High Performance Data Transfer

SC2006 Tutorial M07

13 November 2006

Phillip Dykstra
Chief Scientist
WareOnEarth Communications Inc.
phil@sd.wareonearth.com

Online Copy

- This tutorial can be found at
  - http://www.wcisd.hpc.mil/~phil/sc2006/

Motivation

If our networks are so fast, how come my ftp is so slow?

Unique HPC Environment

- The Internet has been optimized for
  - millions of users behind low speed connections
  - thousands of high bandwidth servers serving millions of low speed streams
- Single high-speed to high-speed flows get little commercial attention

Tutorial Focus

- Moving a lot of data
  - Quickly
  - Securely
  - Error free
- Linux / Unix systems
- Open Source tools

Objectives

- Look at current high performance networks
- Fundamental understanding of delay, loss, bandwidth, routes, MTU, windows
- Learn what is required for high speed data transfer and what to expect
- Look at many useful tools and how to use them for performance testing and debugging
- Examine TCP dynamics and TCP's future
- Look at alternatives to TCP and higher level approaches to data transfer

Topics

Topics

Lessons from top500.org

Lessons from top500.org

- Operating Systems
  - Linux, 73%
  - Unix, 20%
  - Mac OS, 1%
  - Windows, 0.4%
- HPC is a Linux/Unix world (93%)
  - Yet Windows has perhaps a 90% share of the desktop market

Lessons from top500.org

- Interconnects
  - Gigabit Ethernet, 51%
  - Myrinet, 17% (Myricom)
  - SP Switch, 8% (IBM)
  - InfiniBand, 7%
  - Proprietary, 5%
  - Quadrics, 3%

Ethernet

- On nearly every computer in the world
- 1 Gbps Ethernet is now ~ $10/port
  - 10/100/1000 on copper (cat5)
- 10 Gbps Ethernet is now about
  - $2000 on fiber
  - $1000 on copper (CX4)
- Router interfaces cost $$$ more

Myricom

Myri-10G PCI-Express NIC
dual-protocol
10-Gigabit Ethernet and 10-Gigabit Myrinet

- MX/Ethernet does Myricom Express (MX) over standard Ethernet
- Claim 2.4 usec application latency (“5 to 10 times lower than TCP/IP”)
  Done via kernel bypass, application to NIC
- Myrinet-2000 is 2 Gbps

InfiniBand

Single, Double, Quad Data Rates

Link Aggregation

	SDR	DDR	QDR
1X	2 Gbps	4 Gbps	8 Gbps
4X	8 Gbps	16 Gbps	32 Gbps
12X	24 Gbps	48 Gbps	96 Gbps

- High speed serial interconnect
- Grew out of Future I/O (Compaq, IBM, HP) and NGIO (Intel, Microsoft, Sun) projects
- Some are doing InfiniBand over the WAN

Quadrics

- QsNet is 10 bit parallel copper, 900 MBps data rate (7.2 Gbps)

- QsTenG uses 10GigE on copper (CX4)

Lessons from top500.org

- HPC is a Linux/Unix world
  - Fortunately Linux has great networking

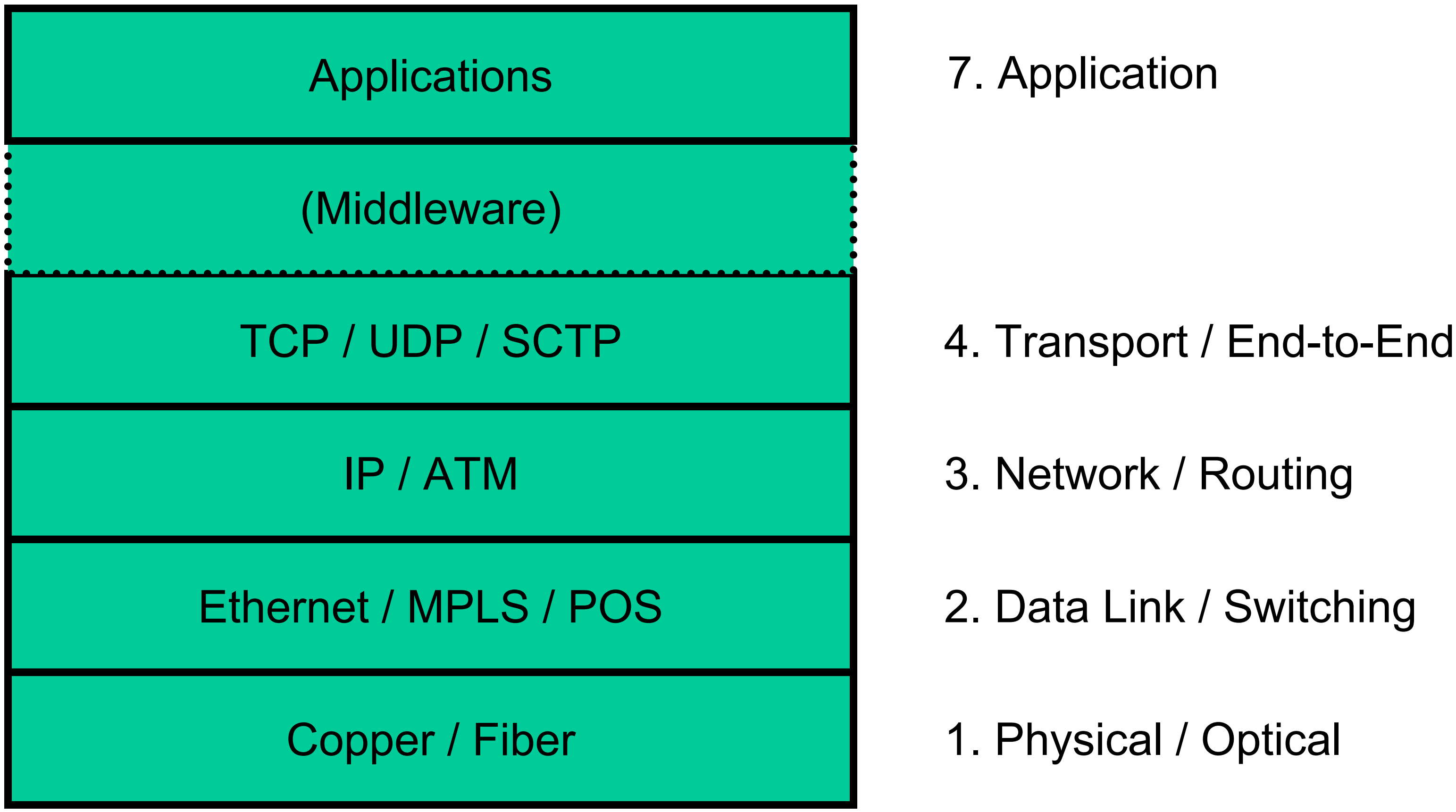
"*Based on all the measurements I'm aware of, Linux has the fastest and most complete [networking] stack of any OS*"

Van Jacobson, Jan 2006

Lessons from top500.org

- All top systems are parallel
  - Smallest was 32 processors, median ~1024
  - Even CPU's are going parallel (multi core)
  - Networks have gone parallel also (WDM)
- 1 Gbps and 10 Gbps Ethernet is taking over the world
  - So why aren't WANs Ethernet (yet)?

Quick Layer Review



Internet Protocols

Web
Email
SSH
FTP
P2P
…

DNS
NTP
VOIP
Multicast
Streaming Media
…

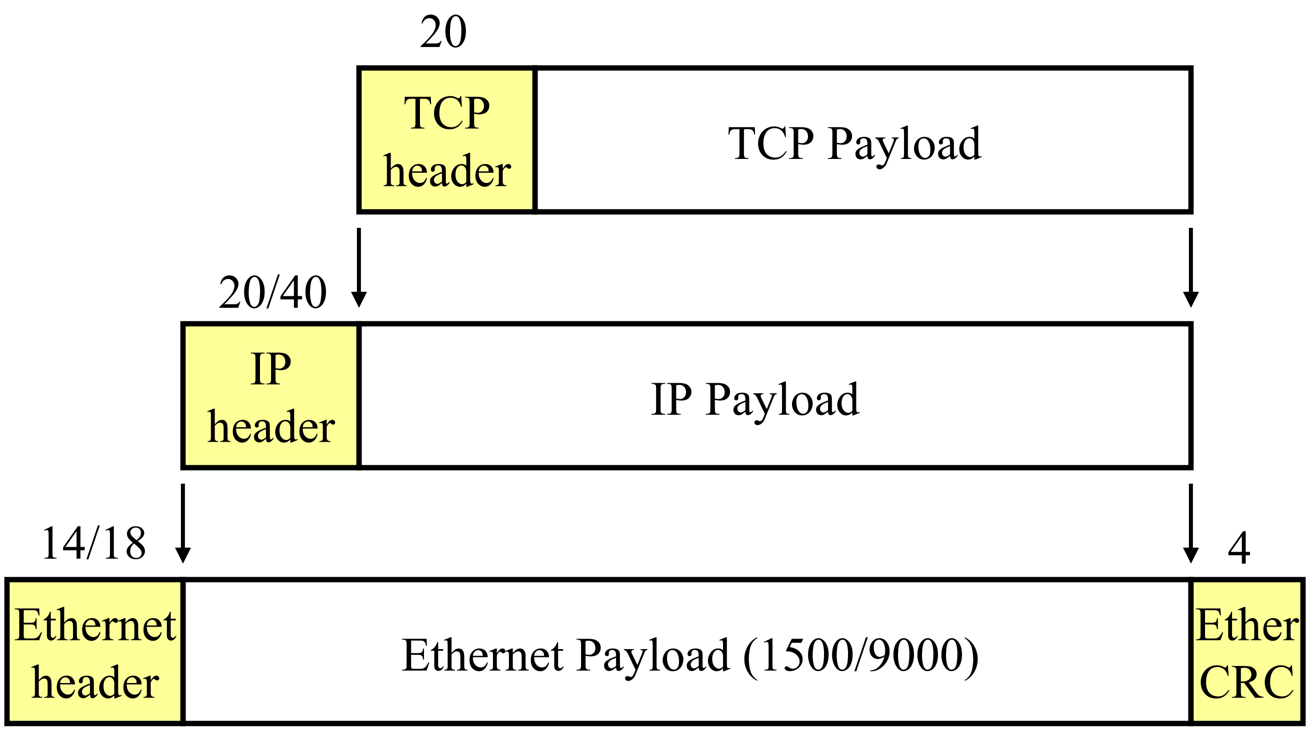
TCP_6	UDP_{17}
IP (v4 or v6)	

End-to-End Layer

Routing Layer

MPLS, ATM, Ethernet, POS, GFP, UCLP, etc.

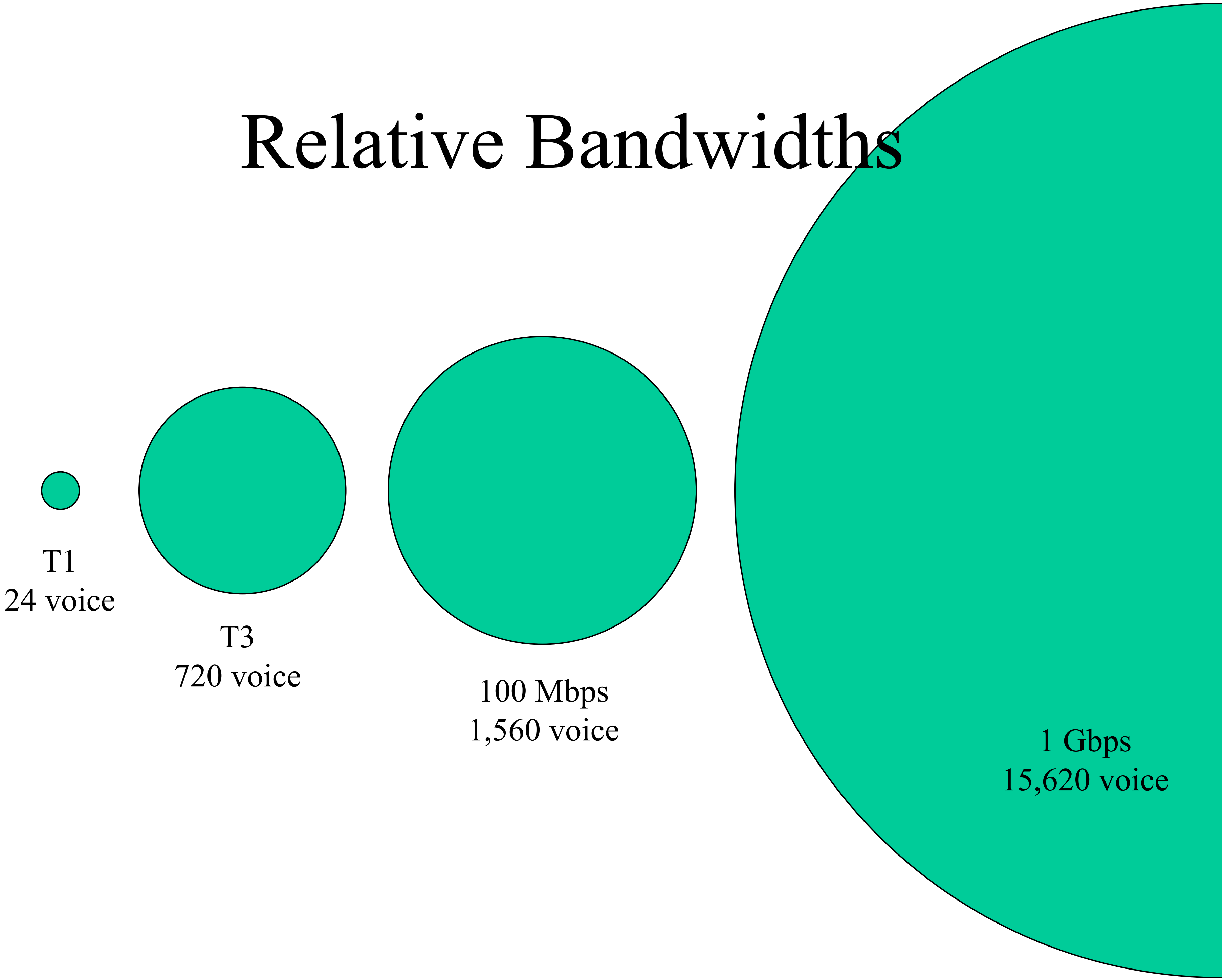
Encapsulation



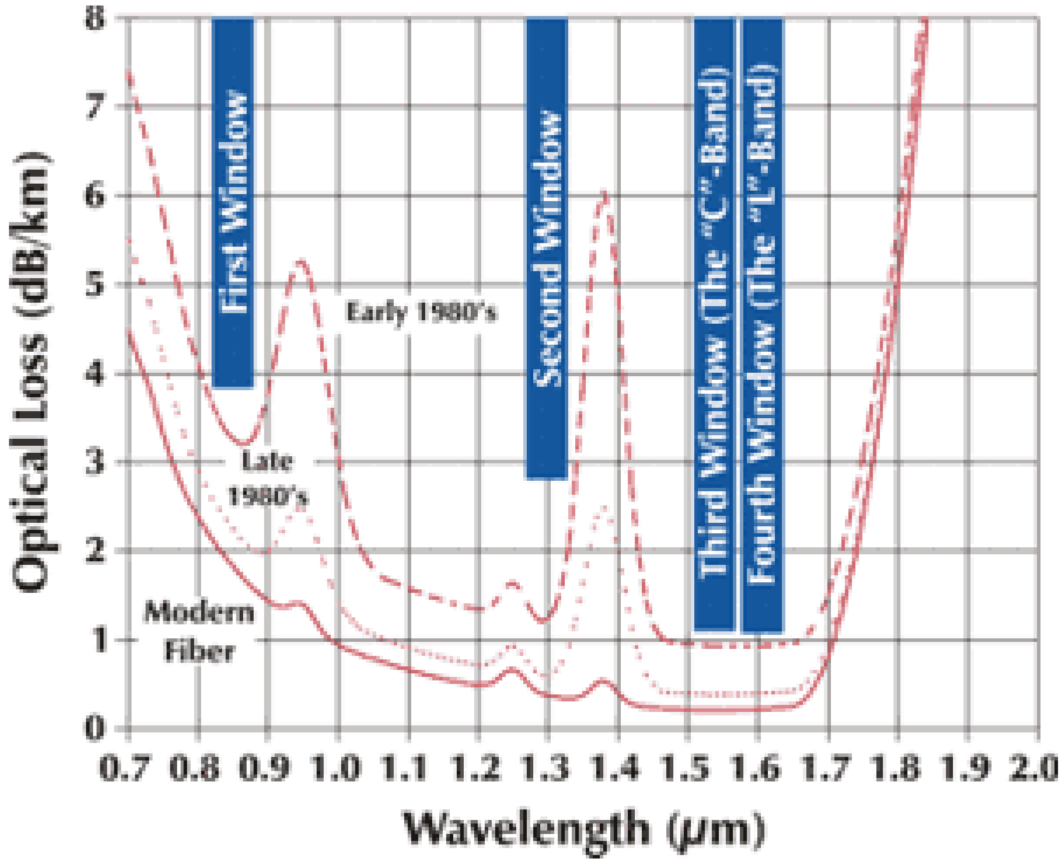
WAN Circuits

- TDM – Time Division Multiplexing (“copper”)
  - T1, 1.544 Mbps
  - T3, 44.736 Mbps
- SONET – Synchronous Optical Network (“fiber”)
  - OC3, 155.520 Mbps
  - OC12, 622.080 Mbps
  - OC48, 2.488 Gbps
  - OC192, 9.953 Gbps (“10 Gbps”)
  - OC768, 39.813 Gbps (“40 Gbps”)

Relative Bandwidths

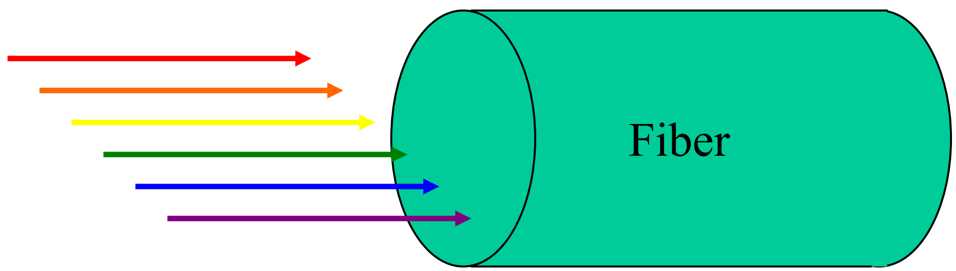


Fiber Optic Attenuation



David Goff, Fiber Optic Reference Guide

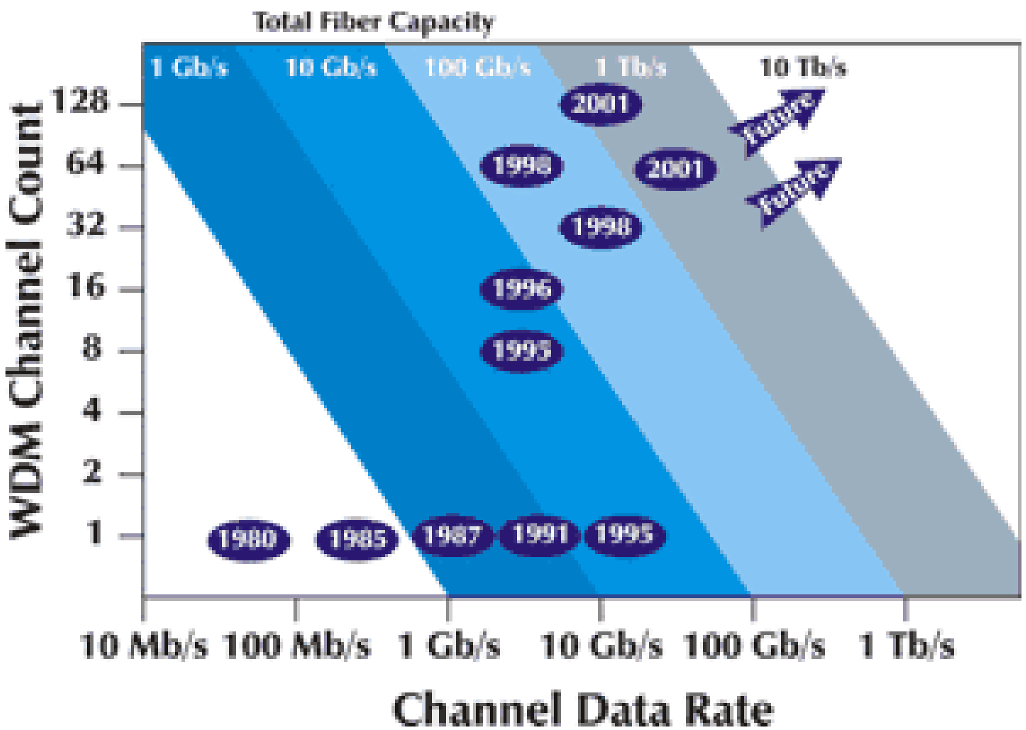
Wave Division Multiplexing (WDM)



- Use color / frequency to separate channels
- Course (CWDM) and Dense (DWDM) varieties
- Usually in the 1550 nm band

WDM Evolution

Toward 100 Tbps per fiber

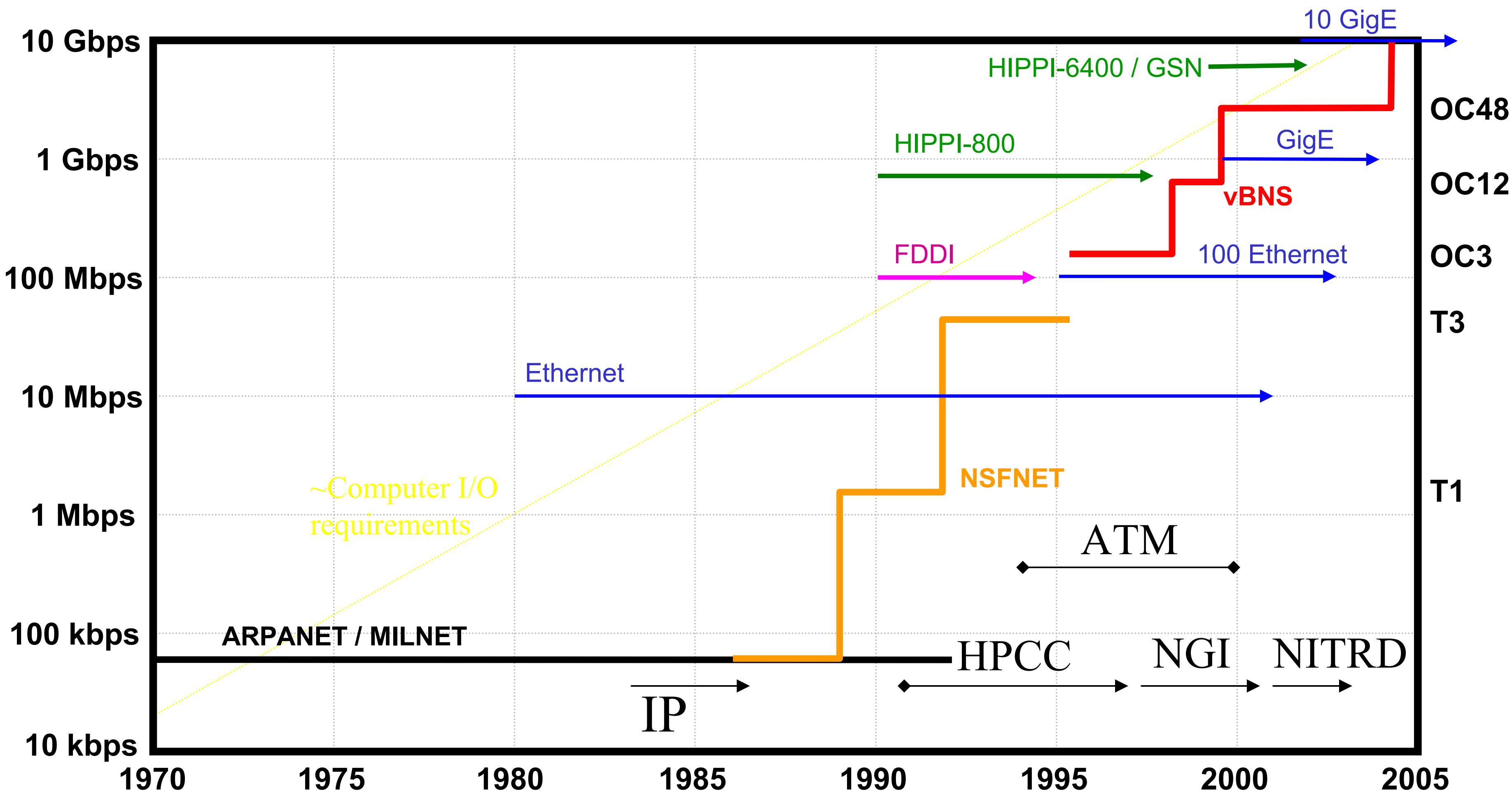


David Goff, Fiber Optic Reference Guide

DWDM Progress

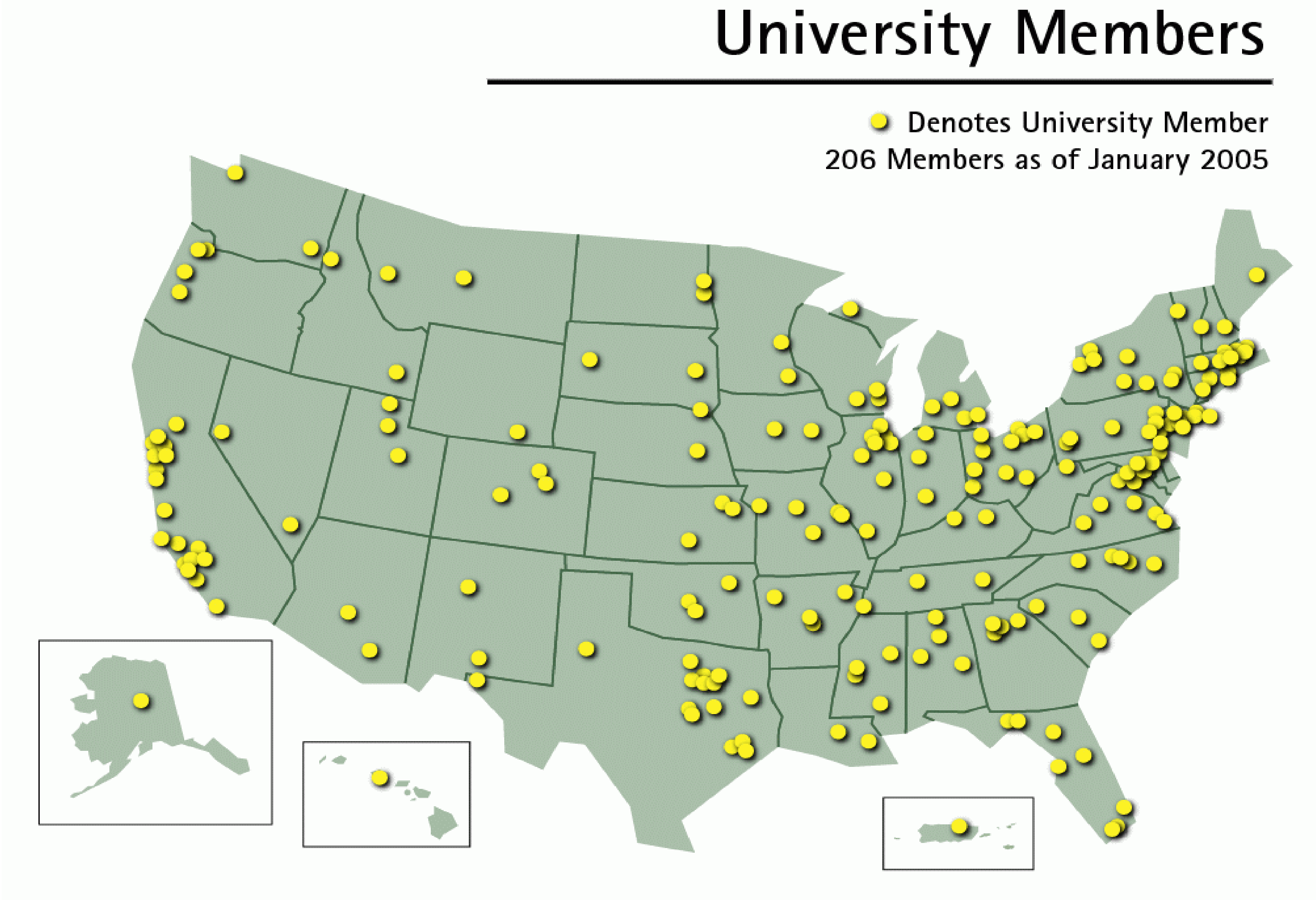
- Example DWDM Technology Deployment
  - 1996, 8 channels, 2.4 Gbps each (19 Gbps)
  - 2000, 40 channels, 10 Gbps each (400 Gbps)
  - 2005, 80 channels, 40 Gbps each (3.2 Tbps)
  - 2010, 320 channels, 160 Gbps each (51 Tbps)
- Cost is linear with # of channels
- ~2.5x cost for 4x bps within a TDM channel
- The return of wider channels (for higher rates)?
- A lot of fiber in the ground today will be obsolete

Network Speeds Over Time



High Performance Networks in the USA

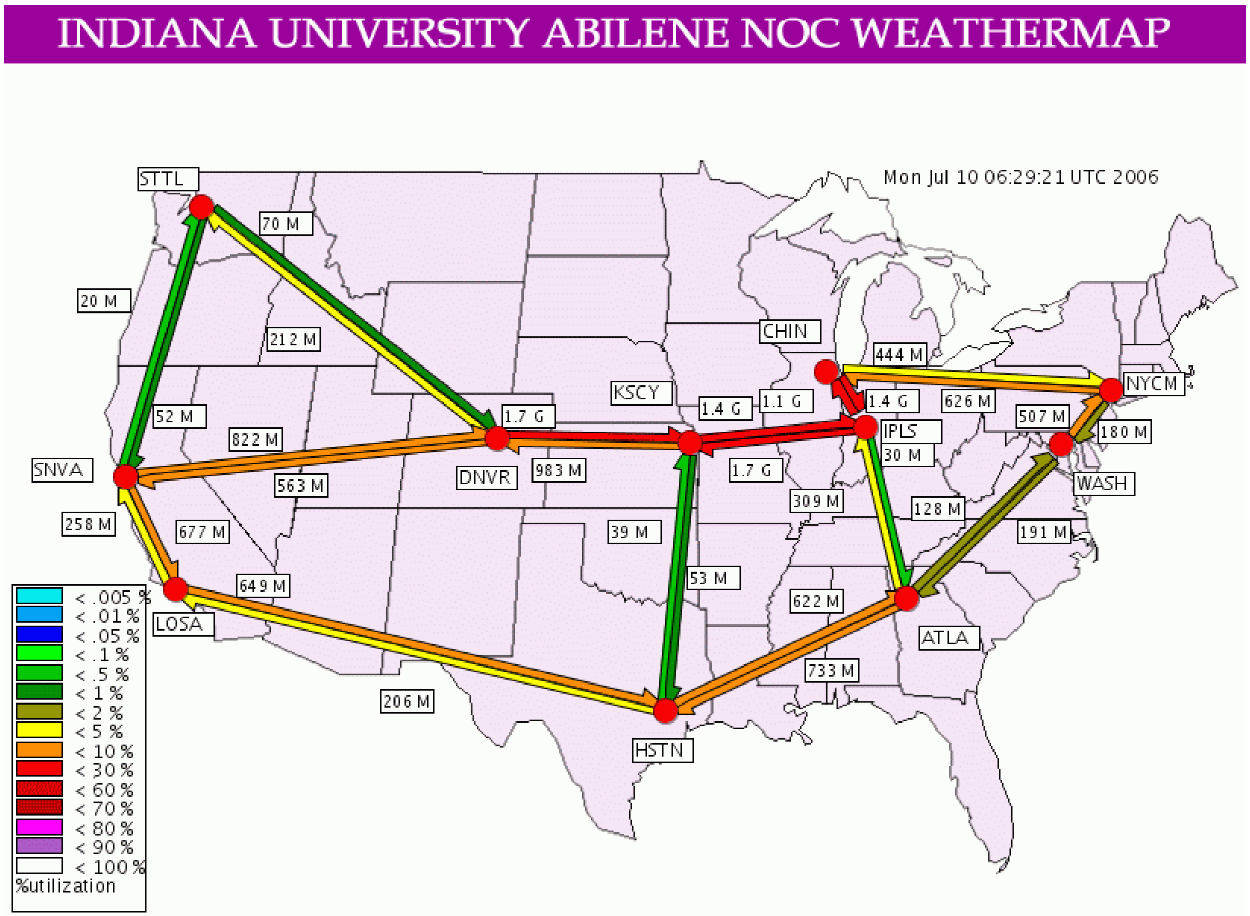
Internet2


University Members
Denotes University Member
206 Members as of January 2005

Internet2 Abilene Network

- Serves the Internet2 member institutions
- Built on Qwest fiber, Nortel optics, Juniper routers
- History
  - IOC on Feb 1999
  - OC48 (2.4 Gbps) backbone 1999
  - OC192 (10 Gbps) upgrade 2003
  - Ends October 2007

