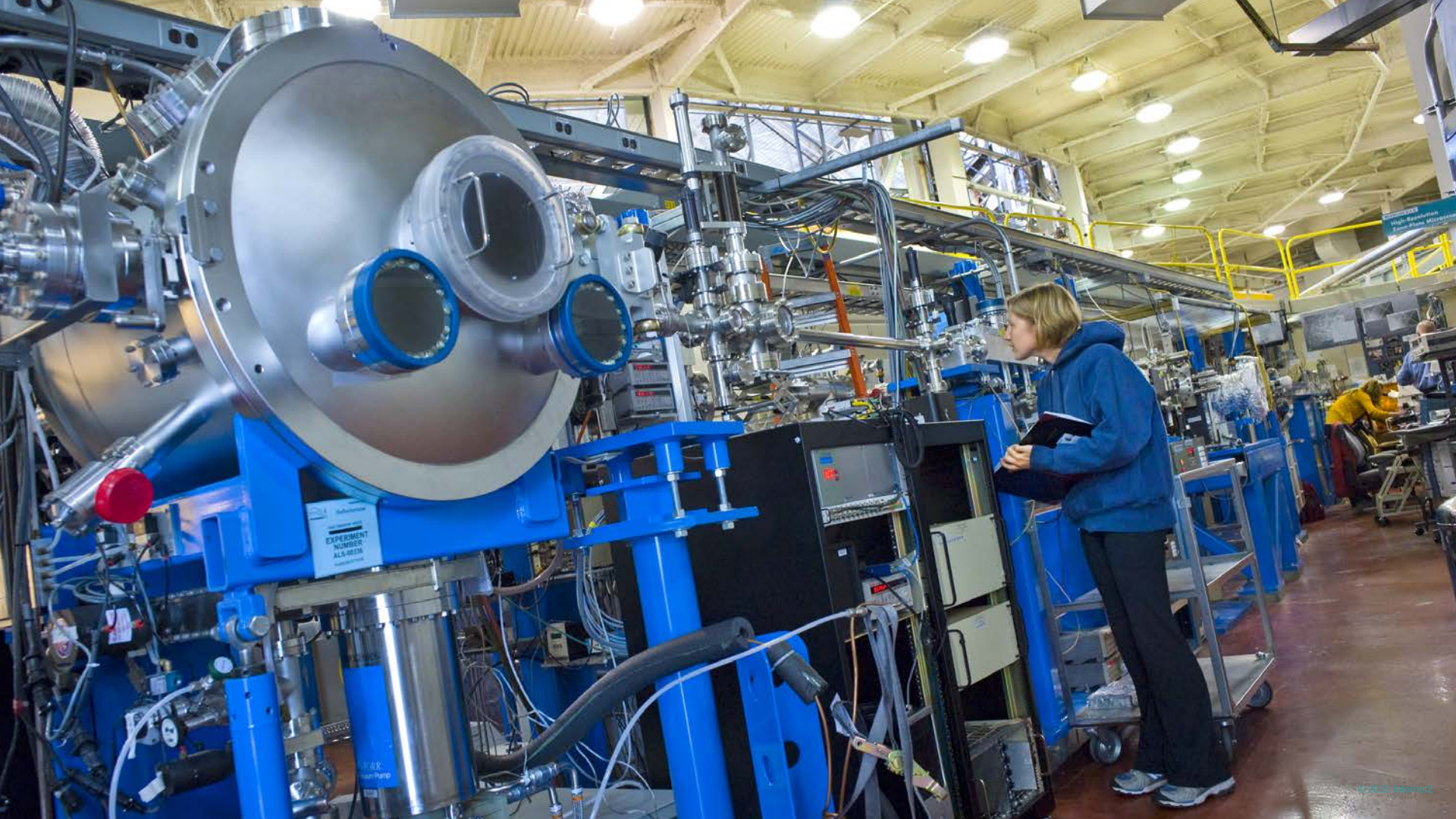
EVALUATING NETWORK BUFFER SIZE REQUIREMENTS

for Very Large Data Transfers

Michael Smitasin
Lawrence Berkeley National Laboratory (LBNL)

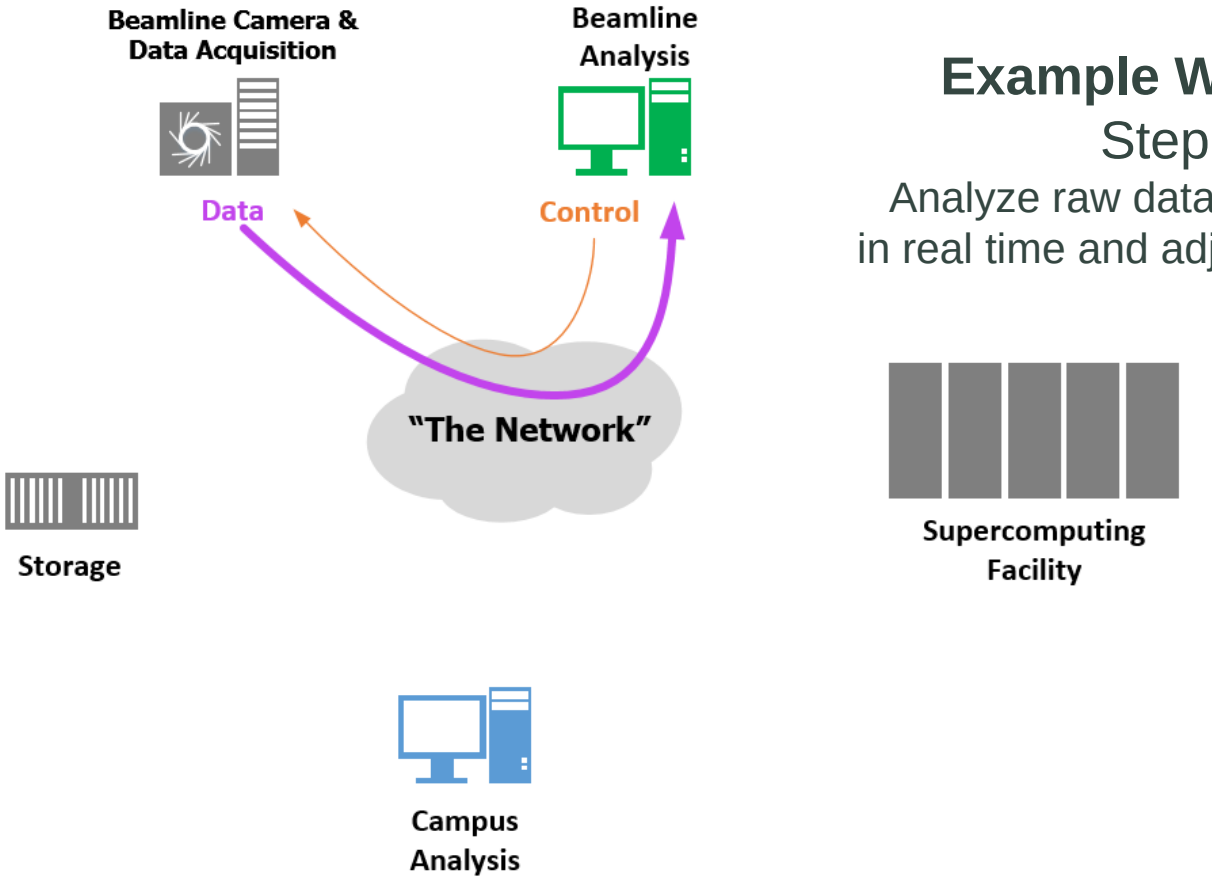
Brian Tierney
Energy Sciences Network (ESnet)


EXPERIMENT
NUMBER
ALS-8038

Example Workflow

Step 1

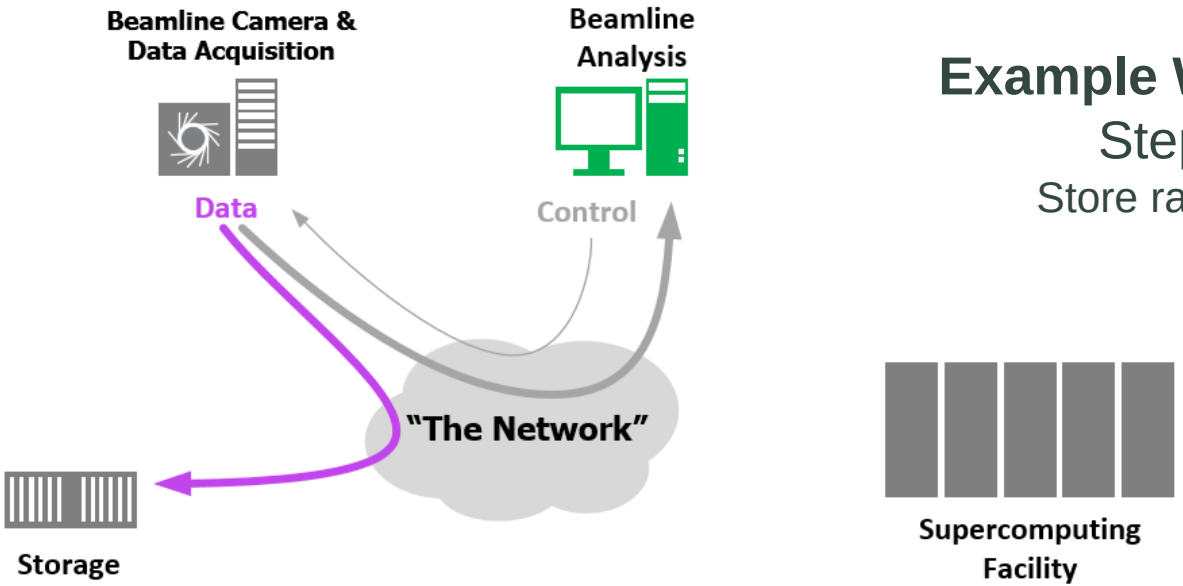
Analyze raw data at Beamline
in real time and adjust experiment



