March 11th 2013 – USCMS/ATLAS Session @ OSG AHM 2013

Jason Zurawski - Senior Research Engineer

Networking Potpourri

Current World View For Networking

"In any large system, there is always something broken."

Jon Postel

- Consider the technology:
  - 100G (and larger soon) Networking
  - Changing control landscape (e.g. SDN, be it OSCARS or OpenFlow, or something new)
  - Smarter applications and abstractions
- Consider the realities:
  - Heterogeneity in technologies
  - Mutli-domain operation
  - "old applications on new networks" as well as "new applications on old networks"

Outline

- Performance Topics
  - perfSONAR Release
  - OSG + perfSONAR
  - ATLAS/CMS Debugging
- DYNES Topics
  - Status
  - Some Things We Learned
  - Grant Completion
  - Follow Ons

perfSONAR Release

- Version 3.3 Is set to release in March (RC2 Available Now)
  - 'Mesh Config' Was the largest update
  - New LS Infrastructure (one that works this time)
  - CentOS 6
  - Litany of Bugs Fixed
- On Deck (Future Work):
  - Web10G instead of Web100 (new NDT, probably no more NPAD)
  - "Mesh" GUIs
  - 1G/10G 'Overdriving'
  - Scheduling OWAMP and BWCTL at the same time

OSG & perfSONAR Interactions

- Interactions are continuing (see Shawn's update for more Info)
  - Debugging Guide
    - Shawn/Jason started this earlier this year
    - Comprehensive guide to address performance problems using tools packaged with VDT (OWAMP, BWCTL, NDT)
  - Current Dashboard
    - Still going – BNL maintaining
    - Will go away once we get more assistance (hint)
  - Next Gen Alarming/Reporting via MyOSG
    - OSG team looking into how the dashboard works, as well as how NAGIOS plugins were designed
    - Ideal approach would be to grab data from mesh configured hosts at a site (no need to have multiple machines per VO) and build that into reports

ATLAS/CMS Debugging

- What I feel like most days:
  - Problems aren't hard, but time consuming considering the multi-domain aspect (and occasional lack of cooperation by the operations community)

ATLAS/CMS Debugging

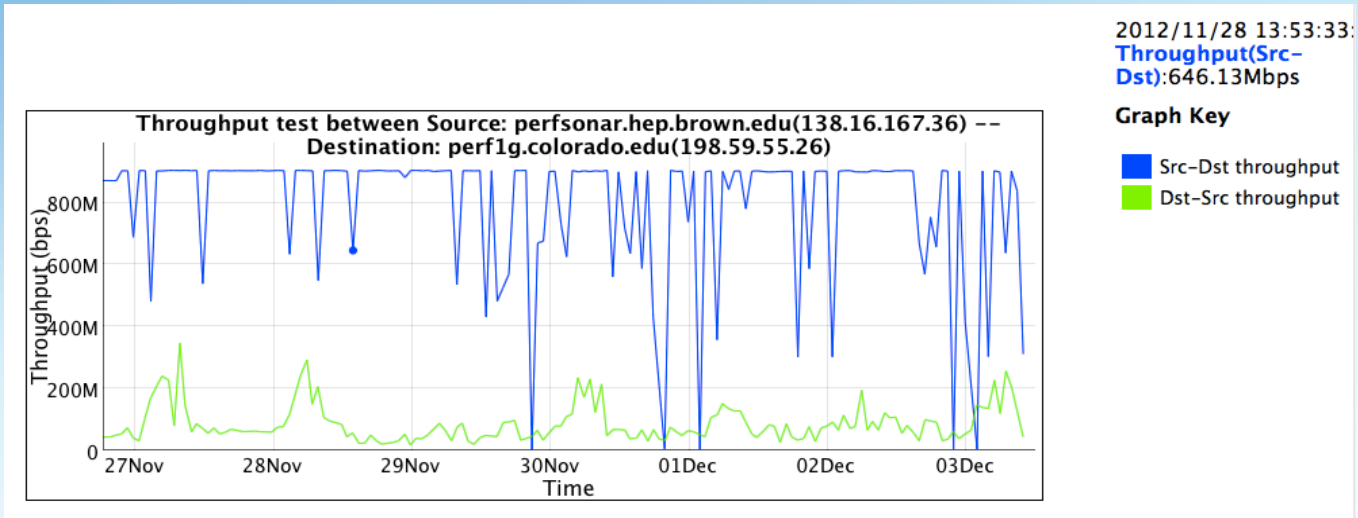
- Lots since the last update
  - Brown University Physics - Ongoing
    - The Firewall is not your friend
  - Florida & Vanderbilt – Done
    - Asymmetric routes to the EU, one route was doing pretty poorly
  - UMich/BNL/UTA – Ongoing
    - Weird issue – seems to be a problem going 'through' Starlight in Chicago
    - Still waiting on local engineer help
  - US to EU Packet Loss – Ongoing
    - Various reports of congestion and low throughput
    - Some VLAN juggling was required
    - New links across Atlantic 'helped'
    - Still some lingering packet loss in one LAG bundle that is being looked at by GEANT

Brown Firewall Case Examined

- Security is at constant odds with performance
  - Ports for communication
  - Slowing of otherwise un-interrupted flows
- Firewalls are a good example of security implemented in a vacuum, which gives off a 'false' sense of security
  - Security of the system vs. Security of a Component (network)
  - Configuration is challenging, and normally not updated
- Example comes from Brown University, and their Physics Department attempting to access another resource at University of Colorado (Boulder)