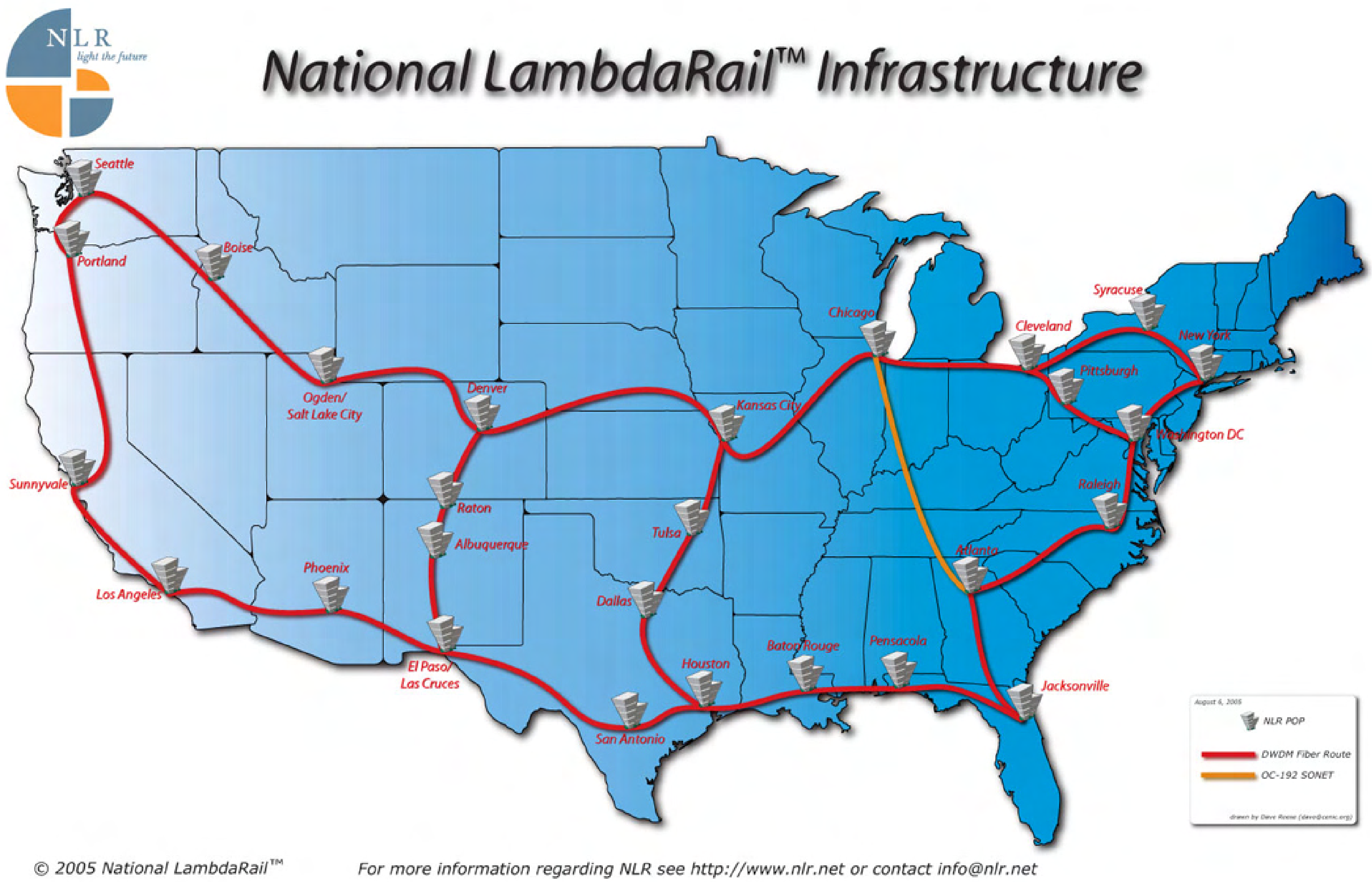
Internet2 Abilene Network



National LambdaRail (NLR)

- Controls a large collection of national fiber
  - 15,000 route miles, from Level3 and WilTel
  - Cisco 15808 (up to 40 wavelengths) and 15454 (up to 32 wavelengths) switches
- Offers members multi layered services
  - WaveNet – point to point 10GigE or OC192
  - FrameNet – Ethernet, shared or private
  - PacketNet – IPv4 and IPv6, CRS-1 routers

National LambdaRail Map

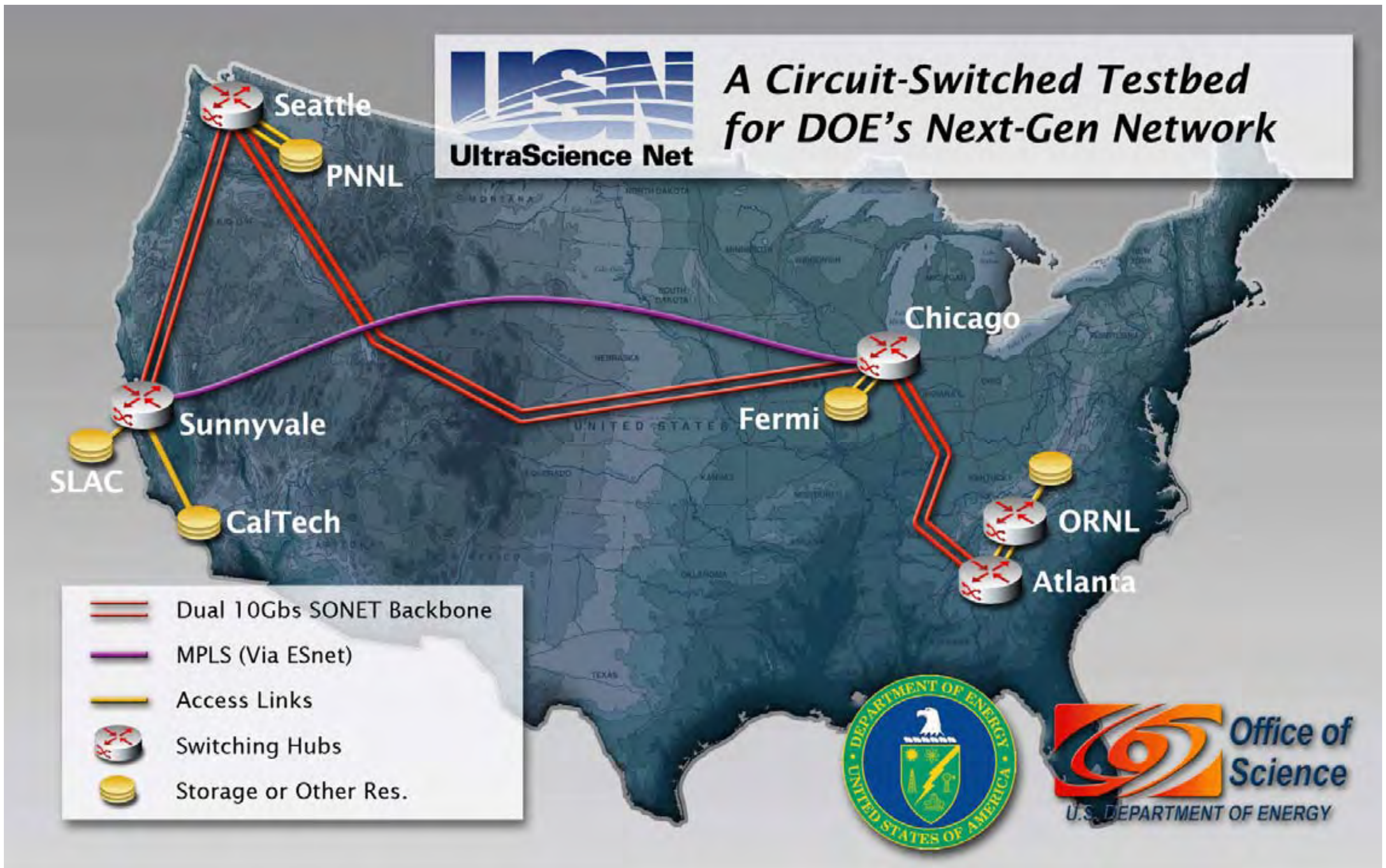


Projects on NLR

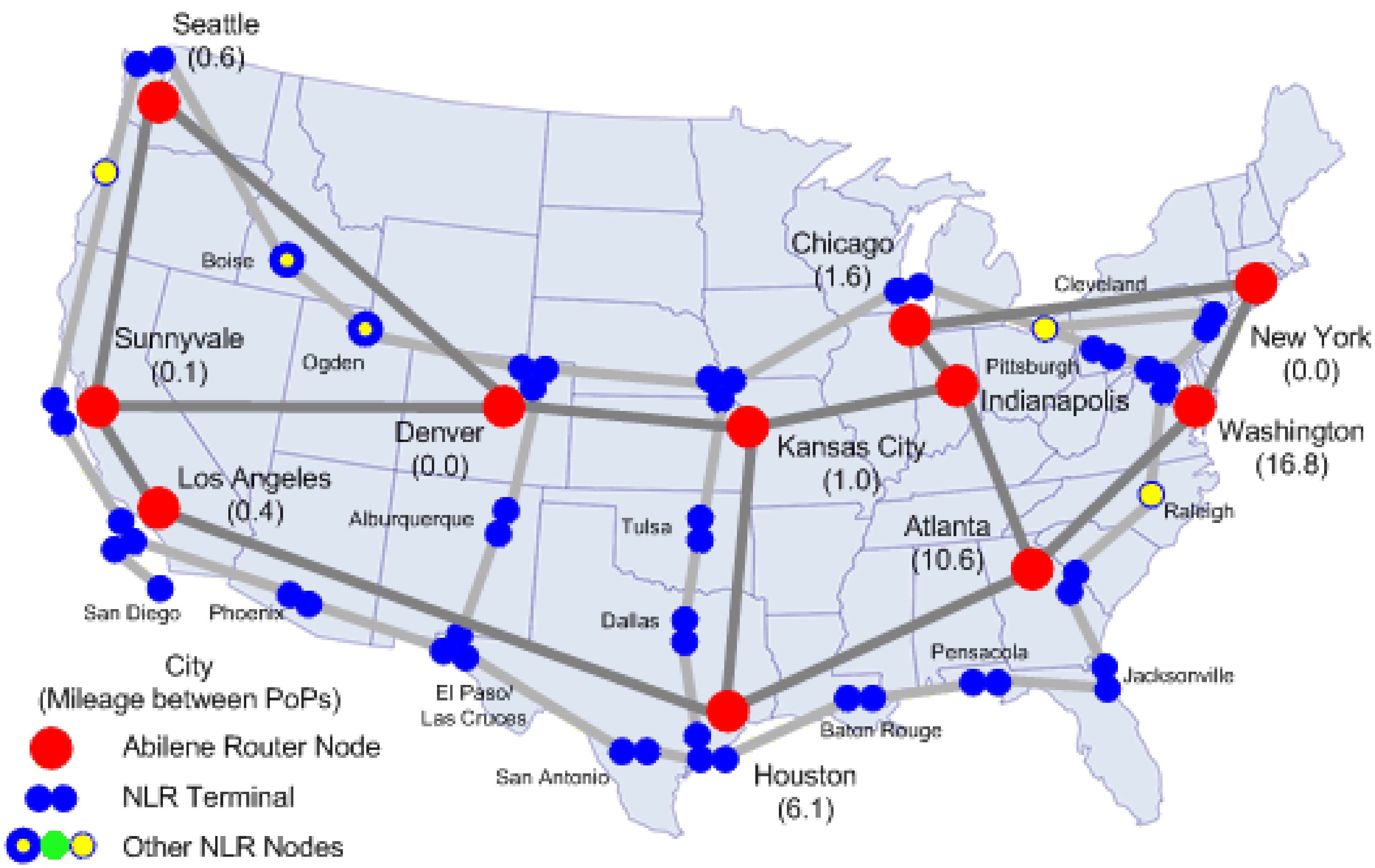
- NSF Extensible Terascale Facility (ETF)
- OptIPuter consortium
- DOE UltraScience Net (USN)
- Internet2 Hybrid Optical and Packet Infrastructure (HOPI)

TeraGrid / ETF

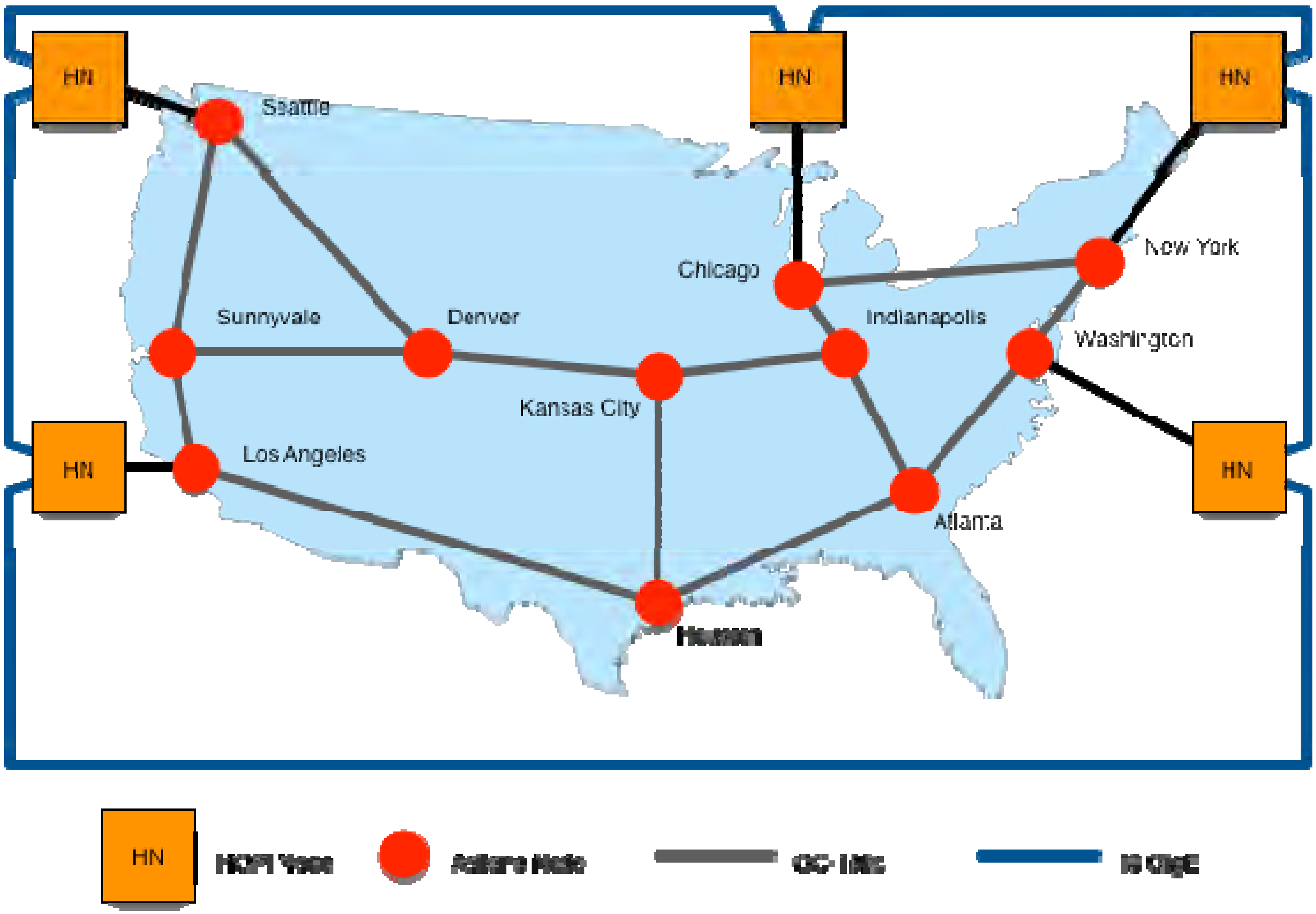
UltraScience Net



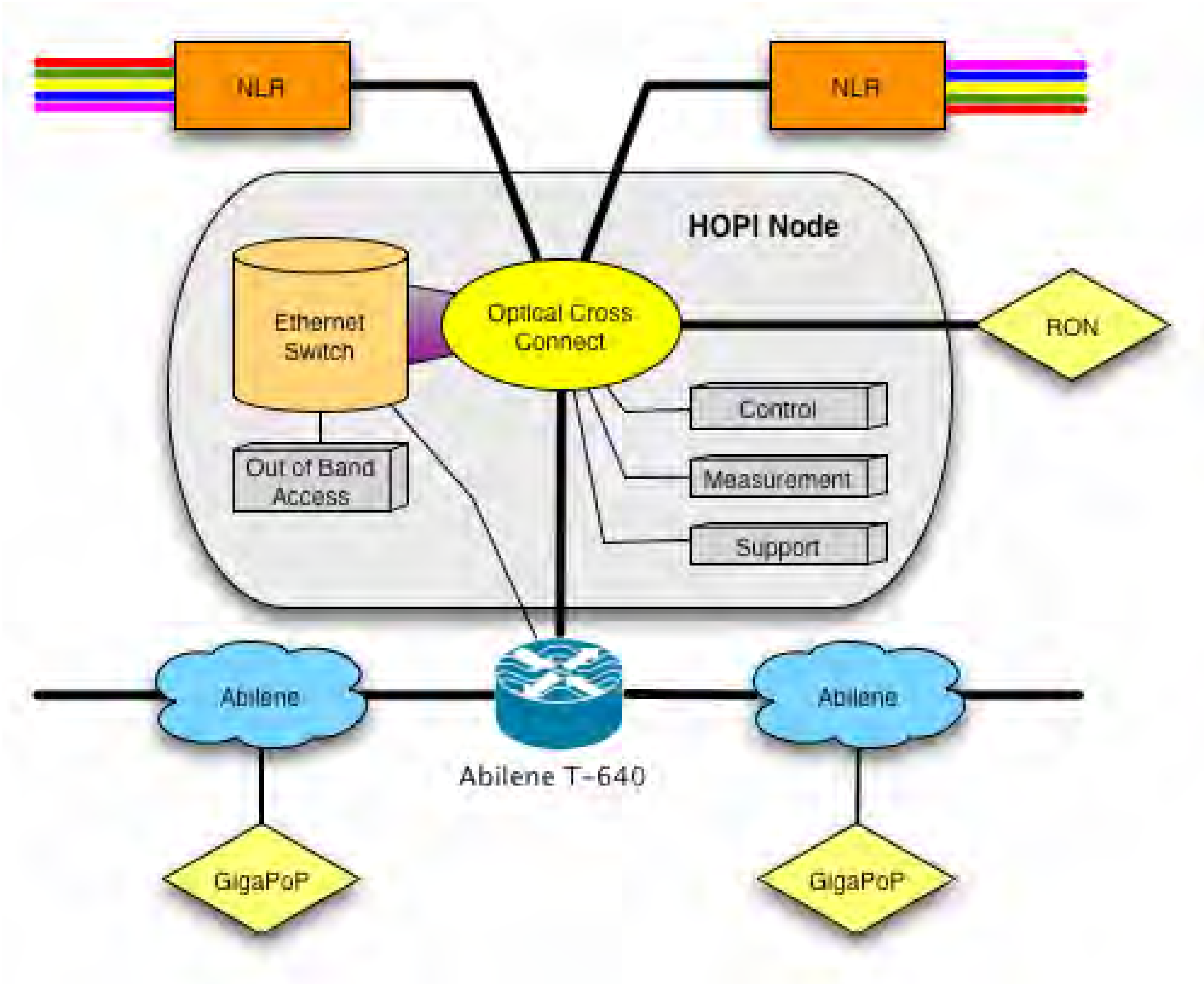
HOPI Network



HOPI Topology



HOPI Node

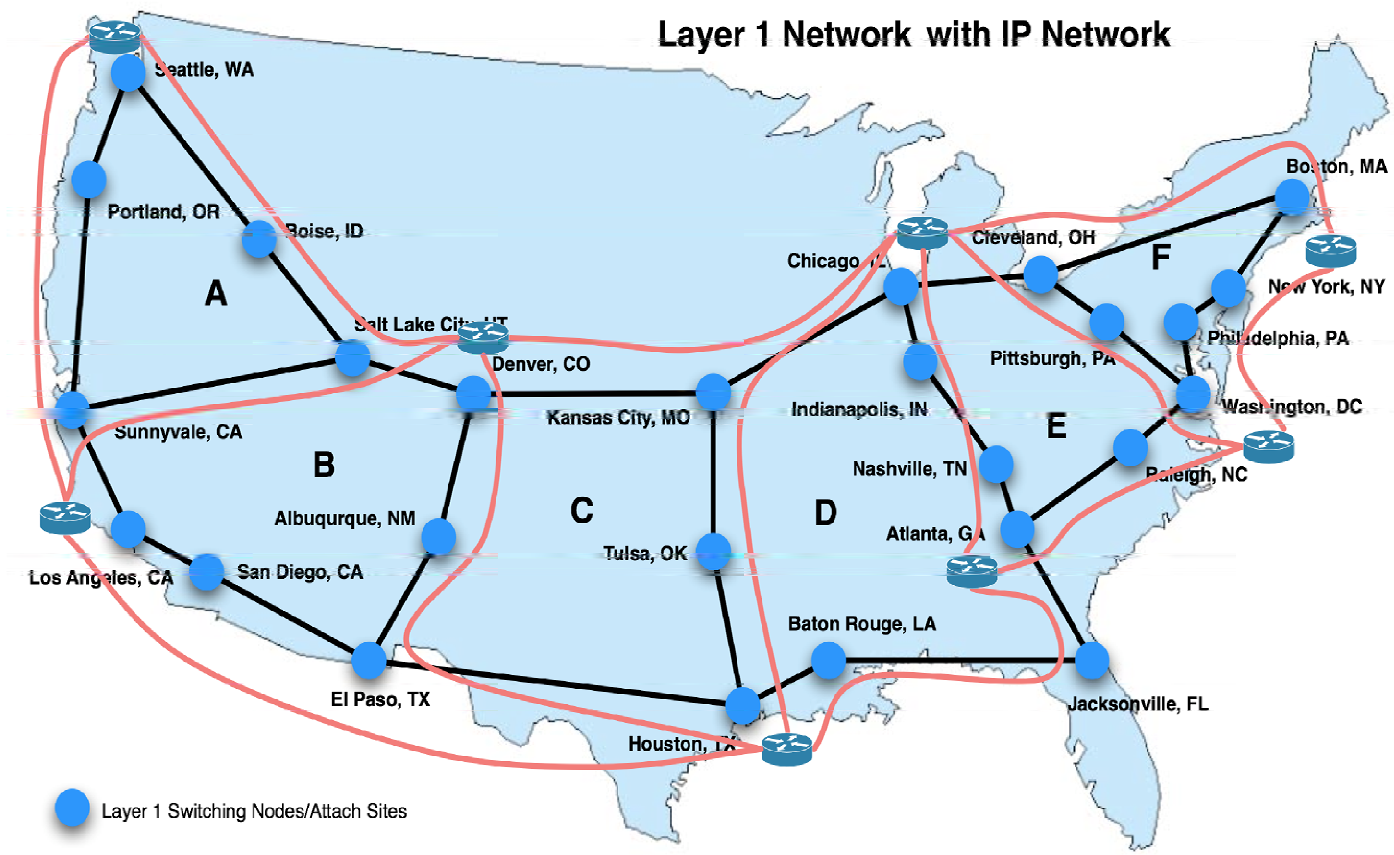

NLR
NLR
HOPI Node
Ethernet Switch
Optical Cross Connect
RON
Control
Out of Band Access
Measurement
Support
Abilene
Abilene
Abilene T-640
GigaPoP
GigaPoP

Internet2 NewNet

- Abilene's Qwest agreement ends October 2007
- NewNet will be built on
  - Level3's Dedicated Wave System (DWS)
    - 10 x 10 Gbps initially
    - Up to 80 x 40 Gbps or 80 x 100 Gbps (8 Tbps)
  - Infinera optical switches
  - Juniper routers
- Expect ~20 Regional Optical Networks (RONs) to connect 10 Gbps IP + 10 Gbps optical

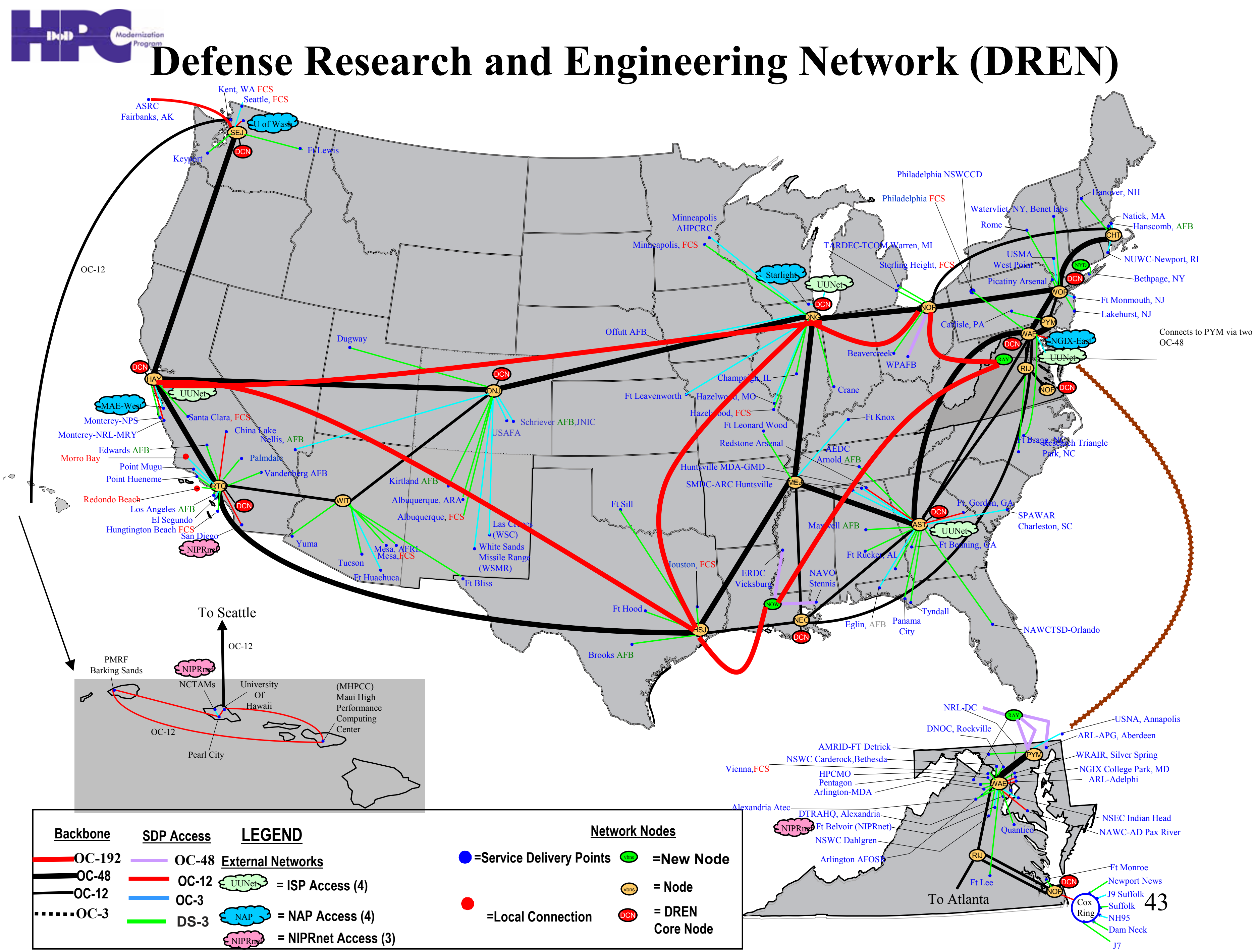
Internet2 NewNet

Provisional Topology, June 2006

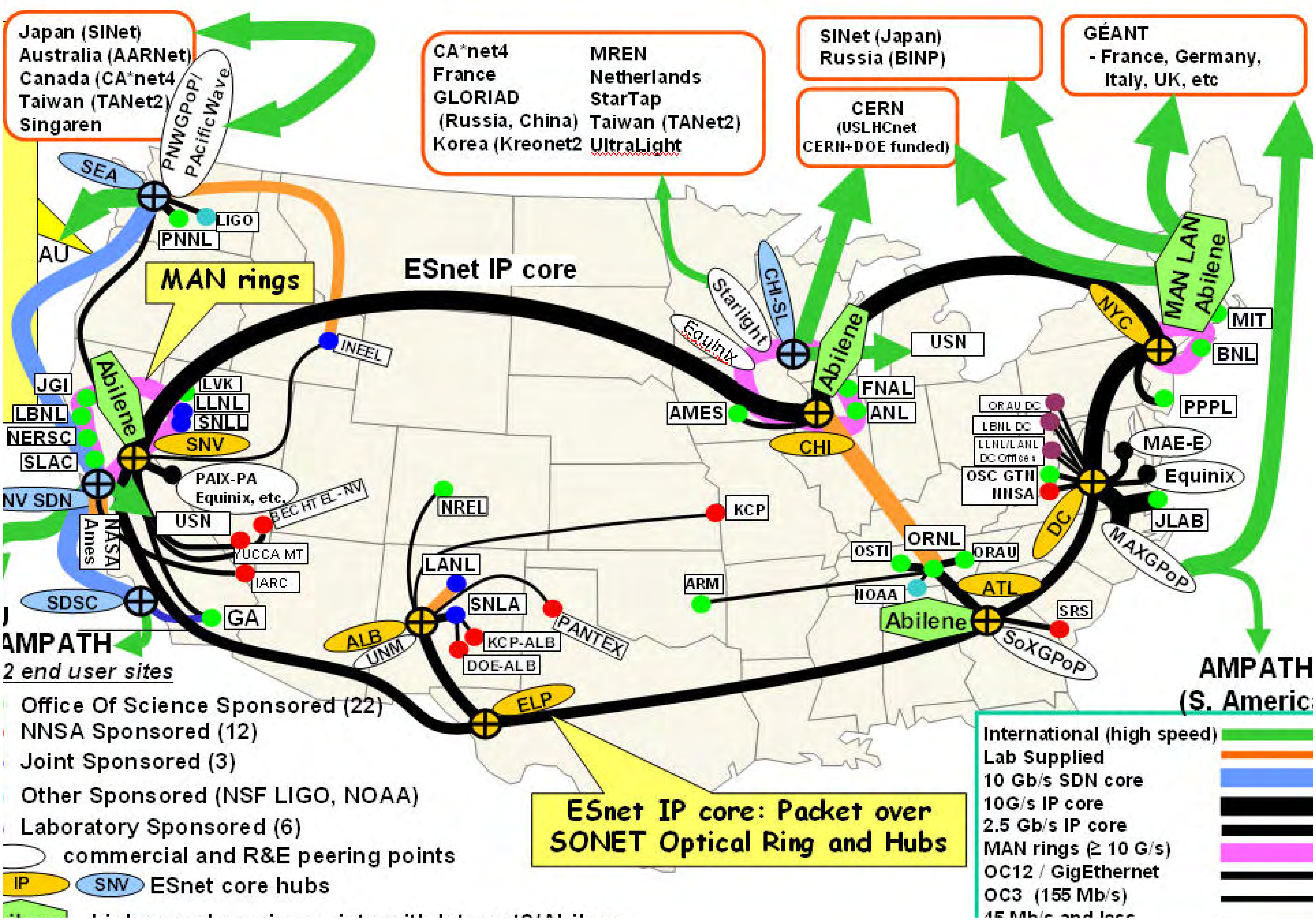


NASA Research and Engineering Network (NREN)

Defense Research and Engineering Network (DREN)



➢ESnet's Physical Connectivity (Spring 2006)



Commercial Networks

The top ten networks defined by address space

Rank	IP Space	ASN	Description
1	89,608,854	721	DISA CONUS
2	43,568,261	3356	Level 3 Communications, LLC
3	32,882,064	701	UUNET Technologies, Inc.
4	29,077,394	4134	No.31,Jin-rong Street
5	28,725,288	17676	APNIC ASN block
6	25,691,444	7018	AT&T WorldNet Services
7	23,882,363	174	Cogent Communications
8	23,775,293	7132	SBC Internet Services
9	22,987,395	3352	Internet Access Network of TDE
10	18,428,110	237	Merit Network Inc.