Example Workflow

Step 2

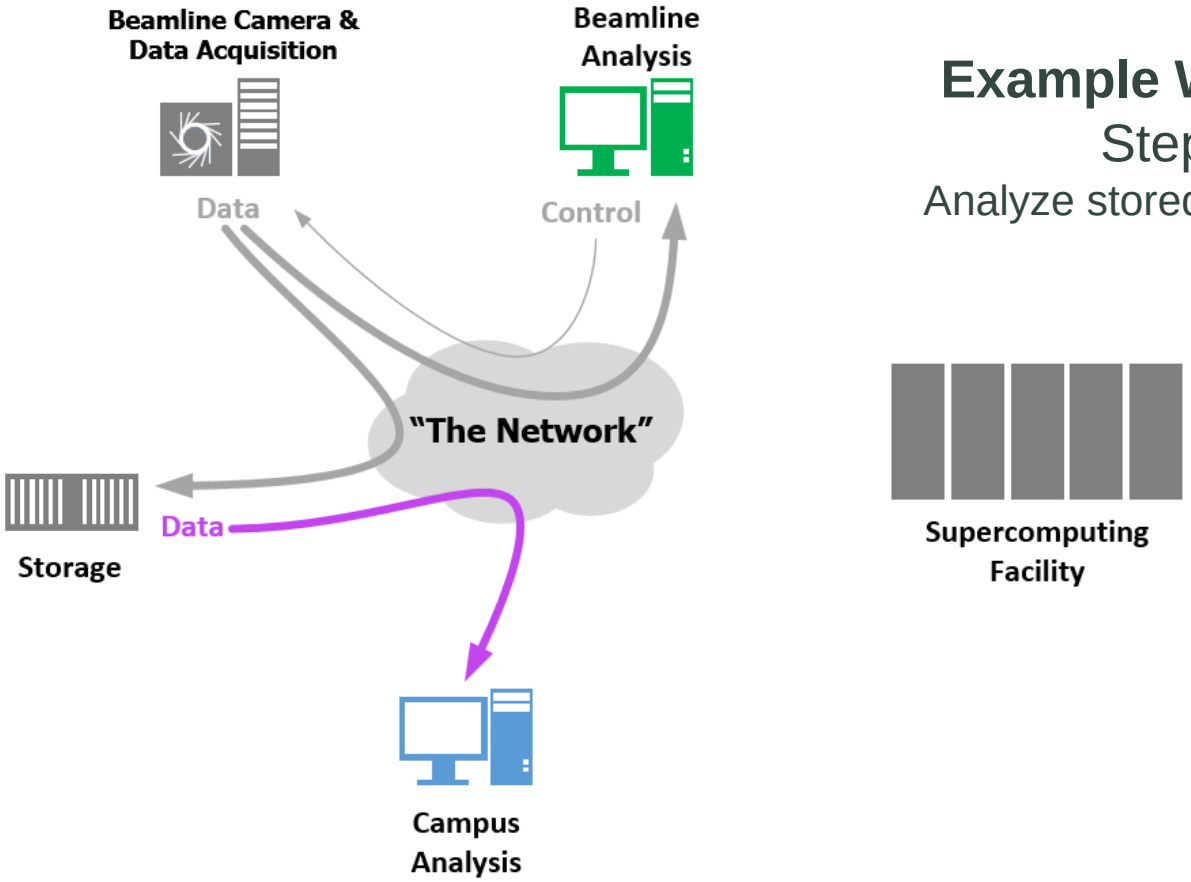
Store raw data



Example Workflow

Step 3

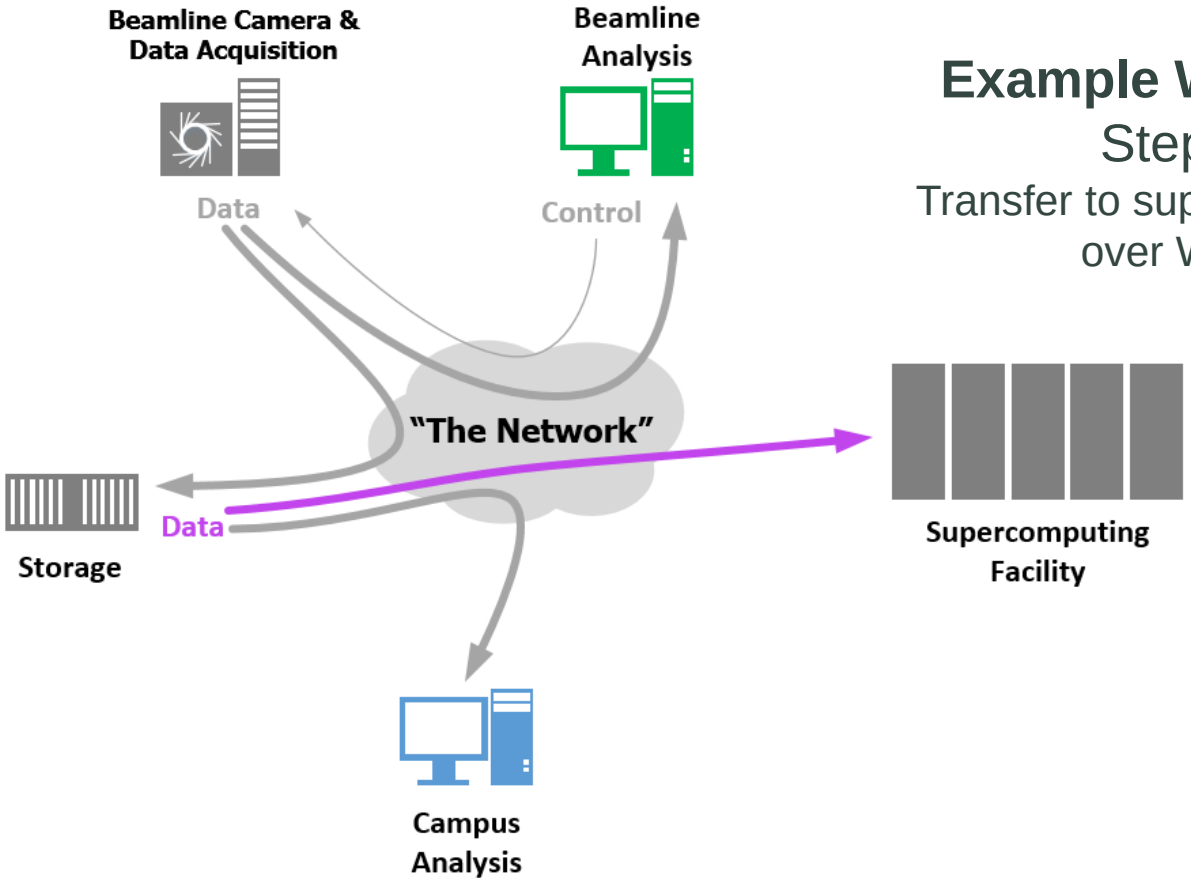
Analyze stored data locally



Example Workflow

Step 4

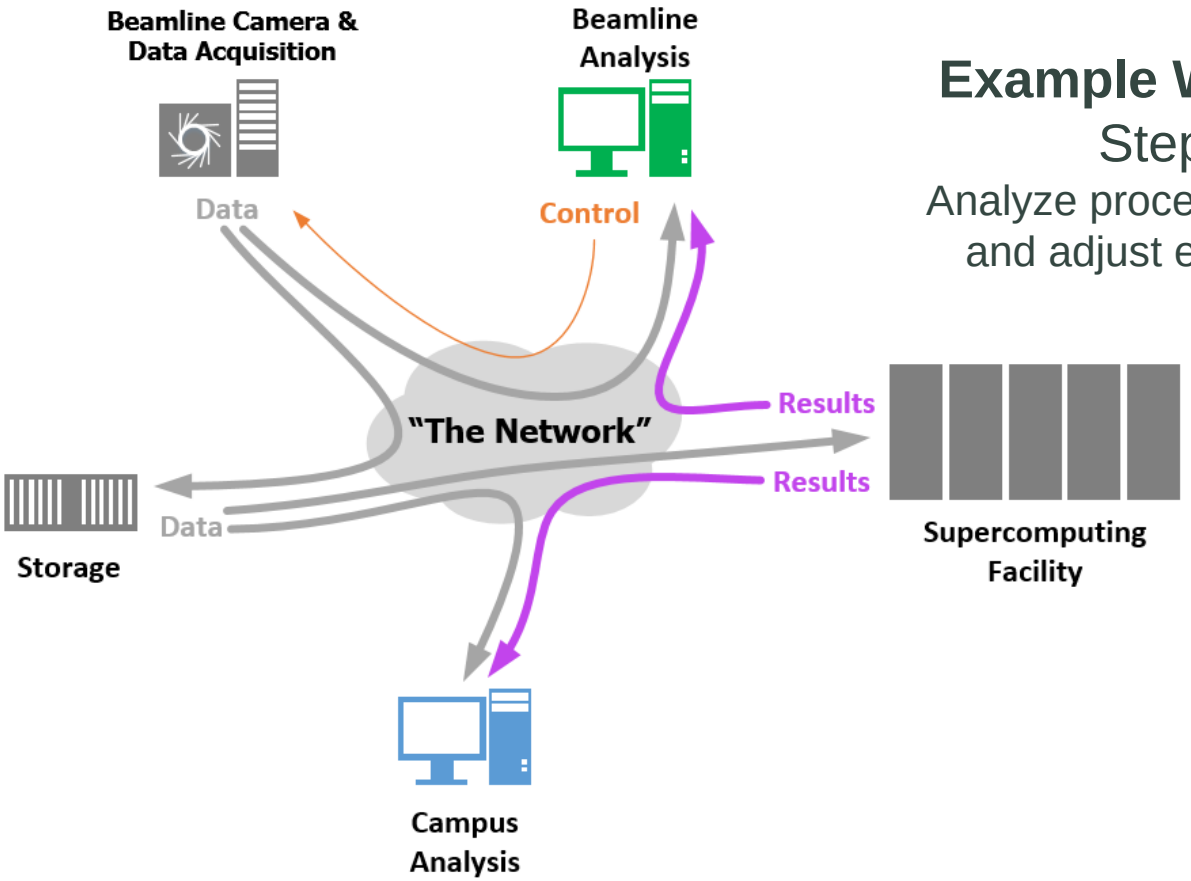
Transfer to supercomputers over WAN

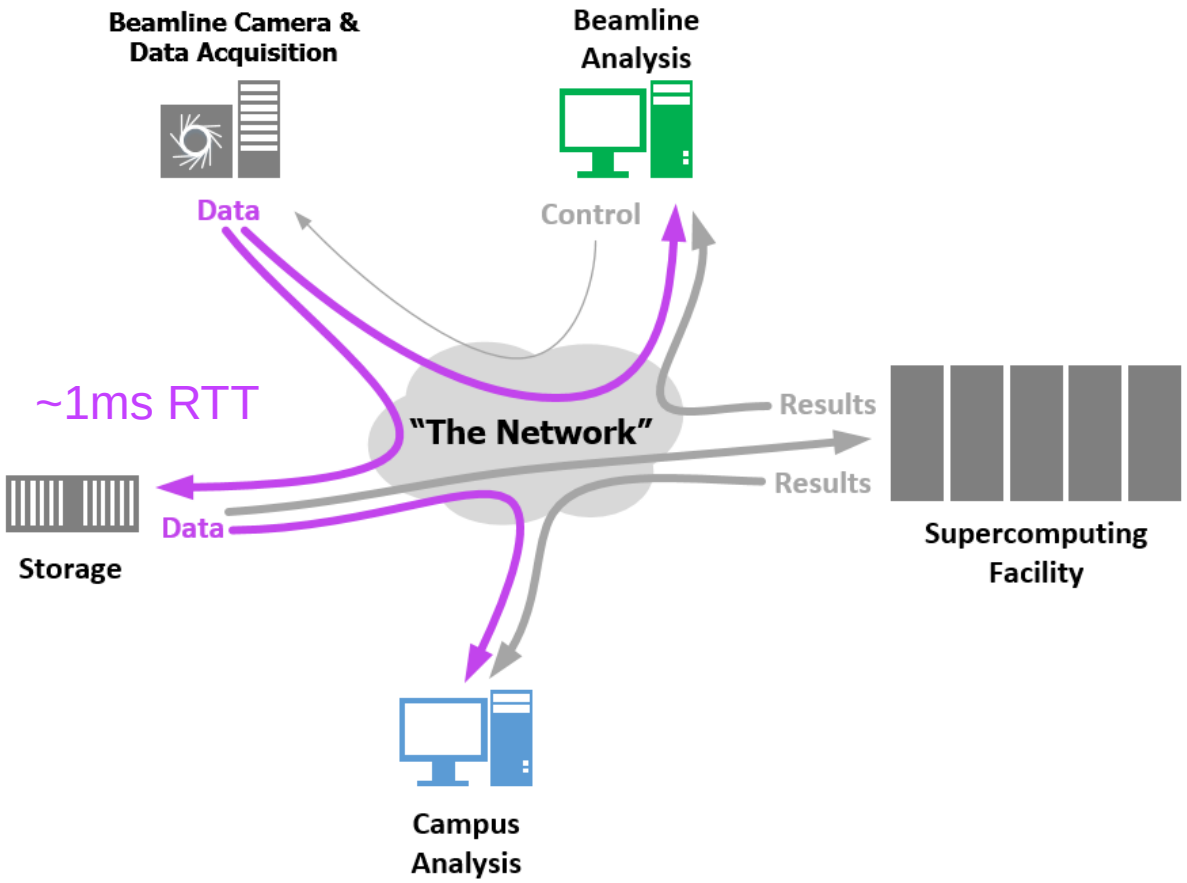


Example Workflow

Step 5

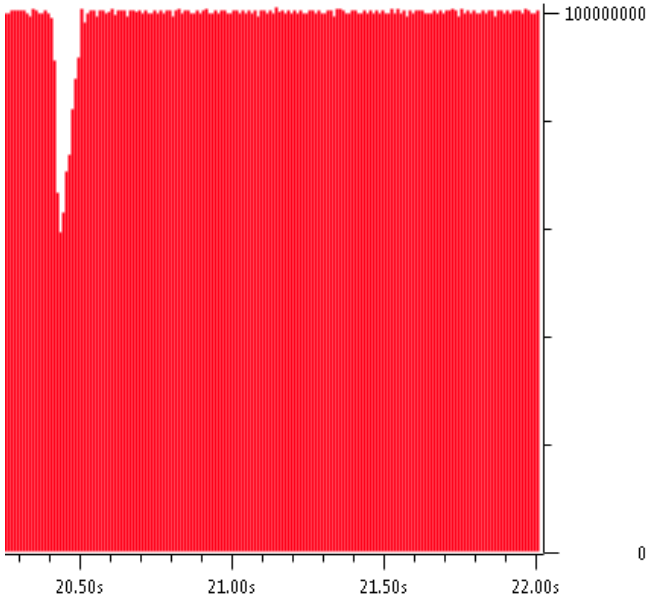
Analyze processed results and adjust experiment