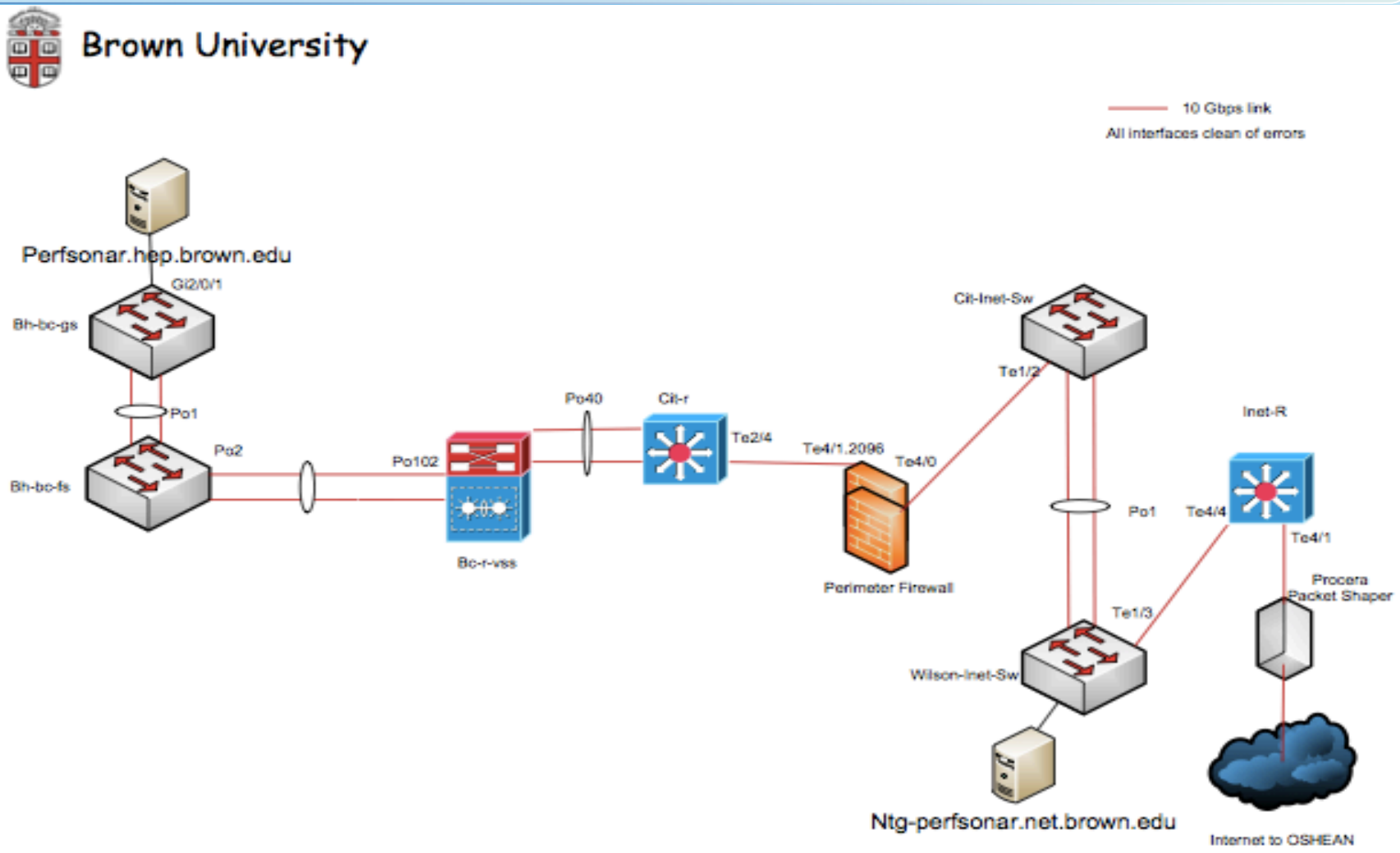
Observation From Inside of the Firewall

- End to End Bandwidth is Low:



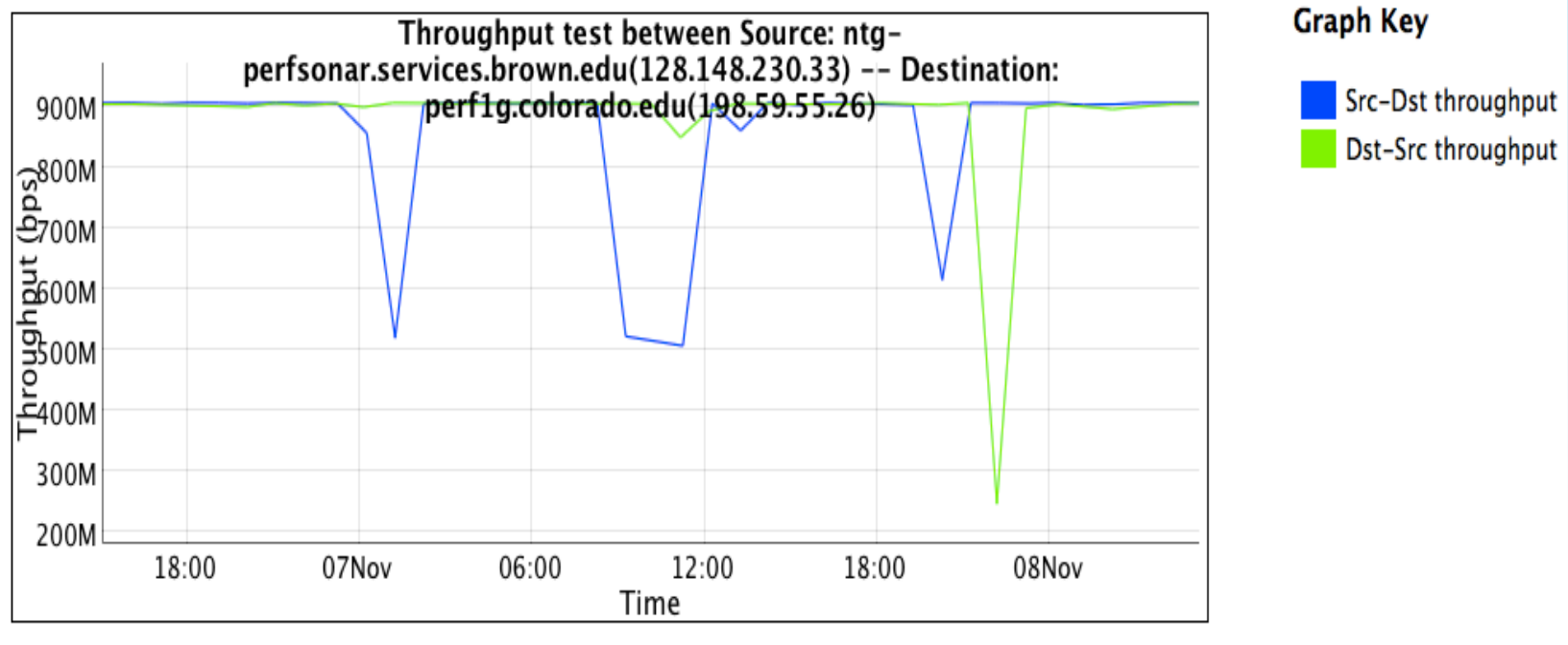
- "Outbound" from Brown University is fine (near 1G for a 1G tester)
- "Inbound" from Colorado to Brown is not (this is the direction the Firewall is patrolling)