Source: www.fixedorbit.com, Sep 2006

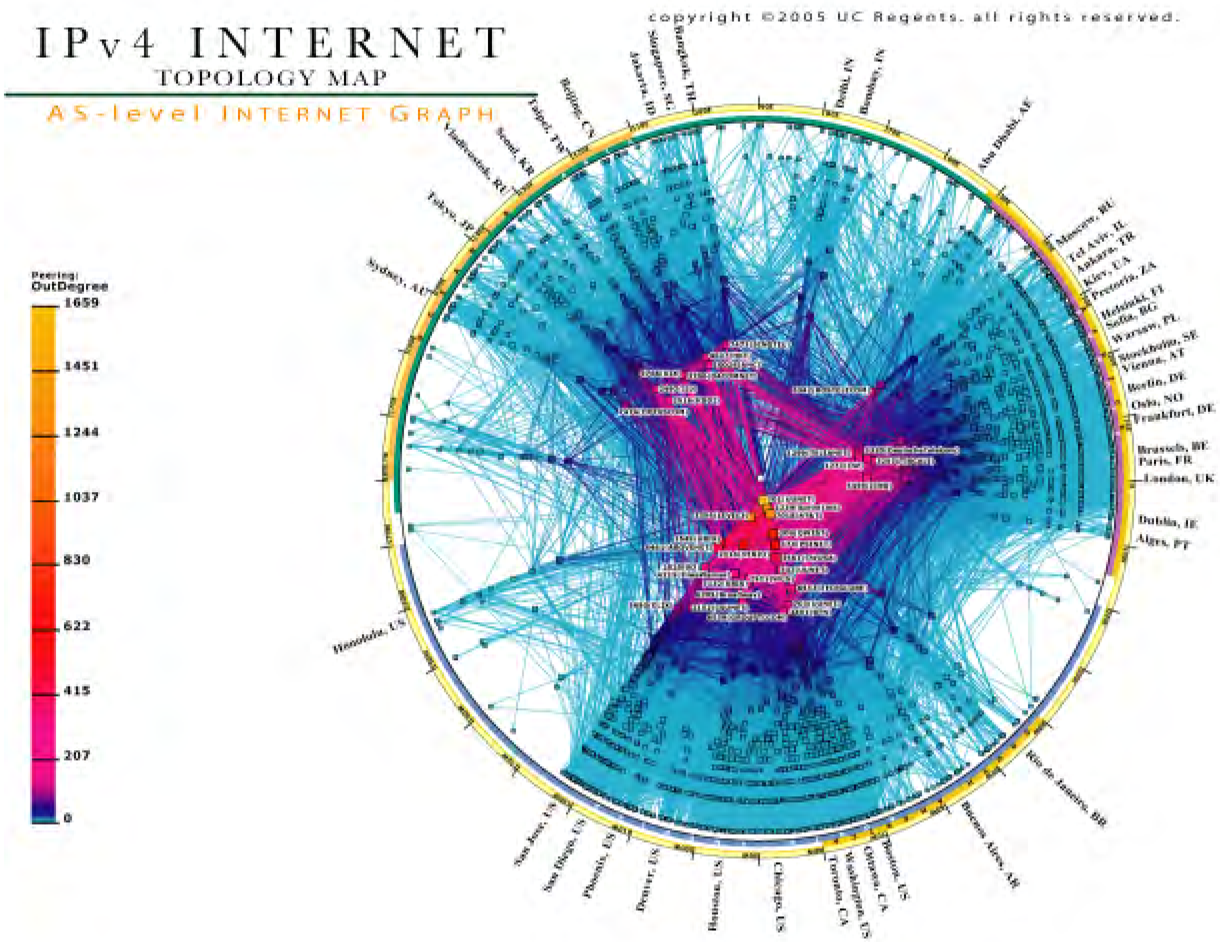
Commercial Networks

The top ten networks defined by number of peers

Rank	Peers	ASN	Description
1	2,402	701	UUNET Technologies, Inc.
2	2,025	7018	AT&T WorldNet Services
3	1,720	1239	Sprint
4	1,302	3356	Level 3 Communications, LLC
5	1,210	174	Cogent Communications
6	1,176	209	Qwest
7	739	3549	Global Crossing
8	715	4323	Time Warner Telecom, Inc.
9	701	6461	Abovenet Communications, Inc
10	655	7132	SBC Internet Services

Source: www.fixedorbit.com, Sep 2006

AS Core Visualization



Apr 2005

www.caida.org/analysis/topology/as_core_network/

Network References

- Abilene, abilene.internet2.edu
- DREN, www.hpcmo.hpc.mil/Htdocs/DREN
- ESnet, www.es.net
- NLR, www.nlr.net
- NREN, www.nren.nasa.gov
- vBNS+, www.vbns.net

Delay

a.k.a. Latency

Speed Limit

The speed of light is a constant

It seems slower every year

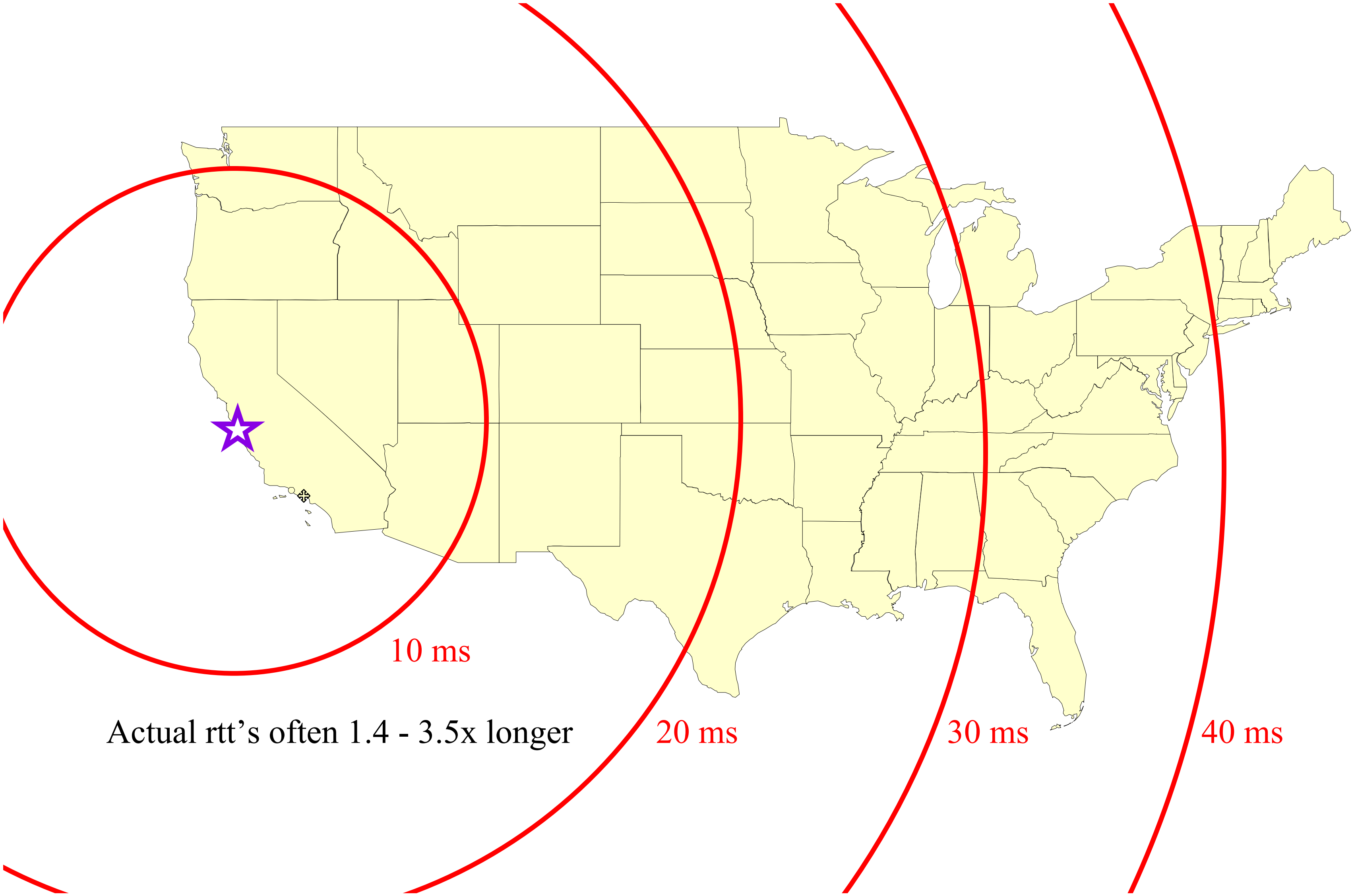
Speed of Light in Media

- ~3.0×10^8 m/s in free space
- ~2.3×10^8 m/s in copper
- ~2.0×10^8 m/s in fiber = 200 km / ms

  [100 km of distance = 1 ms of round trip time]

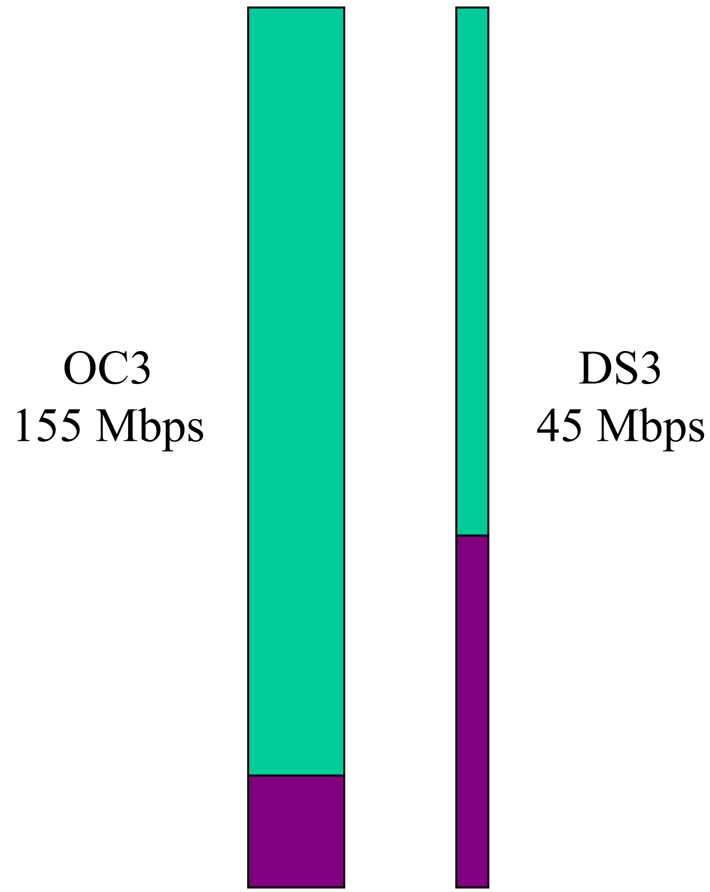
We gave up 1/3 of our speed to use fiber

Light Speed Delay in Fiber



Actual rtt’s often 1.4 - 3.5x longer

High “~~Speed~~” Capacity Networks



Packet Durations and Lengths

1500 Byte Packets in Fiber

	Mbps	pps	sec/pkt	length
56k	0.056	4.7	214 ms	42857 km
T1	1.544	129	7.8 ms	1554 km
Eth	10	833	1.2 ms	240 km
T3	45	3750	267 us	53 km
FEth	100	8333	120 us	24 km
OC3	155	13k	77 us	15 km
OC12	622	52k	19 us	3859 m
GigE	1000	83k	12 us	2400 m
OC48	2488	207k	4.8 us	965 m
10GigE	10000	833k	1.2 us	240 m

Observations on Packet Lengths

- A 56k packet could wrap around the earth!

- A 10GigE packet fits in the convention center

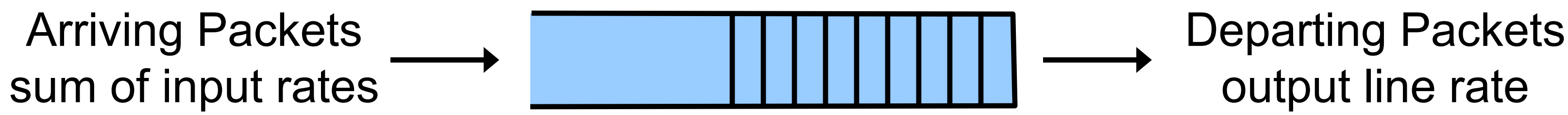
Observations on Packet Lengths

- Each store and forward hop adds the packet duration to the delay
  - In the old days (< 10 Mbps) such hops dominated delay
  - Today (> 10 Mbps) store and forward delays on WANs are minimal compared to propagation

Observations on Packet Lengths

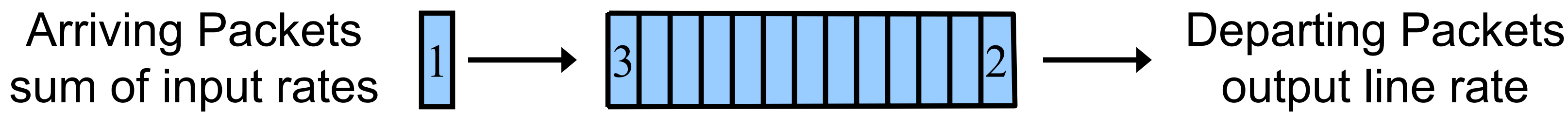
- ATM cells (and TCP ACK packets) are ~1/30th as long, 30x as many per second
  - One of the reasons we haven't seen OC48 SAR until recently (2002)
- Jumbo Frames (9000 bytes) are 6x longer, 1/6th as many per second

Router Queues



- The major source of variable delay
- Handle **temporary** inequalities between arrival rate and output interface speed
- Small queues minimize delay variation
- Large queues minimize packet drop

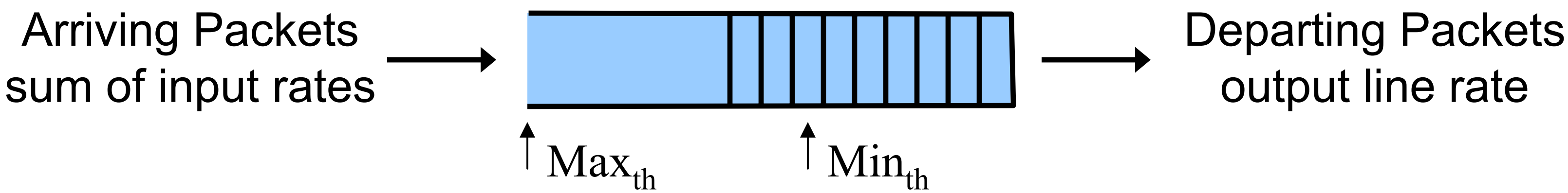
Passive Queue Management (PQM)



When full, choose a packet to drop

1. Tail-Drop – arriving packet
2. Drop-From-Front – packet at front of queue
3. Push-Out – packet at back of queue
4. Random-Drop – pick any packet

Active Queue Management (AQM)



- Addresses lock-out and full queue problems
- Incoming packet drop probability is a function of average queue length
- Random Early Detection (RED)
  - Recommended Practice, RFC 2309, Apr 98
  - Min_{th}, Max_{th}, Min_{drop}, Max_{drop}, w
  - www.icir.org/floyd/red.html

Measuring Delay: Ping

$ **ping –s 56 cisco.com**
PING cisco.com (198.133.219.25) from 63.196.71.246 : 56(84) bytes of data.
64 bytes from www.cisco.com (198.133.219.25): icmp_seq=1 ttl=241 time=25.6 ms
64 bytes from www.cisco.com (198.133.219.25): icmp_seq=2 ttl=241 time=25.5 ms
64 bytes from www.cisco.com (198.133.219.25): icmp_seq=3 ttl=241 time=25.1 ms
64 bytes from www.cisco.com (198.133.219.25): icmp_seq=4 ttl=241 time=26.1 ms
64 bytes from www.cisco.com (198.133.219.25): icmp_seq=5 ttl=241 time=25.0 ms
64 bytes from www.cisco.com (198.133.219.25): icmp_seq=6 ttl=241 time=25.8 ms
64 bytes from www.cisco.com (198.133.219.25): icmp_seq=8 ttl=241 time=25.4 ms
64 bytes from www.cisco.com (198.133.219.25): icmp_seq=9 ttl=241 time=25.1 ms
64 bytes from www.cisco.com (198.133.219.25): icmp_seq=10 ttl=241 time=26.2 ms

--- cisco.com ping statistics ---
10 packets transmitted, 9 received, 10% loss, time 9082ms
rtt min/avg/max/mdev = 25.053/25.585/26.229/0.455 ms

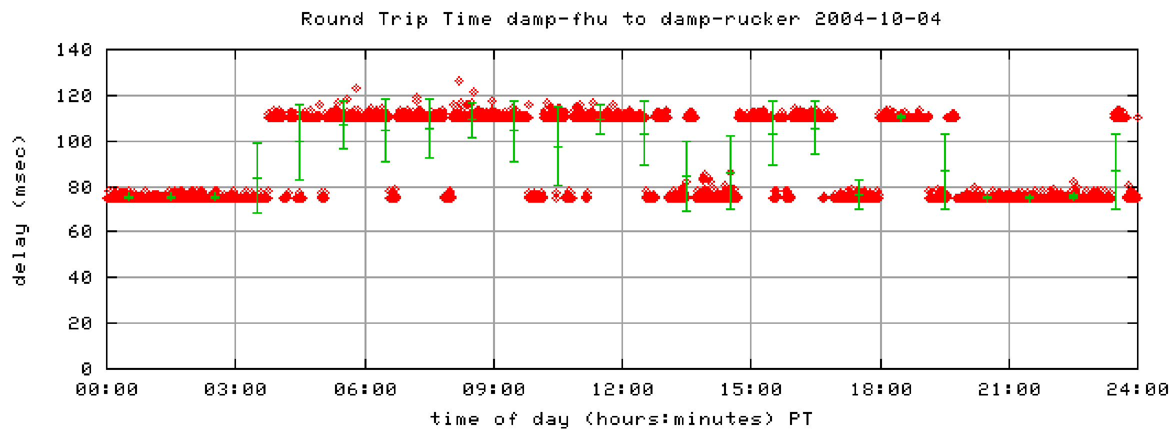
Ping Observations

- Ping packet = 20 bytes IP + 8 bytes ICMP + "user data" (first 8 bytes = timestamp)
- Default = 56 user bytes = 64 byte IP payload = 84 total bytes
- Small pings (-s 8 = 36 bytes) take less time than large pings (-s 1472 = 1500 bytes)

Ping Observations

- TTL = 241 indicates 255-241 = 14 hops
- Delay variation indicates congestion or system load
- Not good at measuring small loss
  - An HPC network should show zero ping loss
- Depends on ICMP ECHO which is sometimes blocked for “security”

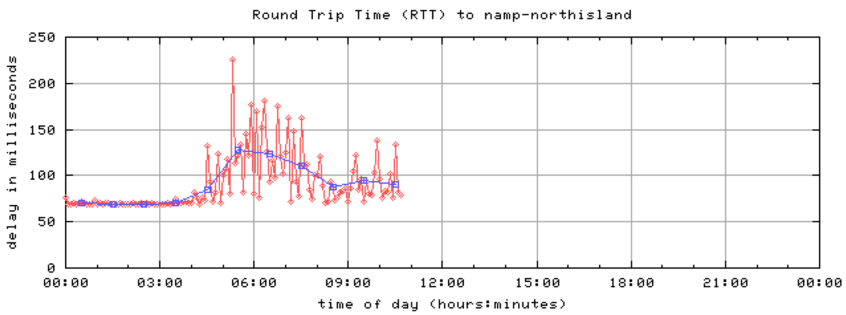
Delay Changes



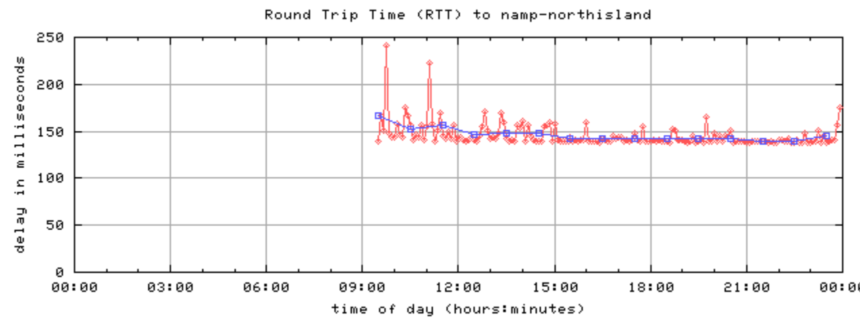
- Delay jumping between two distinct values
- Does **not** show up in traceroute as a route change
- Layer 2 problem: SONET protect failover

Delay Creep

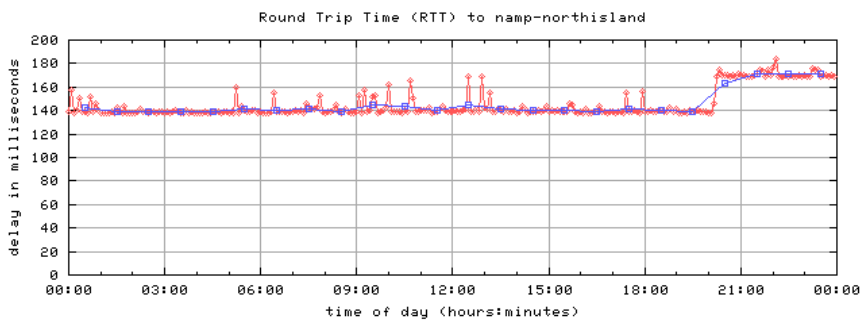
When layer 2 "heals" itself



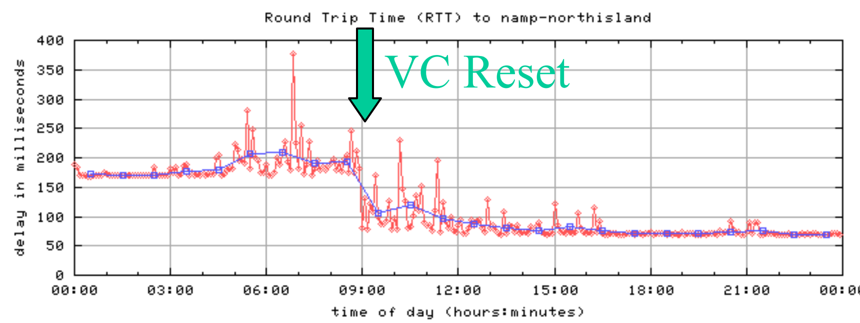
May 4th: 69 msec min



May 5th: 138 msec min



May 7th: 168 msec min



May 9th: 168 to 68 msec drop

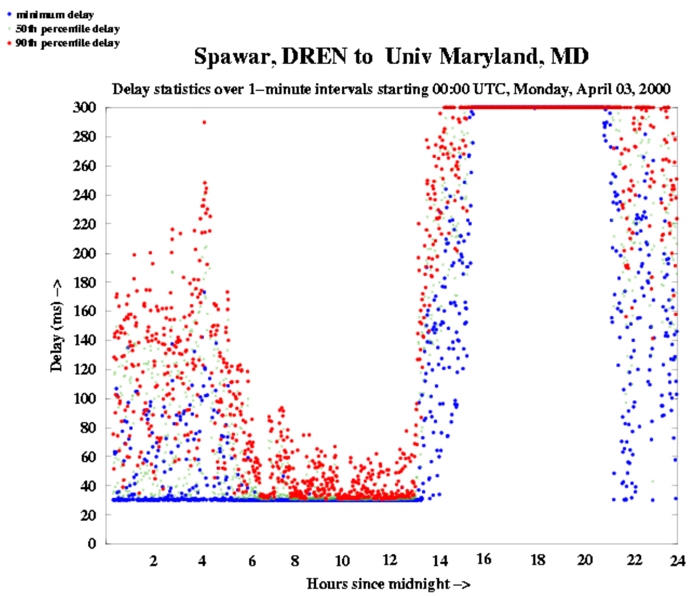
One-Way Active Measurement Protocol (OWAMP)

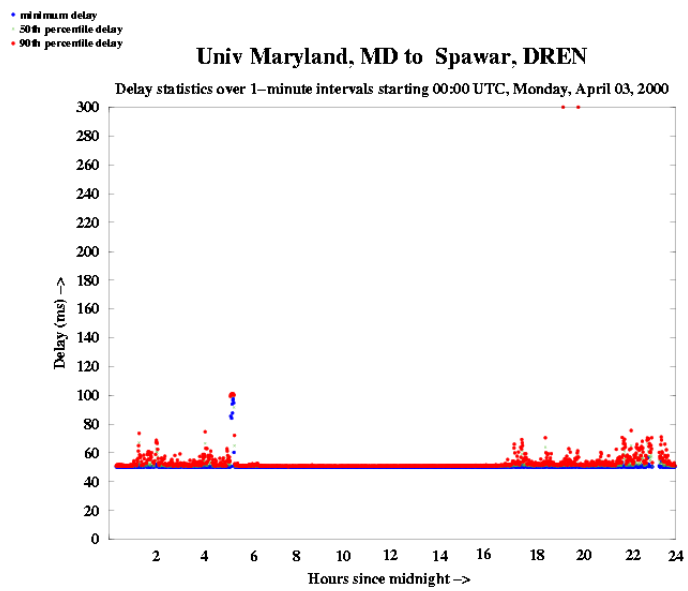
http://e2epi.internet2.edu/owamp

```
$ owping damp-arl

--- owping statistics from [sd.wareonearth.com]:33529 to [damp-arl-ge]:37545 ---
SID: 8a121503c4ab3560b8ca3e7df7886da8
100 packets transmitted, 0 packets lost (0.0% loss)
one-way delay min/median = 43.473/45.152 ms  (precision 0.0018921 s)
no reordering

--- owping statistics from [damp-arl-ge]:37546 to [sd.wareonearth.com]:33530 ---
SID: 3fc447f6c4ab3560c54b599ab1a6185e
100 packets transmitted, 0 packets lost (0.0% loss)
one-way delay min/median = 48.429/48.880 ms  (precision 0.0018921 s)
1-reordering = 4.040404%
no 2-reordering
```

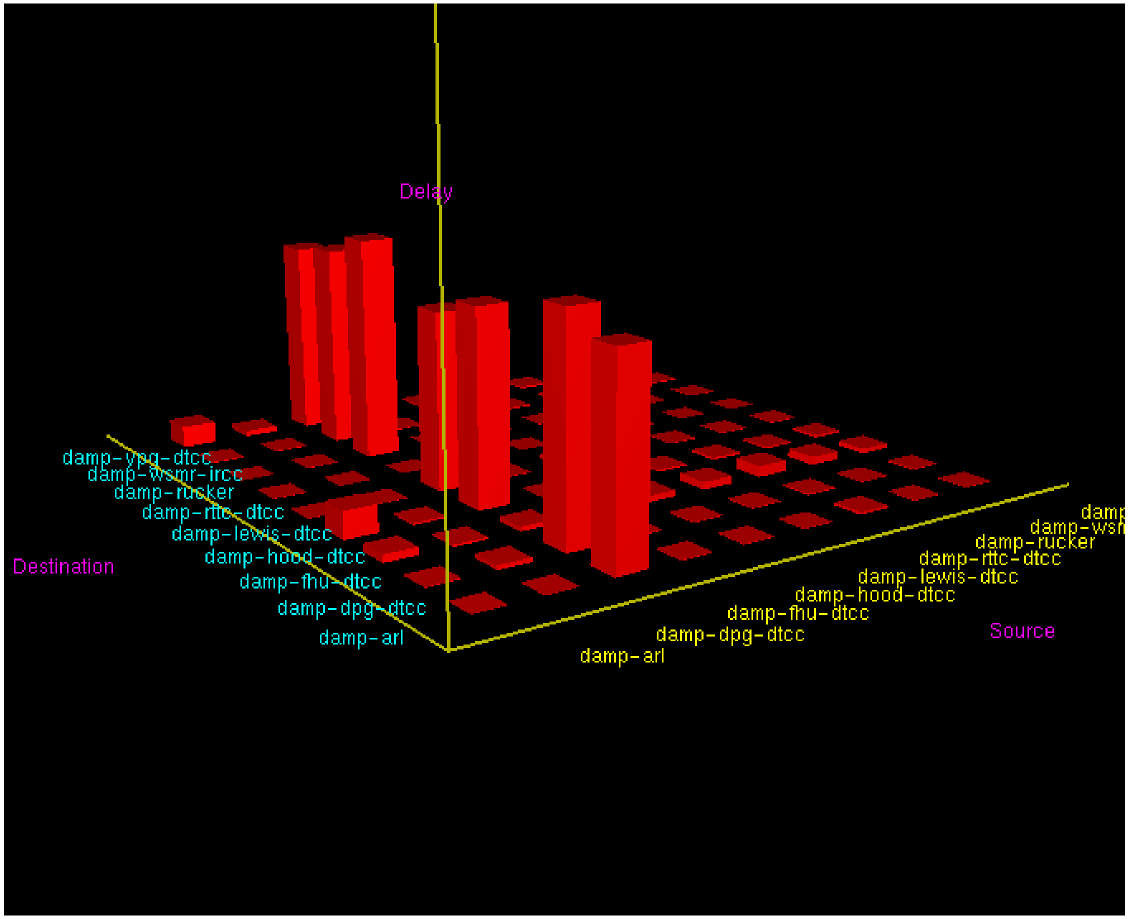
Surveyor

- Showed that one-way is interesting

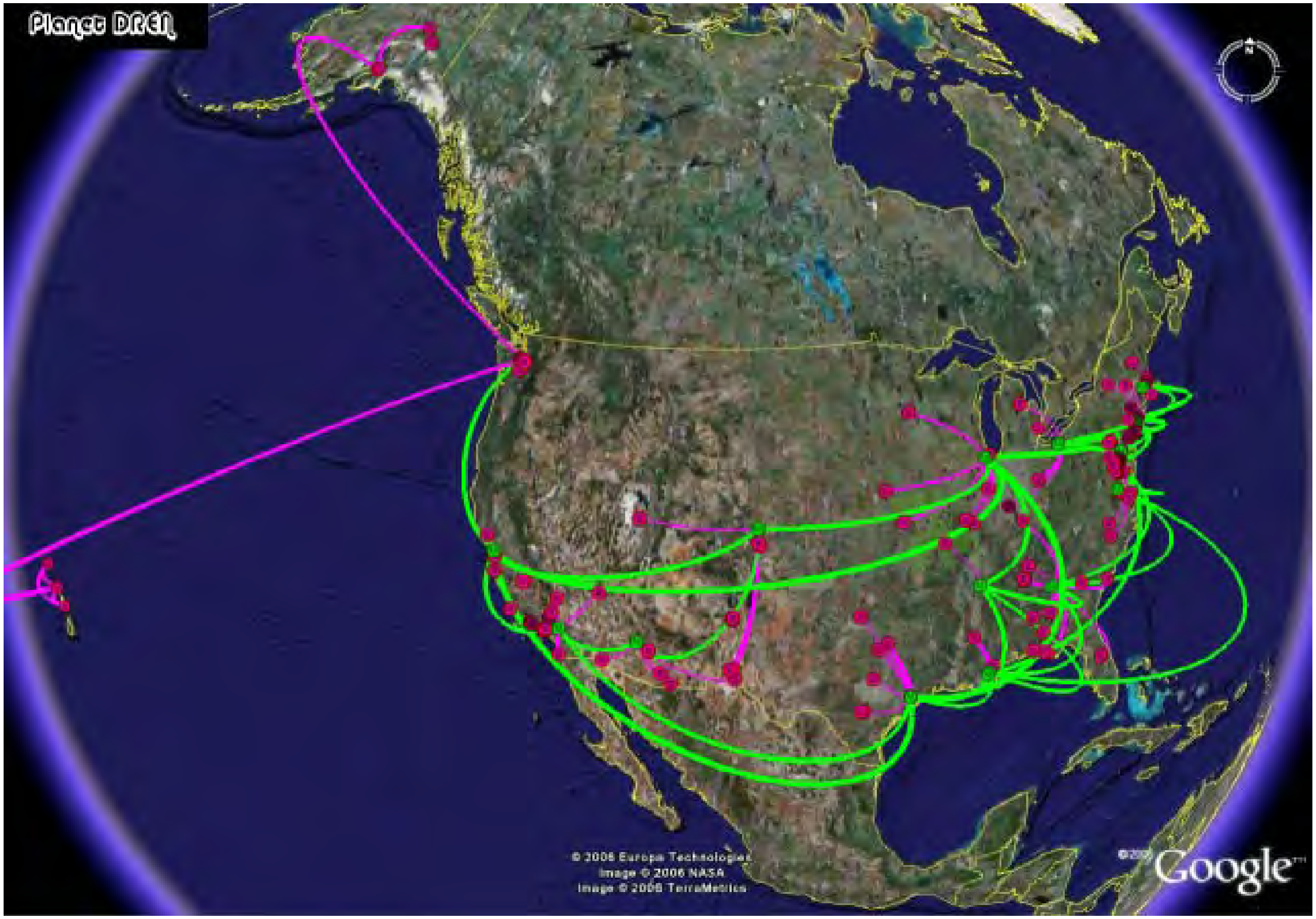




Real-Time Delay



Planet DREN



Planet DREN
© 2006 Europa Technologies
Image © 2006 NASA
Image © 2006 TerraMetrics
Google