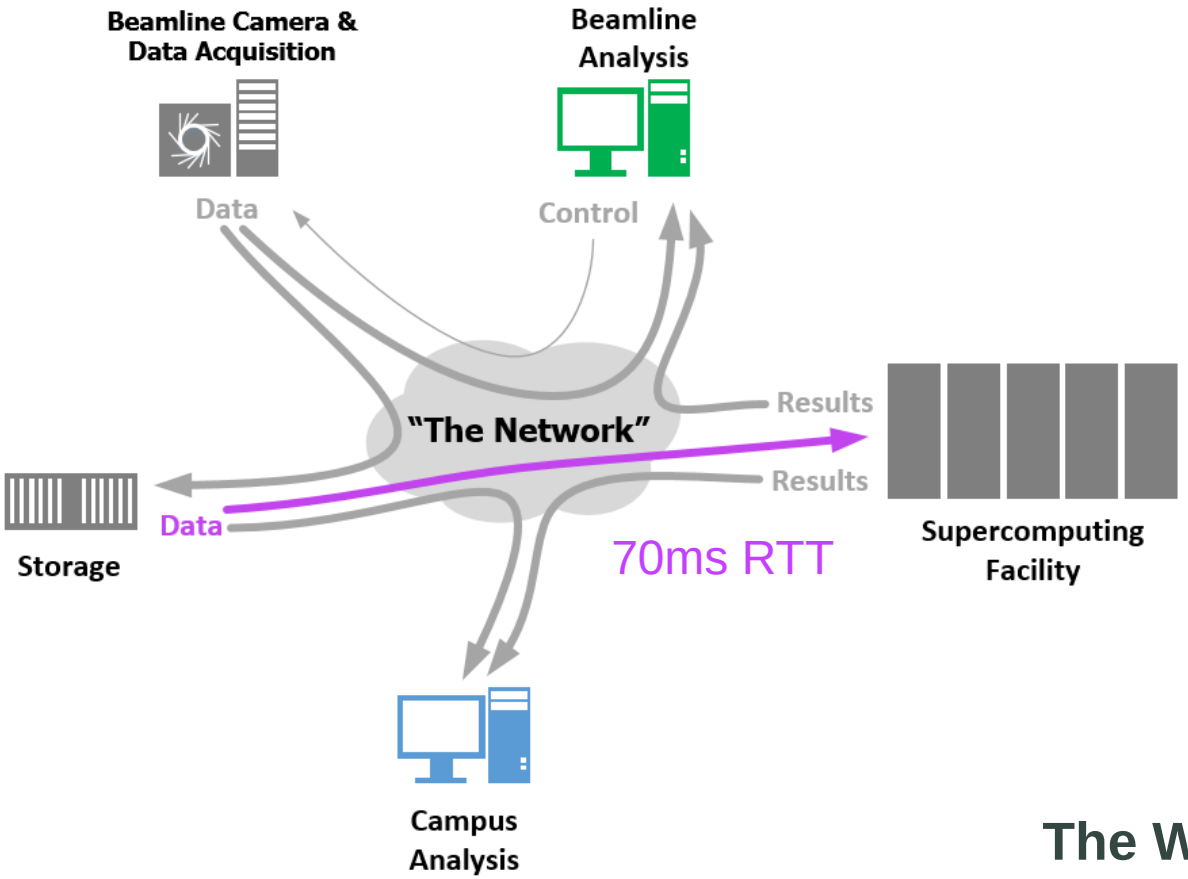


LAN Transfer

Bits per 10 millisecs[1]

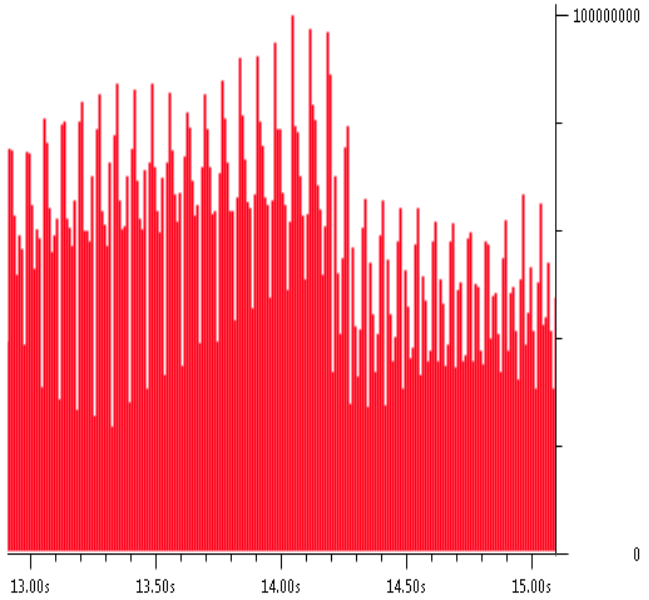


[1]100Mb/0.01sec = 10Gbps



WAN[1] Transfer

Bits per 10 millisecs[2]



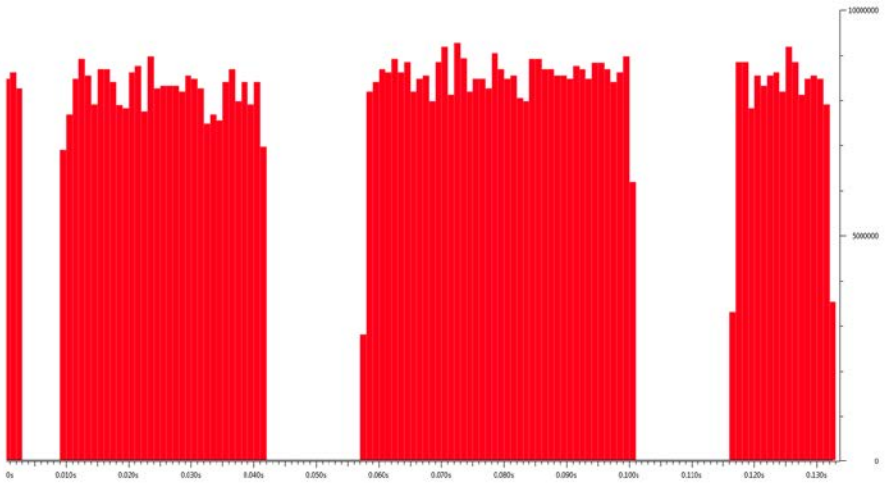
The WAN just sucks, right?

[1] 70ms simulated RTT via netem
[2] 100Mb/0.01sec = 10Gbps

Uncongested vs Congested WAN Transfers

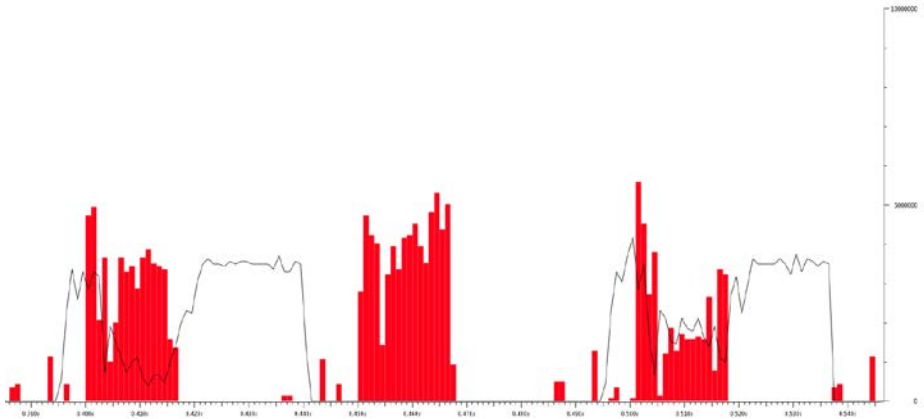
Bits per 1 millisec[2]
captured via passive tap

Uncongested



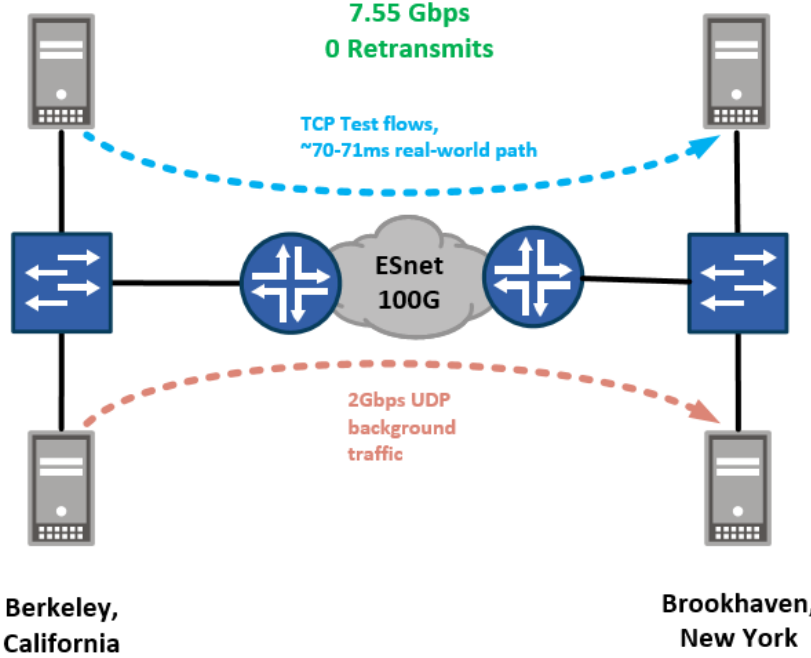
Congested[3]

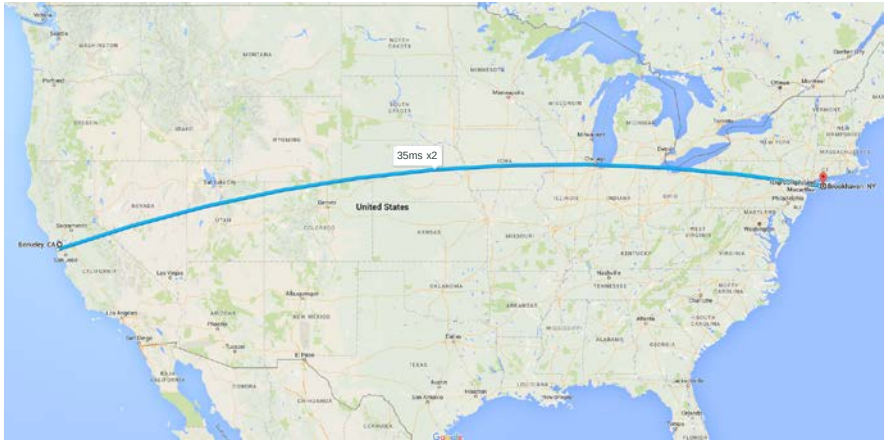
Data Transfer (TCP)
Background Traffic (UDP)



[1] 70ms simulated RTT via netem
[2] 10Mb/0.001sec = 10Gbps
[3] Adding 2Gbps UDP Traffic

