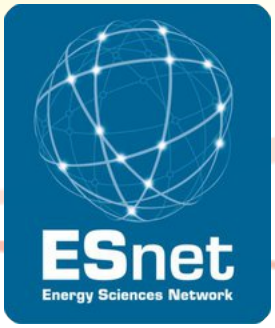


July 22nd 2013, XSEDE Network Performance Tutorial

Jason Zurawski – Internet2/ESnet

Kathy Benninger - Pittsburgh Supercomputing Center

Performance Primer

Agenda

- Expectations and Realities
- TCP Basics
- What We Can Determine
- The Tools

Expectations and Realties

- Setting expectations is an important part of managing relationships with users
  - As we saw from the previous motivating example, users may have high expectations
  - Operators ***may*** try to lower them
  - The correct answer is somewhere in between – it is perfectly possible to get ***near*** line rate speeds on clean networks
    - Congestion is a reality that may slow things down, architectural and traffic management solutions can help
    - Non-congestive packet loss can be completely eliminated

J. Zurawski – zurawski@es.net & K. Benninger - benninger@psc.edu

Reality Check: Capabilities

- The following show current data movement application capabilities*:
- scp**
  - 140 Mbps (17.5 MB/s)
- HPN patched scp, 1 disk
  - 760 Mbps (95 MB/s)
- HPN patched scp, RAID disk
  - 1.2 Gbps (150 MB/s)
- GridFTP, 1 stream, 1 disk
  - 760 Mbps (95 MB/s)
- GridFTP, 1 stream, RAID disk
  - 1.4 Gbps (175 MB/s)
- GridFTP, 4 streams, RAID disk
  - 5.4 Gbps (675 MB/s)
- GridFTP, 8 streams, RAID disk
  - 6.6 Gbps (825 MB/s)

****Assumes ~50ms of Latency between hosts.***

*****Note that rsync, another typical application, uses SSH under the hood***

J. Zurawski – zurawski@es.net & K. Benninger - benninger@psc.edu

Reality Check: Expectations

- The following shows how long it (should*) take to transfer 1 Terabyte of data across various speed networks**:

- 10 Mbps network :
  - 300 hrs (12.5 days)
- 100 Mbps network :
  - 30 hrs
- 1 Gbps network :
  - 3 hrs
- 10 Gbps network :
  - 20 minutes

****Can your network do this?***

*****Assumes running at 100% efficiency, no performance problems***

Agenda

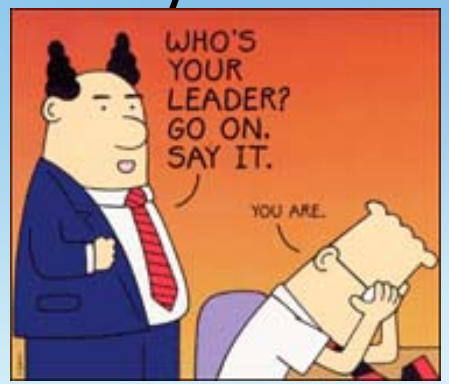
- Expectations and Realities
- TCP Basics
- What We Can Determine
- The Tools

TCP

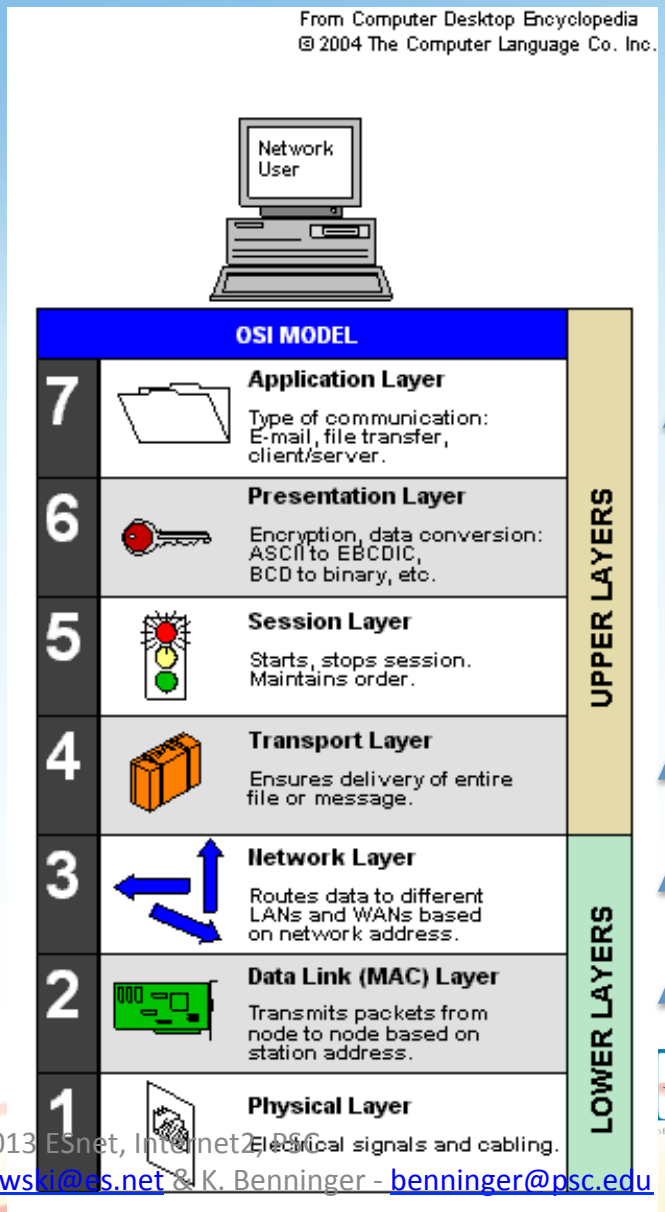
- Transmission Control Protocol
  - One of the core protocols of the Internet Protocol Suite (along with IP [Internet Protocol])
  - This is a *Transport Layer* (Layer 4) protocol, IP is a *Network Layer* (Layer 3) protocol.
- HTTP, SSH are built on top of this
- Constructed by V. Cerf and B. Khan
- Reliable delivery, fairness to all.
- What's not to love?

The OSI Protocols

Layer 8



Dealing with Funding Bodies



HTTP, SSH, etc.

TCP, UDP

IPv4, IPv6, ICMP

ARP, OSPF, Ethernet

TCP

- There is actually a lot not to love:
  - TCP doesn't relay when things are going wrong via the OS Kernel (e.g. a lost packet is re-transmitted without any knowledge to the application).
  - Loss is actually "***required***" for TCP to work, this is how it is able to enforce fairness (e.g. Loss means congestion [as well as physical flaws], therefore back off).
  - No distinction between congestive and non-congestive losses
  - Not optimized for modern networks (LFN) by default. Latency has a pretty profound effect on performance …
  - Pretty easy to use, but also pretty easy to use poorly from an application point of view