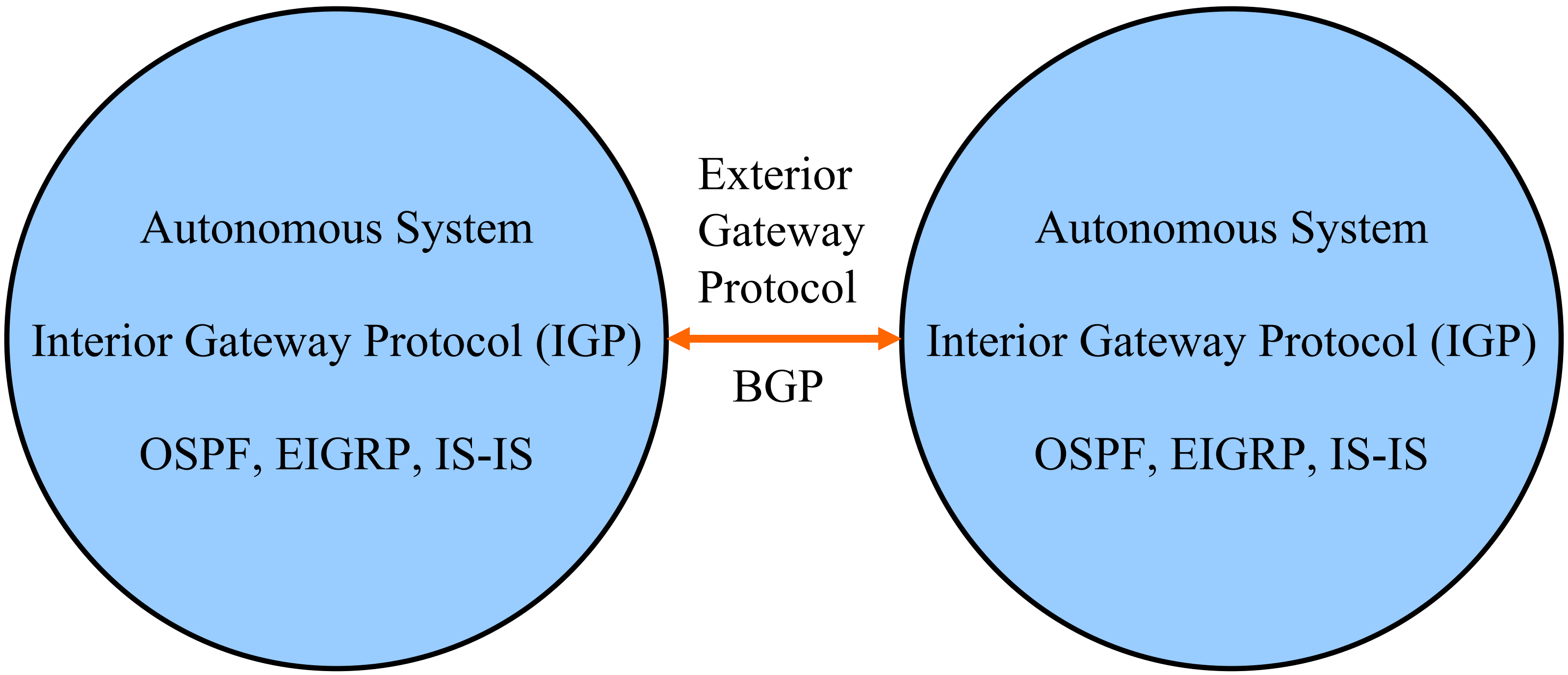
Routes

The path taken by your packets

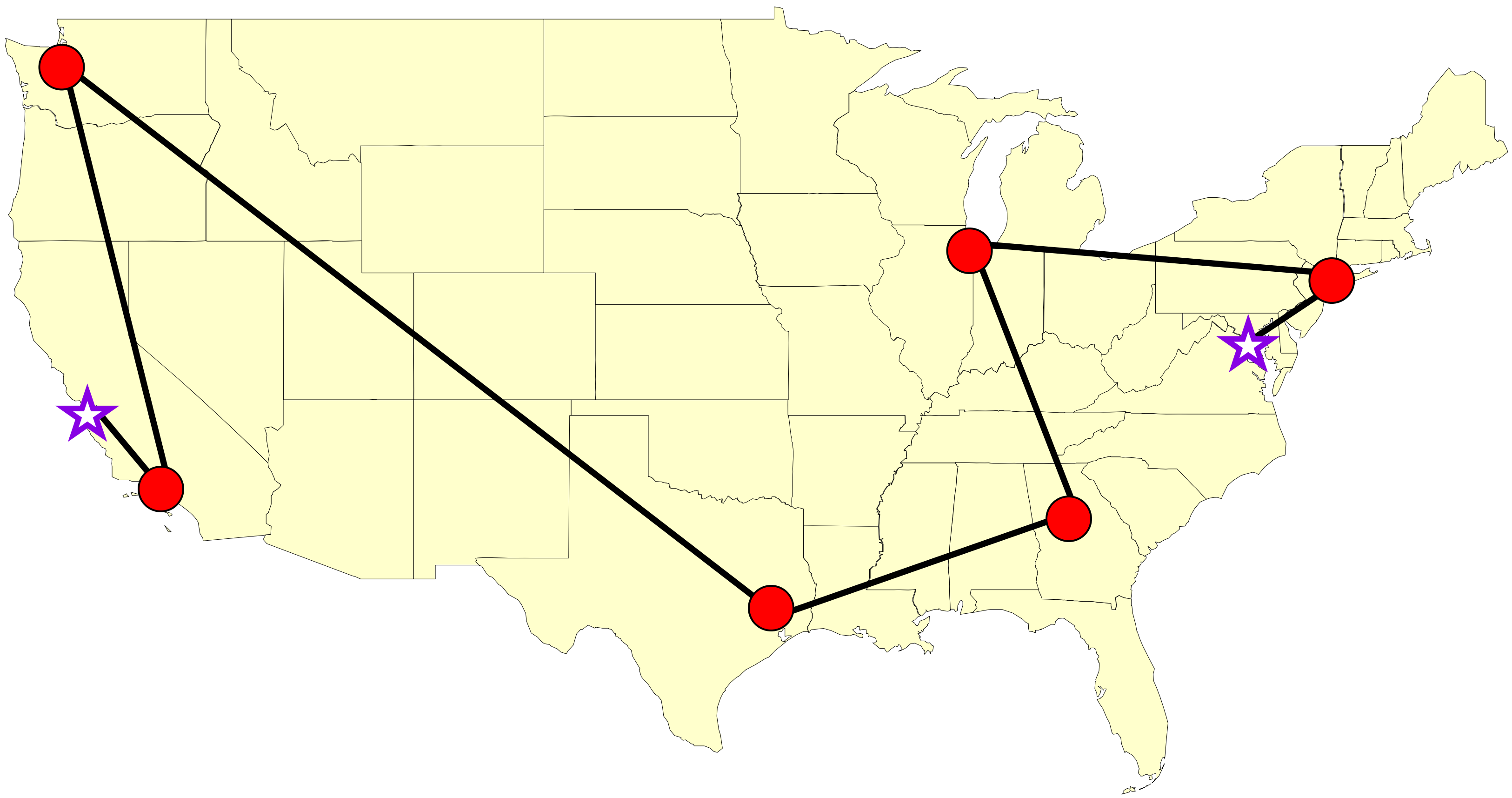
How Routers Choose Routes

- Within a network
  - Smallest number of hops
  - Highest bandwidth paths
  - Usually ignore latency and utilization
- From one network to another
  - Often “hot potato” routing, i.e. pass to the other network ASAP

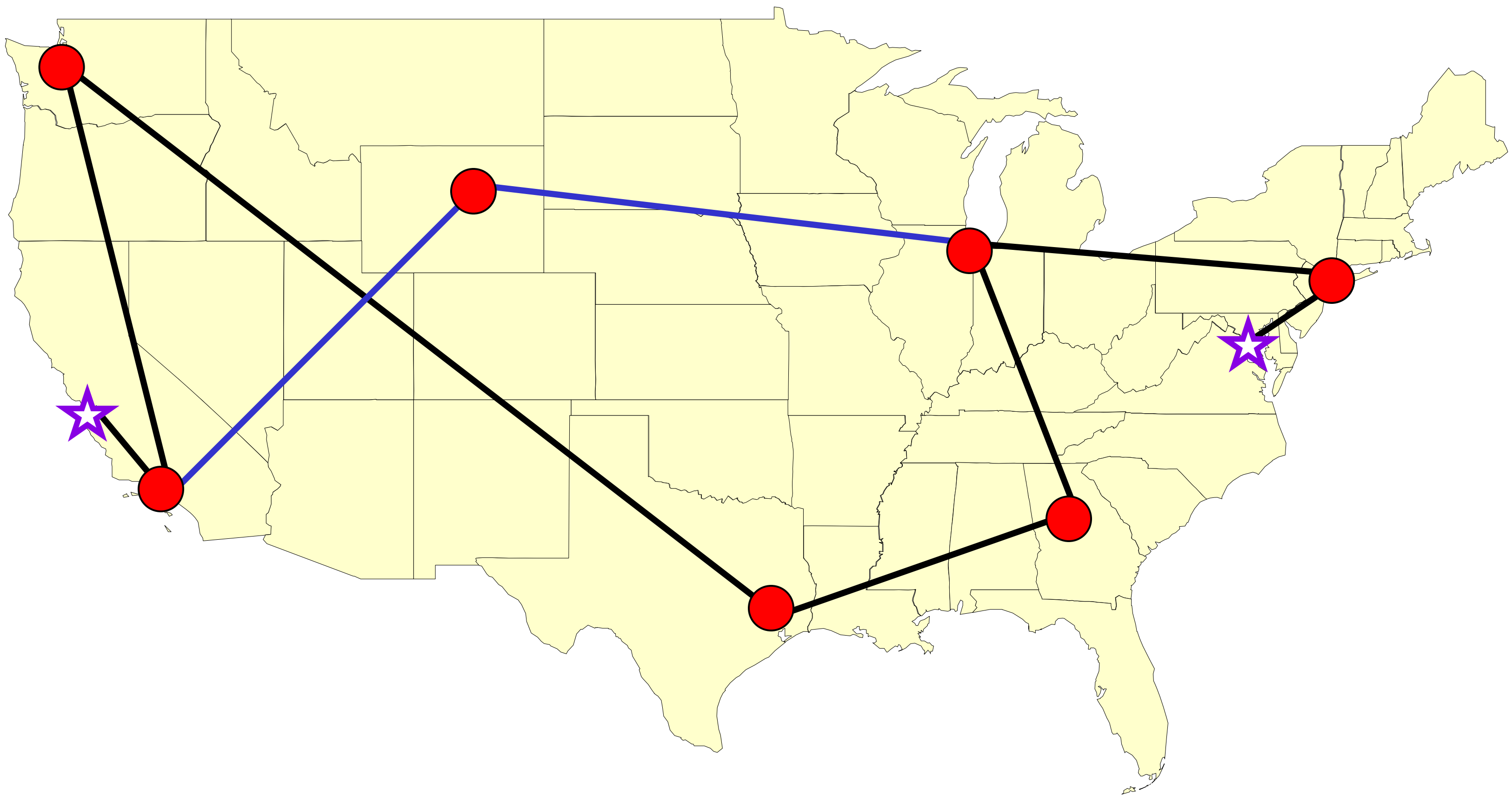
IP Routing Hierarchy

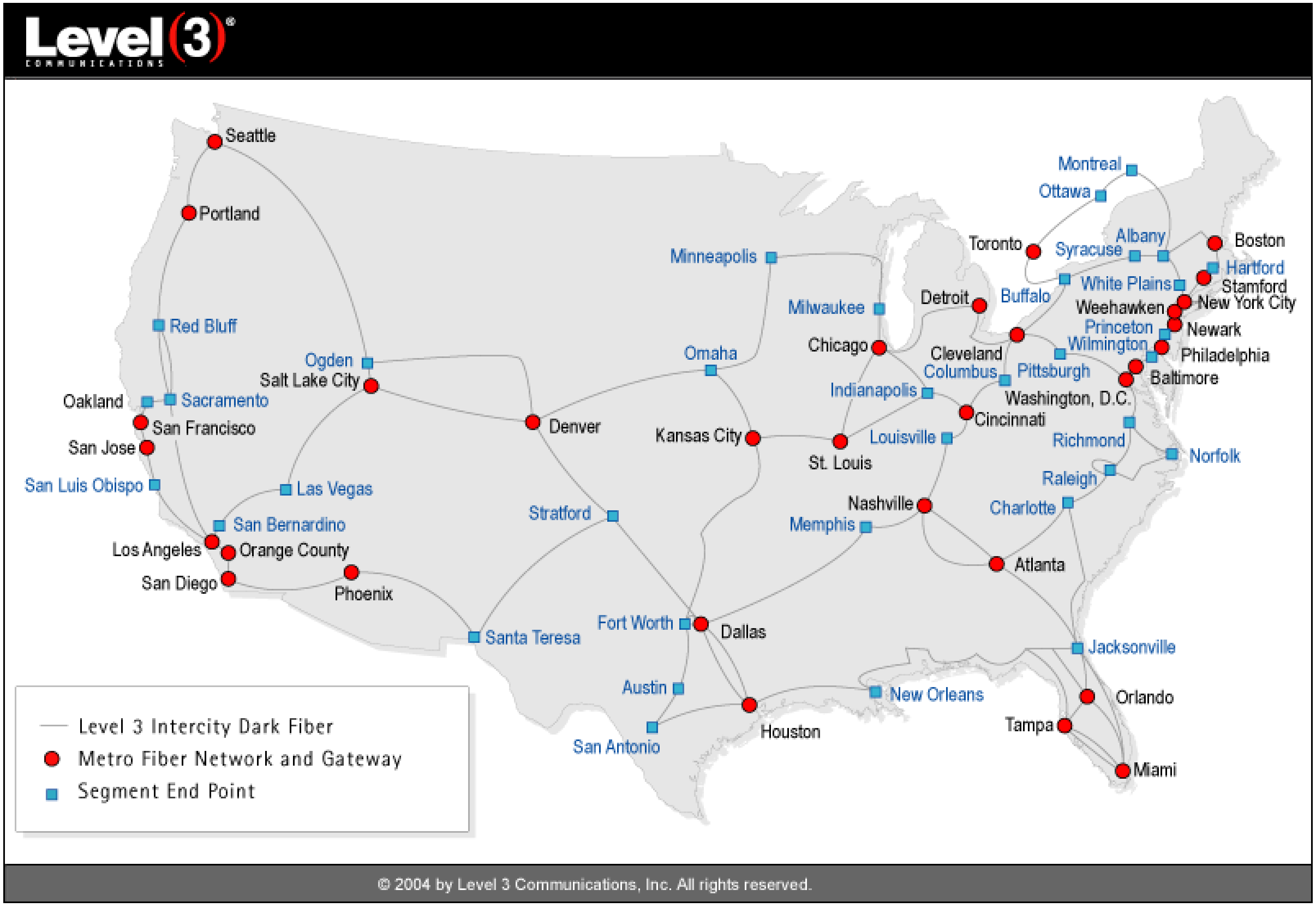


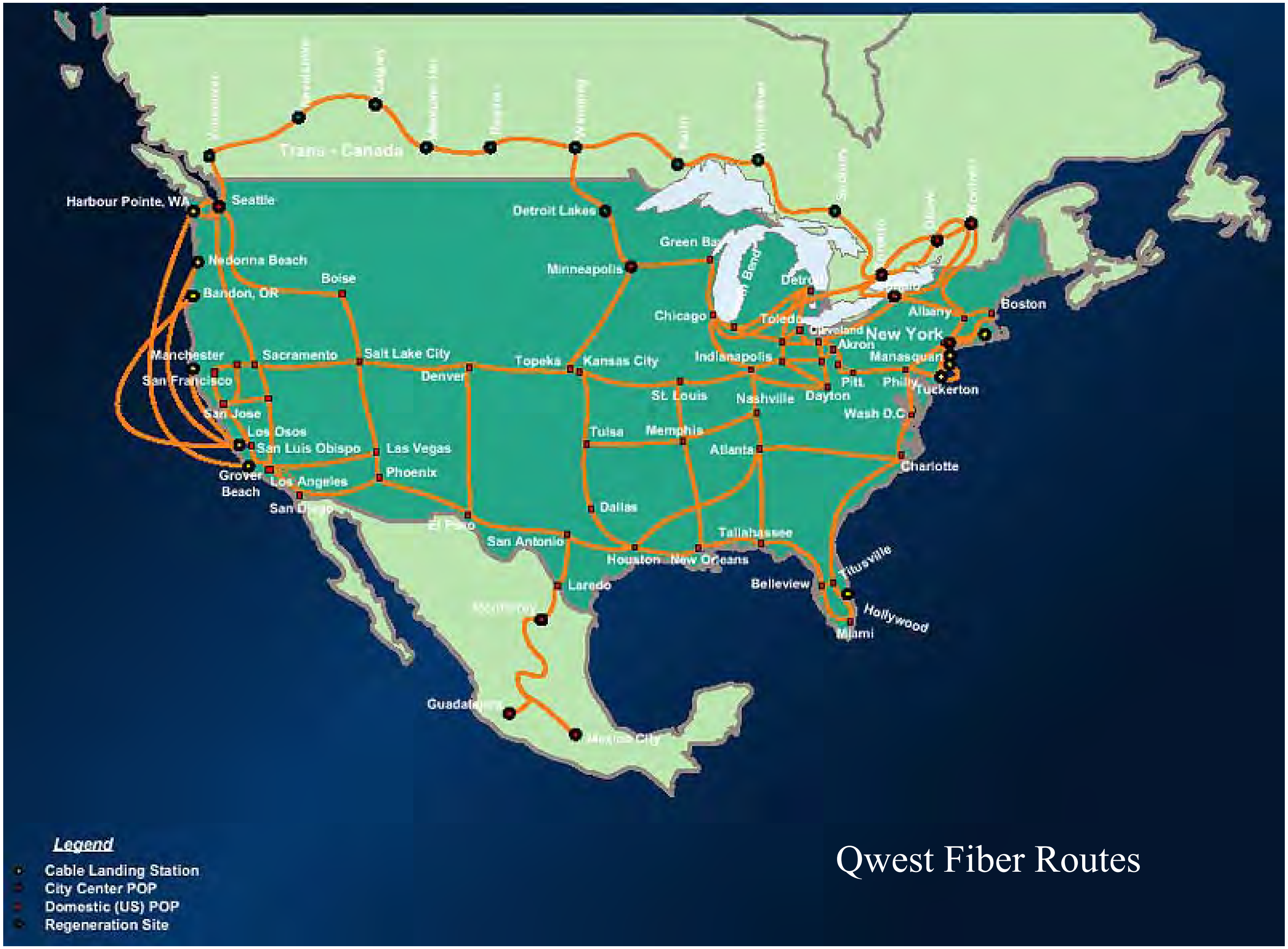
“Scenic” Routes

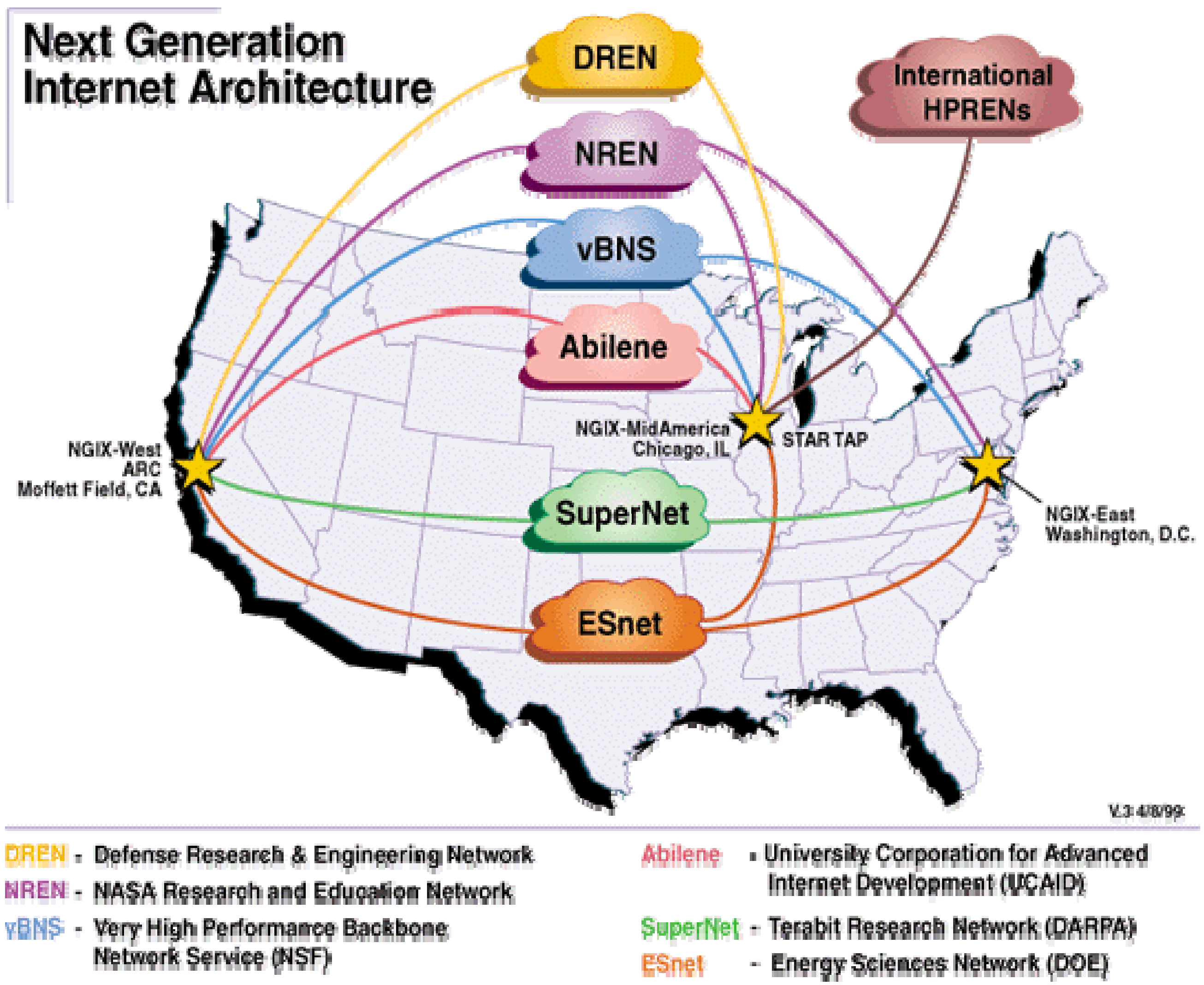


Asymmetric Routes



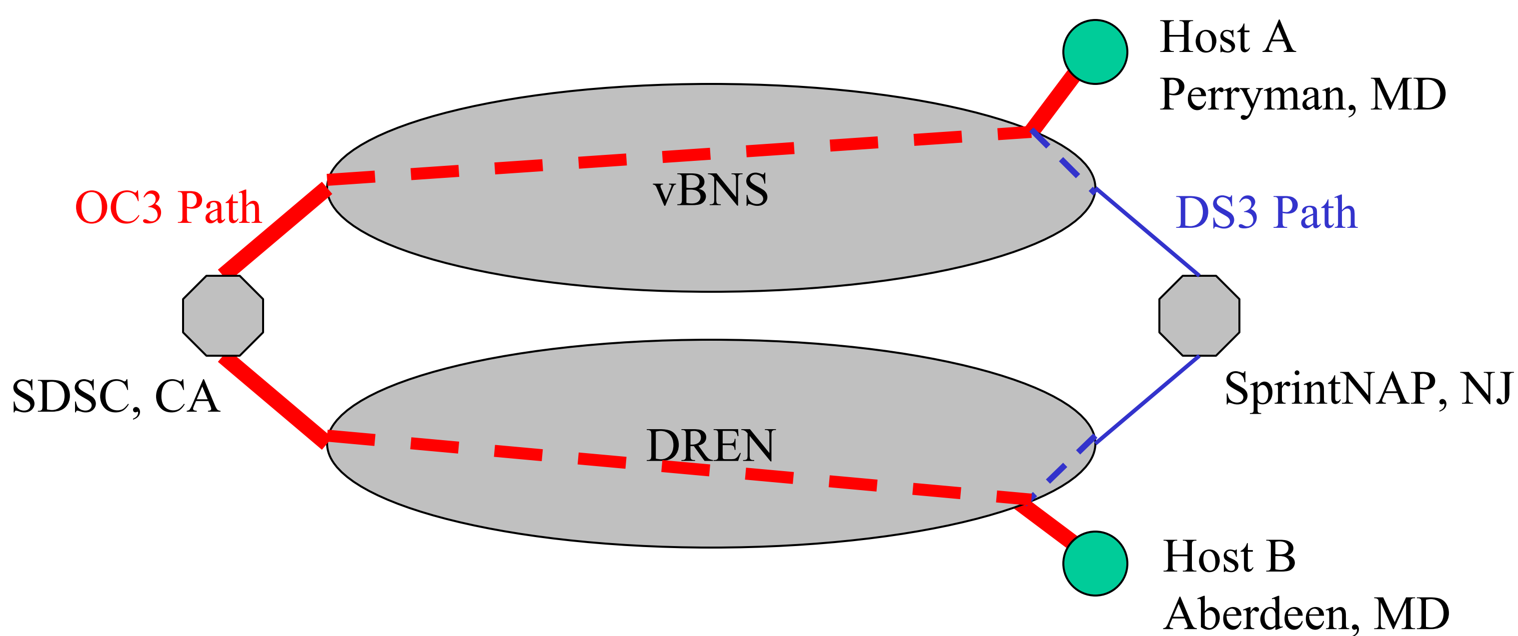

Level(3) COMMUNICATIONS
Seattle
Portland
Red Bluff
Sacramento
Oakland
San Francisco
San Jose
San Luis Obispo
Ogden
Salt Lake City
Las Vegas
San Bernardino
Los Angeles
Orange County
San Diego
Phoenix
Denver
Stratford
Santa Teresa
Minneapolis
Omaha
Kansas City
Fort Worth
Dallas
Austin
San Antonio
Houston
Milwaukee
Chicago
Indianapolis
St. Louis
Louisville
Nashville
Memphis
New Orleans
Detroit
Toronto
Montreal
Ottawa
Syracuse
Albany
Boston
Hartford
Stamford
White Plains
Buffalo
Weehawken
New York City
Newark
Princeton
Wilmington
Philadelphia
Cleveland
Columbus
Pittsburgh
Baltimore
Washington, D.C.
Cincinnati
Richmond
Norfolk
Raleigh
Charlotte
Atlanta
Jacksonville
Orlando
Tampa
Miami
Level 3 Intercity Dark Fiber
Metro Fiber Network and Gateway
Segment End Point


Trans - Canada
Harbour Pointe, WA
Seattle
Detroit Lakes
Green Bay
Minneapolis
Nedonna Beach
Boise
Bandon, OR
Chicago
Detroit
Toledo
Cleveland
New York
Albany
Boston
Akron
Manchester
Sacramento
Salt Lake City
Topeka
Kansas City
Indianapolis
Manasquan
San Francisco
Denver
St. Louis
Nashville
Dayton
Pitt.
Philly
Tuckerton
San Jose
Wash D.C.
Los Osos
San Luis Obispo
Las Vegas
Tulsa
Memphis
Atlanta
Charlotte
Grover Beach
Los Angeles
Phoenix
San Diego
Dallas
El Paso
San Antonio
Tallahassee
Houston
New Orleans
Titusville
Laredo
Belleview
Hollywood
Miami
Guadalajara
Mexico City
Legend
Cable Landing Station
City Center POP
Domestic (US) POP
Regeneration Site
Qwest Fiber Routes


Next Generation
Internet Architecture
DREN
NREN
vBNS
Abilene
SuperNet
ESnet
International
HPRENs
NGIX-West
ARC
Moffett Field, CA
NGIX-MidAmerica
Chicago, IL
STAR TAP
NGIX-East
Washington, D.C.
V.3 4/8/99
DREN - Defense Research & Engineering Network
NREN - NASA Research and Education Network
vBNS - Very High Performance Backbone Network Service (NSF)
Abilene - University Corporation for Advanced Internet Development (UCAID)
SuperNet - Terabit Research Network (DARPA)
ESnet - Energy Sciences Network (DOE)

Path Performance: Latency vs. Bandwidth

The highest bandwidth path is not always the highest throughput path!