Real World Testing @ 70ms RTT



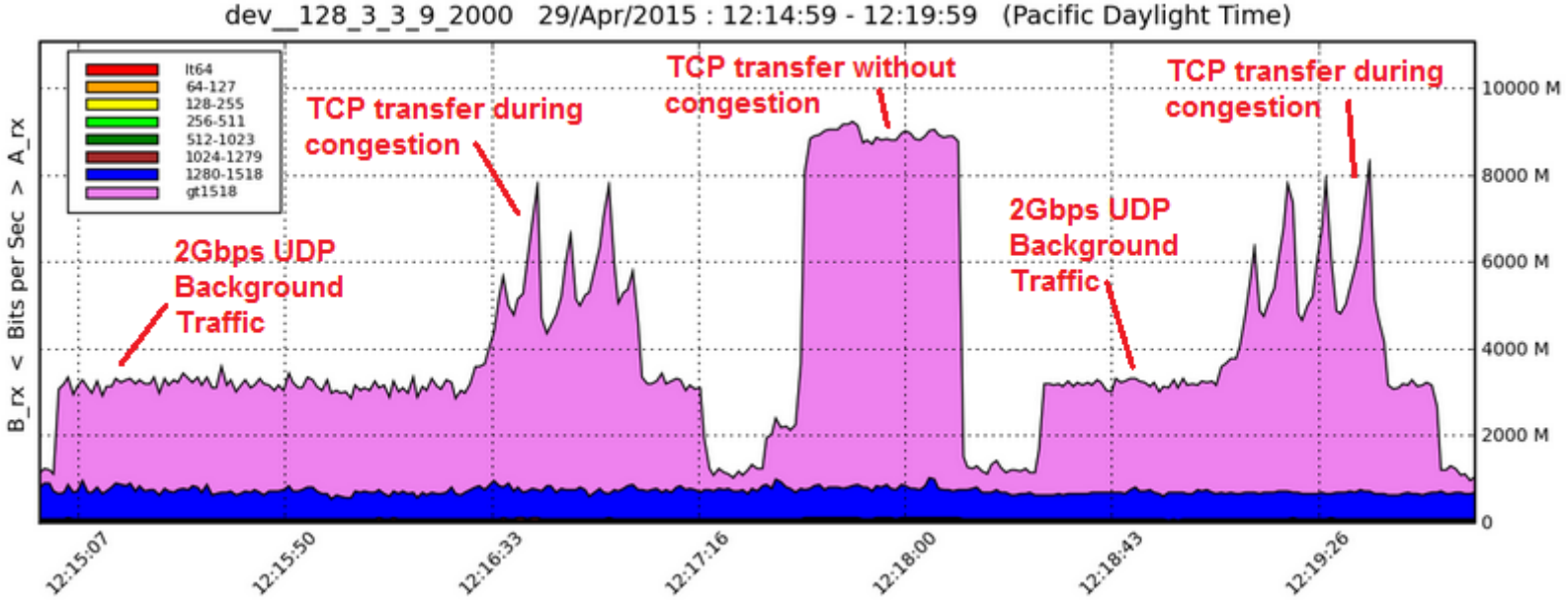


Special thanks to **Mark Lukasczyk** at Brookhaven National Laboratory for providing far-end test servers

Uncongested vs Congested WAN Transfers

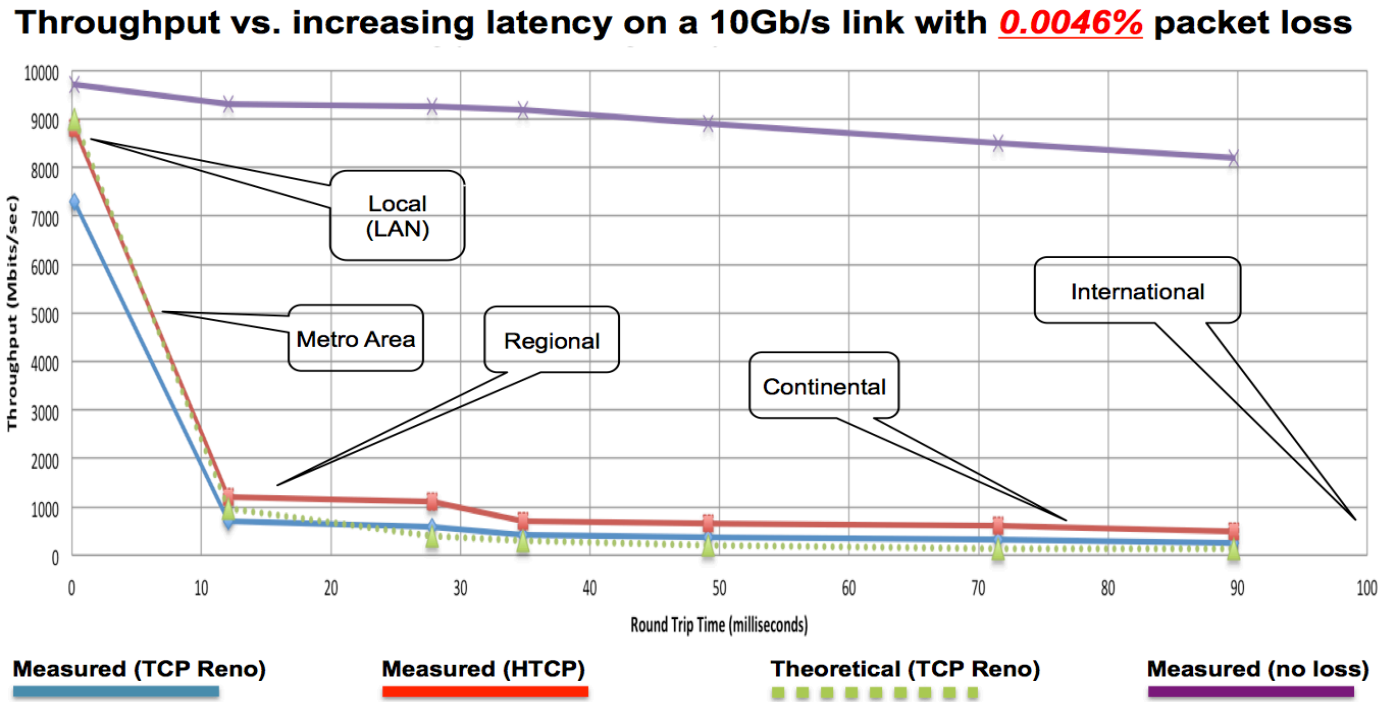
Real World tests California to New York[1]

Optical tap / cPacket @ LBNL border



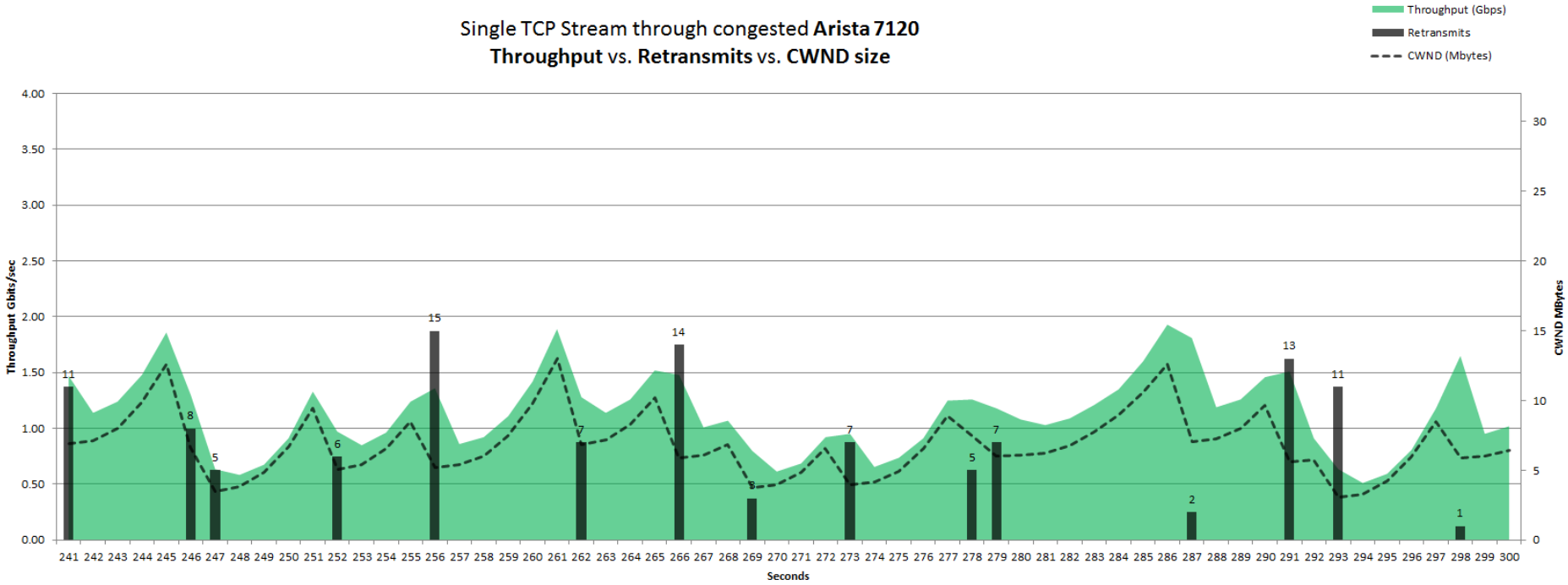
[1] ~70ms real RTT

Impact of packet loss at different distances



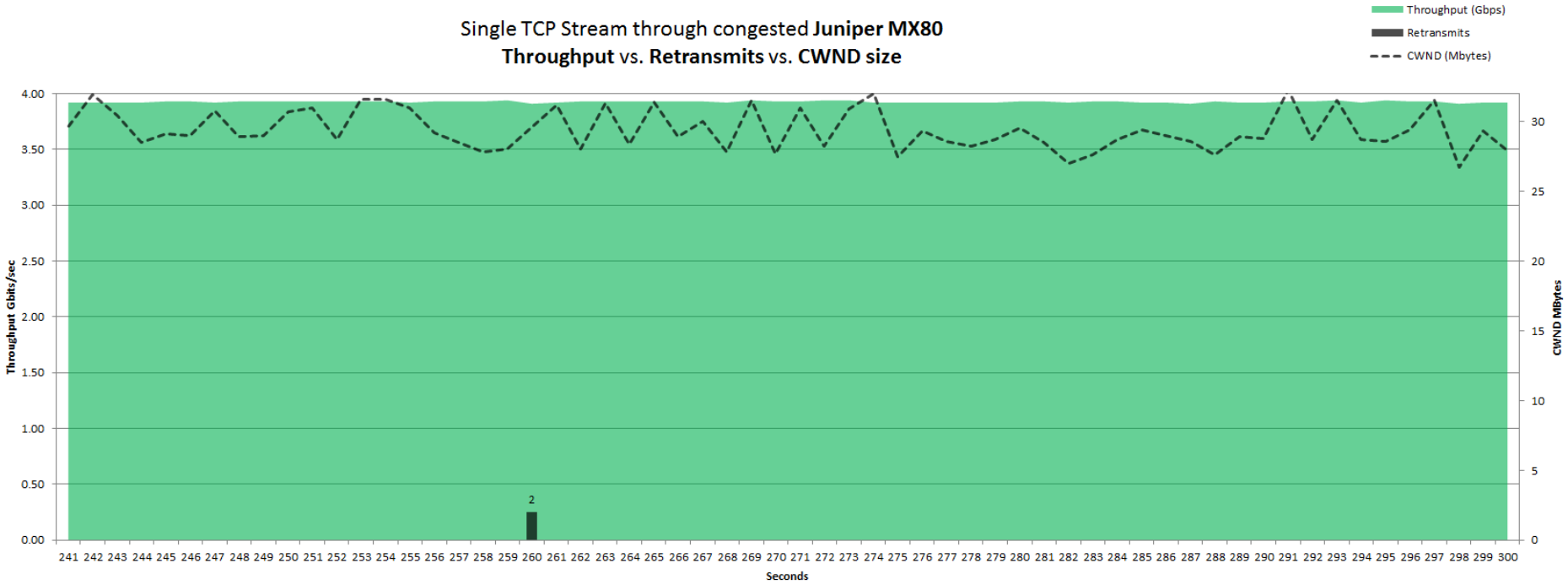
Source: http://fasterdata.es.net

TCP's Congestion Control w/ insufficient buffers



50ms simulated RTT
Congestion w/ 2Gbps UDP traffic
HTCP / Linux 2.6.32

TCP's Congestion Control w/ sufficient buffers



50ms simulated RTT
Congestion w/ 2Gbps UDP traffic
HTCP / Linux 2.6.32

Congestion
= Packet Loss
= Poor Performance
…so what do we do?

Replace (something like) this…

With this?

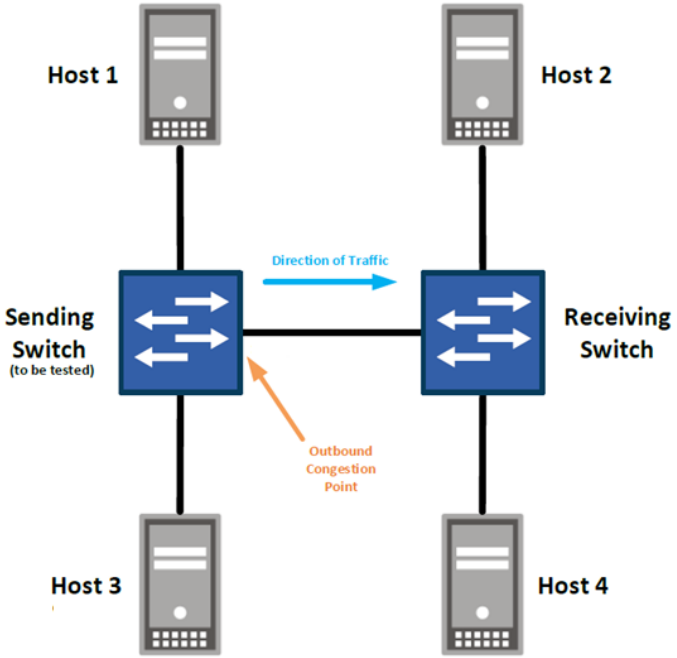
Royalty checks to:
Michael Smitasin
PO Box 919013
Berkeley, CA 94707

Source: http://www.juniper.net

That is a **hard** (… and **expensive**) pill to swallow.