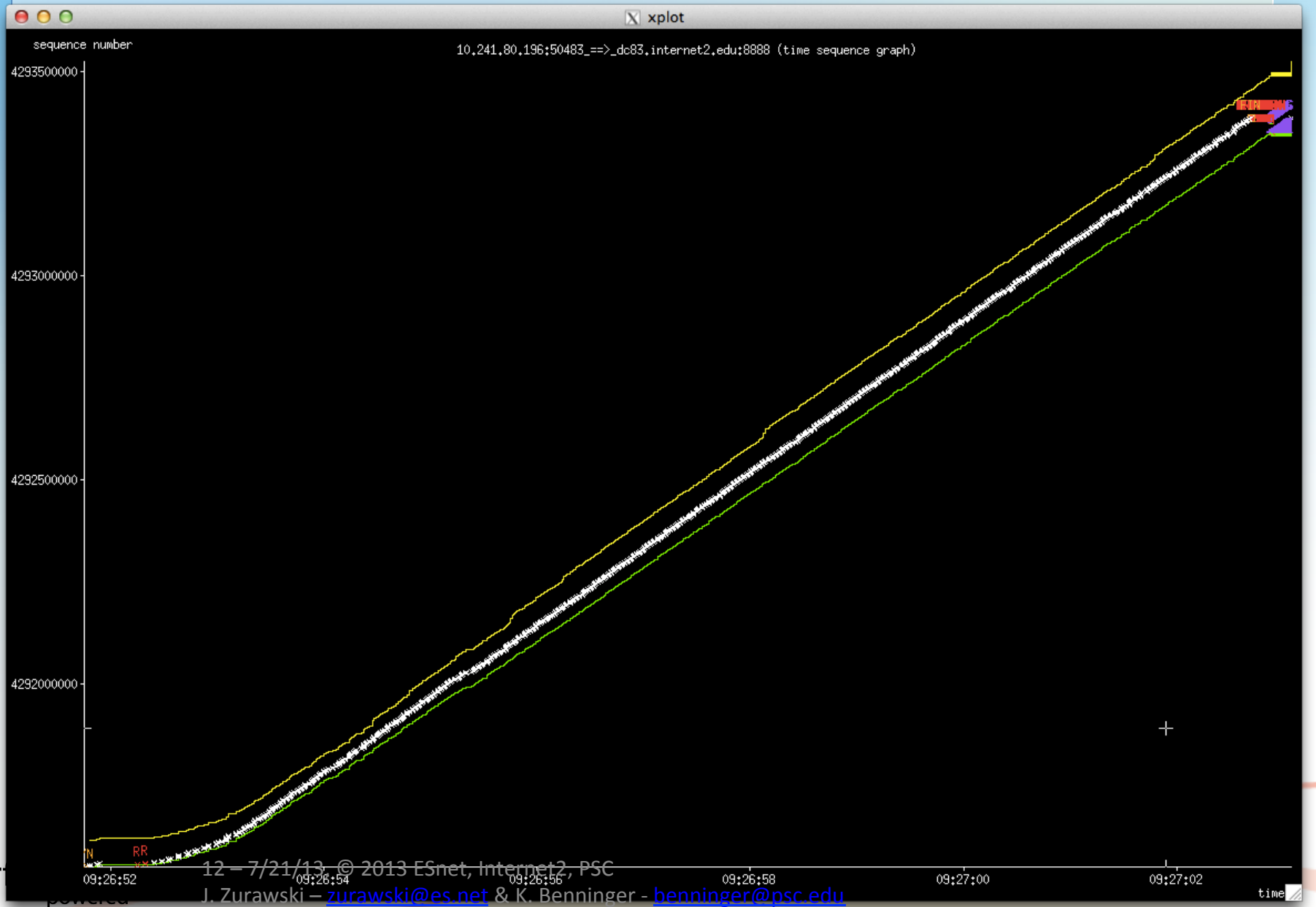
TCP

- Lets introduce some terms, there won't be a quiz.
  - Think about a typical application (a web browser, or moving a file with FTP/SCP) as we discuss this.
  - Imagine all of the work that goes into just moving small amounts of data
- TCP Measurements (from some of the tools we use):
  - Always includes the end system
  - Are sometimes called "memory-to-memory" tests since they don't involve a spinning disk
  - Set expectations for well coded application

TCP

- There are limits of what we can measure
  - TCP ***hides*** details
  - In hiding the details it can obscure what is causing errors
- Many things can limit TCP throughput
  - Loss
  - Congestion
  - Buffer Starvation
  - Out of order delivery
- 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)

TCP – Performance Over Time



TCP – Performance Over Time

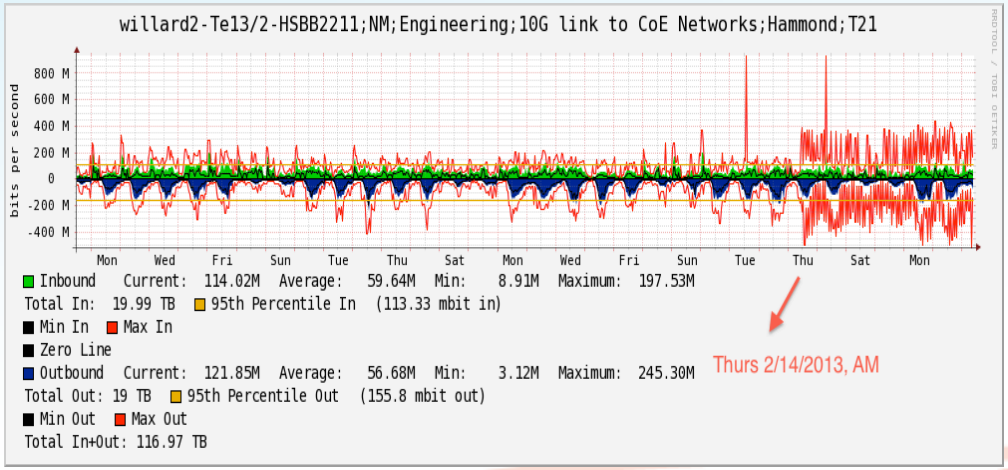
- Graph is illustrating:
  - Flow of ACKs and DATA is linear if there are no problems
  - Window size will grow over time (autotuning)
  - A missing ACK will either go unnoticed (cumulative ACKs) or stall the window if there is a long enough delay
  - A missing data packet will need to be resent
  - Timers as control when we think there has been a misfire between client/server

TCP – Quick Overview

- Data Packet
  - Contains some header overhead, and the broken up chunk of user data
- ACK Packet
  - Acknowledge the receipt of a data packet, "cumulative" in nature
- The "Window"
  - Agreed upon range of data that can be 'in flight' at a point in time. The window can't advance till ACKs are received for the oldest members.
- Timers
  - Placed upon data packets. If we expire a timer, we believe data is lost. We will try to resend specific packets first, or the entire window if need be

TCP – A Word on Window Size

- The TCP header uses a 16 bit field to report the receive window size to the sender. Therefore, the largest window that can be used is 2**16 = ***64K*** bytes.
  - To circumvent this problem, Section 2 of this memo defines a new TCP option, "Window Scale", to allow windows larger than 2**16. This option defines an implicit scale factor, which is used to multiply the window size value found in a TCP header to obtain the true window size. (From RFC 1323, page 2, May 1992)
  - Why do we see this being ignored by security devices in 2013?



J. Zurawski – zurawski@es.net & K. Benninger - benninger@psc.edu

TCP – Quick Overview

- SACK Packet
  - Selective acknowledgement for a specific missing segment
- MTU
  - Maximum Transmission Unit (maximum packet size on a given network segment)
  - 1500 or 9000 bytes (XSEDE uses 9000)
- MSS
  - Maximum Segment Size of TCP packet data payload (MTU = MSS + packet headers)
- Slow Start
  - The name is a bit of a misnomer …
  - Avoid sending more data than the network is capable of consuming. Goal is to reach a loss (establishes window size by relying on acks)
  - Actually really aggressive …

TCP – Quick Overview

- Congestion Control
  - Process of self regulating flow speed due to loss in the network (e.g. making it fair)
- Congestion Avoidance
  - Additive-increase/Multiplicative-decrease [AIMD] scheme to find a fair speed for a TCP flow by adjusting the sending window. Starts low (2 x MSS) and increase
- Fast Retransmit
  - Retransmit the window after receiving 3 duplicate ACKs for the prior numbered segment.
  - Usually indicates that the data is being lost, or delayed significantly

