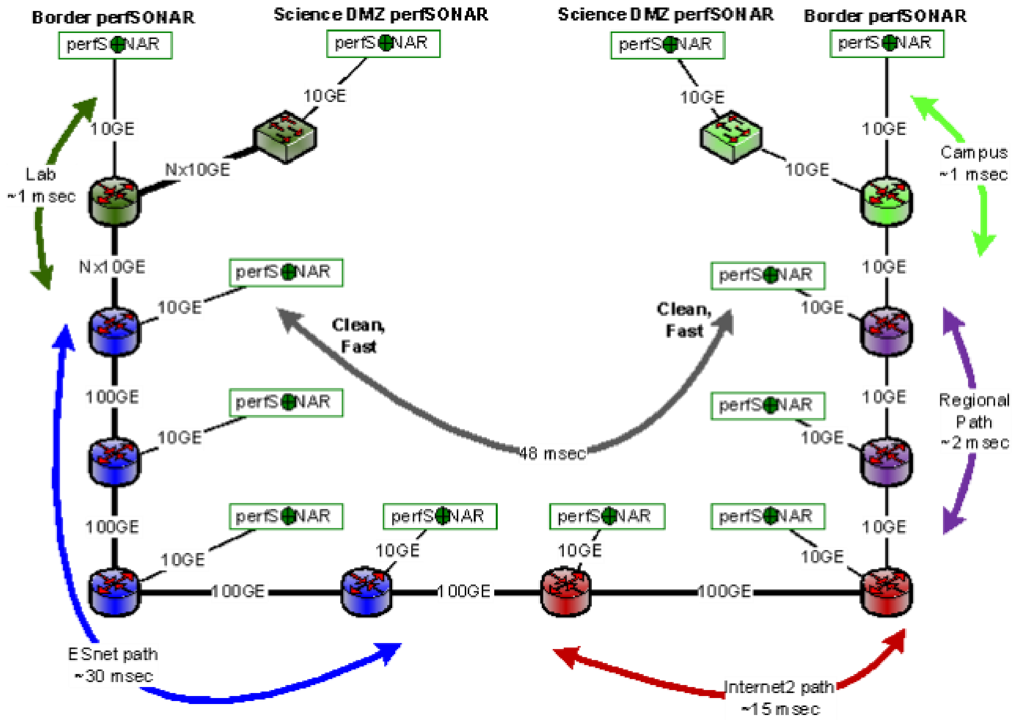


Wide Area Testing – Full Context

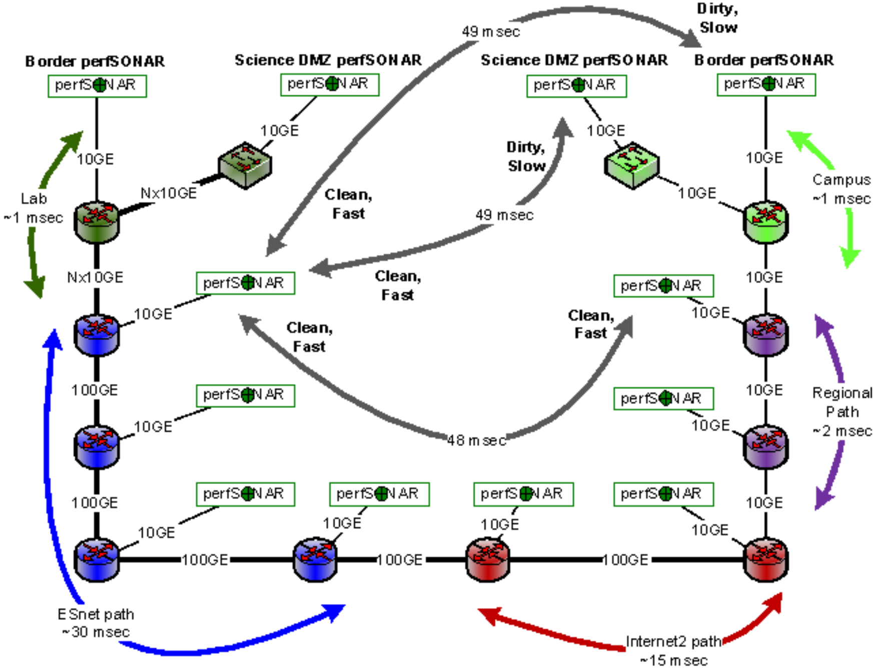


ESnet ENERGY SCIENCES NETWORK GÉANT INDIANA UNIVERSITY INTERNET2 UNIVERSITY OF MICHIGAN

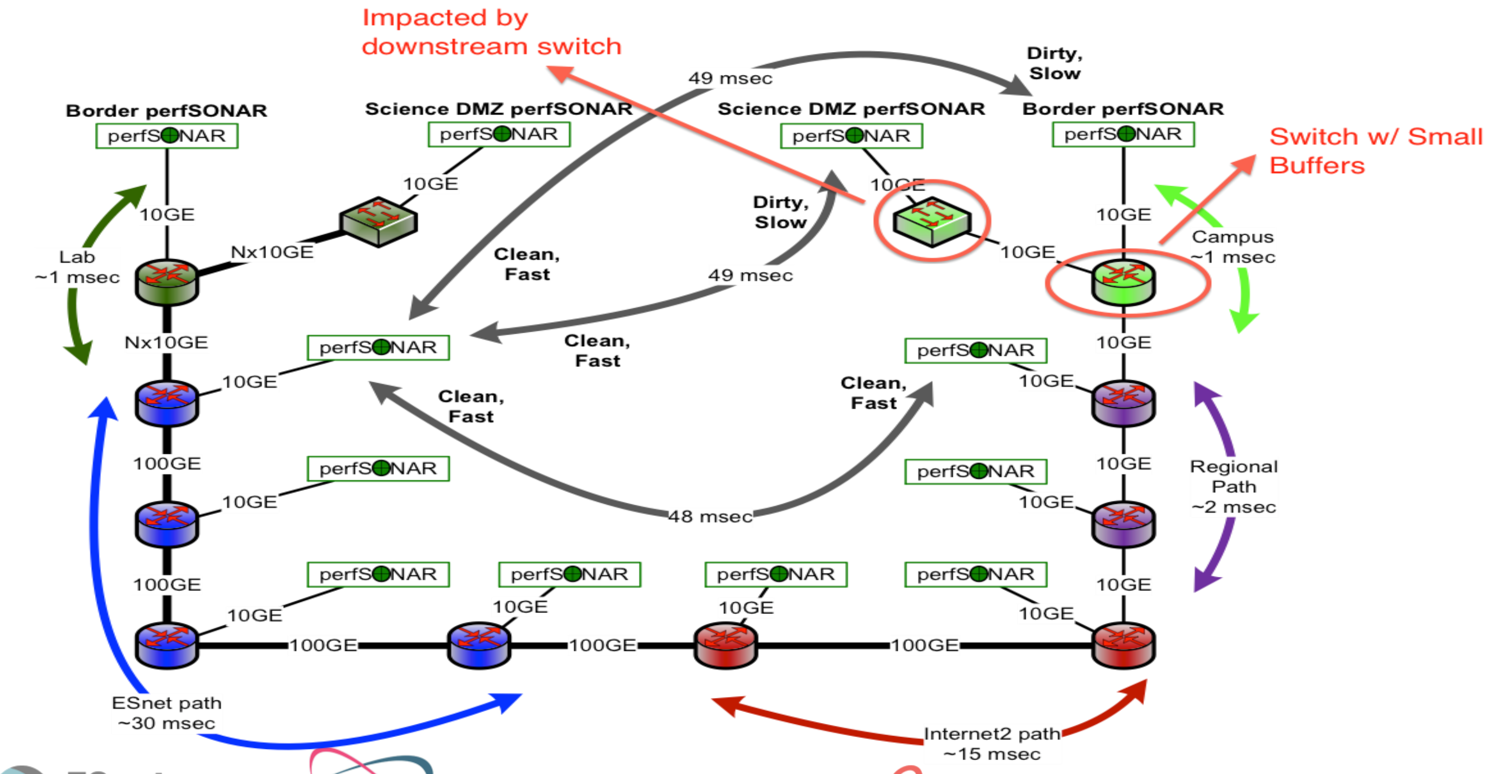
Wide Area Testing – Long Clean Test



Wide Area Testing – Poorly Performing Tests Illustrate Likely Problem Areas



Likely Problem Area



ESnet ENERGY SCIENCES NETWORK GÉANT INDIANA UNIVERSITY INTERNET2 UNIVERSITY OF MICHIGAN