(and it’s not *always* the right choice)

What if you could try before you buy?



- Easy to build test environment
- Open source (free) software
- LAN distances, WAN latencies
- Isolated, controlled (no saturating production links!)
- Quickly compare models, vendors, configs, etc.

Hardware

- 4x Servers w/ 10G NICs
  - We used Dell R320s and Intel X520s
- 1x “Receiving” switch w/ 3x 10G ports
  - Doesn’t need to be expensive, this is where data “fans out” so no congestion on this side.
- 5x SFP+ Direct-Attached Copper Cables (cheapest) OR 6x 10G Optics + 5x fiber cables (flexible options)
  - We used SR optics + 50µm multimode fiber
  - Wanted flexibility for testing models w/ X2, XENPAK, etc
- “Sending” switch(es) w/ 3x 10G ports (to test)

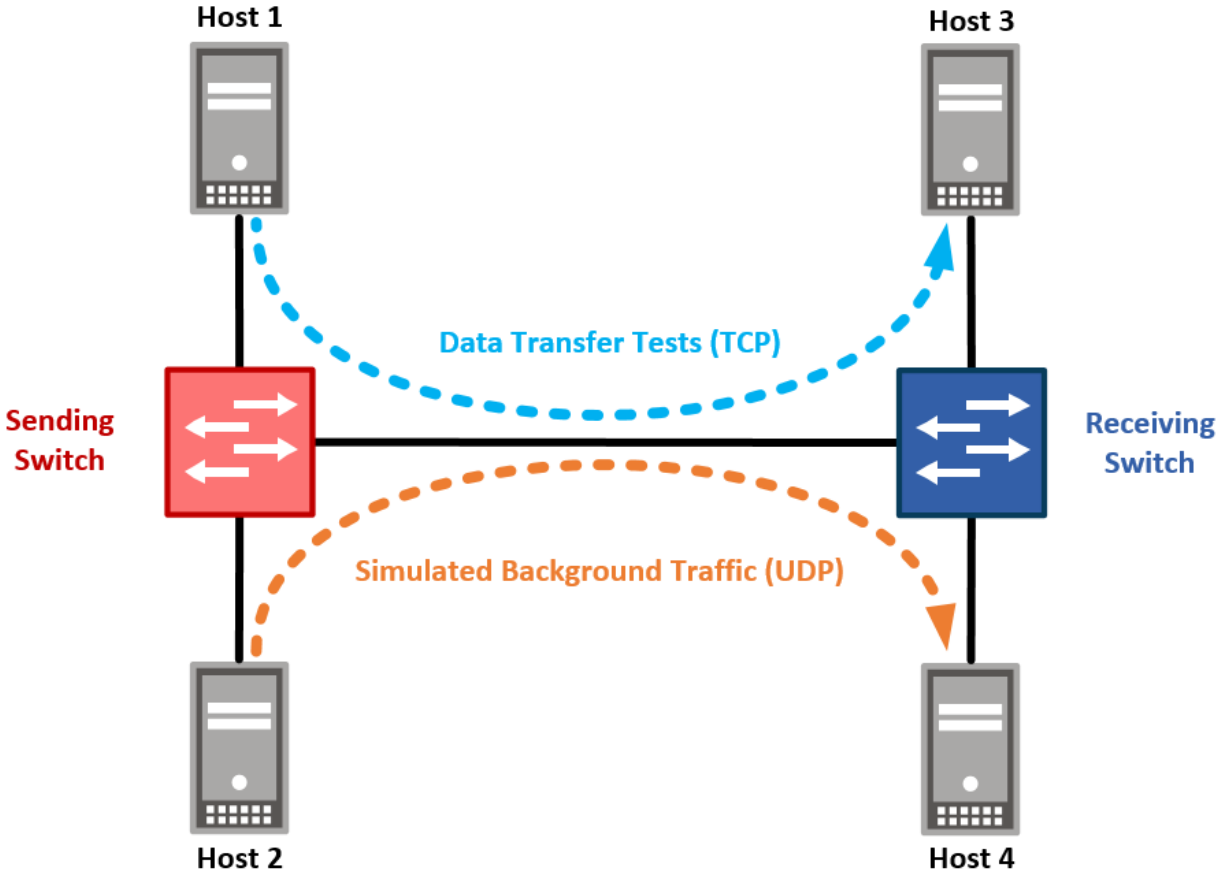
Software / Configuration

perfSONAR

- Linux (distro generally your preference)
  - We used CentOS 6 (some Fedora too)
- Install test utilities (or just use perfSonar[1])
  - import Internet2 repo
  - install iperf nuttcp bwctl-client bwctl-server
- Host and NIC tuning per FasterData[2] recommendations:
  - TCP Tuning - /etc/sysctl.conf
  - TX Queue Length
  - TX / RX Descriptors
  - Jumbo Frames

ESnet
FASTERDATA KNOWLEDGEBASE

[1] http://www.perfsonar.net
[2] http://fasterdata.es.net



Test Overview

- Add delay between Host 1 and Host 3 using netem
- Host 3 runs iperf3 server
- Host 4 runs iperf3 server
- Host 2 sends UDP traffic @ 2Gbps
- Host 1 sends TCP traffic and we measure rate
- Sending Switch's outbound interface is congested and buffering occurs

Test Commands

Add delay:

```
host1 # tc qdisc add dev ethN root netem delay 25ms
host3 # tc qdisc add dev ethN root netem delay 25ms
```

Start iperf3 servers to receive data:

```
host3 # iperf3 -s
host4 # iperf3 -s
```

Start background traffic (to cause congestion):

```
host2 # iperf3 -c host4 -u -b2G -t3000
```

Start TCP traffic to simulate data transfer:

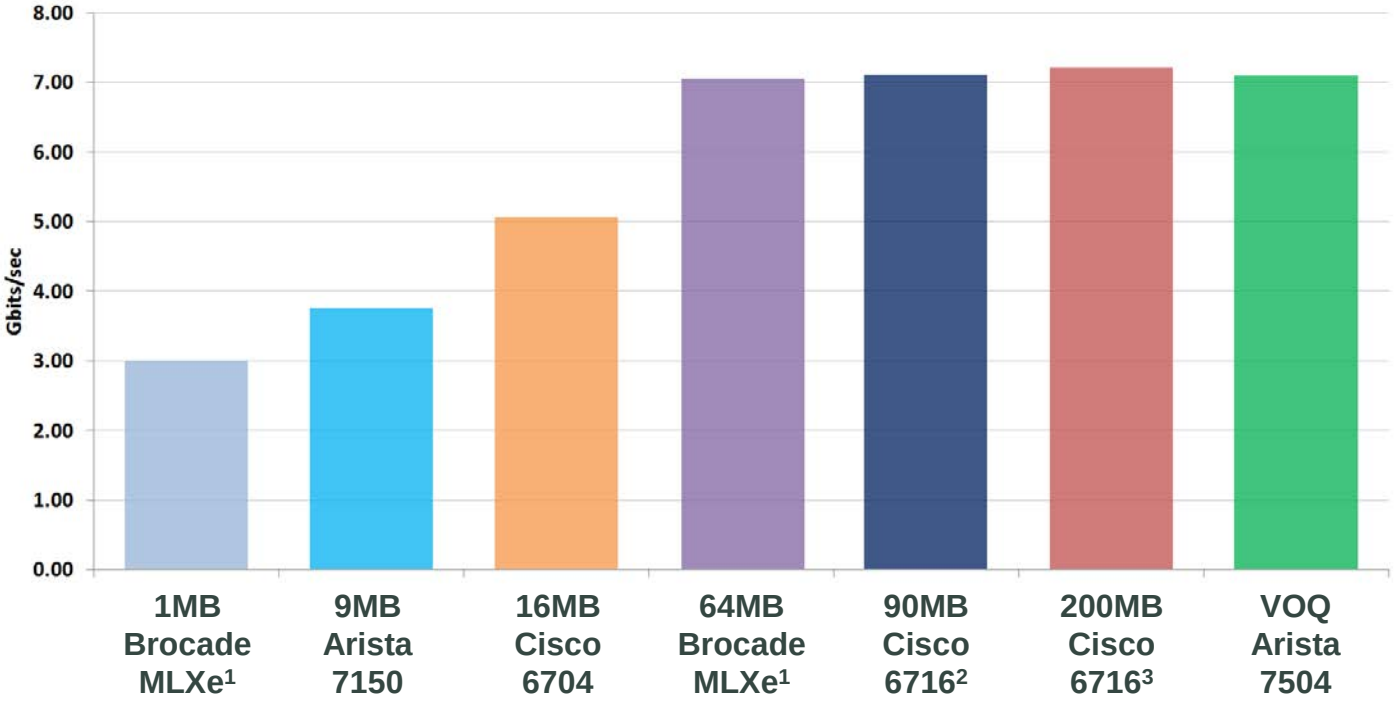
```
host1 # iperf3 -c host3 -P2 -t30 -O5
```

Test Results

[ID]	Interval	Transfer	Bandwidth	Retr	Cwnd
...					
[4]	29.00-30.00 sec	201 MB	1.69 Gbps	0	9.54 MB
[6]	29.00-30.00 sec	126 MB	1.06 Gbps	0	6.05 MB
[SUM]	29.00-30.00 sec	328 MB	2.75 Gbps	0	
- -					
[ID]	Interval	Transfer	Bandwidth	Retr	
[4]	0.00-30.00 sec	5.85 GB	1.68 Gbps	40	sender
[4]	0.00-30.00 sec	5.83 GB	1.67 Gbps		receiver
[6]	0.00-30.00 sec	4.04 GB	1.16 Gbps	39	sender
[6]	0.00-30.00 sec	4.01 GB	1.15 Gbps		receiver
[SUM]	0.00-30.00 sec	9.89 GB	**2.83 Gbps**	**79**	sender
[SUM]	0.00-30.00 sec	9.85 GB	**2.82 Gbps**		receiver

Swap out sending switch and repeat

Average TCP results, various switches



Buffers per 10G egress port

2x parallel TCP streams

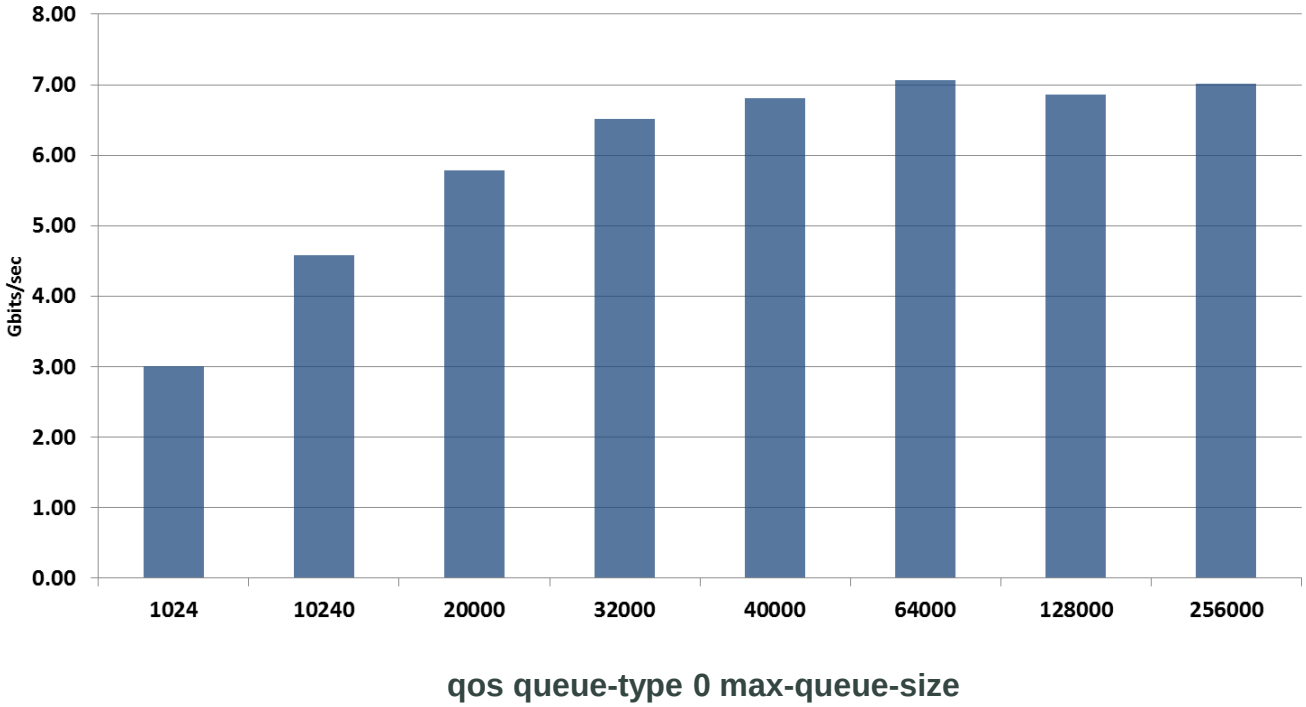
50ms simulated RTT

2Gbps UDP background traffic

15 iterations

[1] NI-MLX-10Gx8-M
[2] Over-subscription Mode
[3] Performance Mode

Tunable Buffers with a Brocade MLXe[1]