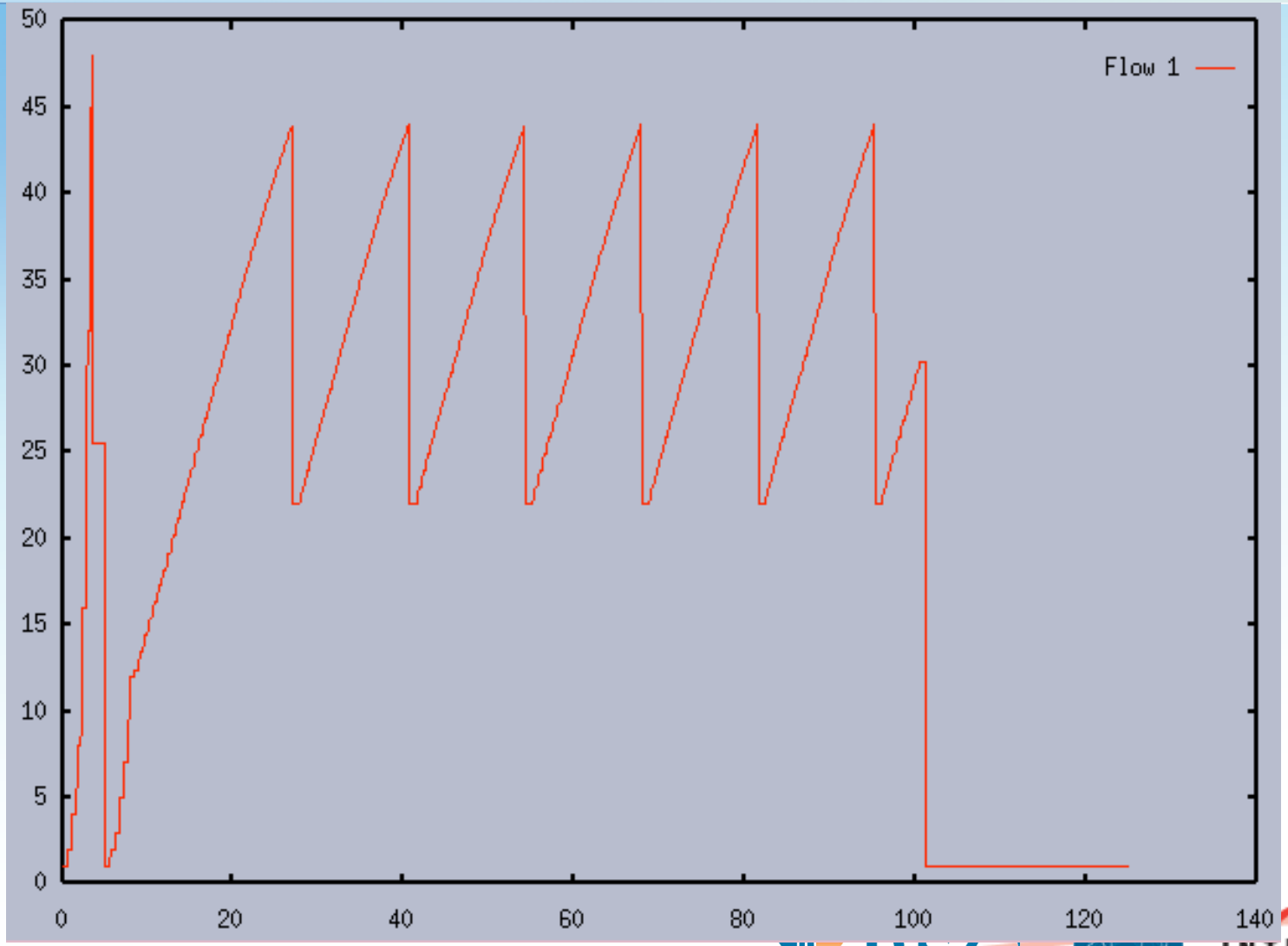
TCP – Quick Overview

- Congestion Control Algorithms (selectable in the Linux Kernel, less options in things like Windows)
  - RENO (Slow Start, Cong. Avoidance, Fast Retransmit, Fast Recovery)
  - Cubic (Optimized for LFN [Long Fat Networks] with large latency, Cubic growth pattern)
  - HTCP (still additive-increase/multiplicative-decrease [AIMD], more agressive as loss decreases on high BDP paths)

TCP – Now What?

- General Operational Pattern
  - SYN/SYN-ACK Packets sent to initiate the stream.
  - Window size starts at a fixed value (around 64k) and may be negotiated to increase (if the hosts support this)
  - Sender application buffers up data and passes to Kernel. Kernel processes this, and plans to send into segments (respect the MSS) and numbers each
  - Packets are sent in order from the window. Window advances as we get ACK packets signifying the 'lower' part of the window has been received.
  - The flow of data and ACK packets will dictate the overall speed of TCP for the length of the transfer

TCP – Overview (Typical Sawtooth)



TCP – Quick Overview

- *The Macroscopic Behavior of the TCP Congestion Avoidance Algorithm,* Mathis, et al. still has relevance.
- TCP throughput bandwidth (BW) in congestion avoidance:

$$BW \propto \frac{MSS}{RTT * \sqrt{p}}$$

Where:

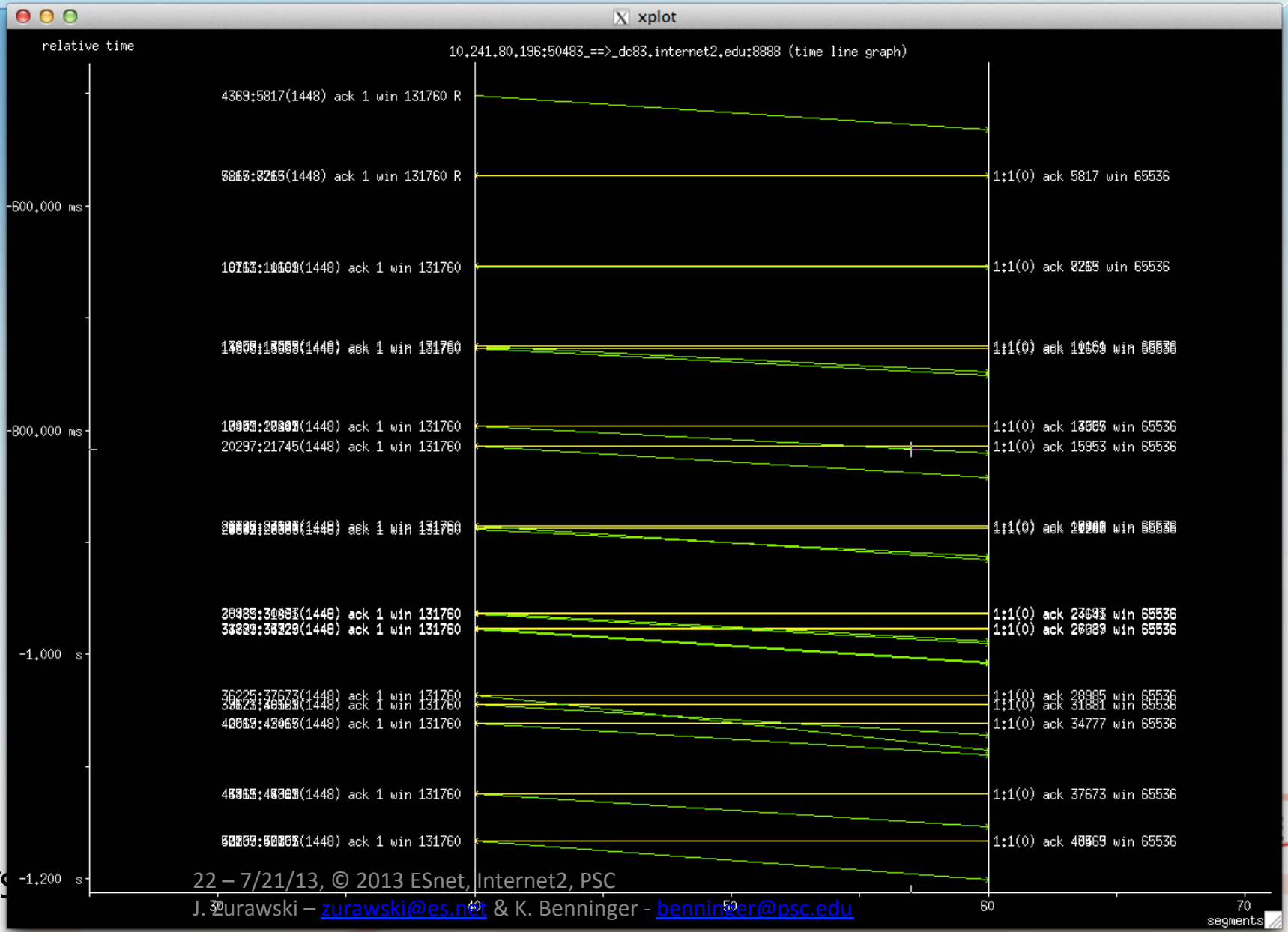
MSS = Maximum Segment Size

RTT = Round Trip Time

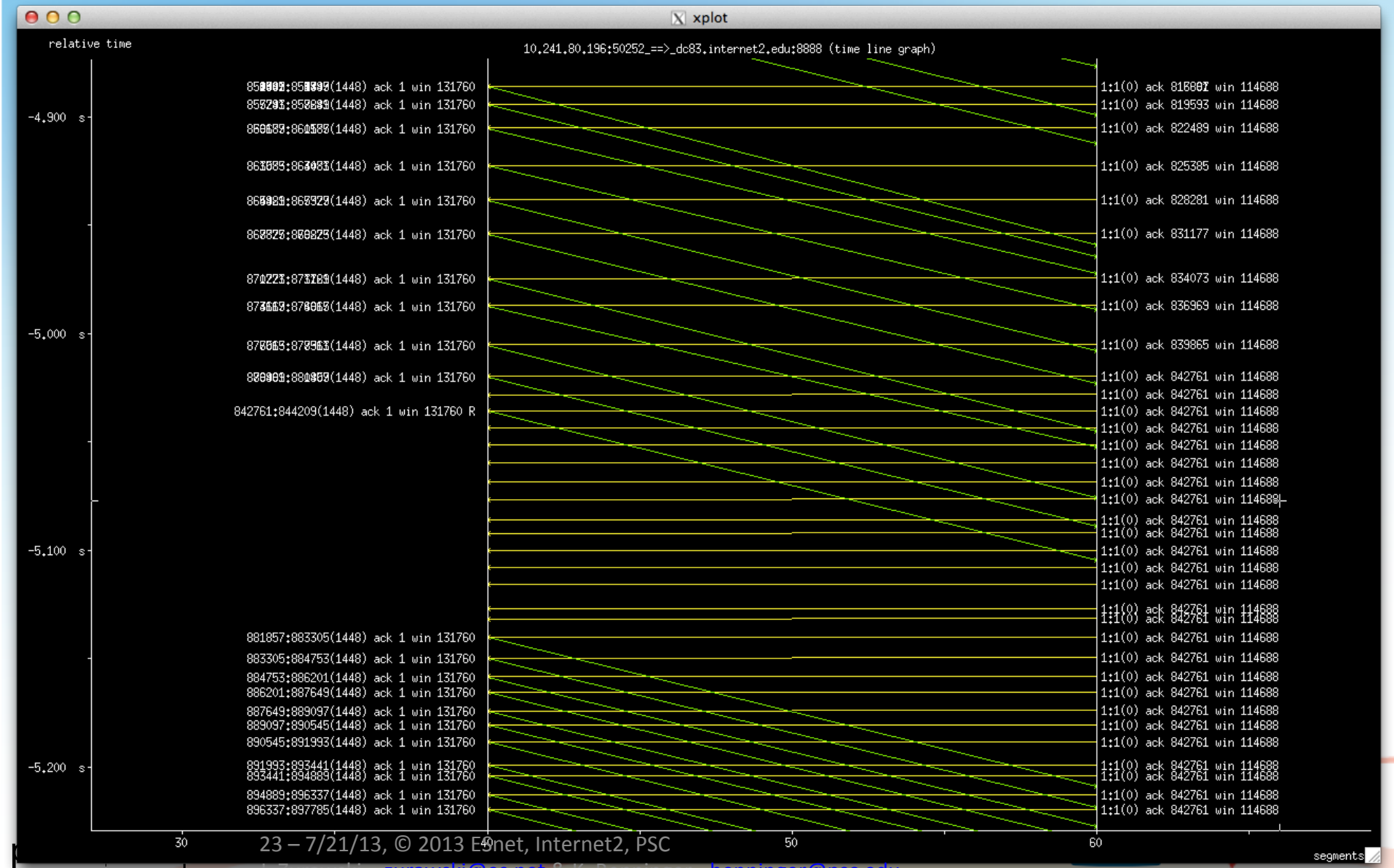
p = Probability of packet loss

J. Zurawski – zurawski@es.net & K. Benninger - benninger@psc.edu

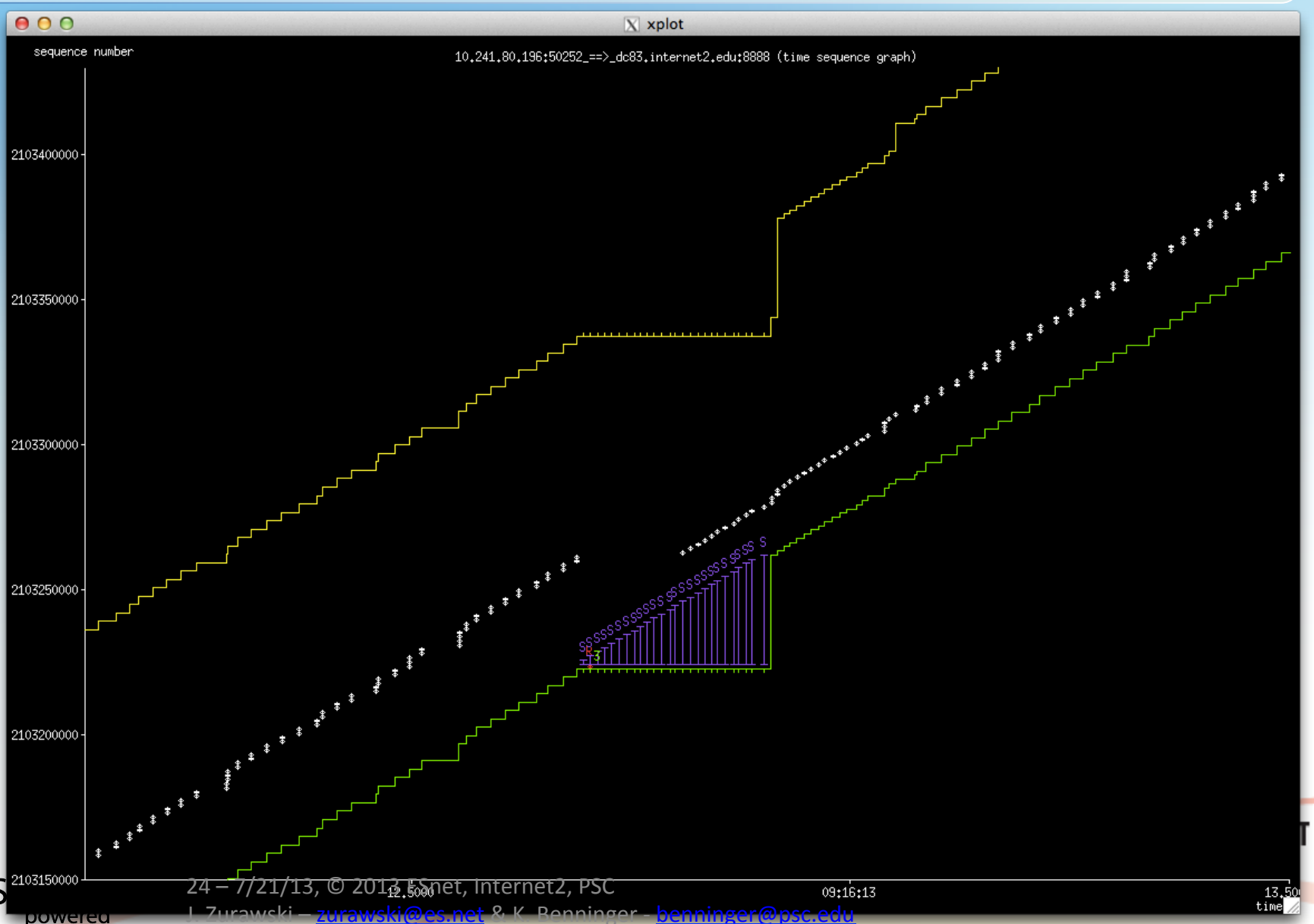
TCP – Data & ACK Feedback



TCP – DUP ACKs + Stalling



TCP – Resending the Window on Loss/Stall



TCP Performance: Parallel Streams

- Parallel streams can help in some situations
  - TCP attempts to be "fair" and conservative
  - Sensitive to loss, but more streams hedge bet
  - Circumventing fairness mechanism
    - 1 stream vs. n background: you get 1/(n+1)
    - X streams vs. n background: you get x/(n+x)
    - Example: 2 background, 1 stream: 1/3 = 33% of available resources
    - Example: 2 background, 8 streams: 8/10 = 80% of available resources
- There is a point of diminishing returns
- To get full TCP performance, the TCP window needs to be large enough to accommodate the ***Bandwidth Delay Product***

Stumbling Blocks – Packet Loss

- 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
- Major OSs default to a base of 64k.
  - For those playing at home, the maximum throughput with a TCP window of 64 KByte for RTTs:
    - 10ms = 50Mbps
    - 25ms = 20Mbps
    - 50ms = 10Mbps