- Host A&B are 15 miles apart
- DS3 path is ~250 miles
- OC3 path is ~6000 miles

The network chose the OC3 path with 24x the rtt, 80x BDP

A Modern Traceroute

- ftp://ftp.login.com/pub/software/traceroute/
- Supports
  - MTU discovery (-M)
  - Report ASNs (-A)
  - Registered owners (-O)
  - Set IP TOS field (-t)
  - Microsecond timestamps (-u)
  - Set IP protocol (-I)

Traceroute Example

[phil@damp-ssc phil]$ **traceroute -A mit.edu**
traceroute to mit.edu (18.7.22.69), 64 hops max, 40 byte packets
1 ge-0-1-0.sandiego.dren.net (138.18.190.1) [AS668] 0 ms 0 ms 0 ms
2 so-0-0-0.ngixeast.dren.net (138.18.1.55) [AS668] 76 ms 75 ms 76 ms
3 Abilene-peer.ngixeast.dren.net (138.18.47.34) [AS668] 76 ms 77 ms 76 ms
4 nycmng-washng.abilene.ucaid.edu (198.32.8.84) [<NONE>] 80 ms 88 ms 80 ms
5 ATM10-420-OC12-GIGAPOPNE.nox.org (192.5.89.9) [<NONE>] 87 ms 86 ms 85 ms
6 192.5.89.90 (192.5.89.90) [<NONE>] 85 ms 86 ms 104 ms
7 W92-RTR-1-BACKBONE.MIT.EDU (18.168.0.25) [AS3] 85 ms 85 ms 85 ms
8 WEB.MIT.EDU (18.7.22.69) [AS3] 86 ms 86 ms 85 ms

- DREN (AS668) to Abilene to MIT (AS3)
- ASN “NONE” results from private/unadvertised address space
- Hop 1 to 2 was over an (invisible) MPLS path

How Traceroute Works

- Sends UDP packets to ports (-p) 33434 and up, TTL of 1 to 30
- Each router hop decrements the TTL
- If the TTL=0, that node returns an ICMP TTL Expired
- The destination host returns an ICMP Port Unreachable
- http://www.caida.org/publications/animations/

Traceroute Observations

- Shows the **return** interface addresses of the **forwarding** path
- You can’t see hops through switches or over tunnels (e.g. ATM VC’s, GRE, MPLS)
- The required ICMP replies are sometimes blocked for “security”, or not generated, or sent without resetting the TTL

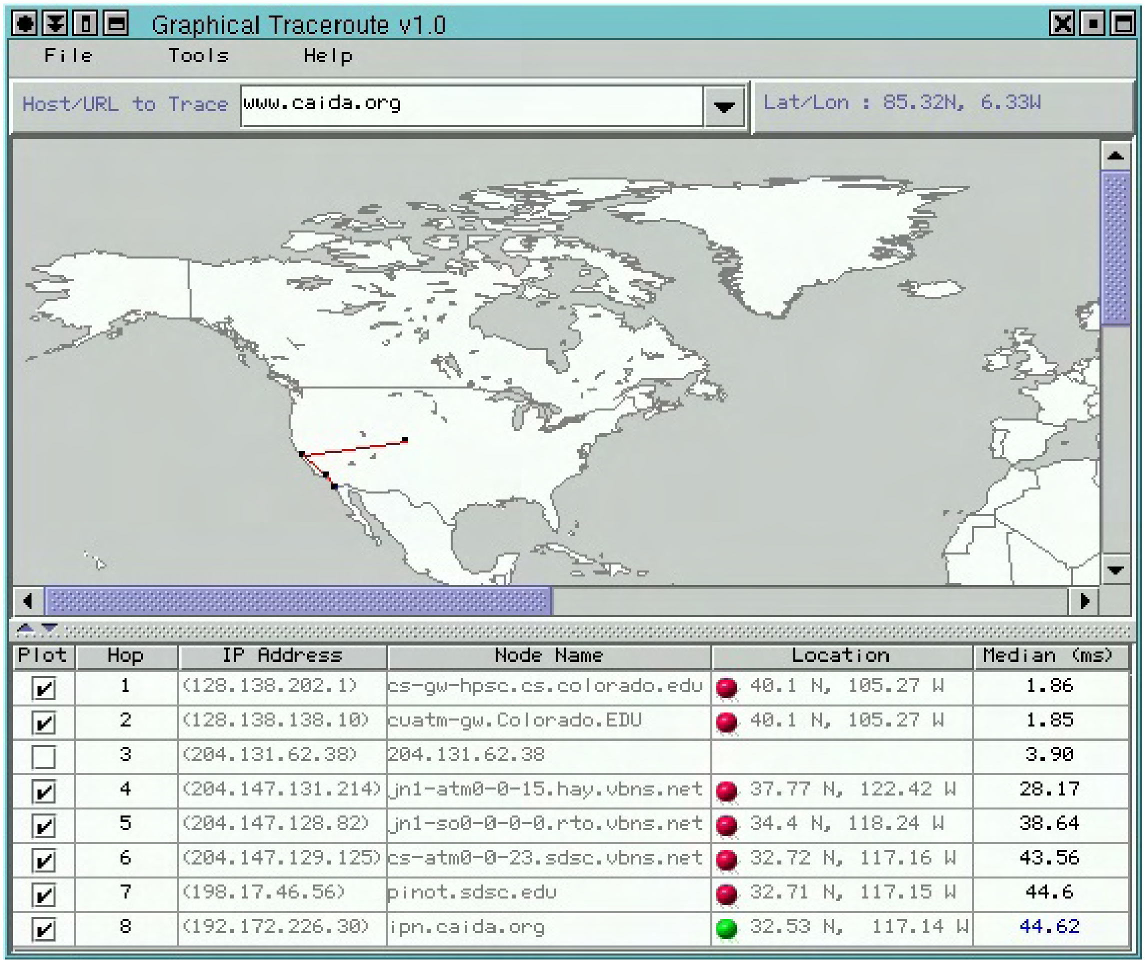
Matt's Traceroute

www.bitwizard.nl/mtr/

```
                         Matt's traceroute  [v0.41]
damp-ssc.spawar.navy.mil                              Sun Apr 23 23:29:51 2000
Keys:  D - Display mode    R - Restart statistics    Q - Quit
                                           Packets               Pings
Hostname                                %Loss  Rcv  Snt  Last Best  Avg  Worst
 1. taco2-fe0.nci.net                      0%   24   24     0    0    0      1
 2. nccosc-bgp.att-disc.net                0%   24   24     1    1    1      6
 3. pennsbr-aip.att-disc.net               0%   24   24    84   84   84     86
 4. sprint-nap.vbns.net                    0%   24   24    84   84   84     86
 5. cs-hssi1-0.pym.vbns.net                0%   23   24    89   88  152    407
 6. jn1-at1-0-0-0.pym.vbns.net             0%   23   23    88   88   88     90
 7. jn1-at1-0-0-13.nor.vbns.net            0%   23   23    88   88   88     90
 8. jn1-so5-0-0-0.dng.vbns.net             0%   23   23    89   88   91    116
 9. jn1-so5-0-0-0.dnj.vbns.net             0%   23   23   112  111  112    113
10. jn1-so4-0-0-0.hay.vbns.net             0%   23   23   135  134  135    135
11. jn1-so0-0-0-0.rto.vbns.net             0%   23   23   147  147  147    147
12. 192.12.207.22                          5%   22   23    98   98  113    291
13. pinot.sdsc.edu                         0%   23   23   152  152  152    156
14. ipn.caida.org                          0%   23   23   152  152  152    160
```

GTrace – Graphical Traceroute

www.caida.org/tools/visualization/gtrace/



tcptraceroute

- http://michael.toren.net/code/tcptraceroute/
- Useful when ICMP is blocked
  - But still depends on TTL expired replies
- Can select the TCP port number
  - Helps locate firewall or ACL blocks
  - Defaults to port 80 (http)

Routing vs. Switching

- IP routing requires a longest prefix match
  - Harder than switching, but now wire speed
  - Inspired IP switching / MPLS
- Switches have gained features
  - Some even route
- Simplicity is good
  - Used switched infrastructure where you can, routers where you need better separation or control

Multi-Protocol Label Switching (MPLS)

- Adds switched (layer 2) paths below IP
  - Useful for traffic engineering, VPN's, QoS control, high speed switching
- IP packets get wrapped in MPLS frames and "labeled"
- MPLS routers switch the packets along Label Switched Paths (LSP's)
- Being generalized for optical switching (GMPLS)

Packet Sizes

MTU

Maximum Transmission Unit (MTU)

- Maximum packet size that can be sent as one unit (no fragmentation)
- Usually "IP MTU" which is the largest IP datagram including the IP header
  - Shown with **ifconfig** on Unix/Linux
- Sometimes a layer 2 MTU
  - IP MTU 1500 = Ethernet MTU 1518 (or 1514, or 1522)

IP MTU Sizes

	Default	Maximum	Reference
IPv4	576	64K	
IPv6	1280	64K – 4GB	RFC 2460
HSSI	4470		
IP on FDDI	4352	4500	RFC 1188
IP on ATM	9180	64K	RFC 2225
IP on FC	65280	64K – 4GB	RFC 2652
POS	4470/9180	64K - Inf	RFC 2615

Ethernet Jumbo Frames

- Non-standard Ethernet extension
  - Usually 9000 bytes (IP MTU)
  - Sometimes anything over 1500
- Usually only used with 1GigE and above
- Requires separate LANs or VLANs to accommodate non-jumbo equipment
- http://sd.wareonearth.com/~phil/jumbo.html

Reasons for Jumbo

- Reduce system and network overhead
- Handle 8KB NFS or SAN packets
- Improve TCP performance!
  - Greater throughput
  - Greater loss tolerance
  - **Reduced** router queue loads

Vendor Jumbo Support

- MTU 9000 interoperability demonstrated:
  - Allied Telesyn, Avaya, Cisco, Extreme, Foundry, NEC, Nortel
- Juniper = 9178 (9192 – 14)
- Cisco (5000/6000) = 9216
- Foundry JetCore 1 & 10 GbE = 14000
- NetGear GA621 = 16728
- http://darkwing.uoregon.edu/~joe/jumbo-clean-gear.html

Path MTU (PMTU)

- Minimum MTU of all hops in a path
- Hosts can do Path MTU Discovery to find it
  - Depends on ICMP replies
- Without PMTU Discovery hosts should assume PMTU is only 576 bytes
  - Some hosts falsely assume 1500!

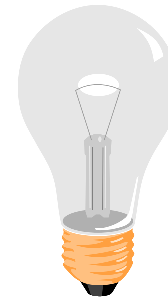
Check the MTU

```
[phil@damp-mhpcc phil]$ tracepath damp-rome
 1?: [LOCALHOST]     pmtu 9000
 1:  ge-0_1_0.mhpcc.dren.net (138.18.203.1)           asymm  2   1.144ms
 2:  so-1_0_0.uhm.dren.net (138.18.4.49)              asymm  3   3.225ms
 3:  so-0_2_0.seattle-m20.dren.net (138.18.4.52)      asymm  4  69.395ms
 4:  ge-0_1_0.seattle.dren.net (138.18.194.33)        asymm  5  69.597ms
 5:  t3-0_0_0.rome.dren.net (138.18.1.11)             asymm  6 163.504ms
 6:  t3-0_0_0.rome.dren.net (138.18.1.11)             162.976ms pmtu 1500
 7:  damp-rome-fe (138.18.25.6)                       asymm  6 158.981ms reached
     Resume: pmtu 1500 hops 7 back 6
```

Using Ping to Check the MTU

damp-navo$ **ping -s 8000 -d -v -M do damp-asc2**
PING damp-asc2-ge (138.18.22.5) from 204.222.177.220 : 8000(8028) bytes of data.
From so-0_0_0.wpafb.dren.net (138.18.1.5) icmp_seq=1 **Frag needed and DF set (mtu = 4352)**
8008 bytes from damp-asc2-ge (138.18.22.5): icmp_seq=1 ttl=60 time=47.6 ms
ping: local error: Message too long, **mtu=4352**

Things You Can Do

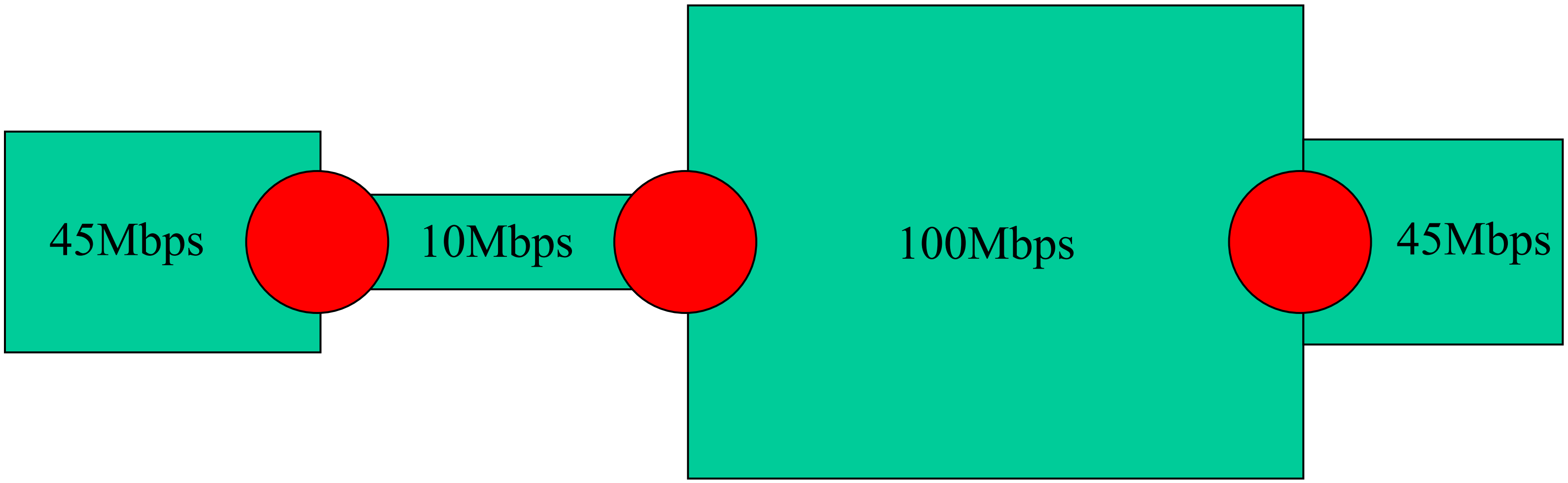


- Use only large MTU interfaces/routers/links
  - Gigabit Ethernet with **Jumbo Frames** (9000)
  - Packet over SONET (POS) (4470, 9000+)
  - ATM CLIP (9180)
- Never reduce the MTU (or bandwidth) on the path between each/every host and the WAN
- Make sure your TCP uses Path MTU Discovery

Bandwidth

and throughput

Hops of Different Bandwidth



- The “Narrow Link” has the lowest bandwidth
- The “Tight Link” has the least **Available** bandwidth
- Queues can form wherever available bandwidth decreases
- A queue buildup is most likely in front of the Tight Link

Throughput Limit

- throughput <= **available** bandwidth

  ("tight link" with the minimum unused bandwidth)

  – A high performance network should be lightly loaded (<50%)
  – *A loaded high speed network is no better to the end user than a lightly loaded slow one*

Bandwidth Determination

- Ask the routers/switches
  - SNMP
  - RMON / NetFlow / sFlow
- Passive Measurement
  - tcpdump
  - OCxmon
- Active Measurement
  - Light weight tests
  - Full bandwidth tests

SNMP

- Simple Network Monitoring Protocol
  - Version 1: RFC 1155-1157 (1990)
  - Version 2c: RFC 1901-1908 (1996)
  - Version 3: RFC 2571-2574 (1999)
- Several Version 2 variations
  - “2c” added community strings
- Router / Switch statistics and control

Net-SNMP

- Net-SNMP – Open Source project
  - Was UCD-SNMP
- Library and utilities
- http://www.net-snmp.org/

SNMP MIBs

- Management Information Base (MIB)
- Defines variables that can be read / set
- Many MIBs exist, both standard and vendor specific
- Tree structure hierarchy
- Numerical strings (OID’s) with text representations

Example MIB Variables

$ snmptranslate -Of SNMPv2-MIB::sysUpTime.0

.iso.org.dod.internet.mgmt.mib-2.system.sysUpTime.0

$ snmptranslate -On SNMPv2-MIB::sysUpTime.0

.1.3.6.1.2.1.1.3.0

$ snmptranslate -On IF-MIB::ifHCInOctets.32

.1.3.6.1.2.1.31.1.1.1.6.32

$ snmptranslate -On IF-MIB::ifHCOutOctets.32

.1.3.6.1.2.1.31.1.1.1.10.32

Net-SNMP Example

$ **snmpget -v 2c -c *public* router.dren.net system.sysUpTime.0**

SNMPv2-MIB::sysUpTime.0 = Time ticks: (1407699551) 162 days, 22:16:35.51

$ **snmpwalk -v 2c -c *public* router.dren.net IF-MIB::ifXTable**

IF-MIB::ifName.1 = STRING: fxp0

IF-MIB::ifName.2 = STRING: fxp1

…

IF-MIB::ifHCInOctets.2 = Counter64: 4049716647

IF-MIB::ifHCOutOctets.2 = Counter64: 3264374754

IF-MIB::ifHighSpeed.2 = Gauge32: 100

IF-MIB::ifAlias.2 = STRING: Router 1 LAN

…

SNMP Counter Wrap

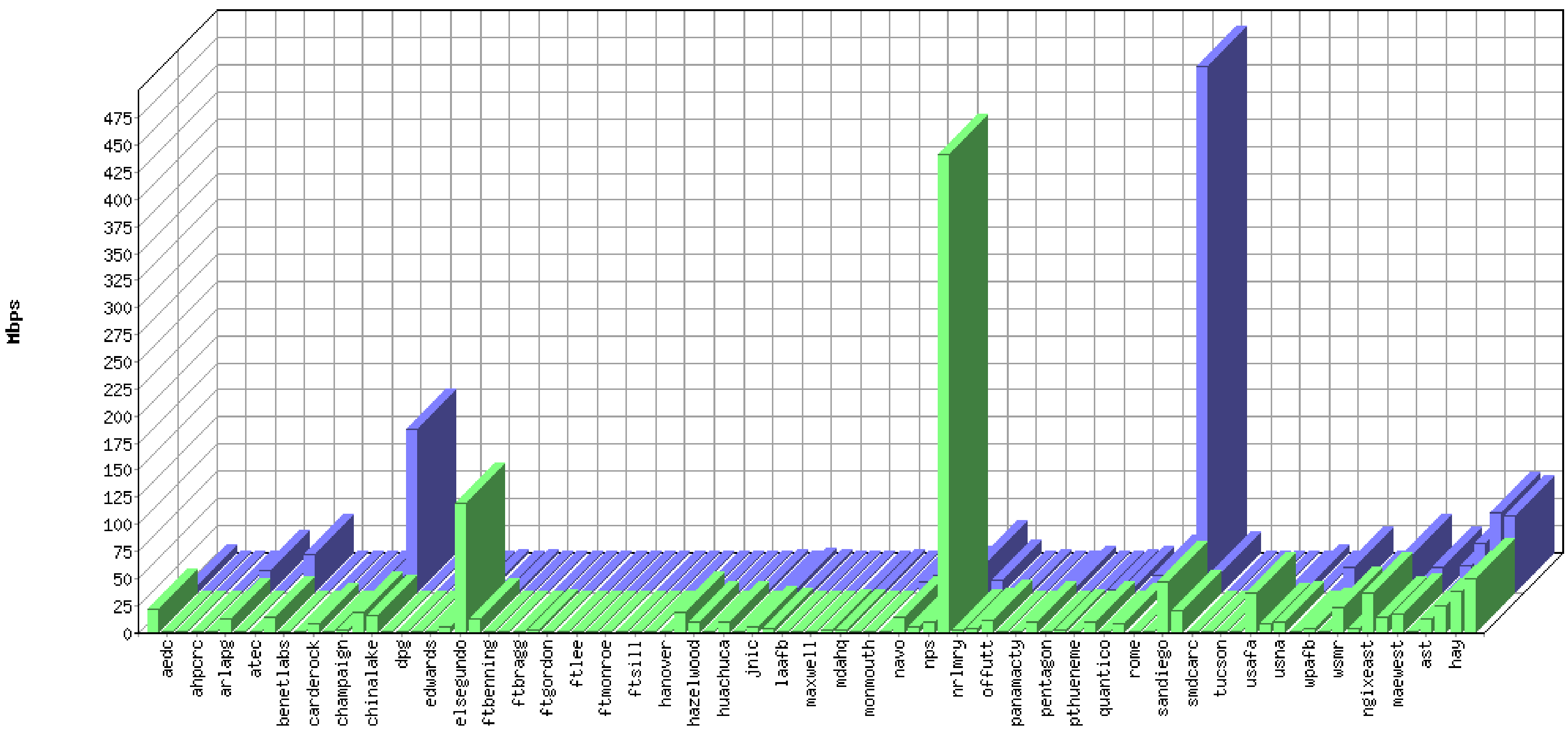
Counter Bits	Data Rate	Wrap Time
32	10 Mbps	57 min
32	100 Mbps	5.7 min
32	1 Gbps	34 sec
32	10 Gbps	3.4 sec
64	10 Gbps	468 years

SNMP Resolution

- Juniper caches interface statistics for five seconds
- Reads provide new data only if cache > 5 seconds old
- No timestamps on the data

Real-Time DREN Traffic

Five second snapshots of DREN traffic



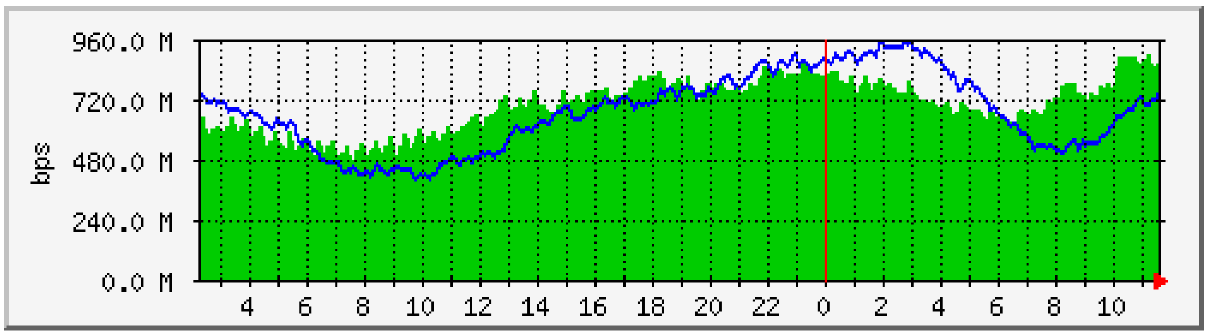
MRTG

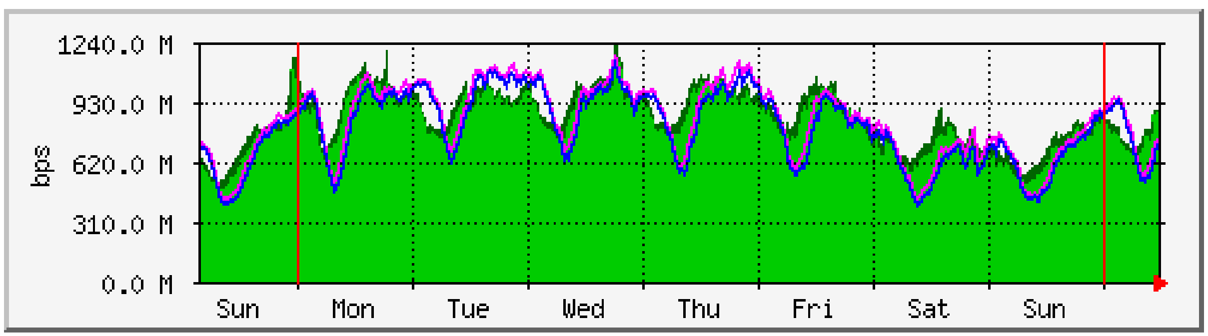
MULTI ROUTER TRAFFIC GRAPHER

- www.mrtg.org
- Extremely popular network monitoring tool
- Most common display:
  - Five minute average link utilizations
  - Green into interface
  - Blue out of interface

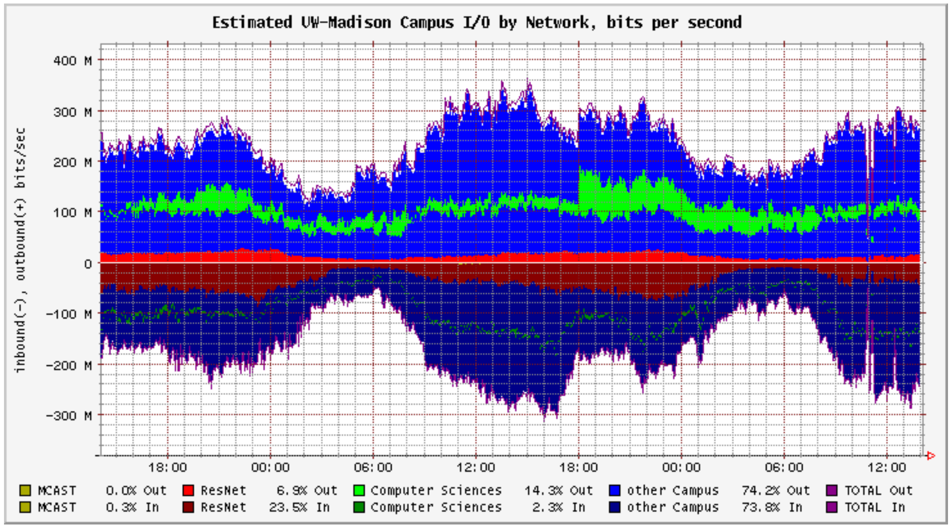
MRTG Example

Abilene, Indianapolis to Kansas City, OC48 link, 7 October 2002



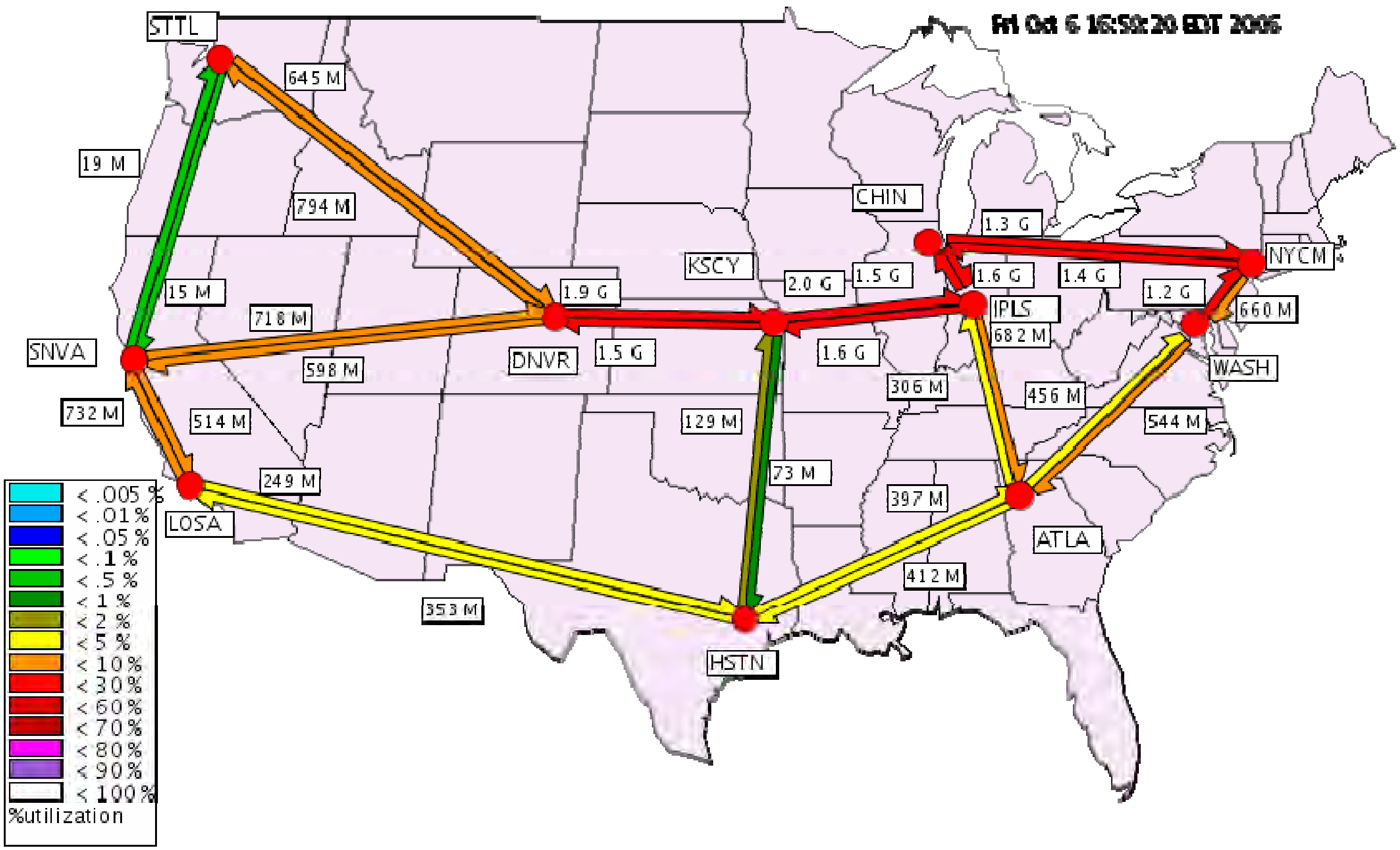


RRDtool



- Round Robin Database Tool
- Reimplementation of MRTG's backend
  - Generalized data storage, retrieval, graphing
  - Many front ends available, including MRTG
- www.rrdtool.org

Abilene Weather Map



http://weathermap.grnoc.iu.edu/abilene_jpg.html

Is SNMP Enough?

- SNMP/MRTG shows you how **much** traffic
- NetFlow and sFlow show you the **kinds** of traffic and **where** it is coming from and going to

What is a Flow?

- A collection of related packets
- Based on selection criteria (the key), examples:
  - Src/dst address, prefix, or ASN
  - Src/dst port
  - Protocol
  - Type of Service (TOS)
  - Input Interface
- Can be bi-directional or unidirectional

Tcpdump example

```
Host100$ tcpdump -w dump.out -s 68 host 192.168.0.200
 tcpdump: listening on eth1
```

```
Host100$ ssh 192.168.0.200 w
  7:54pm  up 78 days,  7:30,  1 user,  load average: 0.01, 0.04, 0.01
USER     TTY      FROM              LOGIN@   IDLE   JCPU   PCPU  WHAT
phil     :0       -                25Sep05   ?     0.00s   ?     -
```

```
 36 packets received by filter
 0 packets dropped by kernel
```

Tcpdump output

```
$ tcpdump -n -q -r dump.out
18:37:33.148979 192.168.0.100.60005 > 192.168.0.200.ssh: tcp 0 (DF) S
18:37:33.149309 192.168.0.200.ssh > 192.168.0.100.60005: tcp 0 (DF) S ack
18:37:33.149331 192.168.0.100.60005 > 192.168.0.200.ssh: tcp 0 (DF) ack
18:37:33.150783 192.168.0.200.ssh > 192.168.0.100.60005: tcp 22 (DF)
18:37:33.150959 192.168.0.100.60005 > 192.168.0.200.ssh: tcp 0 (DF)
18:37:33.151020 192.168.0.100.60005 > 192.168.0.200.ssh: tcp 22 (DF)
18:37:33.151384 192.168.0.200.ssh > 192.168.0.100.60005: tcp 0 (DF)
18:37:33.151397 192.168.0.100.60005 > 192.168.0.200.ssh: tcp 536 (DF)
18:37:33.151738 192.168.0.200.ssh > 192.168.0.100.60005: tcp 0 (DF)
18:37:33.152327 192.168.0.200.ssh > 192.168.0.100.60005: tcp 544 (DF)
18:37:33.152700 192.168.0.100.60005 > 192.168.0.200.ssh: tcp 24 (DF)
18:37:33.155258 192.168.0.200.ssh > 192.168.0.100.60005: tcp 424 (DF)
18:37:33.177305 192.168.0.100.60005 > 192.168.0.200.ssh: tcp 416 (DF)
18:37:33.208861 192.168.0.200.ssh > 192.168.0.100.60005: tcp 736 (DF)
18:37:33.238156 192.168.0.100.60005 > 192.168.0.200.ssh: tcp 16 (DF)
18:37:33.277504 192.168.0.200.ssh > 192.168.0.100.60005: tcp 0 (DF)
18:37:33.277536 192.168.0.100.60005 > 192.168.0.200.ssh: tcp 48 (DF)
18:37:33.277831 192.168.0.200.ssh > 192.168.0.100.60005: tcp 0 (DF)
18:37:33.277832 192.168.0.200.ssh > 192.168.0.100.60005: tcp 48 (DF)
18:37:33.277918 192.168.0.100.60005 > 192.168.0.200.ssh: tcp 64 (DF)
18:37:33.278300 192.168.0.200.ssh > 192.168.0.100.60005: tcp 80 (DF)
18:37:33.278344 192.168.0.100.60005 > 192.168.0.200.ssh: tcp 240 (DF)
18:37:33.280619 192.168.0.200.ssh > 192.168.0.100.60005: tcp 192 (DF)
18:37:33.286025 192.168.0.100.60005 > 192.168.0.200.ssh: tcp 384 (DF)
18:37:33.288273 192.168.0.200.ssh > 192.168.0.100.60005: tcp 32 (DF)
18:37:33.288377 192.168.0.100.60005 > 192.168.0.200.ssh: tcp 64 (DF)
18:37:33.290148 192.168.0.200.ssh > 192.168.0.100.60005: tcp 48 (DF)
18:37:33.290204 192.168.0.100.60005 > 192.168.0.200.ssh: tcp 48 (DF) [tos 0x8]
18:37:33.293550 192.168.0.200.ssh > 192.168.0.100.60005: tcp 48 (DF) [tos 0x8]
18:37:33.333059 192.168.0.100.60005 > 192.168.0.200.ssh: tcp 0 (DF) [tos 0x8]
18:37:33.333314 192.168.0.200.ssh > 192.168.0.100.60005: tcp 384 (DF) [tos 0x8]
18:37:33.333336 192.168.0.100.60005 > 192.168.0.200.ssh: tcp 0 (DF) [tos 0x8]
18:37:33.333892 192.168.0.100.60005 > 192.168.0.200.ssh: tcp 32 (DF) [tos 0x8]
18:37:33.333932 192.168.0.100.60005 > 192.168.0.200.ssh: tcp 0 (DF) [tos 0x8] F
18:37:33.336248 192.168.0.200.ssh > 192.168.0.100.60005: tcp 0 (DF) [tos 0x8] F ack
18:37:33.336265 192.168.0.100.60005 > 192.168.0.200.ssh: tcp 0 (DF) [tos 0x8] ack
```

Example Flows

Src Addr	Src Port	Dst Addr	Dst Port	Packets	Bytes
100	6005	200	22	19	1894
200	22	100	6005	17	2558

Key: Unidirectional IP, port

What Is NetFlow?