Campus Map


Brown University
10 Gbps link
All interfaces clean of errors
Perfsonar.hep.brown.edu
Gi2/0/1
Bh-bc-gs
Po1
Po2
Bh-bc-fs
Po102
Bc-r-vss
Po40
Cit-r
Te2/4
Te4/1.2096
Te4/0
Perimeter Firewall
Cit-Inet-Sw
Te1/2
Po1
Te1/3
Wilson-Inet-Sw
Ntg-perfsonar.net.brown.edu
Inet-R
Te4/4
Te4/1
Procera
Packet Shaper
Internet to OSHEAN

Observation From Outside of the Firewall

- High performance in and out – the firewall is slowing down transmissions inbound:



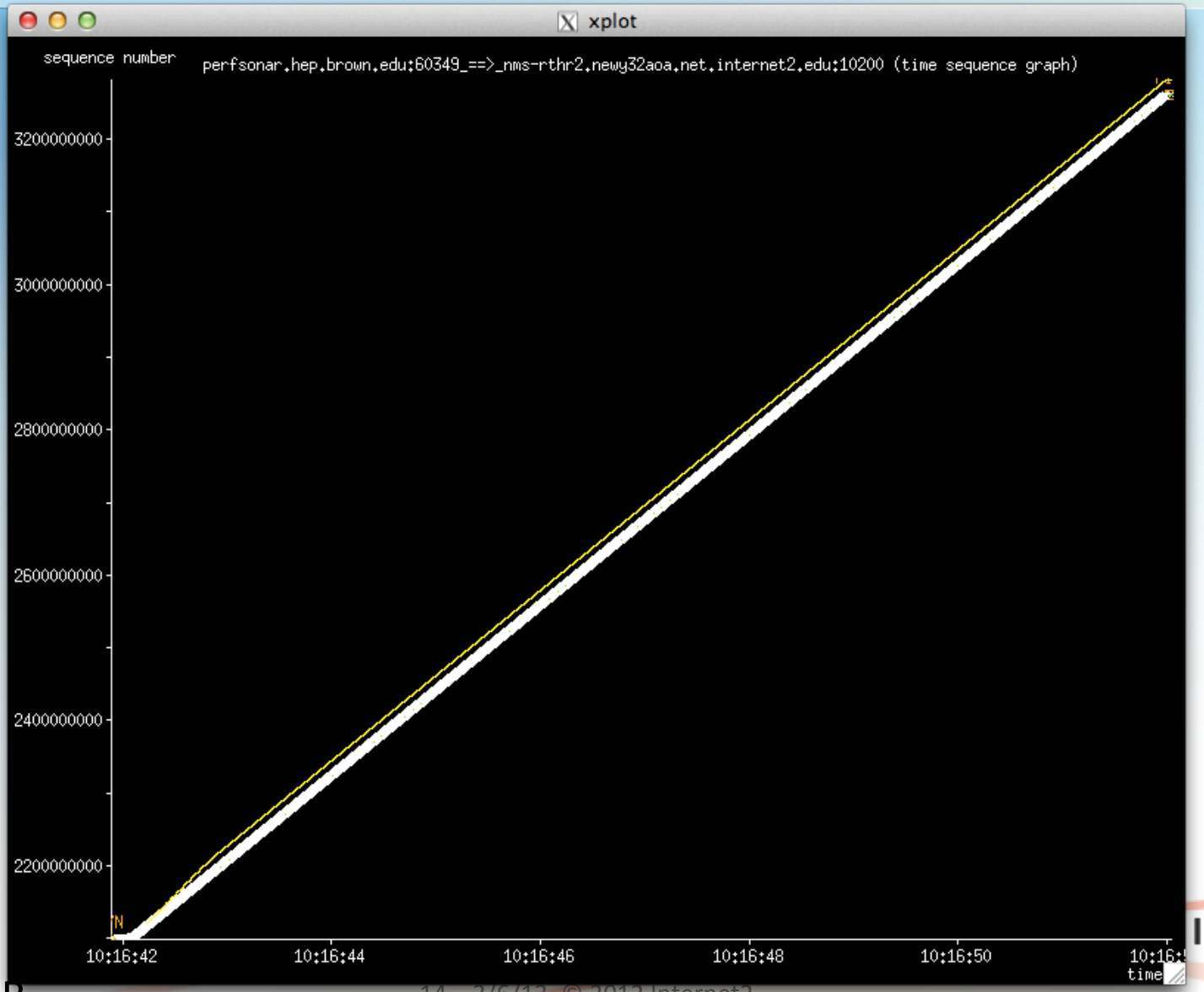
What We are Seeing

- "Outbound" Bypassing Firewall
  - Firewall will normally not impact traffic leaving the domain. Will pass through device, but should not be inspected
- "Inbound" Through Firewall
  - Statefull firewall process:
    - Inspect packet header
    - If on cleared list, send to output queue for switch/router processing
    - If not on cleared list, inspect and make decision
    - If cleared, send to switch/router processing.
    - If rejected, drop packet and blacklist interactions as needed.
  - Process slows down all traffic, even those that match a white list