    - 75ms = 6.67Mbps
    - 100ms = 5Mbps
  - ***Autotuning*** does help by growing the window when needed…

Windows Tuning

- Windows TCP Settings. These can be enabled, disabled, or changed.

```
C:\>netsh int tcp show global

Querying active state...

TCP Global Parameters
----------------------------------------------
Receive-Side Scaling State          : enabled
Chimney Offload State               : automatic
NetDMA State                        : enabled
Direct Cache Acess (DCA)            : disabled
Receive Window Auto-Tuning Level    : normal
Add-On Congestion Control Provider  : none
ECN Capability                      : disabled
RFC 1323 Timestamps                 : disabled
```

Agenda

- Expectations and Realities
- TCP Basics
- What We Can Determine
- The Tools

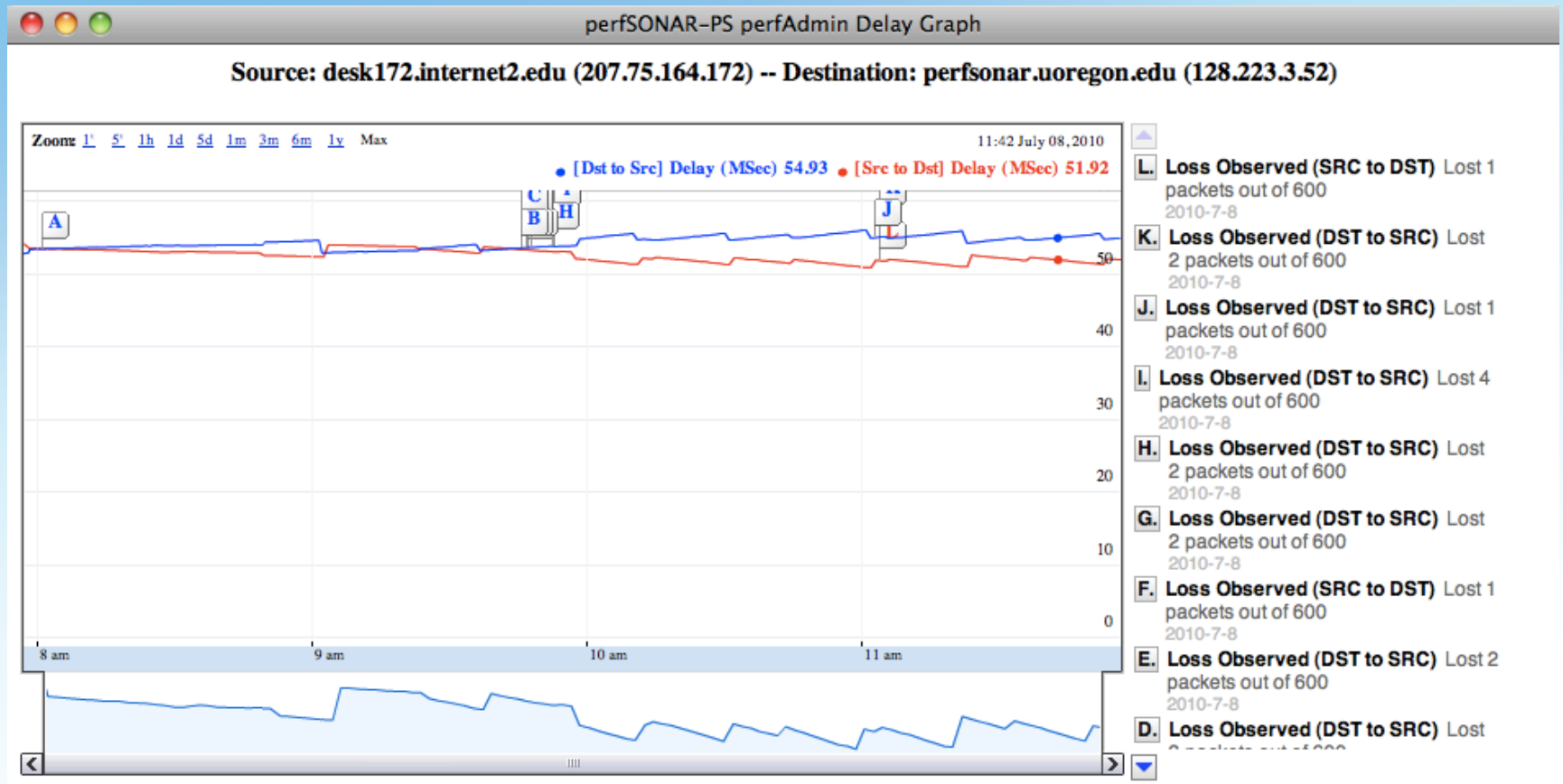
The Metrics

- Use the correct tool for the Job
  - To determine the correct tool, maybe we need to start with what we want to accomplish …
- What do we care about measuring?
  - Latency (Round Trip and One Way)
  - Jitter (Delay variation)
  - Packet Loss, Duplication, out-of-orderness (transport layer)
  - Interface Utilization/Discards/Errors (network layer)
  - Achievable Bandwidth (e.g. “Throughput”)
  - Traveled Route
  - MTU Feedback