- Originally a Cisco *switching method* to accelerate packet forwarding (1997)
- Routers build switching/accounting records for each “flow”
- A flow is all packets with matching:
  - src/dst address, src/dst port, IP protocol, TOS, input interface
- When a flow ends or times out, its accounting information can be exported in a flow record

NetFlow

- Version 5 and 9 formats common today
  - www.cisco.com/go/netflow
  - www.ietf.org/rfc/rfc3954.txt
- Called “cflow” on Juniper
- Many tools
  - flow-tools, www.splintered.net/sw/flow-tools/
  - cflowd, www.caida.org/tools/measurement/cflowd/
  - FlowScan, net.doit.wisc.edu/~plonka/FlowScan/

sFlow

- RFC 3176, Sep 2001
- Switch or router level agent
  - Controlled by SNMP
- Statistical flow samples
  - Flow is one ingress port to one egress port(s)
  - Up to 256 packet header bytes, or summary
- Periodic counter statistics
  - typical 20-120 sec interval
- www.sflow.org, www.ntop.org

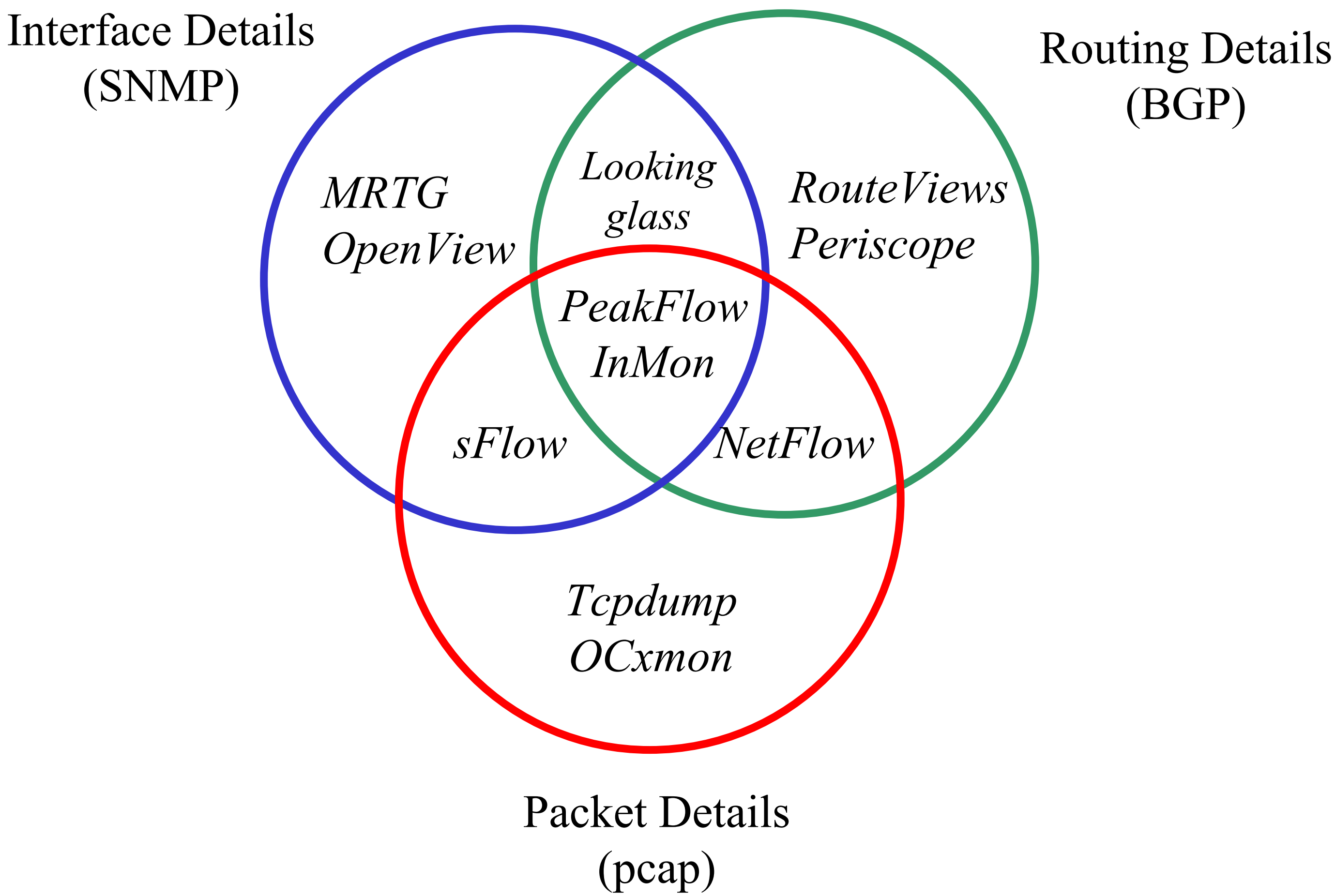
NetFlow / sFlow Comparison

NetFlow	sFlow
Stateful	Stateless
Heavyweight	Lightweight
Start/Stop times	Post processing required
Often full	Usually sampled
IP only	Layer 2 and above
Defined fields only	Raw packet data
More tools available	Cheaper probes

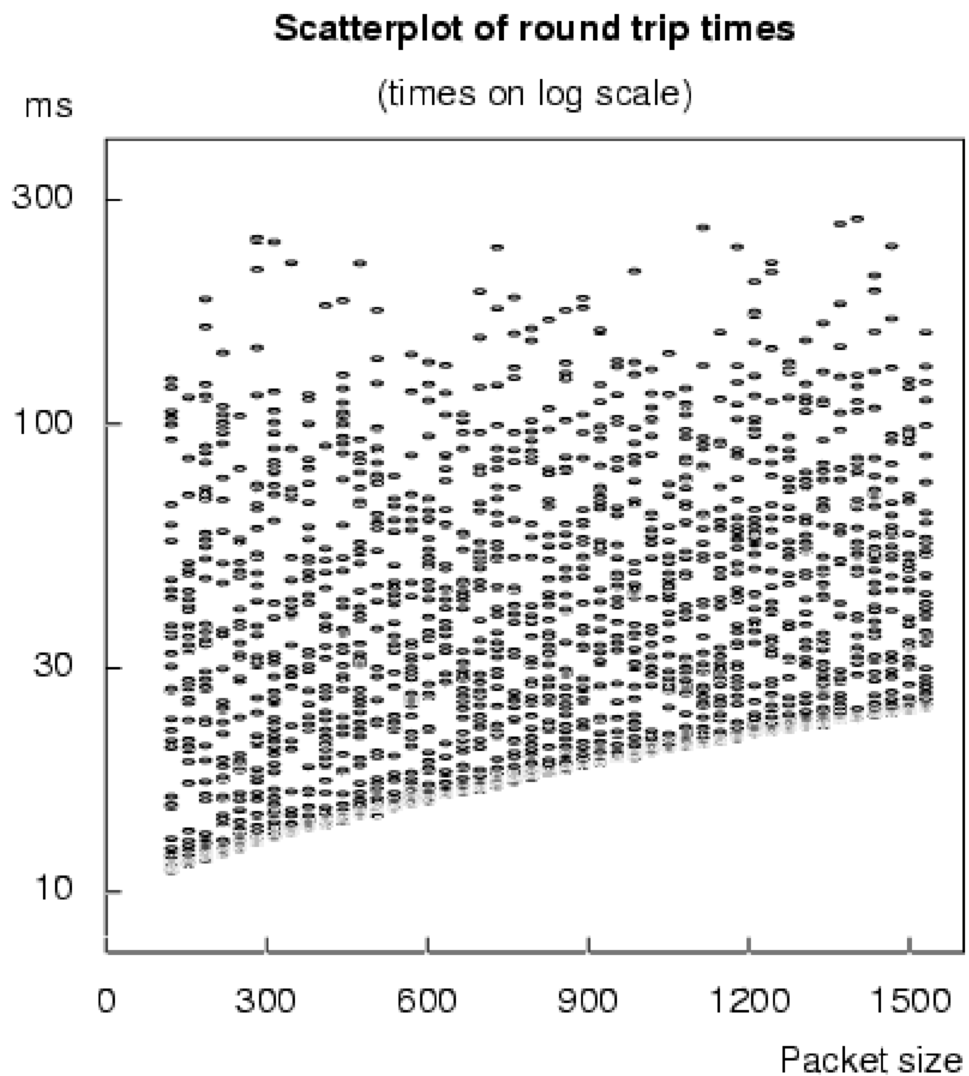
Uses of NetFlow / sFlow Data

- Accounting / billing
  - Who is using the network?
- Security
  - Detection/traceback of scans, (D)DoS attacks, infected hosts, unusual activity, forensics
- Engineering
  - Routing, peering, traffic mix, ToS, applications

Relationships



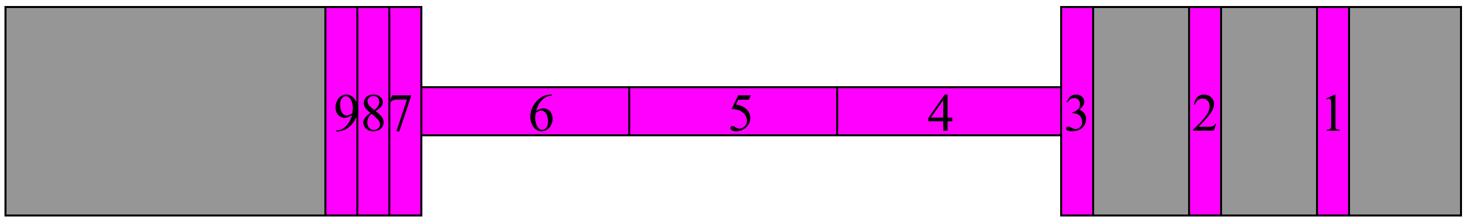
Bandwidth Estimation – Single Packet



- Larger packets take longer
- Delay from intercept
- Bandwidth from slope

From A. Downey

Bandwidth Estimation – Multi Packet



- Packet pairs or trains are sent
- The slower link causes packets to spread
- The packet spread indicates the bandwidth

Bandwidth Measurement Tools

- pathchar – Van Jacobson, LBL
  - ftp://ftp.ee.lbl.gov/pathchar/
- clink – Allen Downey, Wellesley College
  - http://rocky.wellesley.edu/downey/clink/
- pchar – Bruce A. Mah, Sandia/Cisco
  - http://www.employees.org/~bmah/Software/pchar/

Bandwidth Measurement Tools

- pipechar - Jin Guojun, LBL
  - http://www.didc.lbl.gov/pipechar/
- nettimer - Kevin Lai, Stanford University
  - http://gunpowder.stanford.edu/~laik/projects/nettimer/
- pathrate/pathload - Constantinos Dovrolis, Georgia Tech
  - http://www.cc.gatech.edu/fac/Constantinos.Dovrolis/bwmeter.html

abing

- http://www-iepm.slac.stanford.edu/tools/abing/
- Sends 20 pairs (40 packets) of 1450 byte UDP to a reflector on port 8176, estimates (in ~1 second):
  - ABw: Available Bandwidth
  - Xtr: Cross traffic
  - DBC: Dominating Bottleneck Capacity

```
damp-ssc$ abing -d damp-arl
1095727946 T: 192.168.1.2 ABw-Xtr-DBC: 541.1 104.1 645.3 ABW: 541.1 Mbps
                               RTT: 69.042 69.110 69.322 ms 20 20
1095727946 F: 192.168.1.2 ABw-Xtr-DBC: 364.5 200.5 564.9 ABW: 364.5 Mbps
                               RTT: 69.042 69.110 69.322 ms 20 20
```

Bandwidth*Delay Product

- The number of bytes in flight to fill the entire path
- Includes data in queues if they contributed to the delay
- Example
  - 100 Mbps path
  - ping shows a 75 ms rtt
  - BDP = 100 * 0.075 = 7.5 million bits (916 KB)

Windows

Flow/rate control and error recovery

Windows

- Windows control the amount of data that is allowed to be “in flight” in the network
- Maximum throughput is one window full per round trip time
- The sender, receiver, and the network each determine a different window size

Window Sizes 1,2,3

Data packets go one way

ACK packets come back

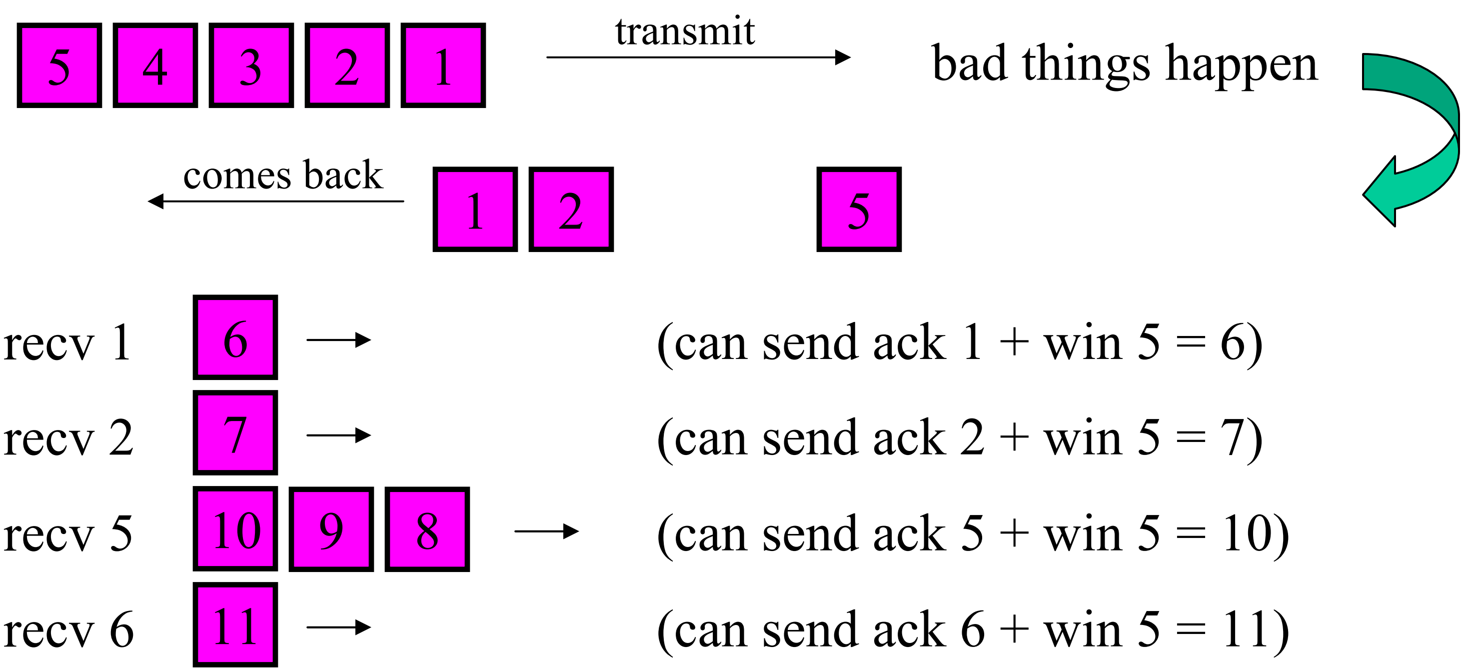
MPing – A Windowed Ping

www.psc.edu/~mathis/wping/

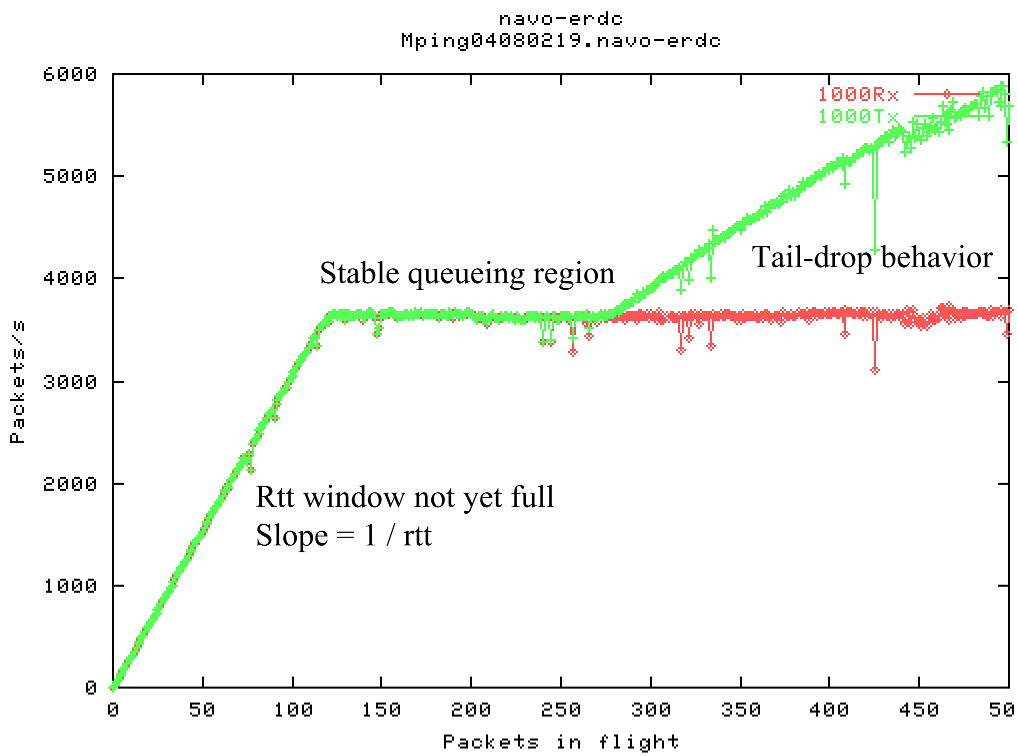
- Excellent tool to view the packet forwarding and loss properties of a path under varying load!
- Sends windows full of ICMP Echo or UDP packets
- Treats ICMP Echo_Reply or Port_Unreachable packets as "ACKs"
- Make sure destination responds well to ICMP
- Consumes a lot of resources: use with care

How MPing Works

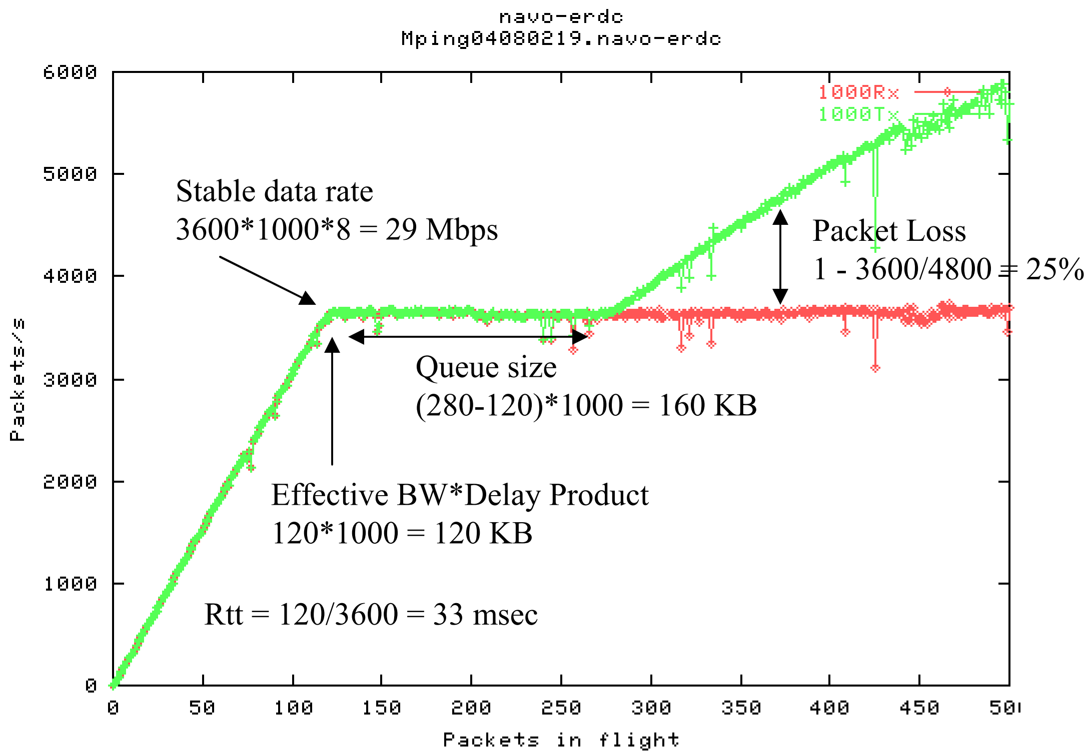
Example: window size = 5



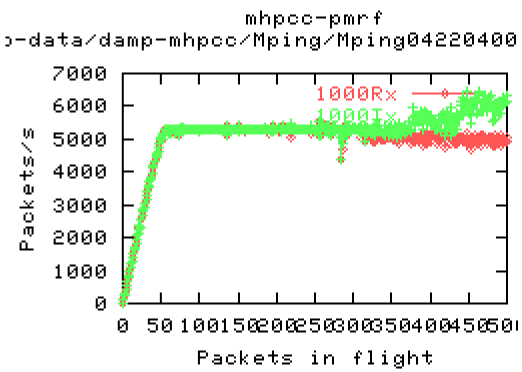
MPing on a "Normal" Path

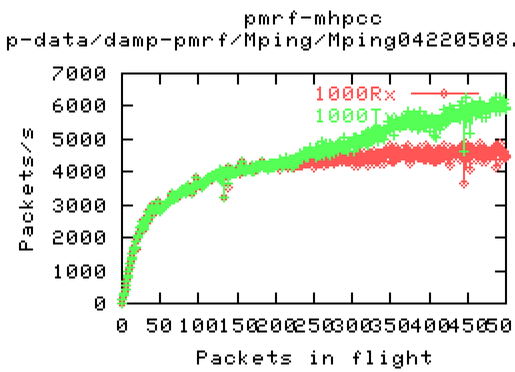


MPing on a "Normal" Path



Some MPing Results #1

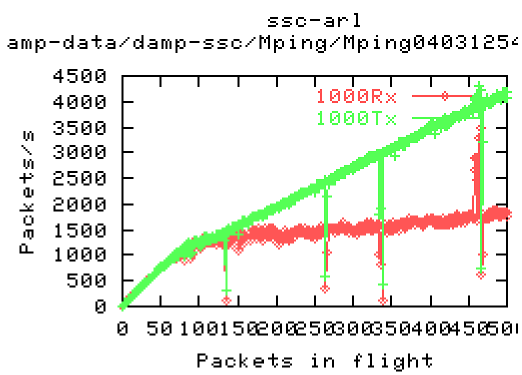


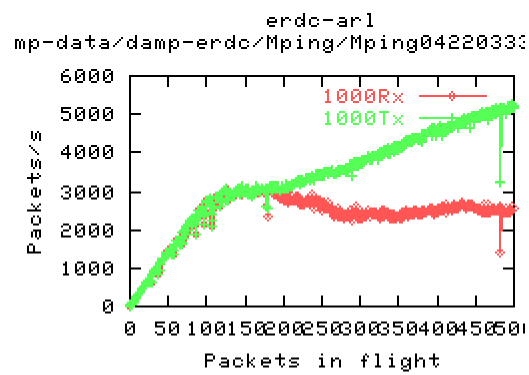


Fairly normal behavior
Discarded packets are costing some performance loss

RTT is increasing as load increases
Slow packet processing?

Some MPing Results #2

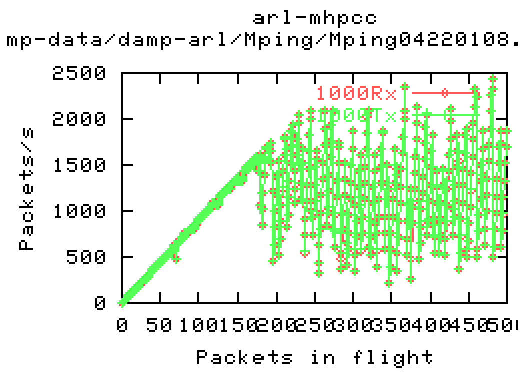


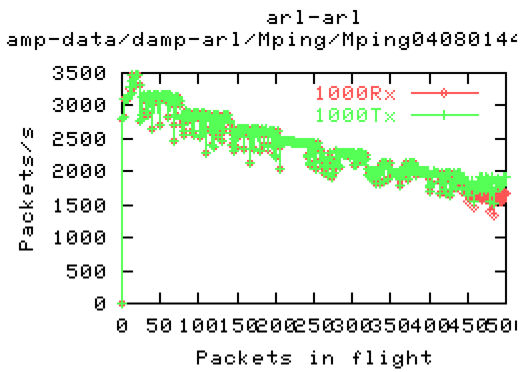


Very little stable queueing
Insufficient memory?
Spikes from some periodic event (cache cleaner?)

Discarding packets comes at some cost to performance
Error logging?

Some MPing Results #3

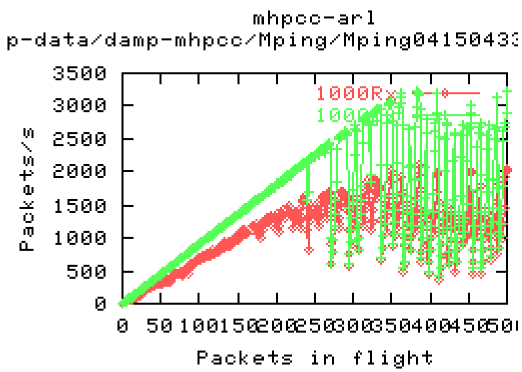


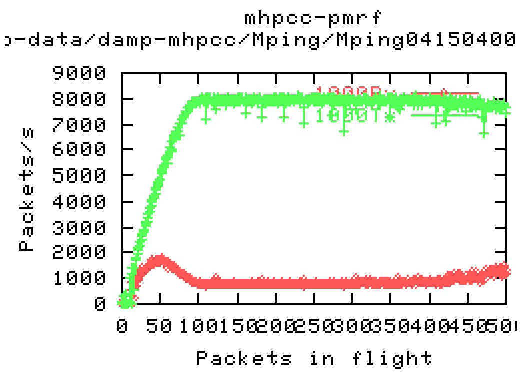


Oscillations with little loss
Rate shaping?

Decreasing performance with increasing queue length
Typical of Unix boxes with poor queue insertion

Some MPing Results #4





Fairly constant packet loss, even under light load

Major packet loss, ~7/8 or 88%
Hump at 50 may be duplex problem

Both turned out to be an auto-negotiation duplex problem
Setting to static full-duplex fixed these!

Ethernet Duplex Problems

An Internet Epidemic!

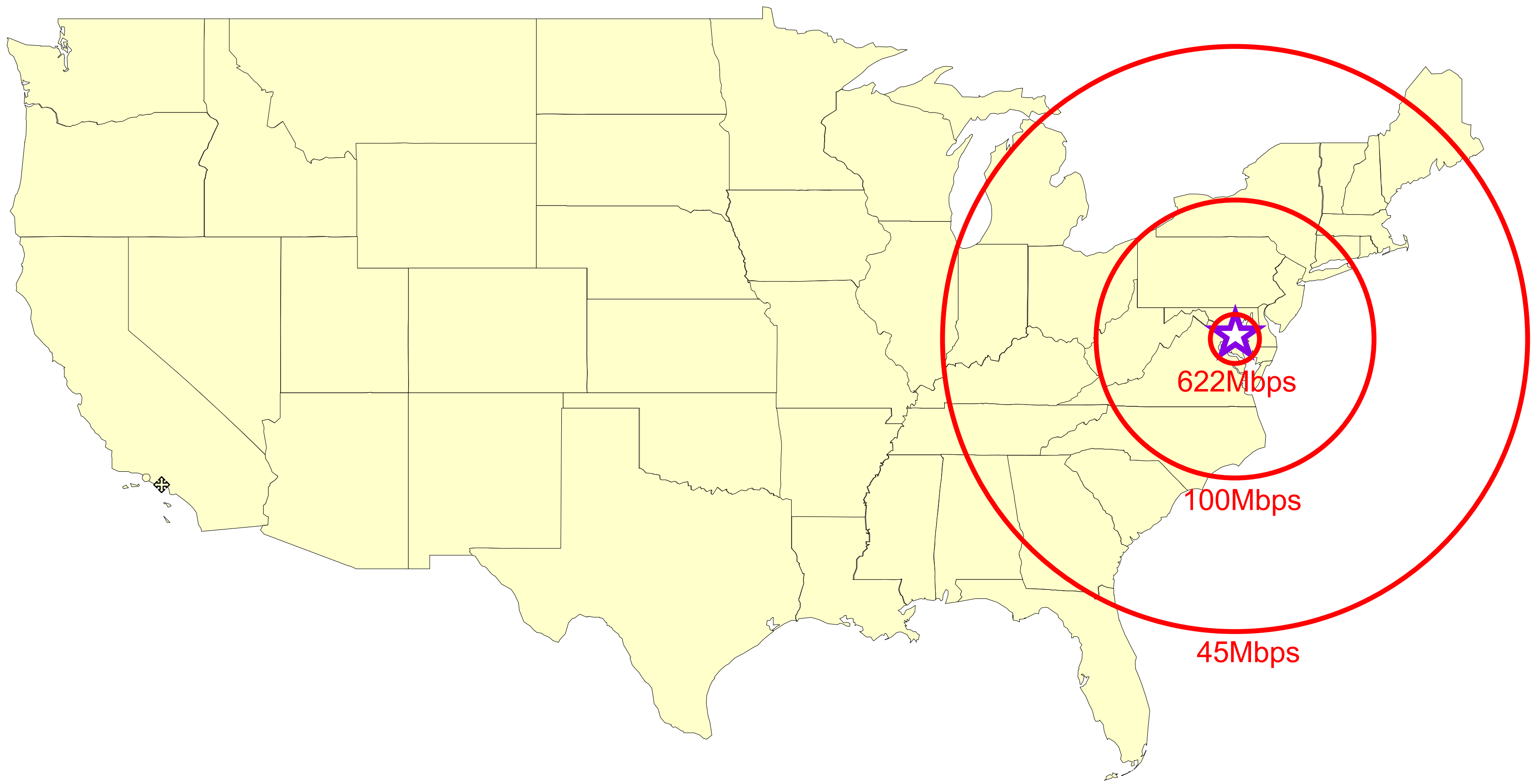
- Ethernet “auto-negotiation” can select the speed and duplex of a connected pair
- If only one end is doing it:
  – It can get the speed right
  – but it will **assume half-duplex**
- Mismatch loss only shows up under load
  – Can’t see it with ping

Windows and TCP Throughput

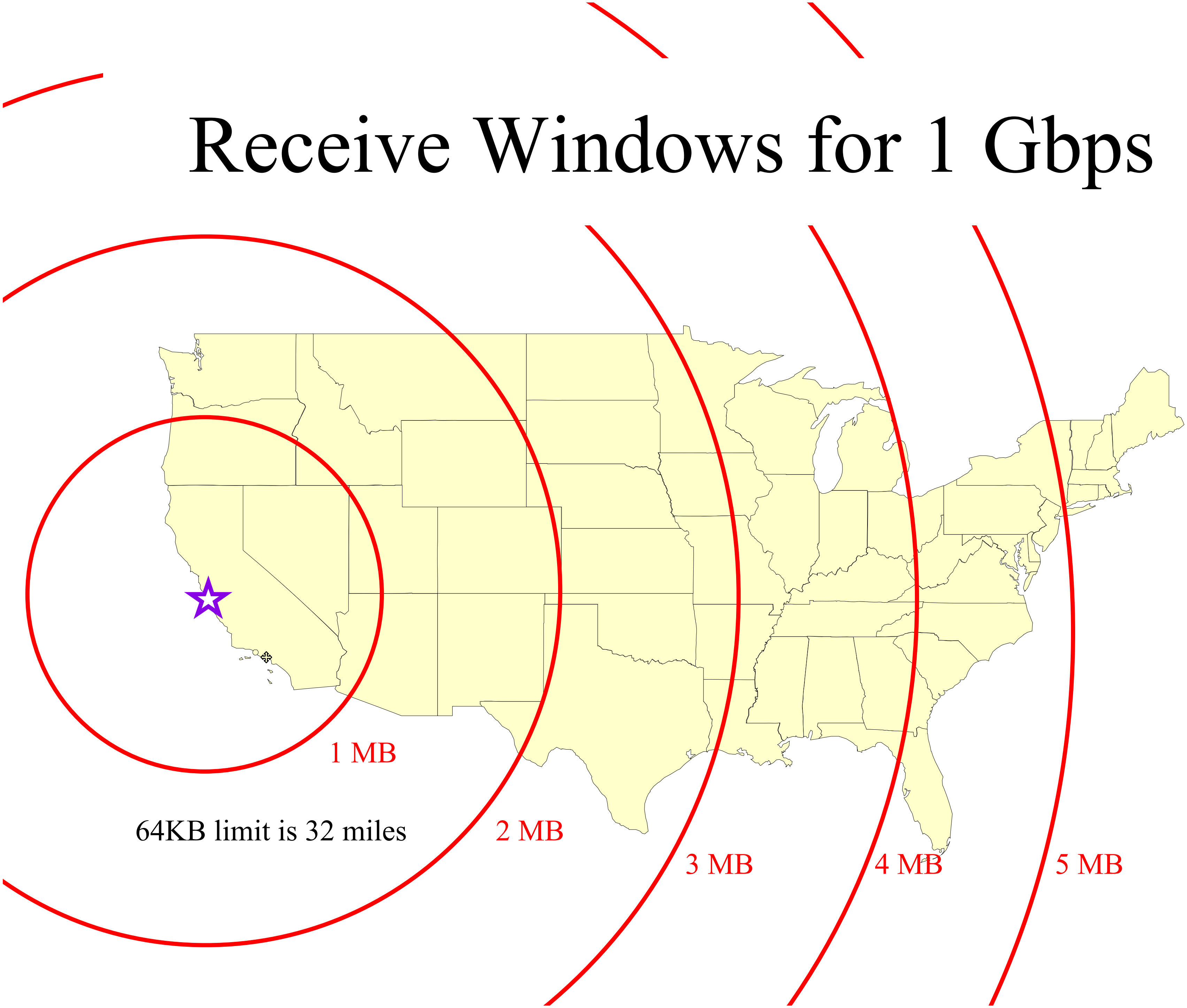
- TCP Throughput is at best one window per round trip time (window/rtt)
- The maximum TCP window of the receiving or sending host is the **most common limiter of performance**
- Examples
  - 8 KB window / 87 msec ping time = 753 Kbps
  - 64 KB window / 14 msec rtt = 37 Mbps