Debugging (Outbound)

- Run “nuttcp” server:
  - `nuttcp -S -p 10200 –nofork`
- Run “nuttcp” client (opposite end of transfer):
  - `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`

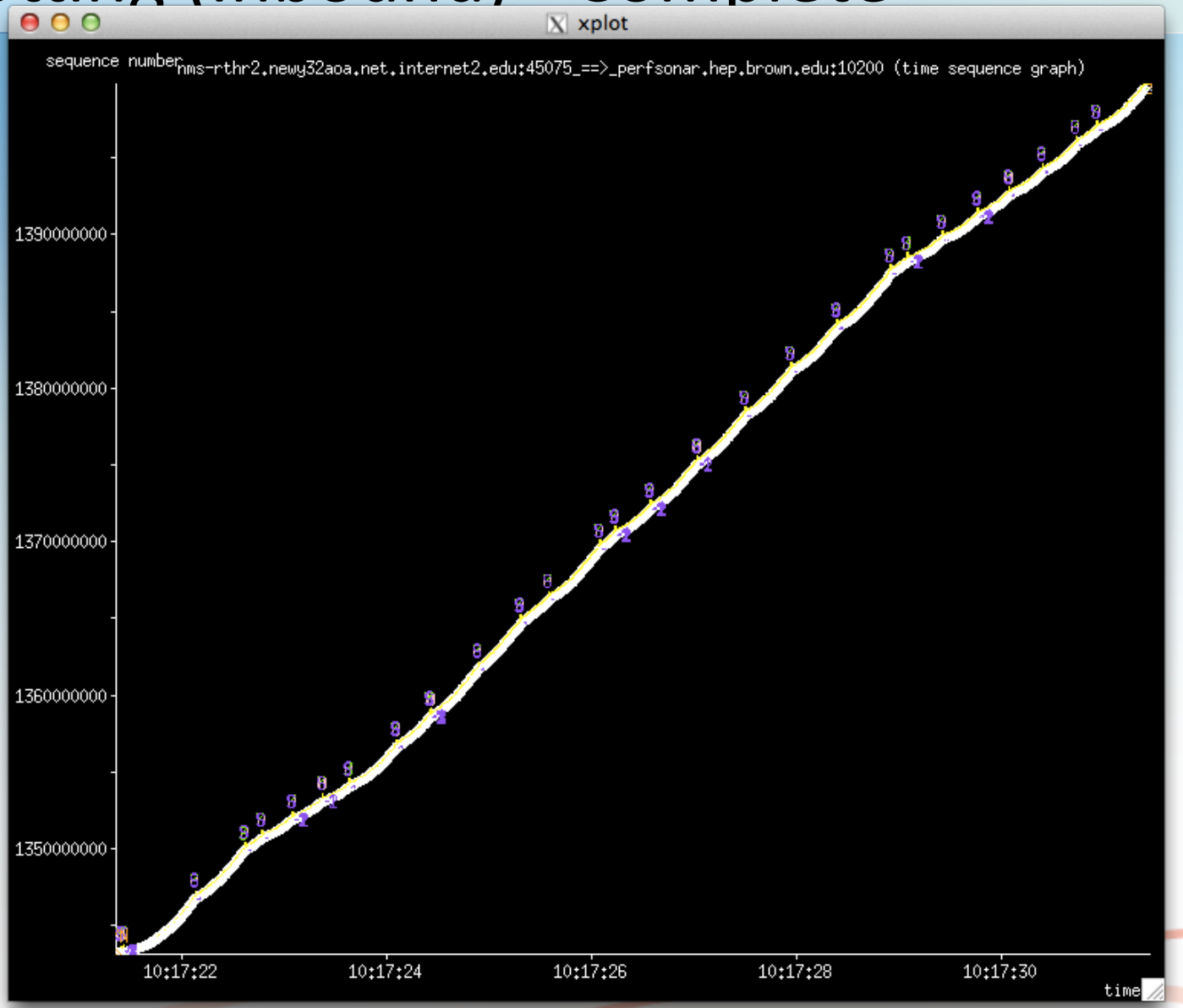
Plotting (Outbound) - Complete



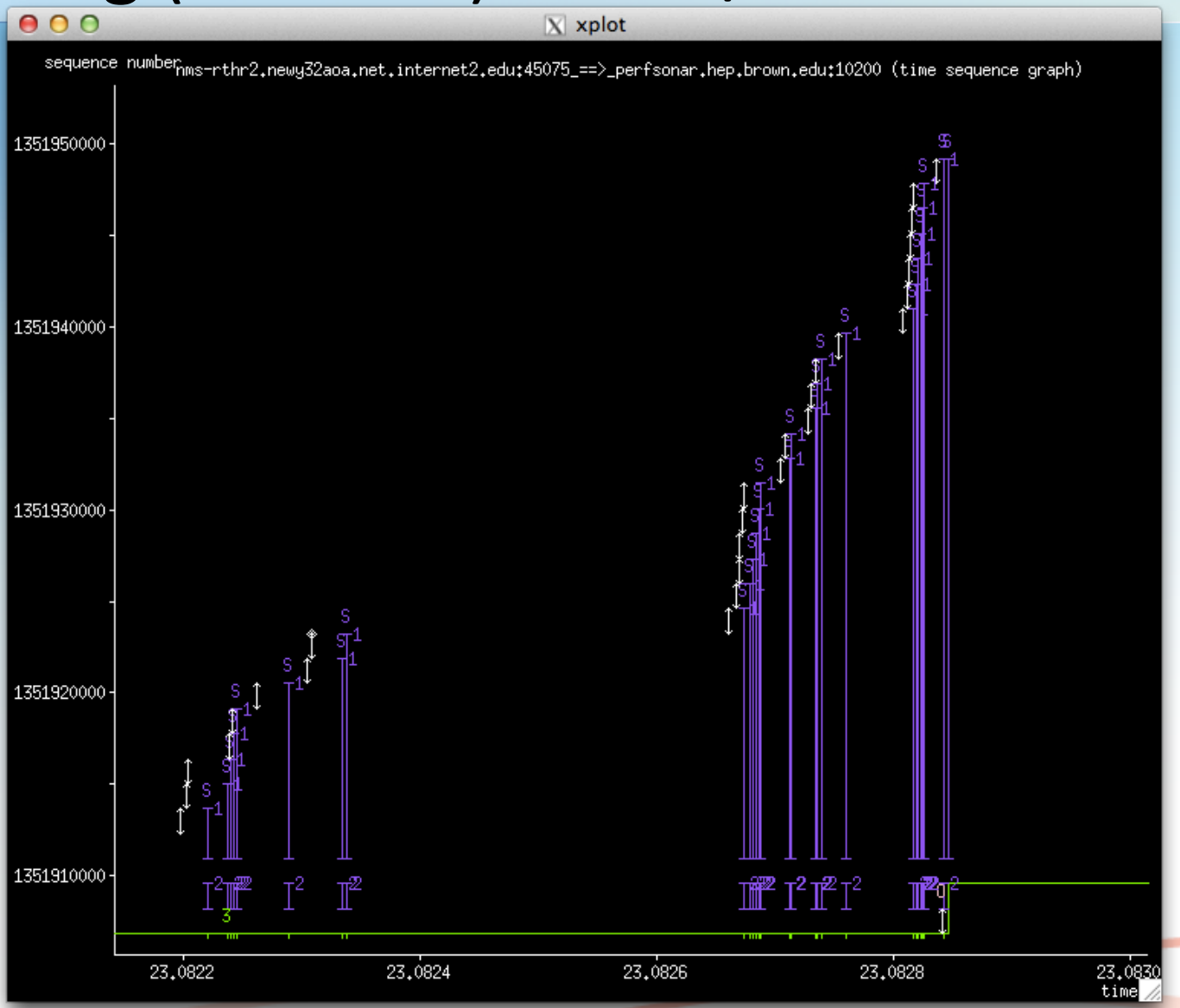
Debugging (Inbound)

- Run “nuttcp” server:
  - `nuttcp -S -p 10200 –nofork`
- Run “nuttcp” client:
  - `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`

Plotting (Inbound) - Complete



Plotting (Inbound) – OOP/Retransmits



perfSONAR powered

INTERNET 2

What Are We Seeing?