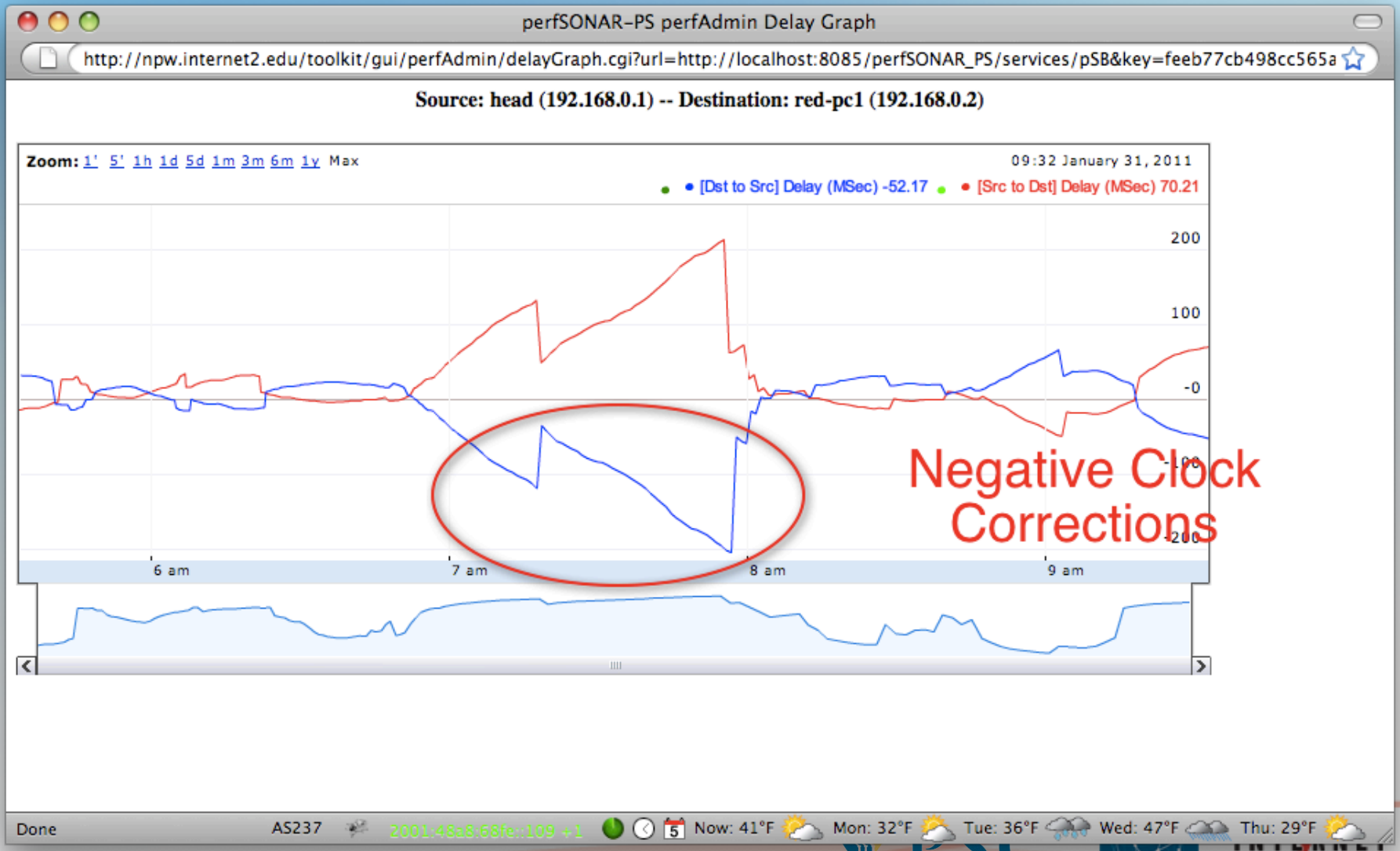
Latency

- Round Trip (e.g. source to destination, and back)
  - Hard to isolate the direction of a problem
  - Congestion and queuing can be masked in the final measurement
  - Can be done with a single ‘beacon’ (e.g. using ICMP responses)
- One Way (e.g. measure one direction of a transfer only)
  - Direction of a problem is implicit
  - Detects asymmetric behavior
  - See congestion or queuing in one direction first (normal behavior)
  - Requires ‘2 Ends’ to measure properly

Delay/Loss Plot From OWAMP



Latency Skew in OWAMP Data



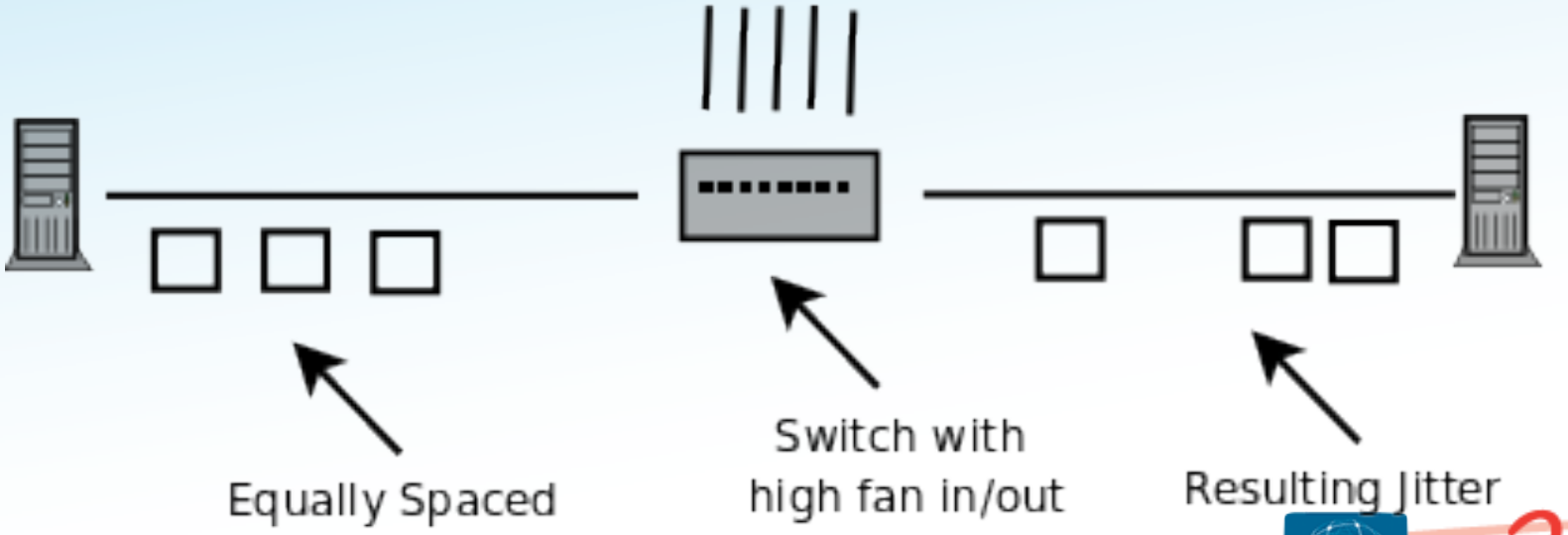
J. Zurawski – zurawski@es.net & K. Benninger - benninger@psc.edu