Maximum TCP/IP Data Rate

With 64KB window



Receive Windows for 1 Gbps



Observed Receiver Window Sizes

- ATM traffic from the Pittsburgh Gigapop
- 50% had windows < 20 KB
  - These are obsolete systems!
- 20% had 64 KB windows
  - Limited to ~ 8 Mbps coast-to-coast
- ~9% are assumed to be using window scale

M. Mathis, PSC, 2002

TCP

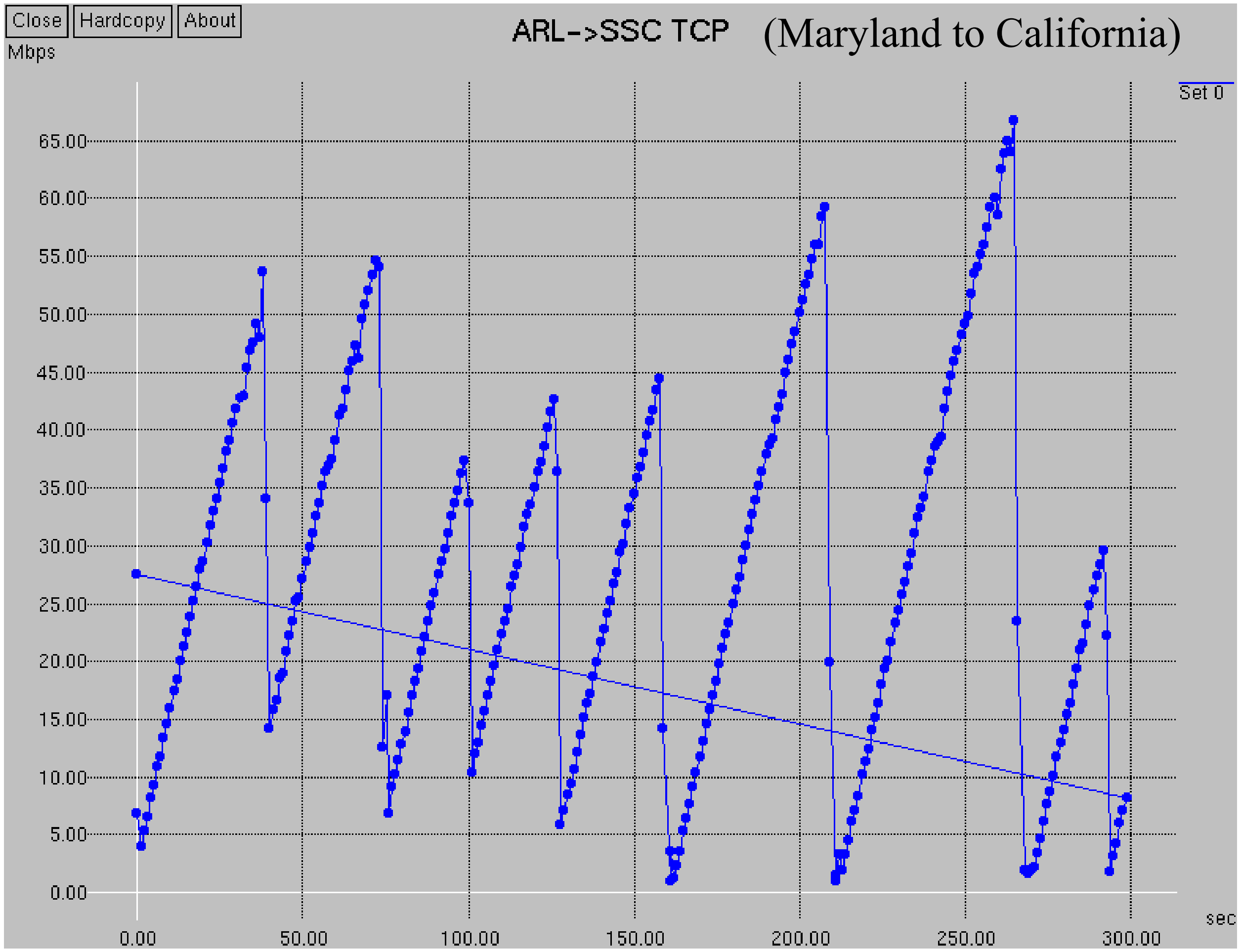
The Internet's transport

Important Points About TCP

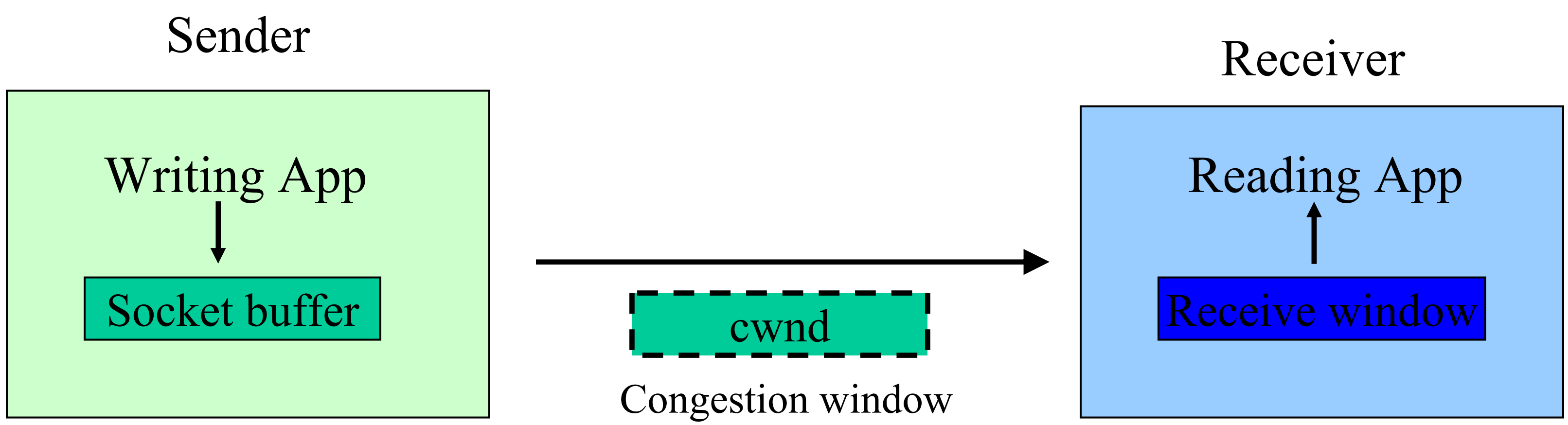
- TCP is *adaptive*
- It is *constantly* trying to go *faster*
- It *slows down* when it detects a *loss*

- *How much* it sends is controlled by *windows*
- *When* it sends is controlled by *received ACK's* (or timeouts)

TCP Throughput vs. Time



The Three TCP Windows



The **smallest** of these three will limit throughput

TCP Receive Window

- A flow control mechanism
- The amount of data the receiver is ready to receive
  - Updated with each ACK
- *Loosely* set by receive socket buffer size
- Application has to drain the buffer fast enough
- System has to wait for missing or out of order data (reassembly queue)

Sender Socket Buffers

- Must hold two round trip times of data
  - One set for the current round trip time
  - One set for possible retransmits from the last round trip time (retransmit queue)
- A system maximum often limits size
- The application must write data quickly enough

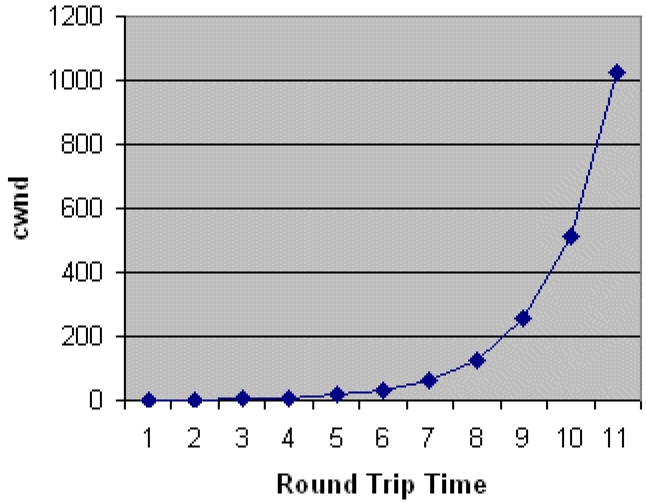
TCP Congestion Window (cwnd)

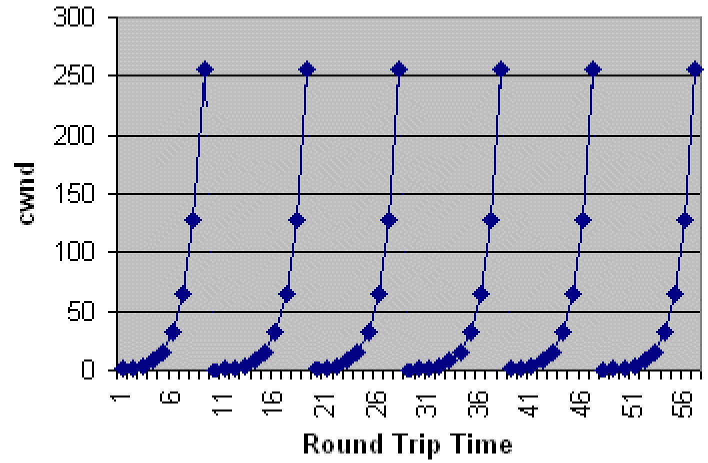
- Flow control, calculated by sender, based on observations of the data transfer (loss, rtt, etc.)
  - cwnd gets larger after every new ACK
  - cwnd get smaller when loss is detected
- cwnd amounts to TCP's estimate of the available bandwidth at any given time
- In the old days, there was none
  - TCP would send a full receive window in one round trip time

Two Modes of TCP Congestion Control

- TCP can operate cwnd in two modes
  - Slow-start
    - cwnd increases exponentially
  - Congestion-Avoidance ("steady state")
    - cwnd increases linearly

Cwnd During Slow Start



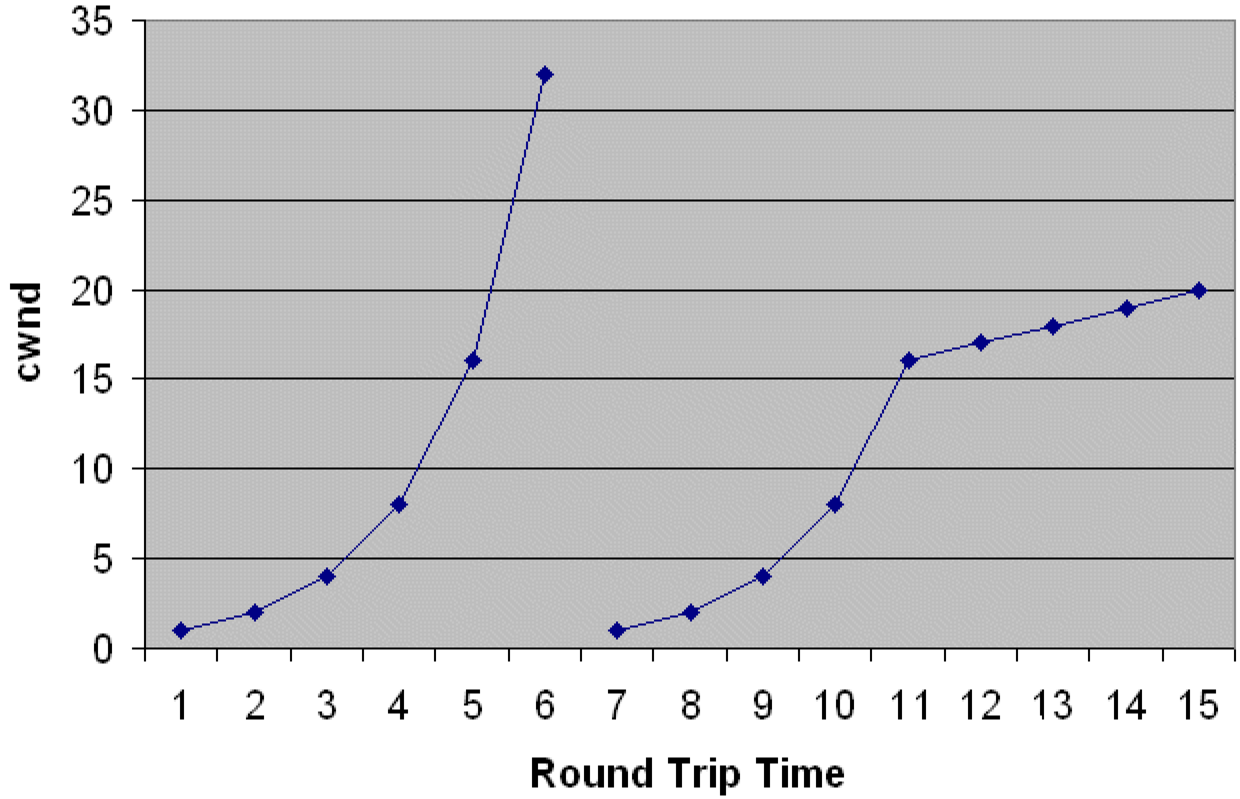


- cwnd increased by one for every new ACK
- cwnd doubles every round trip time (exponential)
- cwnd is reset to zero after a loss

Congestion Avoidance (AIMD)

- Additive Increase, Multiplicative Decrease
  - of cwnd, the congestion window
- The core of TCP's congestion avoidance phase, or "steady state"
  - Standard increase = +1.0 MSS per loss free rtt
  - Standard decrease = *0.5 (i.e. halve cwnd on loss)
- Avoids congestion collapse
- Promotes fairness among flows

Slow Start and Congestion Avoidance Together

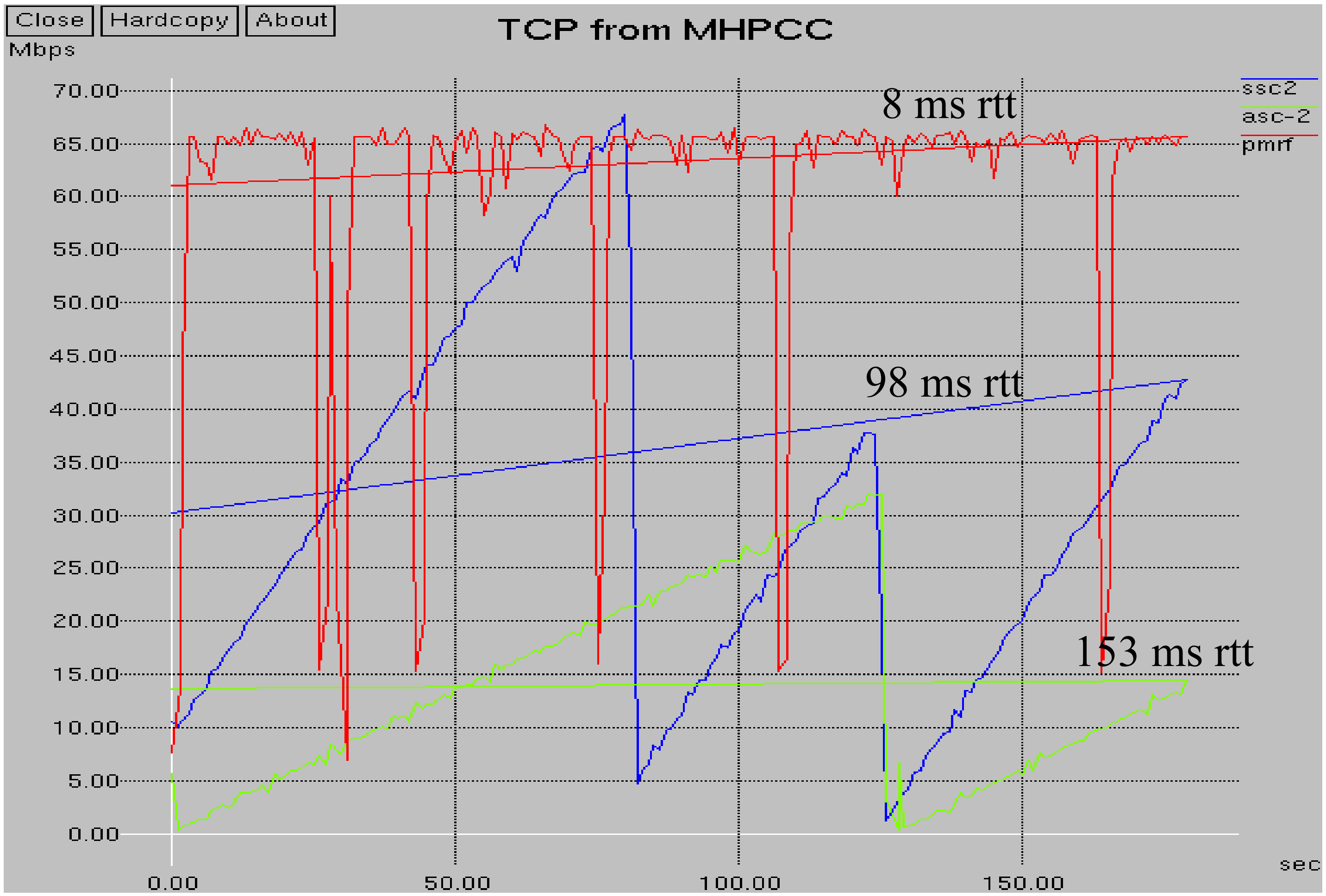


Iperf : TCP California to Ohio

```
damp-ssc2$ iperf -c damp-asc2 -p56117 -w750k -t20 -i1 -fm
------------------------------------------------------------
Client connecting to damp-asc2, TCP port 56117
TCP window size:  1.5 MByte (WARNING: requested  0.7 MByte)
------------------------------------------------------------
[  3] local 192.168.25.74 port 35857 connected with 192.168.244.240 port 56117
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0- 1.0 sec   1.7 MBytes  14.2 Mbits/sec
[  3]  1.0- 2.3 sec   1.5 MBytes   9.5 Mbits/sec
[  3]  2.3- 3.2 sec   1.2 MBytes  11.2 Mbits/sec
[  3]  3.2- 4.1 sec   1.2 MBytes  12.2 Mbits/sec
[  3]  4.1- 5.1 sec   1.6 MBytes  12.5 Mbits/sec
[  3]  5.1- 6.1 sec   1.6 MBytes  13.6 Mbits/sec
[  3]  6.1- 7.0 sec   1.6 MBytes  14.6 Mbits/sec
[  3]  7.0- 8.2 sec   2.0 MBytes  14.7 Mbits/sec
[  3]  8.2- 9.1 sec   1.6 MBytes  15.4 Mbits/sec
[  3]  9.1-10.1 sec   2.0 MBytes  16.7 Mbits/sec
[  3] 10.1-11.1 sec   2.0 MBytes  17.0 Mbits/sec
[  3] 11.1-12.0 sec   2.0 MBytes  18.5 Mbits/sec
[  3] 12.0-13.1 sec   2.4 MBytes  18.8 Mbits/sec
[  3] 13.1-14.1 sec   2.4 MBytes  19.1 Mbits/sec
[  3] 14.1-15.1 sec   2.4 MBytes  20.8 Mbits/sec
[  3] 15.1-16.1 sec   2.4 MBytes  20.6 Mbits/sec
[  3] 16.1-17.0 sec   2.4 MBytes  22.0 Mbits/sec
[  3] 17.0-18.1 sec   2.8 MBytes  22.2 Mbits/sec
[  3] 18.1-19.1 sec   1.6 MBytes  13.6 Mbits/sec
[  3] 19.1-20.1 sec   1.6 MBytes  12.3 Mbits/sec
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0-20.6 sec  38.2 MBytes  15.5 Mbits/sec
```

82 msec rtt

TCP Examples from Maui HI



TCP Acceleration (MSS/rtt^2)

(Congestion avoidance rate increase, MSS = 1448)

rtt (msec)	Mbps/s	0-100Mbps (sec)
5	463	0.216
10	116	0.864
20	29	3.45
50	4.6	21.6
100	1.16	86.4
200	0.29	345

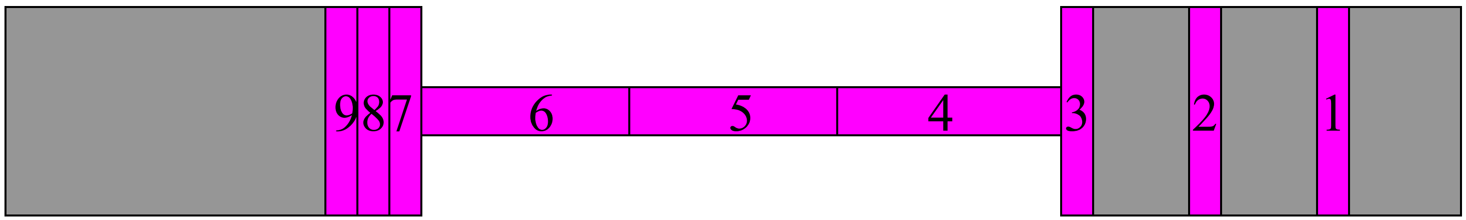
Bandwidth*Delay Product and TCP

- TCP needs a **receive window** (rwin) equal to or greater than the BW*Delay product to achieve maximum throughput
- TCP needs **sender side socket buffers** of 2*BW*Delay to recover from errors
- You need to send about 3*BW*Delay bytes for TCP to reach maximum speed

Delayed ACKs

- TCP receivers send ACK’s:
  - after every **second** segment
  - after a delayed ACK timeout
  - on every segment after a loss (missing segment)
- A new segment sets the delayed ACK timer
  - Typically 0-200 msec
- A second segment (or timeout) triggers an ACK and clears the delayed ACK timer

ACK Clocking



- A queue forms in front of a slower speed link
- The slower link causes packets to spread
- The spread packets result in spread ACK's
- The spread ACK's end up clocking the source packets at the slower link rate

Detecting Loss

- Packets get discarded when queues are full (or nearly full)
- Duplicate ACK's get sent after missing or out of order packets
- Most TCP's retransmit after the third duplicate ACK ("triple duplicate ACK")
  - Windows XP now uses 2nd dup ACK

TCP Throughput Model

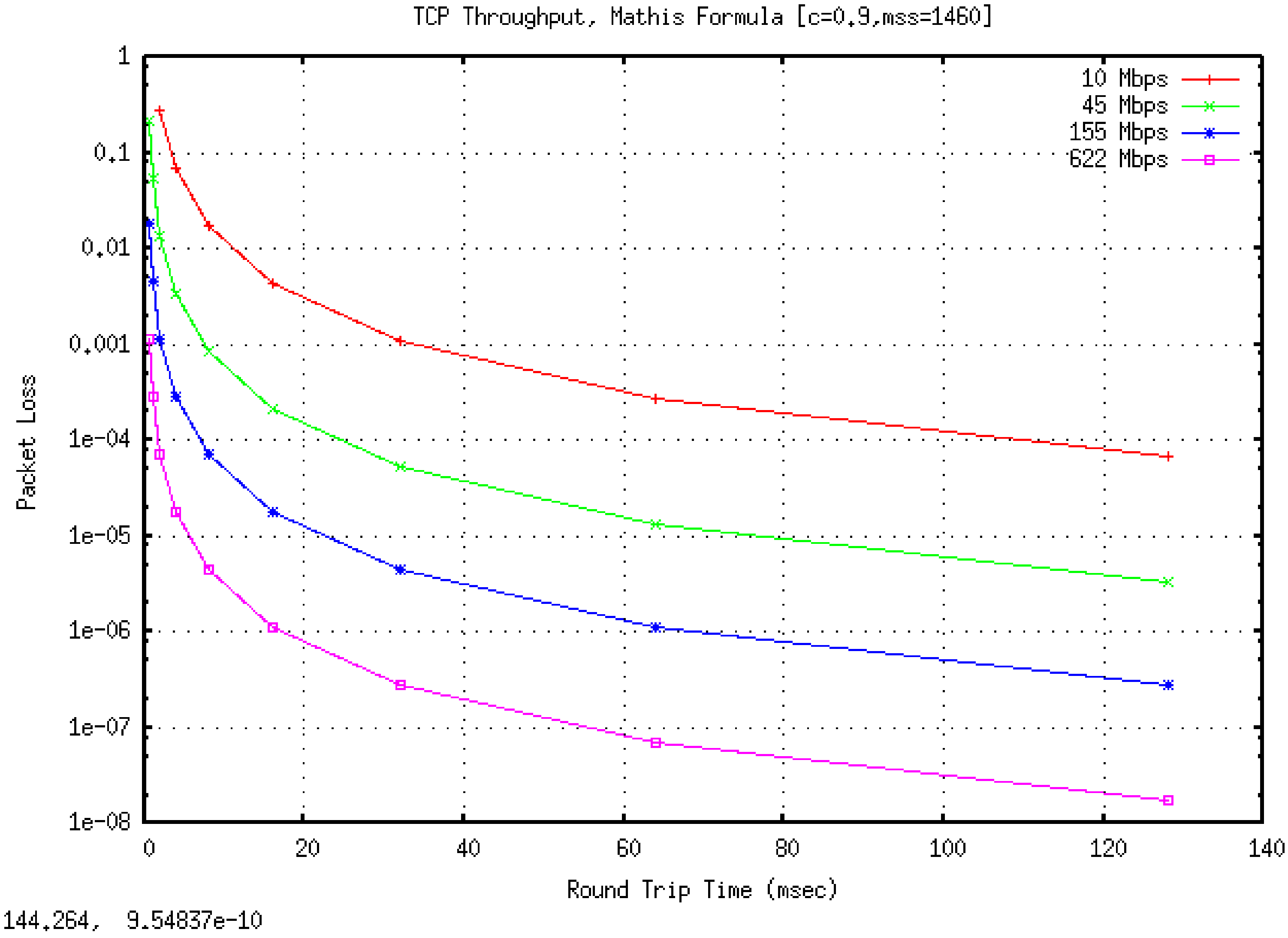
Once recv window size and available bandwidth aren't the limit

$$\text{Rate} = \frac{\sim 0.7 * \text{Max Segment Size (MSS)}}{\text{Round Trip Time} * \text{sqrt[pkt\_loss]}}$$

M. Mathis, et al.

- Double the MTU, double the throughput
- Halve the latency, double the throughput
  - shortest path matters
- Halve the loss rate, 40% higher throughput
- www.psc.edu/networking/papers/model_abstract.html

Example Mathis Predictions



Round Trip Time (RTT)

rate = 0.7 * MSS / (**rtt** * sqrt(p))

- If we could halve the delay we could double throughput!
- Most delay is caused by speed of light in fiber (~200 km/msec)
- "Scenic routing" and fiber paths raise the minimum
- Congestion (queueing) adds delay

Max Segment Size (MSS)

rate = 0.7 * **MSS** / (rtt * sqrt(p))

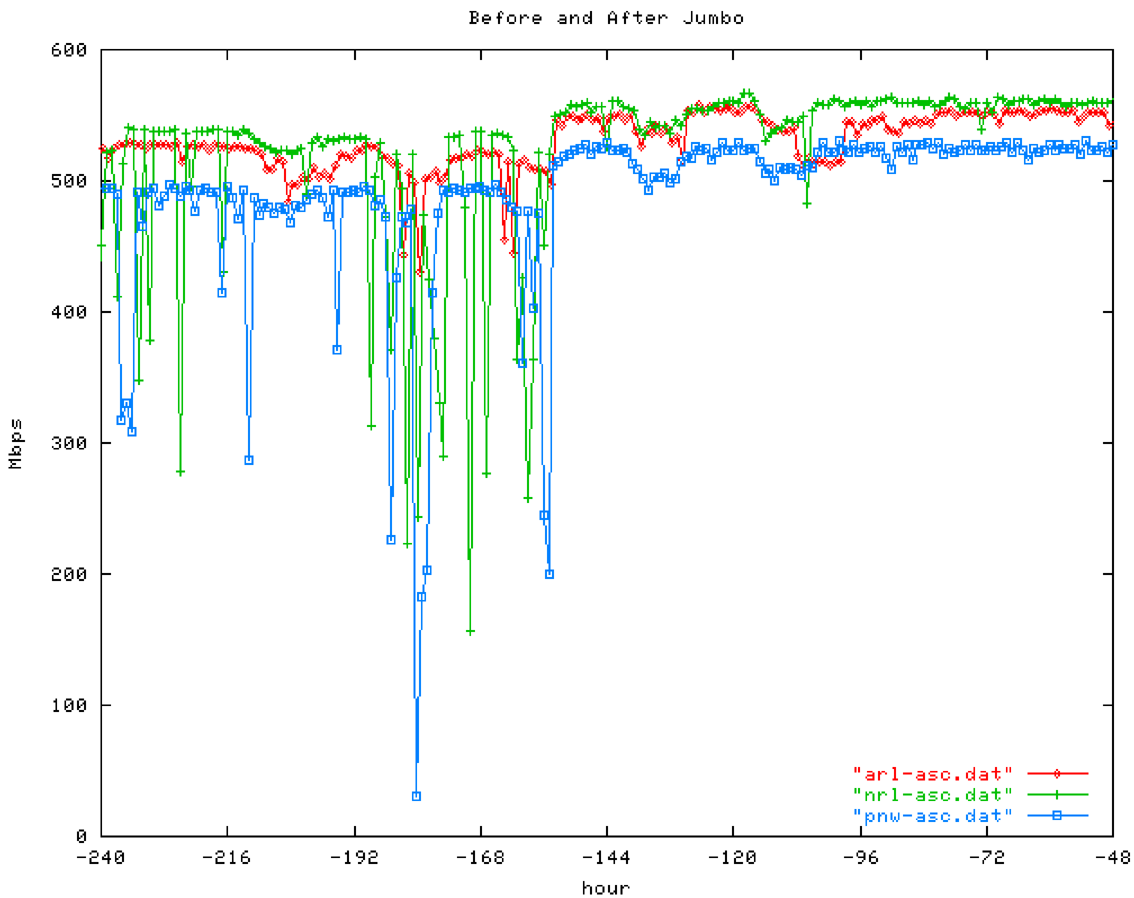
- MSS = MTU – packet headers
- Most often, MSS = 1448 or 1460
- Jumbo frame => ~6x throughput increase
- The Abilene and DREN WANs support MTU 9000 everywhere
  - Most sites still have a lot of work to do

DREN MTU Increase

- IP Maximum Transmission Unit (MTU) on the DREN WAN is now 9146 bytes
- After much debate, exactly 9000 is being used to the sites with GigE interfaces
  - Sites choose 1500, 4470, or 9000; others by exception
- Sites are encouraged to support 9000 on their GigE infrastructures