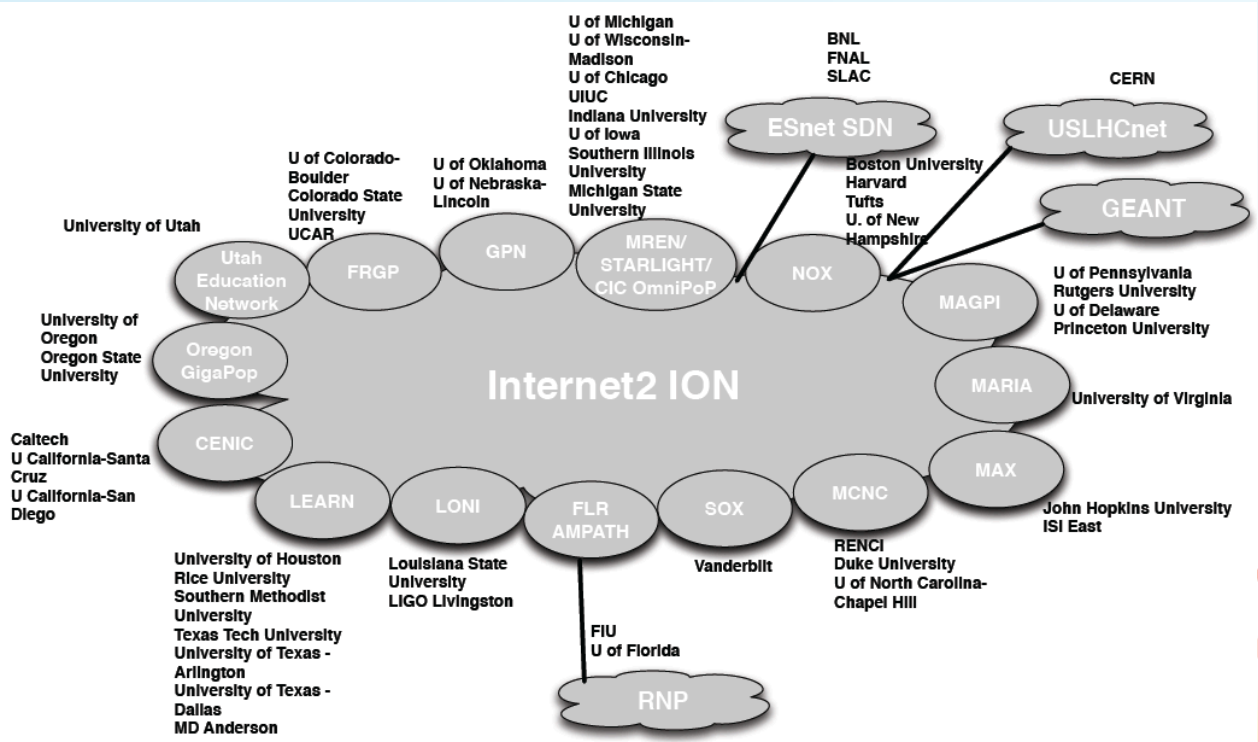
- Packets take a long time to process on ingress queue of FW – note we are not actually dropping traffic, but the delay ***feels*** like that
- Sending end's TCP timer starts to go off if it doesn't see ACKs. Retransmissions start
- Eventually packets make it through to receiver, and ACKs start
- Retransmissions start to make it through too … duplicate ACKs are sent from receiver
- 3 duplicate ACKs = Fast Retransmit/SACK process – e.g. "It's All Over", and we will never do well again
- Flow is never able to recover, an this seems to happen every couple of seconds