Jitter

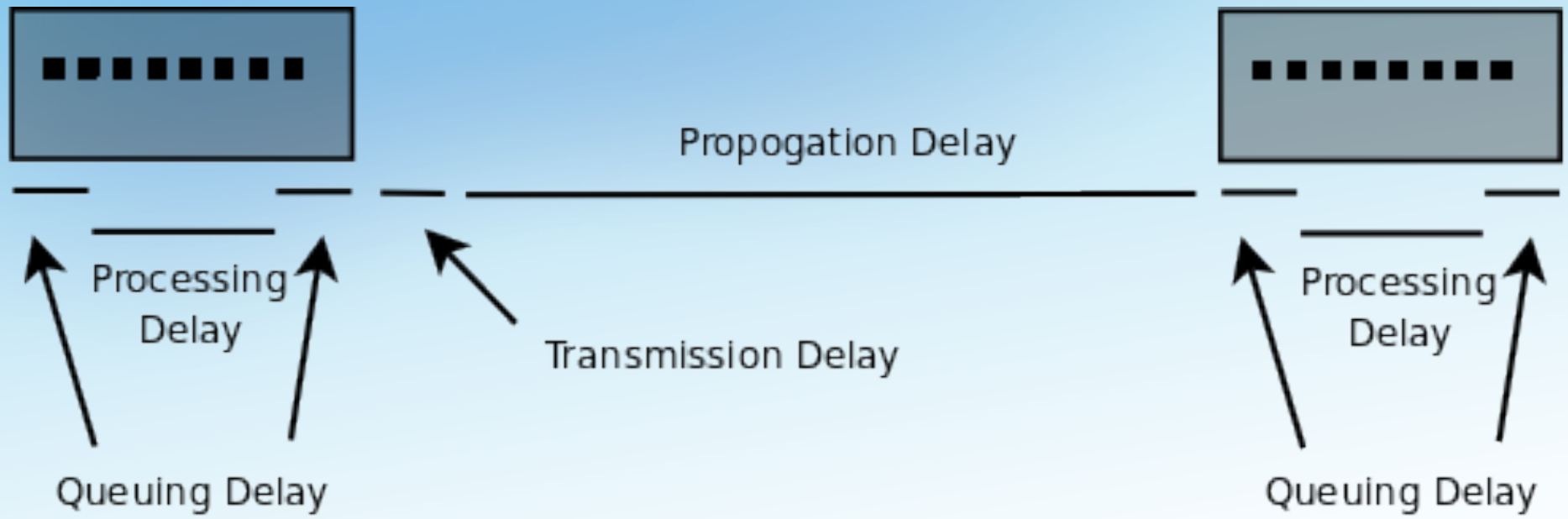
- To Quote Wikipedia: "***undesired deviation from true periodicity***"
- Computer people usually avoid the classic definition and use "***packet delay variation***" (PDV) instead
- In layman's terms:
  - Packet trains should be well spaced to aid in processing. The sending application/host ***should*** throttle/pace itself initially.
  - Bursts can cause queuing on devices (followed by periods of inactivity)
  - Jitter is a calculation of this variation in distances between packets. High jitter indicates things are consistently not well spaced

Jitter - Example

- Packets start well spaced at the host (usually)
- Devices in the path have processing per packet
  - Waiting in buffers, being processed (header info, or for danger info [firewalls])
  - Traversing switching fabric
- They don't always end up well spaced after…



Jitter - Example

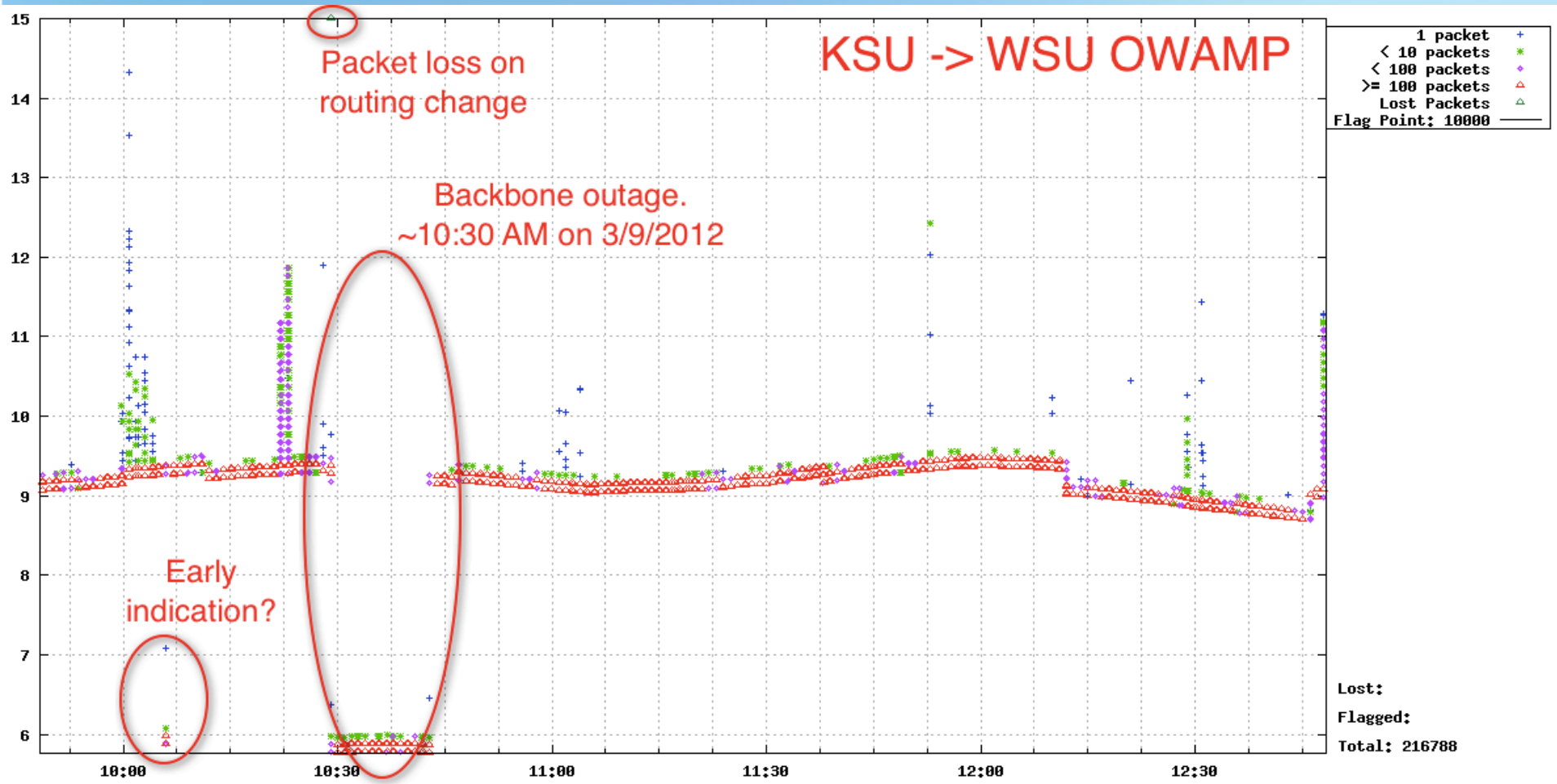


- **Processing Delay**: Time to process a packet
- **Queuing Delay**: Time spent in ingress/egress queues to device
- **Transmission Delay**: Time needed to put the packet on the wire
- **Propagation Delay**: Time needed to travel on the wire

Lost/Duplicated/Out of Order Packets

- We care about these on the transport (UDP/TCP) layer for the most part.
  - Network Layer loss is a bit different (and covered in other tools)
- Loss
  - Congestive (e.g. buffers are full on a device, it drops it on the floor)
  - Non-Congestive, (e.g. an error is introduced in transmission and the garbled packet is dropped)
- Duplication
  - If we don't receive an ACK from the other side, our window may stall and we may end up re-sending a packet.
- Out Of Order
  - Packets take a different path to a destination
  - Aggregation at the interface layer may shuffle up packets in a buffer. When they are delivered to the application they are not in the same order, which causes stalling