Buffers per 10G egress port

2x parallel TCP streams

50ms simulated RTT

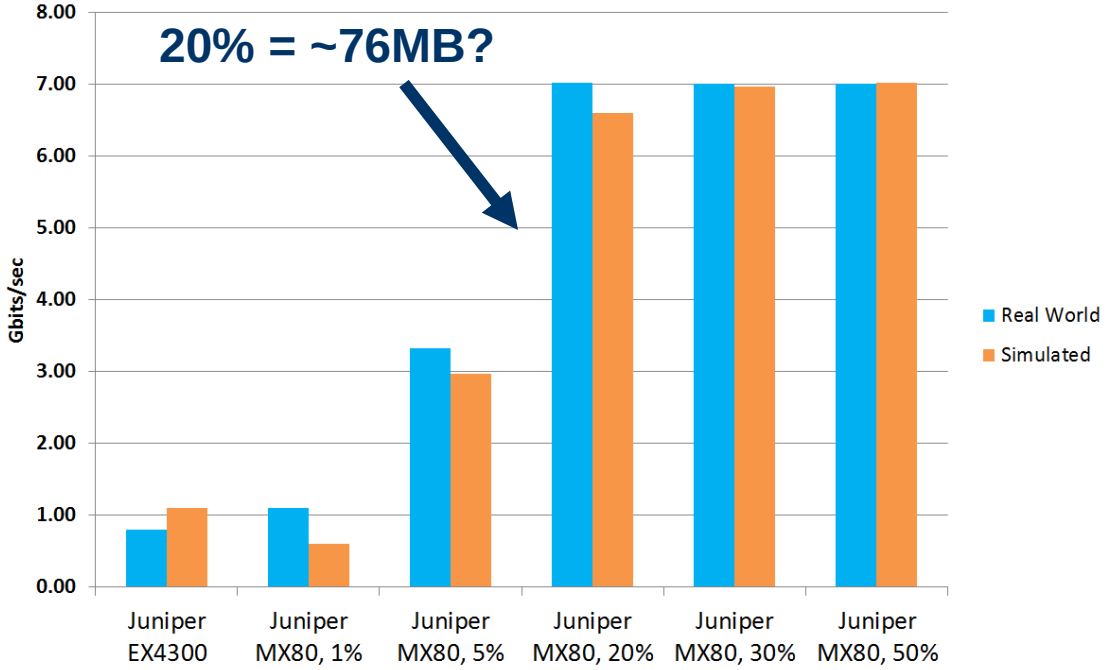
2Gbps UDP background traffic

15 iterations

[1] NI-MLX-10Gx8-M

Is netem accurate?

Real World RTT vs Simulated RTT



70ms RTT

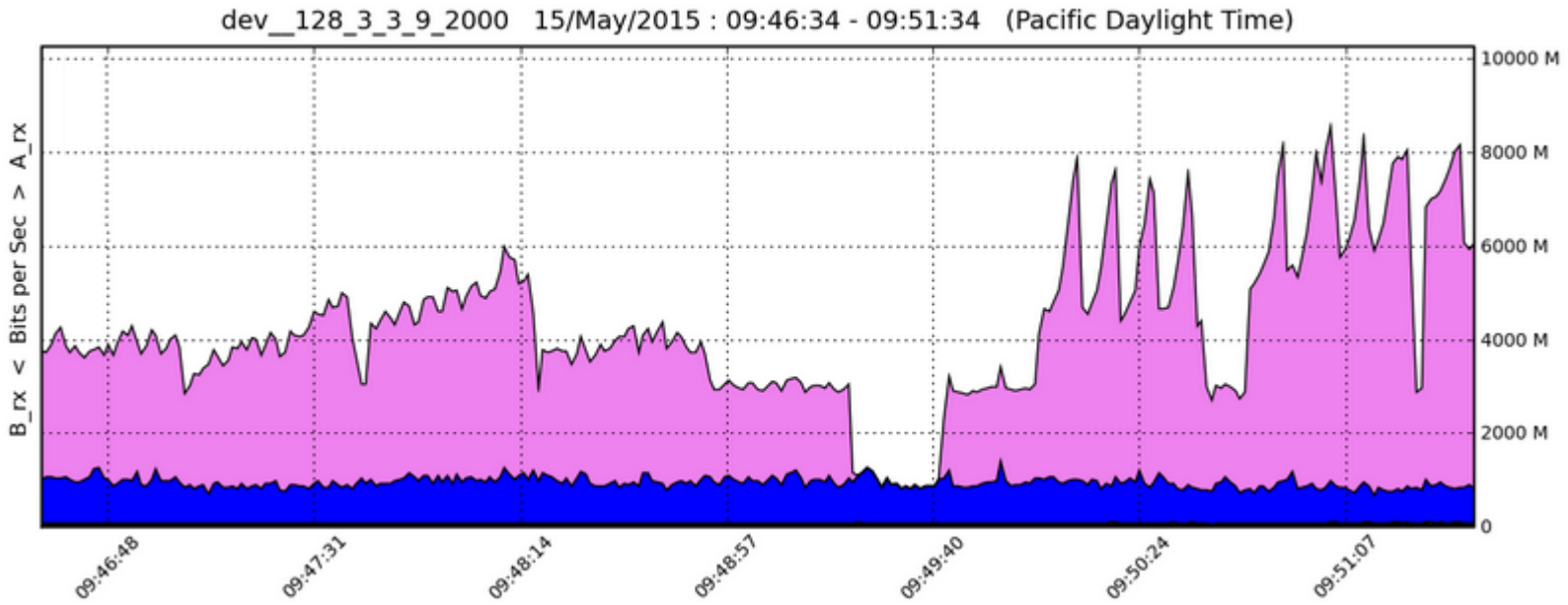
2x parallel TCP streams

2Gbps UDP background traffic

Juniper MX80 config:
class-of-service scheduler
buffer-size percent

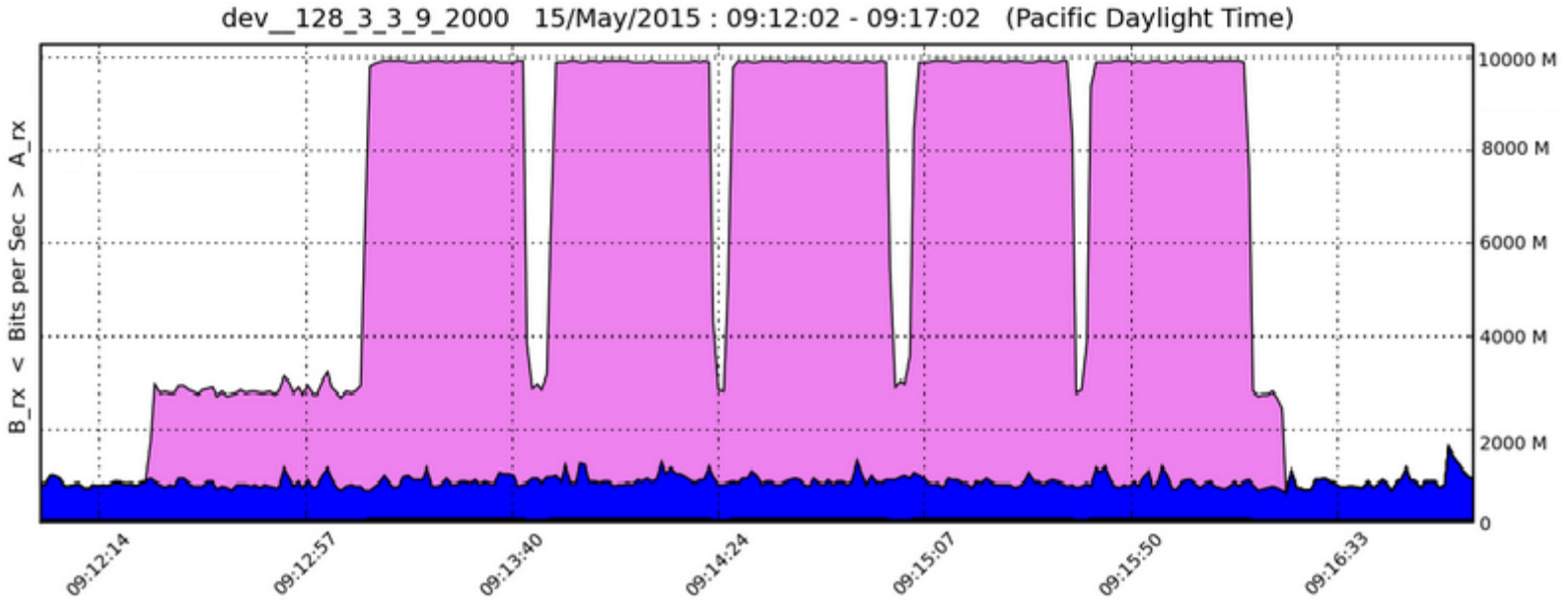
Special thanks to **Mark Lukasczyk** at Brookhaven National Laboratory for providing far-end test servers

Tap from LBNL border – CA to NY



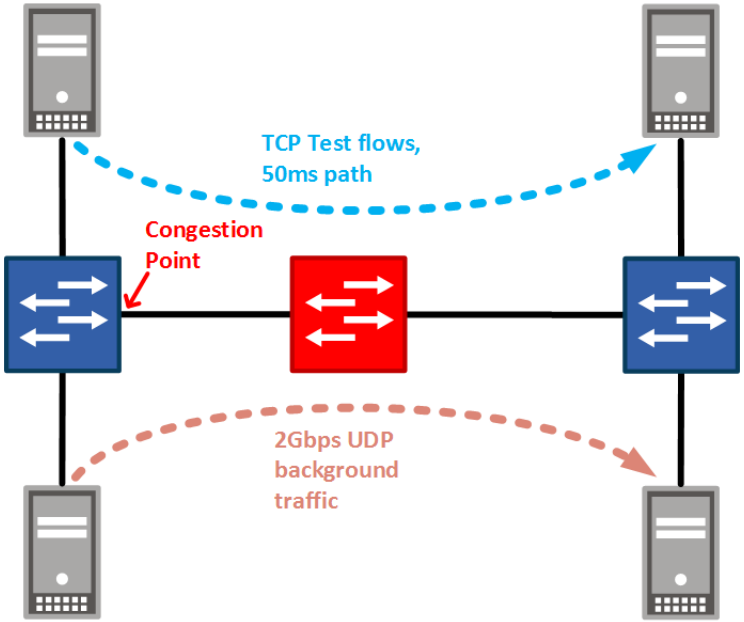
MX80 w/ 1% buffer **MX80 w/ 5% buffer**

Tap from LBNL border – CA to NY

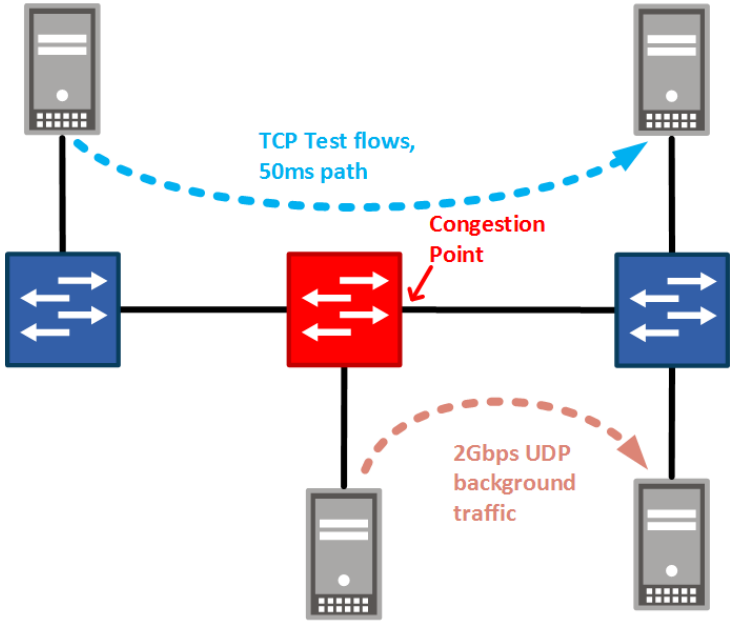


MX80 w/ 50% buffer

What if there's a small buffered switch upstream?