Outline

- Performance Topics
  - perfSONAR Release
  - OSG + perfSONAR
  - ATLAS/CMS Debugging
- DYNES Topics
  - Status
  - Some Things We Learned
  - Grant Completion
  - Follow Ons

Status

- Running total here: http://www.internet2.edu/ion/status.html
- Basically 42% or so are Done, 48% are In-Progress (of those 60% are late additions to the project), and 10% are stalled for some other reason beyond our control.
- Map:



Some Things We Learned

- Software Related
  - OSCARS switched to a new release midway through our project, and this complicated things (upgrades and downgrades)
  - DRAGON no longer had support
  - OESS (OpenFlow control software) was too new for our hardware
- Task List Related
  - Its hard to send out 50+ sets of hardware, ordered by 4 different parties, and get it to work in a remote fashion
  - More time should have been spent on instructions for end sites vs. direct intervention
- Infrastructure Related
  - Internet2 ION needs more capacity (in progress)
  - QoS can hurt TCP if you don't do it 'right'

QoS Hurts TCP

- Circuit is implemented on top of packet networks using QoS
  - Different queues for different traffic
    - Circuit = Expedited
    - IP = Best Effort
    - “Scavenger” = Less Than Best Effort
  - The latter queue is used for traffic that goes beyond circuit reservation

TCP Use

- TCP doesn't have a notion of 'pace', so it will just send all traffic into the network at once:

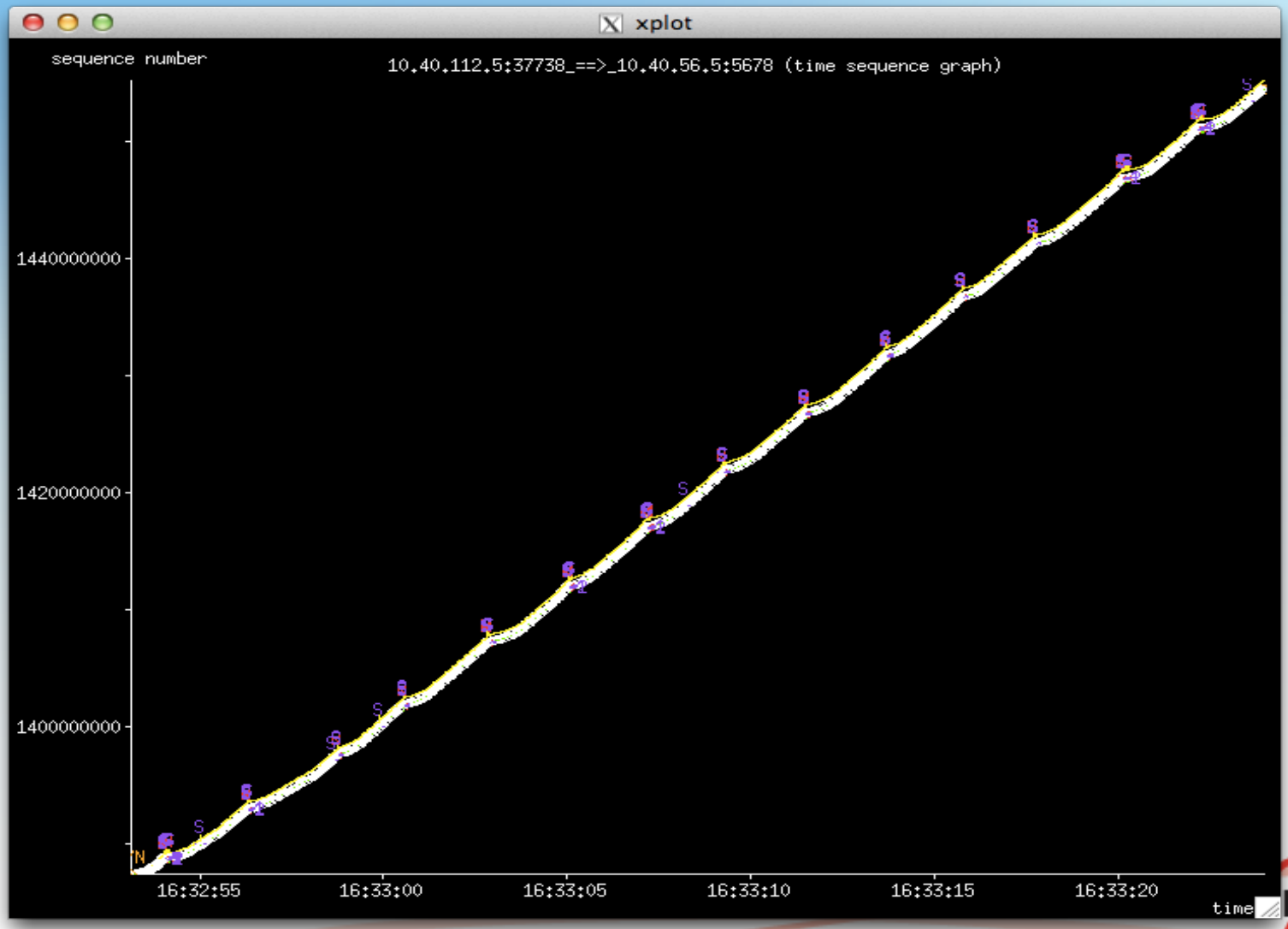
```
[dynes@fdt-wisc ~]$ nuttcp -T 30 -i 1 -p 5678 -P 5679 10.40.56.5
    1.2500 MB /   1.00 sec =   10.4844 Mbps    15 retrans
    1.4375 MB /   1.00 sec =   12.0587 Mbps     0 retrans
    2.2500 MB /   1.00 sec =   18.8749 Mbps     2 retrans
    1.5000 MB /   1.00 sec =   12.5825 Mbps     0 retrans
    1.7500 MB /   1.00 sec =   14.6808 Mbps     0 retrans
    2.0625 MB /   1.00 sec =   17.3013 Mbps     2 retrans
    2.5625 MB /   1.00 sec =   21.4956 Mbps     0 retrans
    1.7500 MB /   1.00 sec =   14.6804 Mbps     1 retrans
    2.5000 MB /   1.00 sec =   20.9711 Mbps     0 retrans
    2.0625 MB /   1.00 sec =   17.3016 Mbps     3 retrans
    1.9375 MB /   1.00 sec =   16.2526 Mbps     0 retrans
    2.4375 MB /   1.00 sec =   20.4475 Mbps     2 retrans
    2.0625 MB /   1.00 sec =   17.3018 Mbps     0 retrans
    2.7500 MB /   1.00 sec =   23.0675 Mbps     4 retrans
    1.6250 MB /   1.00 sec =   13.6318 Mbps     0 retrans
    2.6250 MB /   1.00 sec =   22.0196 Mbps     1 retrans
    1.6250 MB /   1.00 sec =   13.6316 Mbps     0 retrans
    2.5625 MB /   1.00 sec =   21.4963 Mbps     0 retrans
    1.6250 MB /   1.00 sec =   13.6313 Mbps     3 retrans
    2.5625 MB /   1.00 sec =   21.4961 Mbps     0 retrans
    2.0625 MB /   1.00 sec =   17.3014 Mbps     3 retrans
    2.4375 MB /   1.00 sec =   20.4473 Mbps     0 retrans
    2.0625 MB /   1.00 sec =   17.3010 Mbps     4 retrans
    2.5000 MB /   1.00 sec =   20.9719 Mbps     0 retrans
    1.8125 MB /   1.00 sec =   15.2046 Mbps     1 retrans
    2.3125 MB /   1.00 sec =   19.3979 Mbps     0 retrans
    2.5625 MB /   1.00 sec =   21.4959 Mbps     3 retrans
    1.5000 MB /   1.00 sec =   12.5834 Mbps     0 retrans
    2.6250 MB /   1.00 sec =   22.0201 Mbps     2 retrans
    1.3125 MB /   1.00 sec =   11.0100 Mbps     0 retrans

   64.0112 MB /  30.77 sec =   17.4531 Mbps 0 %TX 0 %RX 46 retrans 36.68 msRTT
```

Explanation

- TCP will blast packets into the network during “slow start”
  - Tries to find the limit of the network
  - Buffering implemented by QoS could be small (128K on Dell’s, larger on something like a Juniper T1600)
  - This lack of buffer causes our first hit
- As TCP window grows, and more data is sent into the network, queue use goes from E to E and LBE
  - Causes OOP to occur
  - Delays in receiving all data in the window, forces SACK/ Fast Retransmit behavior

XPlot of TCP Flow



Possible Solutions

- Application Pacing
  - Instruct application to pace traffic to a set BW or Buffer size
  - Challenging to do – Kernel gets to pick things even after application requests
- Host QoS (Linux TC)
  - Implemented on sending interface – can set a specific rate to limit/smooth traffic

```
sudo /usr/sbin/tc qdisc del dev eth0.3123 root
sudo /usr/sbin/tc qdisc add dev eth0.3123 handle 1: root htb
sudo /usr/sbin/tc class add dev eth0.3123 parent 1: classid 1:1 htb rate 112.5mbps
sudo /usr/sbin/tc filter add dev eth0.3123 parent 1: protocol ip prio 16 u32 match ip src 10.10.200.20/32 flowid 1:1
```