KanREN Monitoring – When Links Die



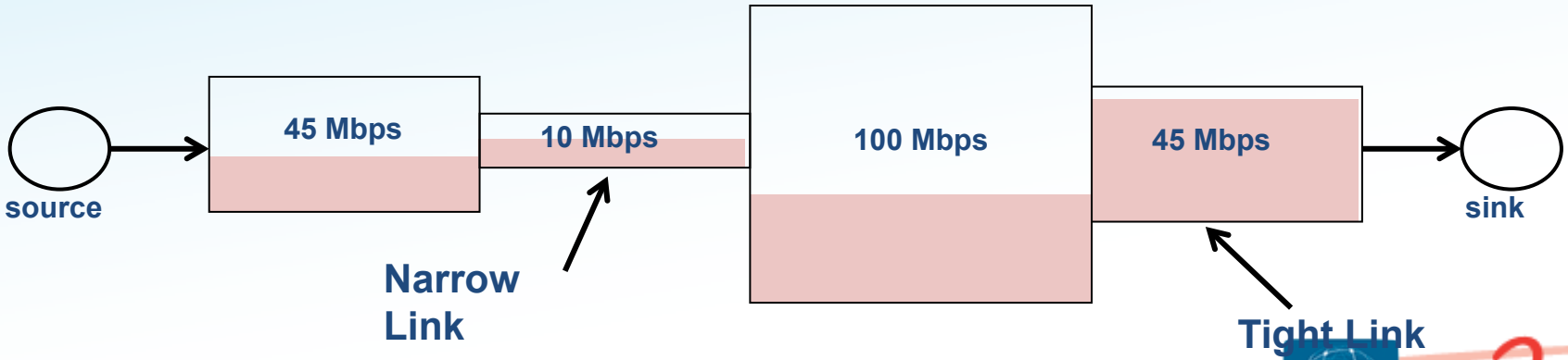
J. Zurawski – zurawski@es.net & K. Benninger - benninger@psc.edu

SNMP Metrics

- Network Layer metric.
- Lets us know what is passing through a specific interface on a specific device
- Utilization
  - How many bits, packets, etc. have passed through
- Discards
  - Packets dropped due to full buffers (or physical flaws on the device)
- Errors
  - Packets that fail a checksum

Throughput? Bandwidth?

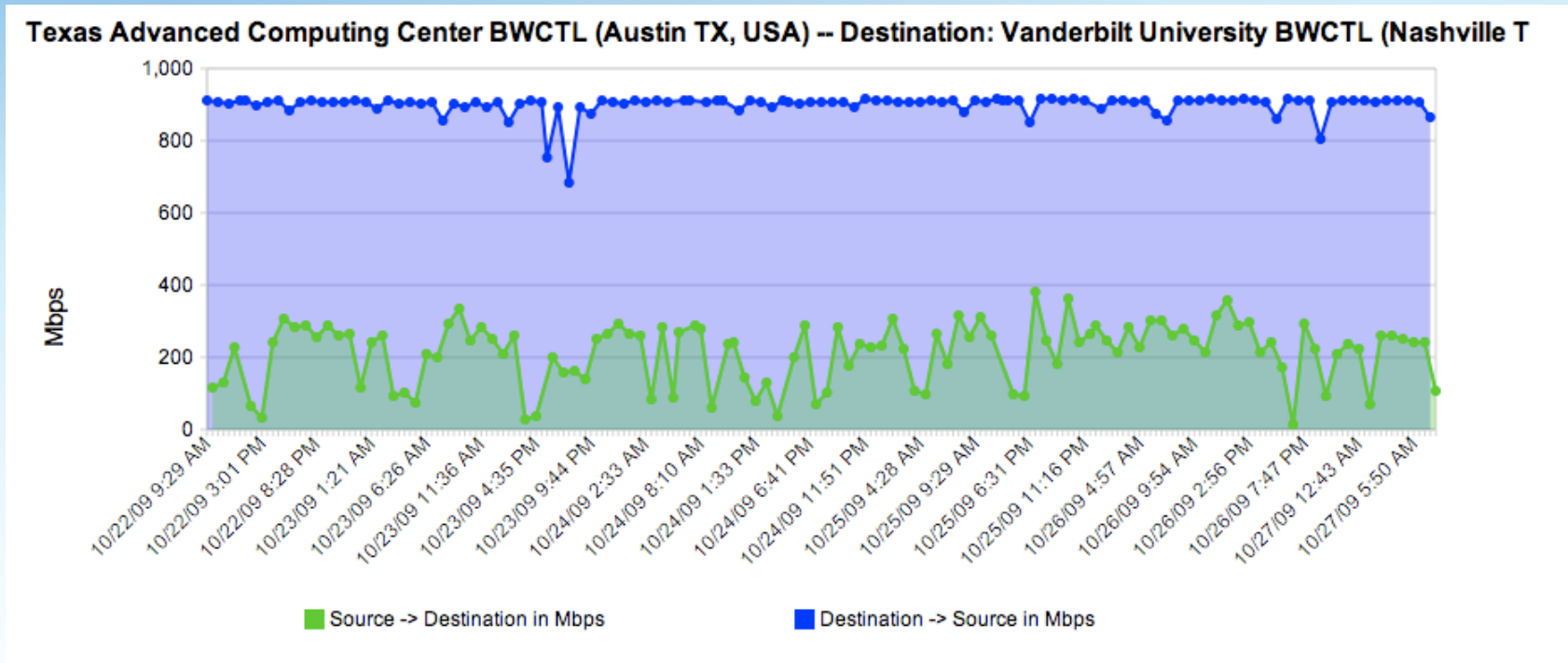
- The term "throughput" is vague
  - Capacity: link speed
    - Narrow Link: link with the lowest capacity along a path
    - Capacity of the end-to-end path = capacity of the narrow link
  - Utilized bandwidth: current traffic load
  - Available bandwidth: capacity – utilized bandwidth
    - Tight Link: link with the least available bandwidth in a path
  - Achievable bandwidth: includes protocol and host issues



(Shaded portion shows background traffic)

J. Zurawski – zurawski@es.net & K. Benninger - benninger@psc.edu

BWCTL Graphs



J. Zurawski – zurawski@es.net & K. Benninger - benninger@psc.edu

Traveled Route

- Normally confined to the Network Layer (although some tools available for Link Layer and Transport layer)
- Rely on ICMP, sometimes UDP and TCP
- Use TTL tricks to send out a packet for each 'hop' on a route, ask for a response
- Don't trust the latency numbers (especially with ICMP packets). Routers process ICMP on a different processor than 'real' data packets.

MTU Information

- 1500 Byte (total, some of this is header) is the defacto standard for MTU
- XSEDE uses “Jumbo Frames”, around 9000 bytes
  - Note – its usually good to set your devices higher (91xx) to capture ‘slush’
- If you have configured jumbo frames, and you try to pass these packets, two things have to be true for success:
  - The entire path must be enabled
  - Fragmentation has to be supported to break up the larger packets
- If these are not true, you run into “Black Hole” issues where packets are dropped.

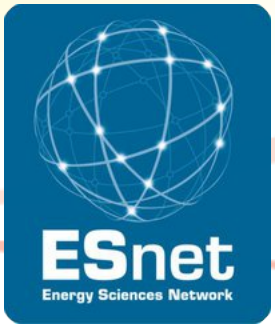
Agenda

- Expectations and Realities
- TCP Basics
- What We Can Determine
- The Tools

Mapping to the Tools

- What do we care about measuring?
  - Latency – Ping, OWAMP, NDT, NPAD [don't trust Traceroute]
  - Jitter – OWAMP, UDP Iperf, UDP Nuttcp, NDT, NPAD
  - Packet Loss, Duplication, out-of-orderness - OWAMP, UDP Iperf, UDP Nuttcp, NDT, NPAD
  - Interface Utilization/Discards/Errors - SNMP
  - Achievable Bandwidth – Iperf, Nuttcp, NDT, NPAD
  - Traveled Route – Traceroute, Tracepath
  - MTU Feedback – Tracepath, Ping, NDT, NPAD

J. Zurawski – zurawski@es.net & K. Benninger - benninger@psc.edu



Performance Primer

July 22nd 2013, XSEDE Network Performance Tutorial

Jason Zurawski – Internet2/ESnet
Kathy Benninger - Pittsburgh Supercomputing Center

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

perfSONAR powered