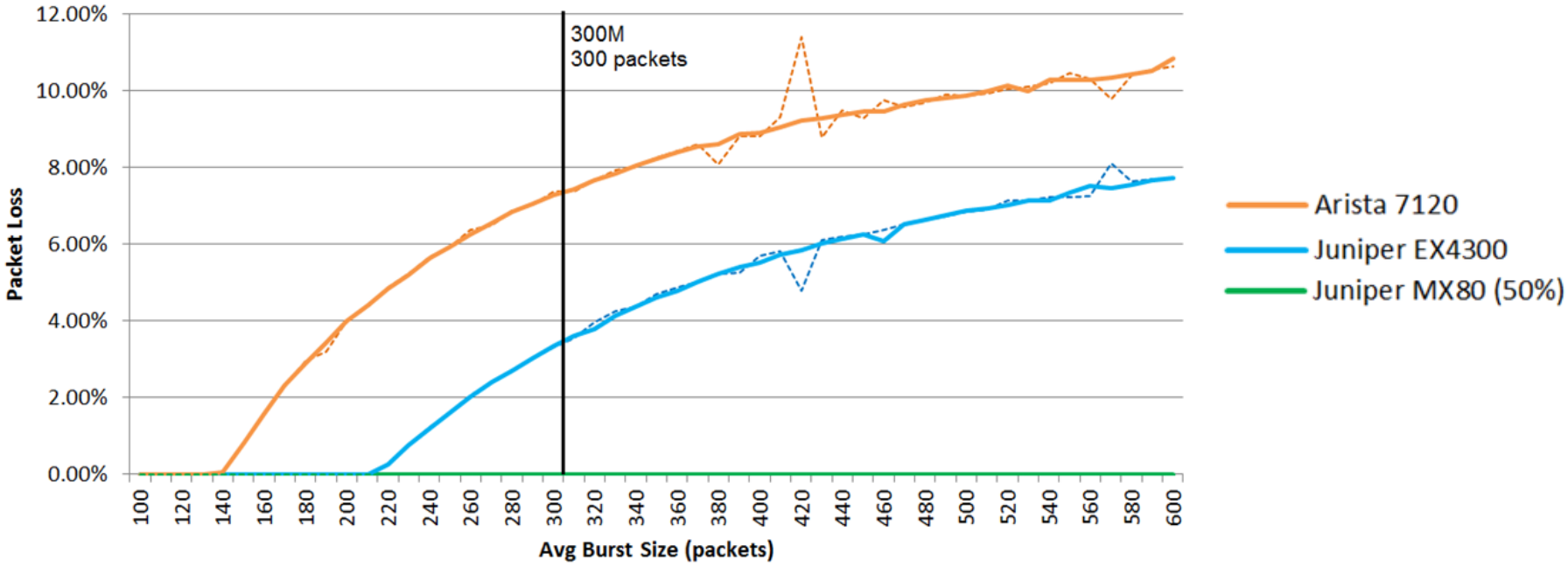


No Congestion on it? No problem



Congestion on it? Many problem

Alternate test - nuttcp



Alternate test – nuttcp commands

- Add delay:

```
host1# tc qdisc add dev eth1 root netem delay 25ms
host2# tc qdisc add dev eth1 root netem delay 25ms
```

- Start 2Gbps UDP flow to add congestion:

```
host4# iperf3 –s
host3# iperf3 -c host4 -u -b2G -t3000
```

- nuttcp basic test parameters[1]:

```
host2# nuttcp -S
host1# nuttcp -l8972 -T30 -u -w4m –Ri300m/X –i1 host2
```

[1] https://fasterdata.es.net/performance-testing/network-troubleshooting-tools/nuttcp/

nuttcp conclusion

This will probably have no packet loss on smaller buffer switches:

```
nuttcp -l8972 -T30 -u -w4m -Ri300m/65 -i1
```

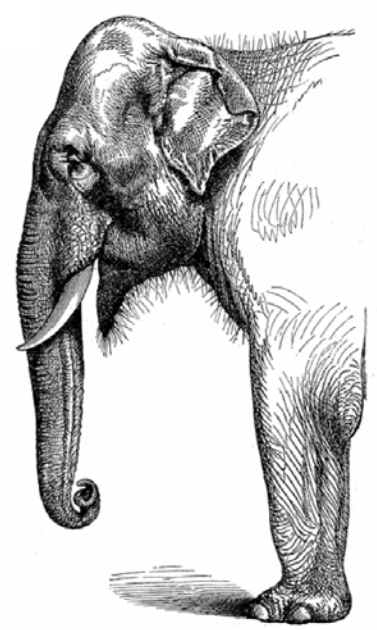
While this will probably have some:

```
nuttcp -l8972 -T30 -u -w4m -Ri300m/300 -i1
```

BUT only applies to where there is congestion. “Small” buffer switch that isn’t congested won’t be detectable with this method.

Then “Big” Buffers = good?

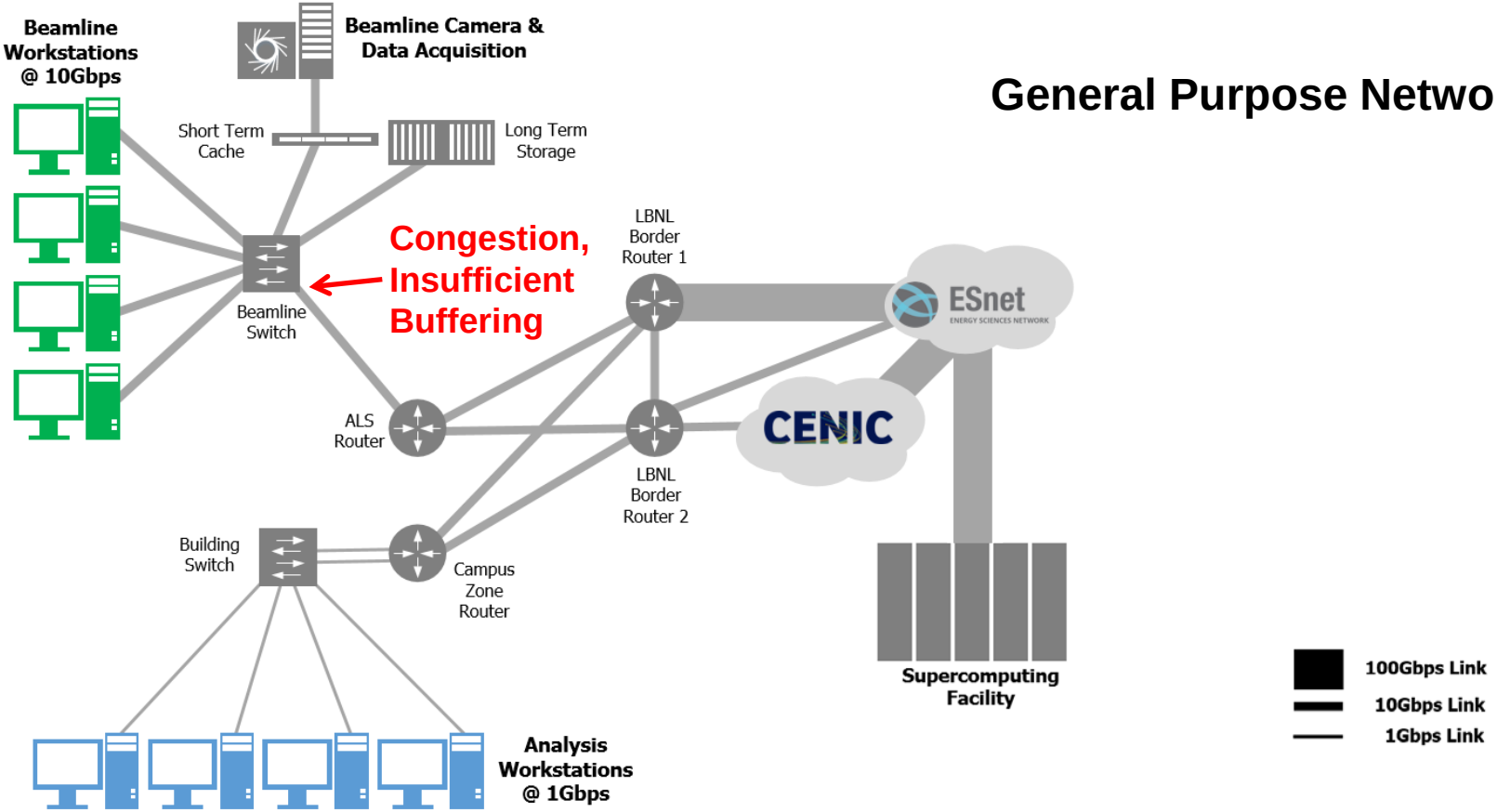
- Only in the context of these **Elephant flows**
  - Very large data transfers **(Terabytes, Petabytes)**
  - Large pipes **(10 Gbps & up)**
  - Long distances **(50ms+)**
  - Between **small numbers of hosts**



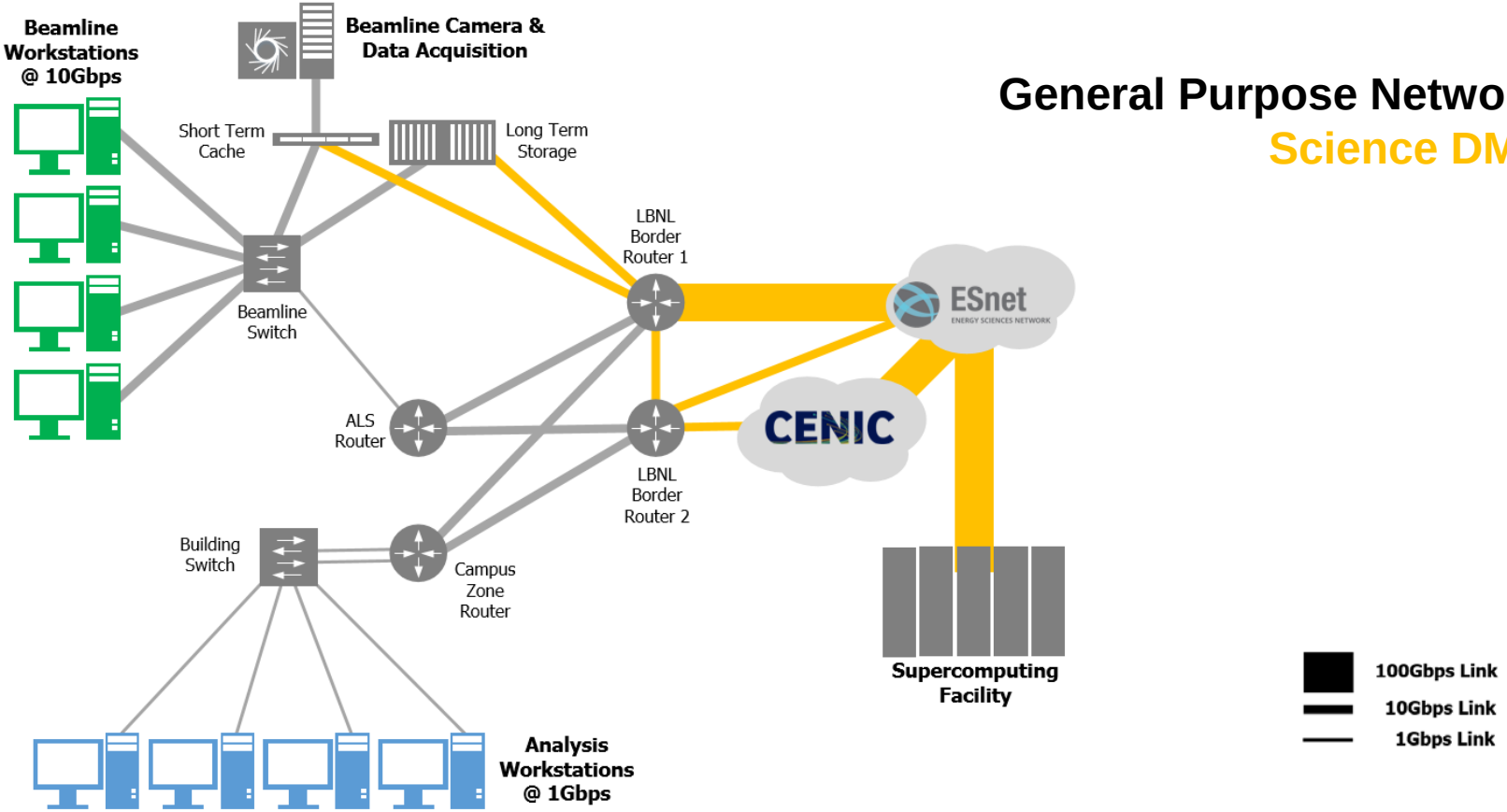
- By “big” we’re talking **MBs** per 10G port, not **GBs**.