TCP w/ TC Results – Much Better

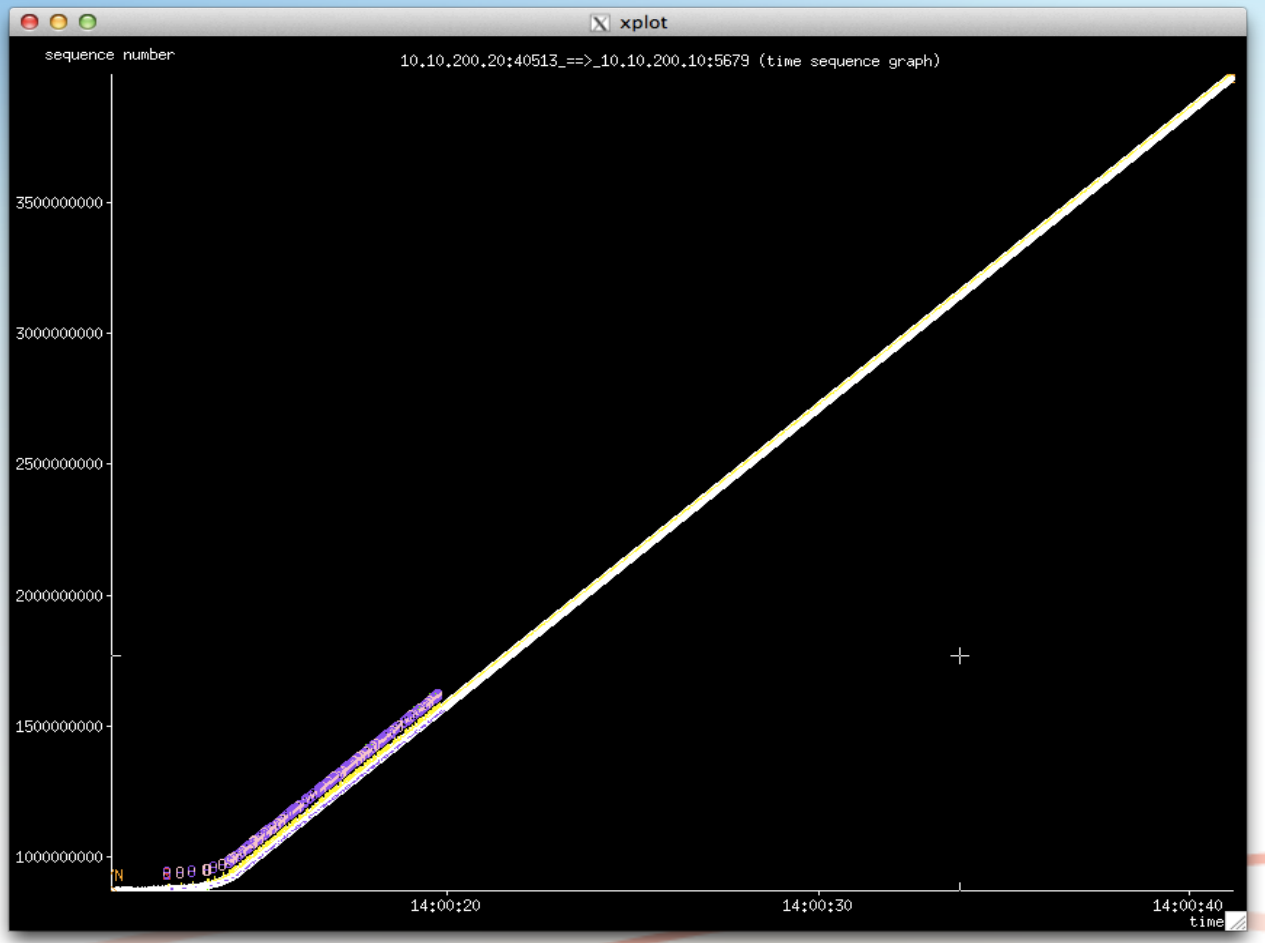
- Key is to smooth to a BW limit below the reservation (900M on a 1G circuit):

```
[dynes@fdt-wisc ~]$ nuttcp -T 30 -i 1 -p 5679 -P 5678 10.10.200.10
    2.1875 MB /   1.00 sec =   18.3486 Mbps     0 retrans
    8.3125 MB /   1.00 sec =   69.7281 Mbps     1 retrans
   28.3125 MB /   1.00 sec =  237.5170 Mbps     0 retrans
   99.1875 MB /   1.00 sec =  832.0559 Mbps     0 retrans
  108.5000 MB /   1.00 sec =  910.1831 Mbps     0 retrans
  108.4375 MB /   1.00 sec =  909.6078 Mbps     0 retrans
  108.4375 MB /   1.00 sec =  909.6706 Mbps     0 retrans
...
  108.4375 MB /   1.00 sec =  909.6397 Mbps     0 retrans
  108.3125 MB /   1.00 sec =  908.5911 Mbps     0 retrans

 2965.6678 MB /  30.12 sec =  825.9052 Mbps 3 %TX 8 %RX 1 retrans 36.73 msRTT
```

Graphical Representation

- We see some loss in the start as we get the window size sorted out



QoS Conclusions

- TCP may not be the correct protocol
  - UDP does pretty well
  - UDT/others may do better
- Old applications – new networks
  - File transfer (e.g. GridFTP) is a target use for circuits, thus TCP will be used
  - Killing the network with parallel streams will not help
  - Host smoothing is the best way to mitigate the badness in TCP in this case – but this is still not ideal

Grant Completion

- July 31st
  - Money needs to be gone, so do final reports
- After July 31st
  - There will still be sites 'not ready'
  - Support email will be passed around to get things working
- "Using" DYNES is a different matter
  - FDT was provided as the default application
  - Still working with Globus Online to integrate DYNES into their list of endpoints
  - Phoebus/XSP (application developed at Indiana) is another alternative that can be used with GridFTP – idea is to take the guesswork out of making a circuit

Follow Ons

- ANSE Grant
  - Caltech, University of Michigan, Vanderbilt University, UT Arlington
  - Use the framework of DYNES to build intelligent applications (e.g. Mods to PanDA/PhEDEx)
  - Expires Dec 31, 2014
- Internet2 Advanced Layer 2 Services
  - E.g. OpenFlow switches for Layer 2 activities
  - Similar to Internet2 ION, just build using newer gear, and can go up to 100G.

Questions/Comments

- ?

Networking Potpourri

March 11th 2013 – USCMS/ATLAS Session @ OSG AHM 2013

Jason Zurawski - Senior Research Engineer

For more information, visit http://www.internet2.edu/research