- Important to have enough buffers to ride out **micro-bursts**. May need to drop 1 or 2 packets to fit available capacity, but to maintain performance we need to keep TCP from getting stuck in loss recovery mode.

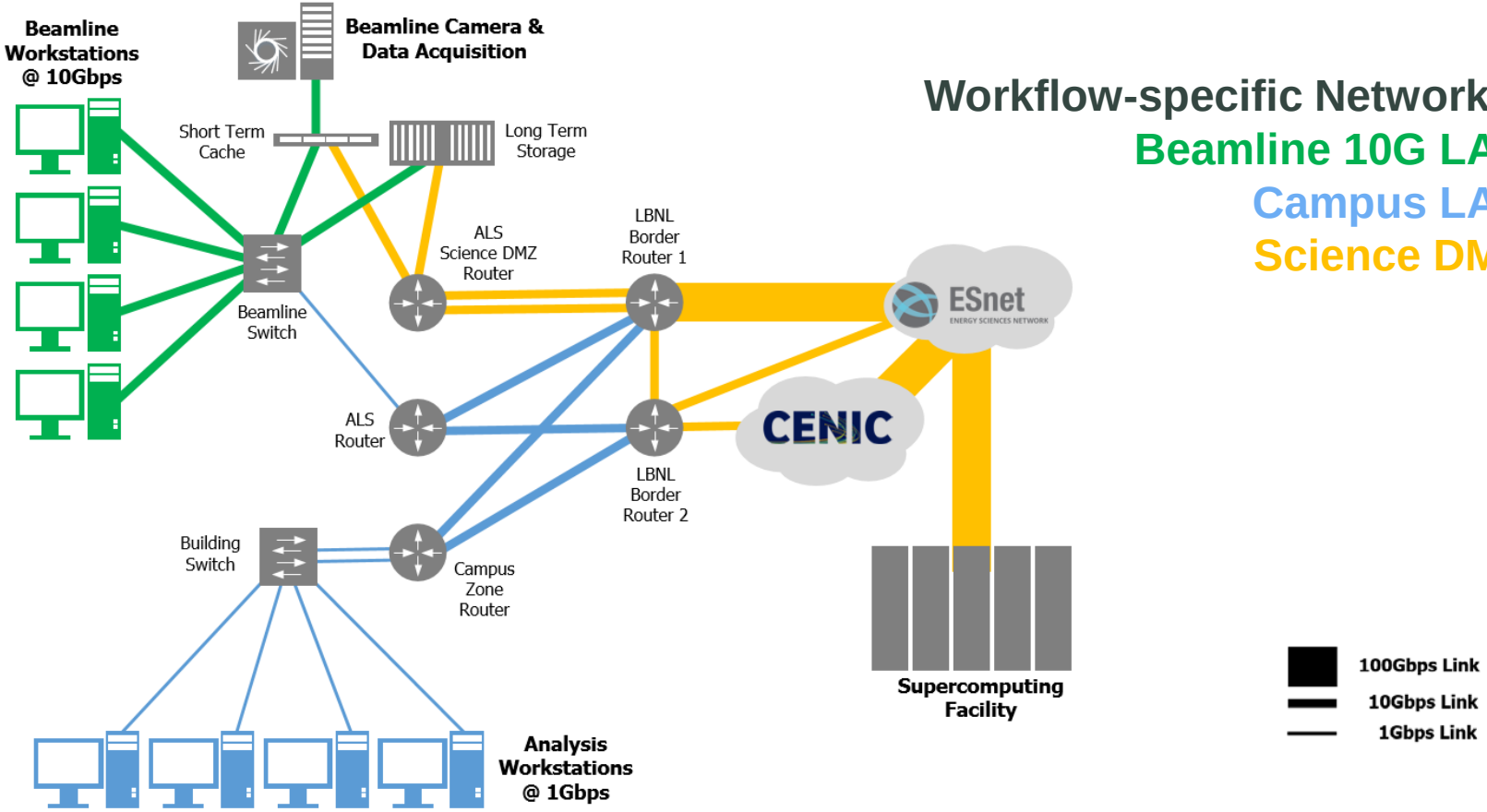
General Purpose Network



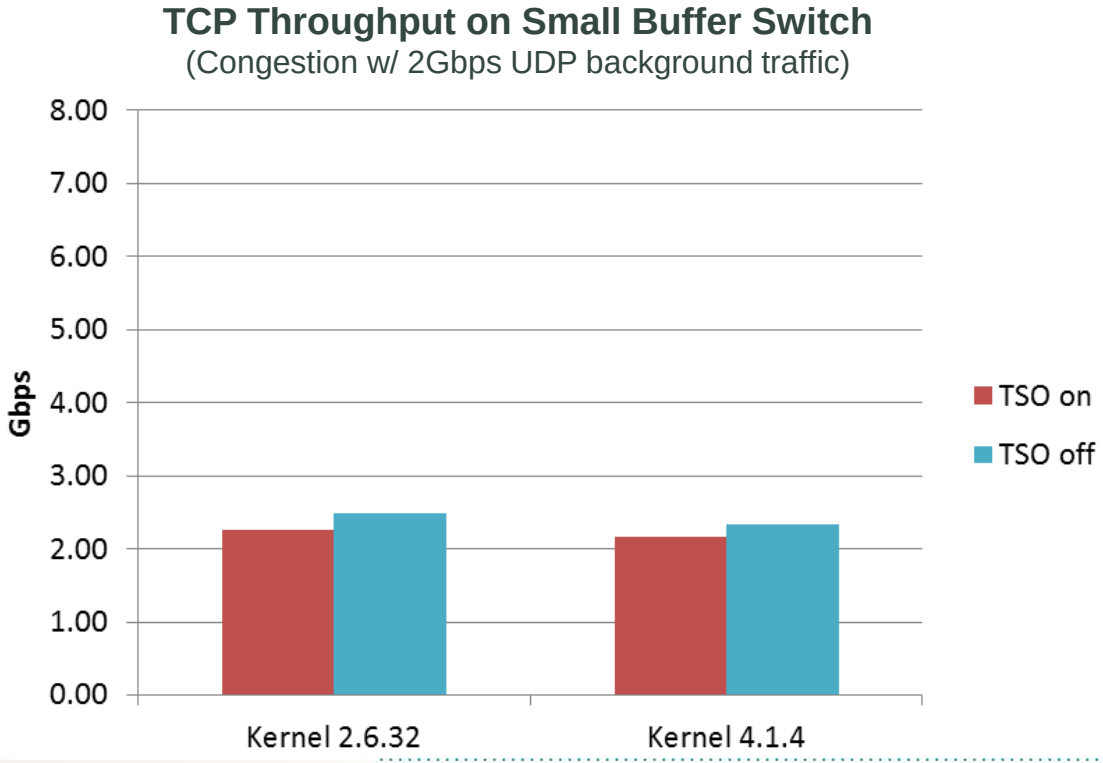
General Purpose Network
Science DMZ



Workflow-specific Networks:
Beamline 10G LAN
Campus LAN
Science DMZ

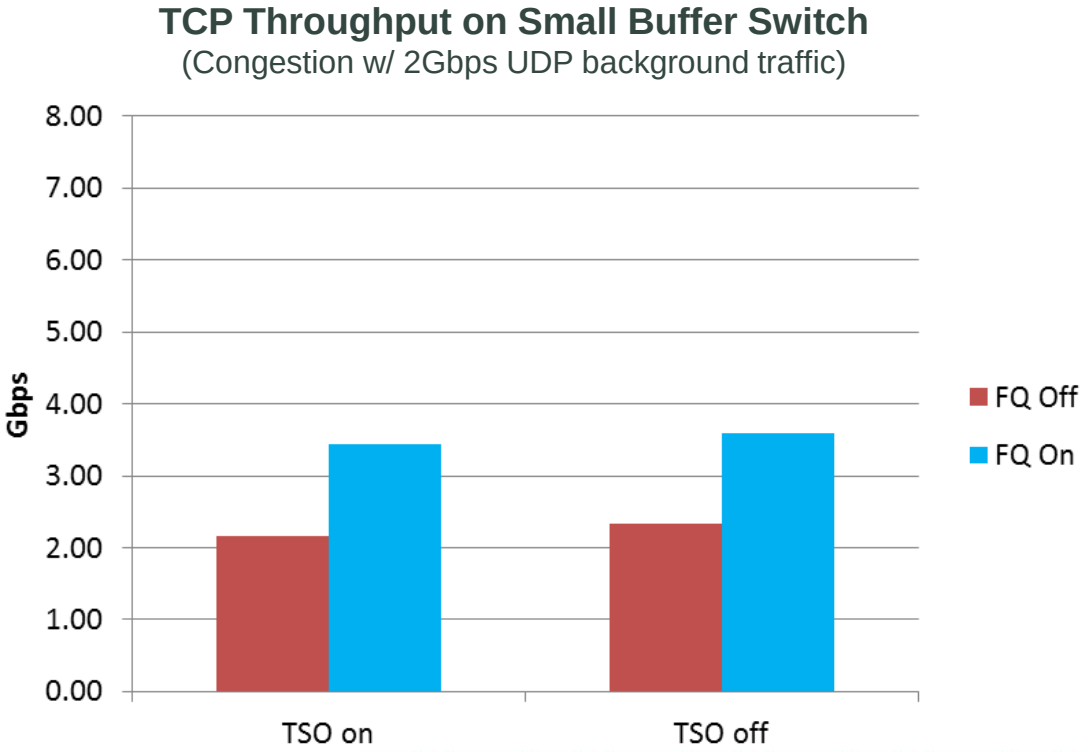


Effects of TCP Segmentation Offload



- `ethtool -k EthN`
  - Show current offload settings
- `ethtool -K EthN tso off`
  - Set TSO off
- TSO Off was negligible on our hosts with Intel X520 @ 10Gbps (~100-200Mbps)
- BUT… may be NIC specific, or more applicable @ 40Gbps+

Fair Queuing and Pacing in Kernel 4.1.4



- Requires newer kernel version
  - not available in 2.6.32
- `tc qdisc add dev EthN root fq`
  - Enable Fair Queuing
- Pacing side effect of Fair Queuing yields ~1.25Gbps increase in throughput @ 10Gbps on our hosts
- TSO differences still negligible on our hosts w/ Intel X520

Additional Information

- **A History of Buffer Sizing**
  http://people.ucsc.edu/~warner/Bufs/buffer-requirements
- **Jim Warner's Packet Buffer Page**
  http://people.ucsc.edu/~warner/buffer.html
- **Faster Data @ ESnet**
  http://fasterdata.es.net
- **Michael Smitasin** – mnsmitasin@lbl.gov
- **Brian Tierney** – bltierney@es.net

EVALUATING NETWORK BUFFER SIZE REQUIREMENTS

for Very Large Data Transfers

Michael Smitasin
Lawrence Berkeley National Laboratory (LBNL)

Brian Tierney
Energy Sciences Network (ESnet)