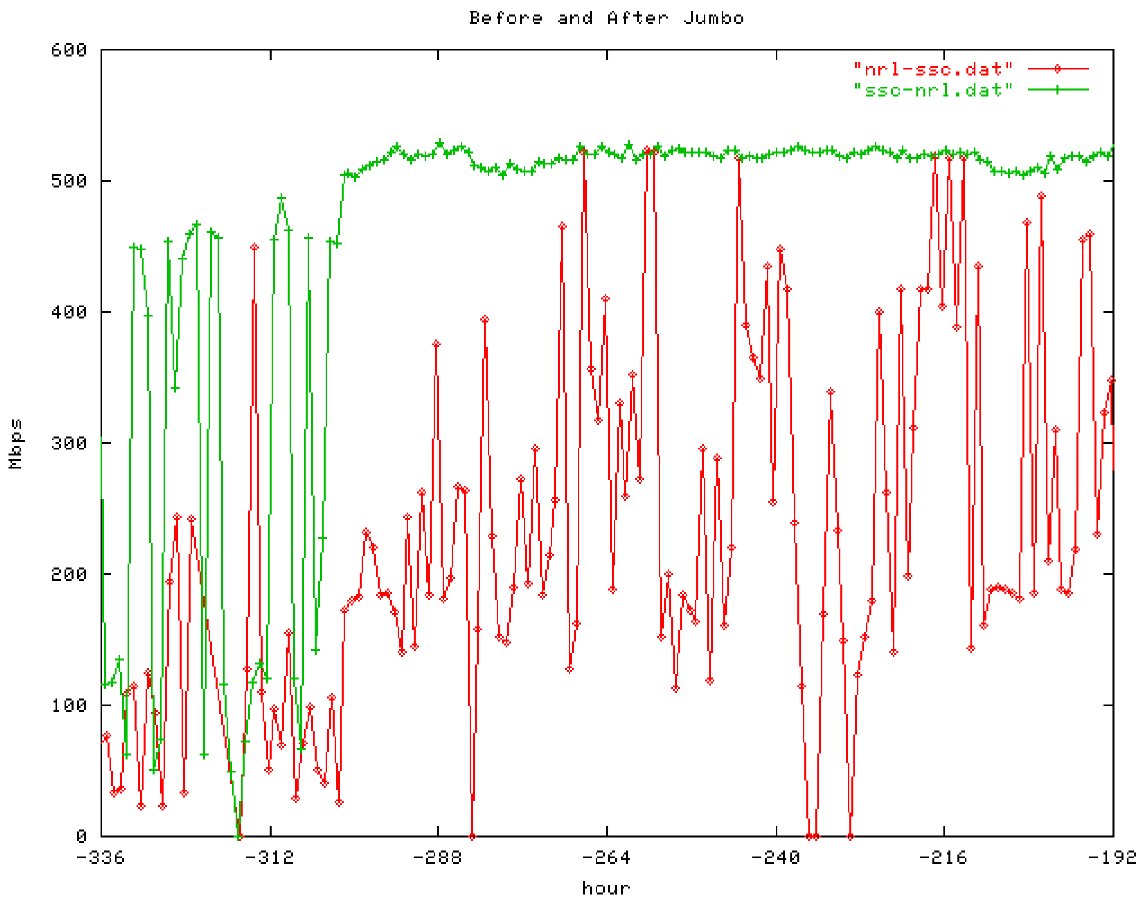
Impact of Jumbo, into Ohio

Jumbo enabled 23 July 2003



Impact of Jumbo, San Diego to DC

Jumbo enabled 17 July 2003

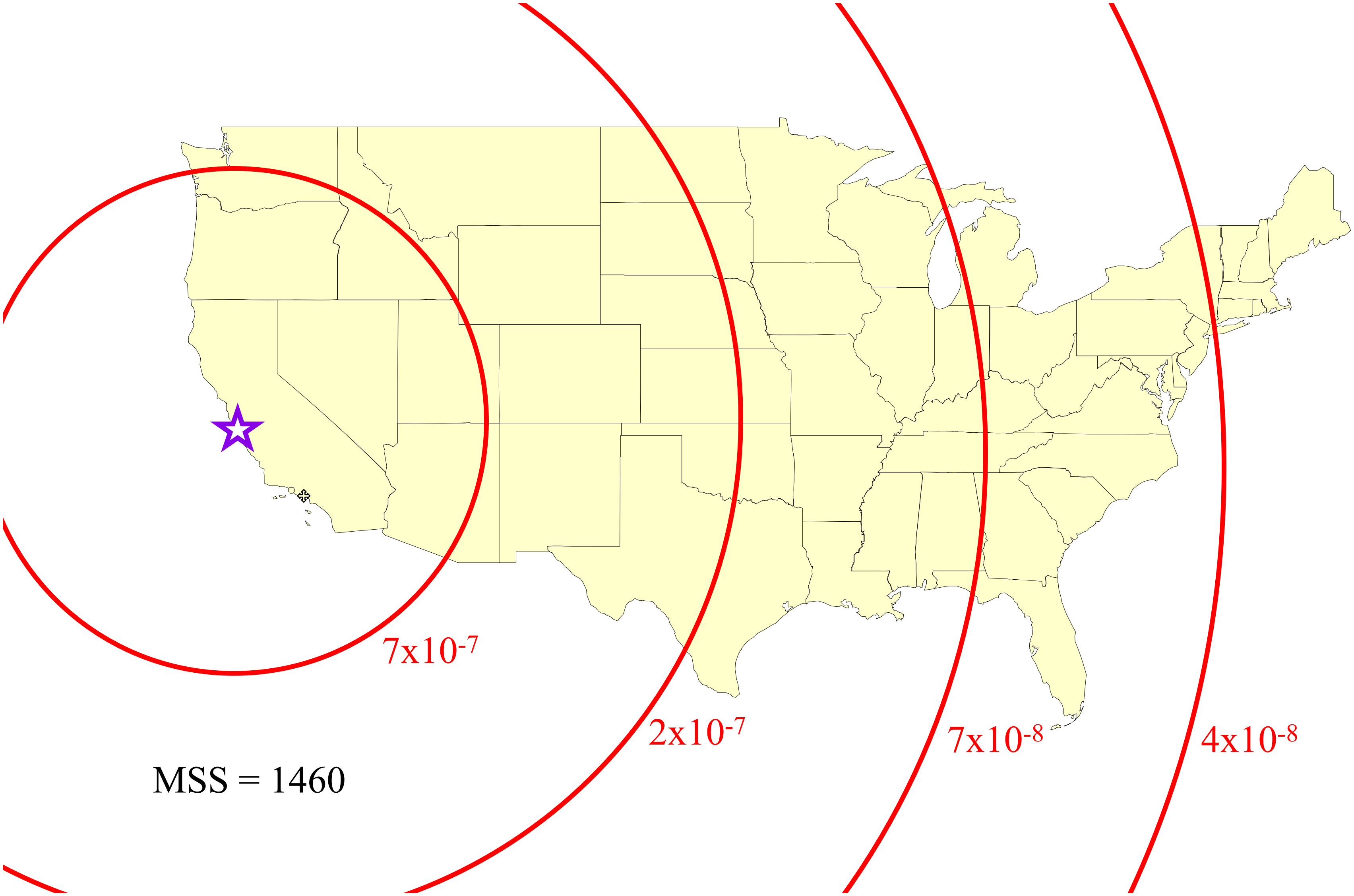


Packet Loss (p)

$$\text{rate} = 0.7 * \text{MSS} / (\text{rtt} * \text{sqrt}(\mathbf{p}))$$

- ***Loss dominates throughput !***
- At least 6 orders of magnitude observed on the Internet
- 100 Mbps throughput requires $O(10^{-6})$
- 1 Gbps throughput requires $O(10^{-8})$

Loss Limits for 1 Gbps



Specifying Loss

- TCP loss limits for 1 Gbps across country are $O(10^{-8})$, i.e. 0.000001% packet loss
  - About 1 "ping" packet every **three years**
  - *Systems like AMP would never show loss*
  - Try to get 10^{-8} in writing from a provider!
  - Most providers won't guarantee < 0.01%

Specifying Throughput

- Require the provider to demonstrate TCP throughput
  – DREN contract requires ½ line rate TCP flow sustained for 10 minutes cross country (e.g. ~300 Mbps on an OC12)
- A low loss requirement comes with this!

Concerns About Bit Errors

- Bit Error Rate (BER) specs for networking interfaces/circuits may not be low enough
  - E.g. 10^{-12} BER => 10^{-8} packet ER (1500 bytes)
  - 10 hops => 10^{-7} packet drop rate

Example Bit Error Rate Specs

- Hard Disk, 10^{-14}
  - *One error in 10 TB*
- SONET Equipment, 10^{-12}
- SONET Circuits, 10^{-10}
- 1000Base-T, 10^{-10}
  - *One bit error every 10 seconds at line rate*
- Copper T1, 10^{-6}

Is 10^{-8} Packet Loss Reasonable?

- Requires a bit error rate (BER) of 10^{-12} or better!
- Perhaps TCP is expecting too much from the network
- Loss isn't always congestion
- Solution: Modify TCP's congestion control

TCP Throughput Model Revisited

- Original formula assumptions
  - constant packet loss rate
  - dominated by congestion from other flows
  - 6 x MTU provides 6 x throughput
- HPC environment (low congestion)
  - Loss may be dominated by bit error rates
  - MSS becomes sqrt(MSS)
  - 6 x MTU provides 2.4 x throughput

More About TCP

Some high performance options

High Performance TCP Features

- Window Scale
- Timestamps
- Selective Acknowledgement (SACK)
- Path MTU Discovery (PMTUD)
- Explicit Congestion Notification (ECN)

TCP Window Scale

- 16-bit window, 65535 byte maximum
- RFC1323 window scale option in SYN packets
- Creates a 32-bit window by left shifting 0 to 14 bit positions
  - New max window is 1 GB (2^{30})
  - Granularity: 1B, 2B, 4B, …, 16KB

TCP Timestamps

- 12 byte option
  - drops MSS of 1460 to 1448
- Allows better Round Trip Time Measurement (RTTM)
- Prevention Against Wrapped Sequence numbers (PAWS)
  - 32 bit sequence number wraps in 17 sec at 1 Gbps
  - TCP assumes a Maximum Segment Lifetime (MSL) of 120 seconds

Selective ACK

- TCP originally could only ACK the last in sequence byte received (cumulative ACK)
- RFC2018 SACK allows ranges of bytes to be ACK'd
  - Sender can fill in holes without resending everything
  - Up to four blocks fit in the TCP option space (three with the timestamp option)
- Surveys have shown that SACK implementations often have errors
  - RFC3517 addresses how to respond to SACK

Duplicate SACK

- RFC2883 extended TCP SACK option to allow duplicate packets to be SACK'd
- D-SACK is a SACK block that reports duplicates
- Allows the sender to infer packet reordering

Forward ACK Algorithm

- Forward ACK (FACK) is the forward-most segment acknowledged in a SACK
- FACK TCP uses this to keep track of outstanding data
- During fast recovery, keeps sending as long as outstanding data < cwnd
- Generally a good performer - recommended

Path MTU Discovery

- Probes for the largest end-to-end unfragmented packet
- Uses don't-fragment option bit, waits for must-fragment replies
- Possible black holes
  - some devices will discard the large packets without sending a must-fragment reply
  - Problems are discussed in RFC2923

Explicit Congestion Notification (ECN)

- RFC3168, Proposed Standard
- Indicates congestion without packet discard
- Uses last two bits of the IP Type of Service (TOS) field
- Black hole warning
  - Some devices silently discard packets with these bits set

TCP Connection Establishment

- Three-way handshake
  - SYN
  - SYN+ACK
  - ACK
- Use tcpdump, look for performance features
  - window sizes, window scale, timestamps, MSS, SackOK, Don't-Fragment (DF)

Tcpdump of TCP Handshake

```
16:08:33.674226 wcisd.hpc.mil.40874 > damp-nrl.56117:
    S 488615735:488615735(0) win 5840
    <mss 1460,sackOK,timestamp 263520790 0,nop,wscale 0> (DF)

16:08:33.734045 damp-nrl.56117 > wcisd.hpc.mil.40874:
    S 490305274:490305274(0) ack 488615736 win 5792
    <mss 1460,sackOK,timestamp 364570771 263520790,nop,wscale 5> (DF)

16:08:33.734103 wcisd.hpc.mil.40874 > damp-nrl.56117:
    . ack 1 win 5840
    <nop,nop,timestamp 263520796 364570771> (DF)
```

SYN Option Check Server

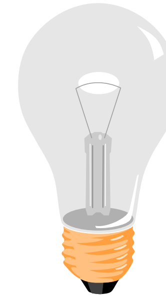
- http://syntest.psc.edu:7961/
- telnet syntest.psc.edu 7960

```
Your IP address is: 192.168.26.200
!
! Check of SYN options
!
!========================================================
! Variable          : Val        : Warning (if any)
!========================================================
SACKEnabled         : 3          :
TimestampsEnabled : 1          :
CurMSS              : 1448       :
WinScaleRcvd        : 2          :
CurRwinRcvd         : 1460       :
!
! End of SYN options
!
```

System Tuning

Interfaces, routes, buffers, etc.

Things You Can Do

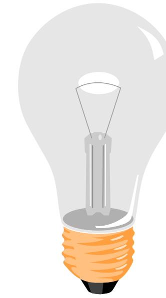


- Throw out your low speed interfaces and networks!

- Make sure routes and DNS report high speed interfaces
- Don't over-utilize your links (<50%)
- Use routers sparingly, host routers not at all

```
routed -q
```

Things You Can Do



- Make sure your HPC apps offer sufficient receive windows and use sufficient send buffers
  - But don't run your system out of memory
  - Find out the rtt with ping, compute BDP
  - Tune system wide, by application, or automatically
- Check your TCP for high performance features
- Look for sources of loss
  - Watch out for duplex problems

TCP Performance

- Maximum TCP throughput is one window per round trip time
- System default and maximum window sizes are usually too small for HPC
- The #1 cause of slow TCP transfers!

Minimum TCP Window Sizes

Throughput\rtt	85 msec	155 msec
40 Mbps	425KB	775KB
140 Mbps	1.5 MB	2.7 MB
570 Mbps	6.0 MB	11 MB
2280 Mbps	24 MB	44 MB
9100 Mbps	97 MB	176 MB

Increase Your **Max** Window Size

Linux example

- Command line:

```
sysctl -w net.core.rmem_max=40000000
sysctl -w net.core.wmem_max=40000000
```

- In **/etc/sysctl.conf** (makes it permanent)

```
net.core.rmem_max = 40000000
net.core.wmem_max = 40000000
```

Increase Your **Default** Window Size

Linux example

- In **/etc/sysctl.conf**

```
net.ipv4.tcp_rmem = 4096 349520 20000000
net.ipv4.tcp_wmem = 4096  65536 20000000
```

- Three values are: minimum, default, maximum for automatic buffer allocation on Linux

Application Buffer Sizes

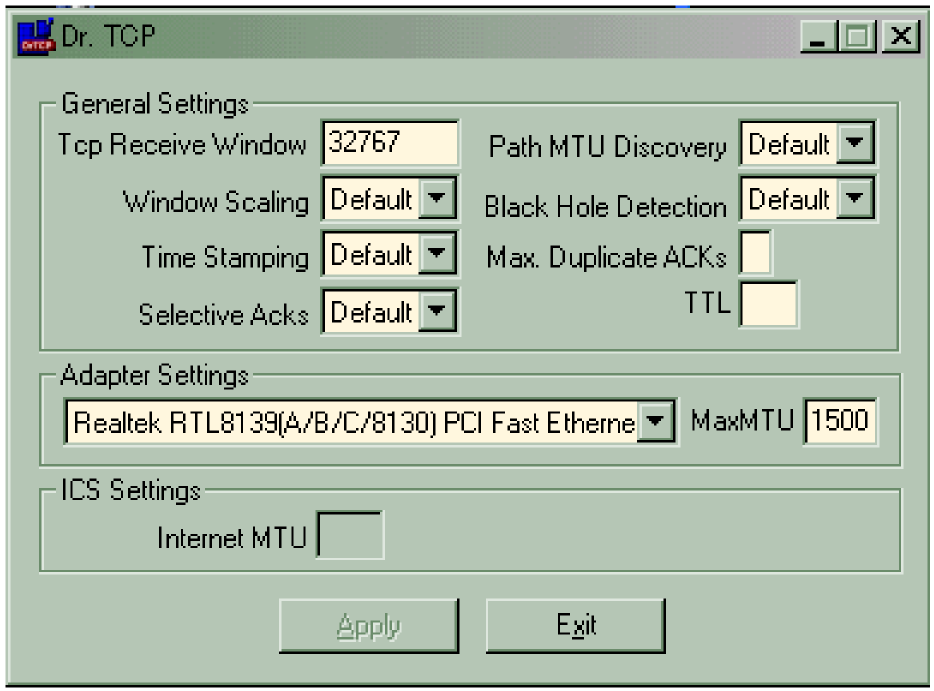
- Before doing a connect or accept

```
setsockopt(fd, SOL_SOCKET, SO_SNDBUF,
  …)
setsockopt(fd, SOL_SOCKET, SO_RCVBUF,
  …)
```

Dr. TCP

A TCP Stack Tuner for Windows

http://www.dslreports.com/front/drtcp.html



- Beware that modem utilities such as DunTweak can reduce performance on high speed nets

Tuning Other Systems

- See “Enabling High Performance Data Transfers”

  http://www.psc.edu/networking/projects/tcptune/

FTP Buffer Sizes

- Many FTP's allow the user to set buffer sizes
  - The commands are different everywhere!
- kftp buffer size commands

  `lbufsize 8388806` (local)

  `rbufsize 8388806` (remote)

  - Other kftp commands
    - protect/cprotect (clear|safe|private)
    - passive – client opens the data channel connection
- For other versions of FTP, see

  http://dast.nlanr.net/Projects/FTP.html

vsftpd FTP server

- Small, fast, secure
- 2.x has SSL / TLS support
- http://vsftpd.beasts.org/

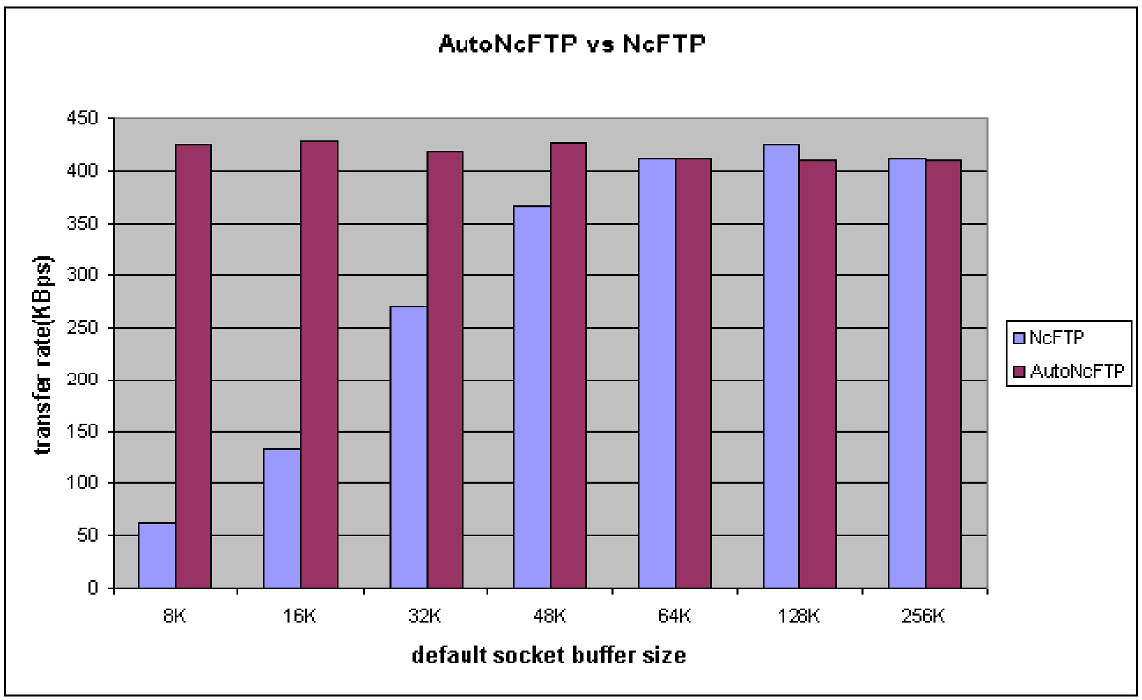
GridFTP

- FTP Protocol Extensions
  - SBUF and ABUF – set / auto buffer sizes
  - SPOR and SPAS – striped port / passive

Autobuf – An Auto-tuning FTP

http://dast.nlanr.net/Projects/Autobuf/

- Measures the spread of a burst of ICMP Echo packets to estimate BDP, sets bufs



High Performance SCP (HPN-SSH)

- The SSH/SCP secure shell applications use tiny internal flow control buffers
- For greatly improved performance over large round trip times, see
  - http://www.psc.edu/networking/projects/hpn-ssh/

Tuning Success Story

- Rome NY to Maui HI, using kftp
- Before tuning: 3.2 Mbps
- After tuning: 40 Mbps (DS3 line rate)

A very common story…

Good Tuning References

- Users Guide to TCP Windows
  www.ncsa.uiuc.edu/People/vwelch/net_perf/tcp_windows.html
- TCP Tuning Guide
  www-didc.lbl.gov/TCP-tuning/
- WAN Tuning and Troubleshooting
  www.internet2.edu/~shalunov/writing/tcp-perf.html
- Enabling High Performance Data Transfers on Hosts
  www.psc.edu/networking/projects/tcptune/

Throughput Tests

Throughput Testing Tools

- ttcp – the original, many variations
  - http://sd.wareonearth.com/~phil/net/ttcp/
- nuttcp – great successor to ttcp (recommended)
  - ftp://ftp.lcp.nrl.navy.mil/pub/nuttcp/
- Iperf – great TCP/UDP tool (recommended)
  - http://dast.nlanr.net/Projects/Iperf/
- netperf – dated but still in wide use
  - http://www.netperf.org/
- ftp – nothing beats a real application

Throughput Testing Notes

- Network data rates (bps) are powers of 10, not powers of 2 as used for Bytes
  - E.g. 100 Mbps ethernet is 100,000,000 bits/sec
  - Some tools wrongly use powers of 2 (e.g. ttcp)
- User payload data rates are reported by tools
  - No TCP, IP, Ethernet, etc. headers are included
  - E.g. 100 Mbps ethernet max is 97.5293 Mbps
    - http://sd.wareonearth.com/~phil/net/overhead/

nuttcp

- My favorite TCP/UDP test tool
- Get the latest .c file
  - ftp://ftp.lcp.nrl.navy.mil/pub/nuttcp/
- Compile it:
  - **cc –O3 –o nuttcp nuttcp-5.5.2.c**
- Start a server:
  - **nuttcp –S**
  - Allows remote users to run tests to/from that host without accounts

A Permanent nuttcp Server

RedHat Linux example

Create a file **/etc/xinetd.d/nuttcp**

```
# default: off
# description: nuttcp
service nuttcp
{
        socket_type     = stream
        wait            = no
        user            = nobody
        server          = /usr/local/bin/nuttcp
        server_args     = -S
        disable         = no
}
```

Nuttcp vs. Iperf

- Iperf is probably better for
  - Parallel stream reporting (-P)
  - Bi-directional tests (-d)
  - MSS control (-M) and reporting (-m)
- Nuttcp is better at
  - Getting server results back to the client
  - Third party support
  - Traceroute (-xt)
  - Setting priority (-xP)
  - Undocumented instantaneous rate limit (-Ri)

Bandwidth Test Controller (BWCTL)

- A remote interface to iperf
- Provides access / policy control
- Keeps tests from overlapping
- Useful for cross network testing
  - See the Performance Measurement Point list
  - http://e2epi.internet2.edu/pipes/pmp/pmp-dir.html
- http://e2epi.internet2.edu/bwctl/

BWCTL Test Example

DREN in San Diego to Abilene in Seattle

```
[phil@damp-ssc phil]$ bwctl -A A AESKEY phil -L 3600 -i 1 -w 8M -c nms1-sttl.abilene.ucaid.edu
Enter passphrase for identity 'phil':
bwctl: 16 seconds until test results available
RECEIVER START
3348416450.867761: /ami/bin/iperf -B 198.32.8.190 -P 1 -s -f b -m -p 5004 -w 8388608 -t 10 -i 1
------------------------------------------------------------
Server listening on TCP port 5004
Binding to local address 198.32.8.190
TCP window size: 16777216 Byte (WARNING: requested 8388608 Byte)
------------------------------------------------------------
[ 14] local 198.32.8.190 port 5004 connected with 138.18.190.5 port 5004
[ 14]  0.0- 1.0 sec  11862696 Bytes  94901568 bits/sec
[ 14]  1.0- 2.0 sec  17112464 Bytes  136899712 bits/sec
[ 14]  2.0- 3.0 sec  17150112 Bytes  137200896 bits/sec
[ 14]  3.0- 4.0 sec  17232648 Bytes  137861184 bits/sec
[ 14]  4.0- 5.0 sec  17251472 Bytes  138011776 bits/sec
[ 14]  5.0- 6.0 sec  17229752 Bytes  137838016 bits/sec
[ 14]  6.0- 7.0 sec  17252920 Bytes  138023360 bits/sec
[ 14]  7.0- 8.0 sec  17281880 Bytes  138255040 bits/sec
[ 14]  8.0- 9.0 sec  17294912 Bytes  138359296 bits/sec
[ 14]  9.0-10.0 sec  17274640 Bytes  138197120 bits/sec
[ 14]  0.0-10.6 sec  177283072 Bytes  133823782 bits/sec
[ 14] MSS size 8948 bytes (MTU 8988 bytes, unknown interface)
RECEIVER END
```

Testing a Path

Example

San Diego CA to Washington DC

OC12 path

Traceroute

```
[phil@damp-ssc phil]$ **nuttcp -xt damp-nrl**
traceroute to damp-nrl-ge (138.18.23.37), 30 hops max, 38 byte packets
 1  ge-0-1-0.sandiego.dren.net (138.18.190.1)  0.255 ms  0.237 ms  0.238 ms
 2  so12-0-0-0.nrldc.dren.net (138.18.1.7)  72.185 ms  72.162 ms  72.164 ms
 3  damp-nrl-ge (138.18.23.37)  72.075 ms  72.069 ms  72.070 ms

traceroute to 138.18.190.5 (138.18.190.5), 30 hops max, 38 byte packets
 1  ge-0-1-0.nrldc.dren.net (138.18.23.33)  0.239 ms  0.199 ms  0.184 ms
 2  so12-0-0-0.sandiego.dren.net (138.18.4.21)  72.210 ms  72.979 ms  72.217 ms
 3  damp-ssc-ge (138.18.190.5)  72.087 ms  72.079 ms  72.068 ms
```

Make sure it goes where you expect!

Check Path RTT and MTU

```
[phil@damp-ssc phil]$ ping damp-nrl
PING damp-nrl-ge (138.18.23.37) from 138.18.190.5 : 56(84) bytes of data.
64 bytes from damp-nrl-ge (138.18.23.37): icmp_seq=1 ttl=62 time=72.0 ms
64 bytes from damp-nrl-ge (138.18.23.37): icmp_seq=2 ttl=62 time=72.0 ms
64 bytes from damp-nrl-ge (138.18.23.37): icmp_seq=3 ttl=62 time=72.0 ms
64 bytes from damp-nrl-ge (138.18.23.37): icmp_seq=4 ttl=62 time=72.0 ms
64 bytes from damp-nrl-ge (138.18.23.37): icmp_seq=5 ttl=62 time=72.0 ms

--- damp-nrl-ge ping statistics ---
5 packets transmitted, 5 received, 0% loss, time 4035ms
rtt min/avg/max/mdev = 72.071/72.082/72.092/0.007 ms
```

```
[phil@damp-ssc phil]$ tracepath damp-nrl
 1?: [LOCALHOST]     pmtu 9000
 1:  ge-0-1-0.sandiego.dren.net (138.18.190.1)            asymm  2   1.060ms
 2:  so12-0-0-0.nrldc.dren.net (138.18.1.7)               asymm  3  72.905ms
 3:  damp-nrl-ge (138.18.23.37)                            72.747ms reached
     Resume: pmtu 9000 hops 3 back 3
```

RTT = 0.072, MSS = 9000-20-20-12 = 8948

Do the Math

- Bandwidth * Delay Product
  - BDP = 622000000/8 * 0.072 = **5.6 MB**
- The TCP receive window needs to be at least this large
- The sender buffer should be twice this size
- We choose 8MB for both
  - Knowing that Linux will double it for us!

Check Buffer Sizes

```
[phil@damp-ssc phil]$ nuttcp -v -t -T10 -w8m damp-nrl
nuttcp-t: v5.1.10: socket
nuttcp-t: buflen=65536, nstream=1, port=5001 tcp -> damp-nrl
nuttcp-t: time limit = 10.00 seconds
nuttcp-t: connect to 138.18.23.37
nuttcp-t: send window size = 16777216, receive window size = 103424
nuttcp-t: 608.0786 MB in 10.00 real seconds = 62288.32 KB/sec = 510.2659 Mbps
nuttcp-t: 9730 I/O calls, msec/call = 1.05, calls/sec = 973.33
nuttcp-t: 0.0user 0.9sys 0:10real 9% 0i+0d 0maxrss 2+0pf 0+0csw

nuttcp-r: v5.1.10: socket
nuttcp-r: buflen=65536, nstream=1, port=5001 tcp
nuttcp-r: accept from 138.18.190.5
nuttcp-r: send window size = 103424, receive window size = 16777216
nuttcp-r: 608.0786 MB in 10.20 real seconds = 61039.55 KB/sec = 500.0360 Mbps
nuttcp-r: 71224 I/O calls, msec/call = 0.15, calls/sec = 6981.97
nuttcp-r: 0.0user 0.5sys 0:10real 5% 0i+0d 0maxrss 3+0pf 0+0csw
```

Things to Check

- The receiver’s (nuttcp-r) receive buffer size
- The transmitter’s (nuttcp-t) send buffer size
- The receiver’s reported throughput
  - The transmitter number is often too high
- The CPU utilization of both sender and receiver
  - Make sure you didn’t run out of CPU

Throughput Test (with -i)

```
[phil@damp-ssc phil]$ nuttcp -t -T10 -i1 -w8m damp-nrl
    1.5531 MB /   1.00 sec =   13.0935 Mbps
   44.3144 MB /   1.00 sec =  371.7978 Mbps        Slow start
   67.9265 MB /   1.00 sec =  569.9192 Mbps
   67.5937 MB /   1.00 sec =  567.1195 Mbps
   67.5339 MB /   1.00 sec =  566.6178 Mbps
   67.5937 MB /   1.00 sec =  567.1184 Mbps
   67.6363 MB /   1.00 sec =  567.4661 Mbps        Stable
   67.5937 MB /   1.00 sec =  567.1195 Mbps
   67.7558 MB /   1.00 sec =  568.4850 Mbps
   67.8753 MB /   1.00 sec =  569.4834 Mbps

  601.0000 MB /  10.19 sec =  494.5603 Mbps 8 %TX 4 %RX
```

Reverse Throughput Test (-r)

[phil@damp-ssc phil]$ **nuttcp -r -T10 -i1 -w8m damp-nrl**

```
  2.2614 MB /   1.00 sec =   18.9842 Mbps
 48.4872 MB /   1.00 sec =  406.7883 Mbps      Slow start
 38.9809 MB /   1.00 sec =  327.0350 Mbps
 29.8672 MB /   1.00 sec =  250.5731 Mbps
 51.3801 MB /   1.00 sec =  431.0547 Mbps
 50.1768 MB /   1.00 sec =  420.9644 Mbps
 24.6874 MB /   1.00 sec =  207.1180 Mbps      Unstable
 15.0616 MB /   1.00 sec =  126.3602 Mbps
 15.8211 MB /   1.00 sec =  132.7315 Mbps
 16.5891 MB /   1.00 sec =  139.1766 Mbps

303.6979 MB /  10.60 sec =  240.3490 Mbps 4 %TX 2 %RX
```

UDP Test (-u –l –R)

[phil@damp-ssc phil]$ **nuttcp -t -u -l8000 -R500m -T10 -i1 –w8m damp-nrl**

```
59.2346 MB /   0.99 sec =  499.6586 Mbps     0 /  7764 ~drop/pkt  0.00 ~%loss
59.6008 MB /   1.00 sec =  500.0570 Mbps     0 /  7812 ~drop/pkt  0.00 ~%loss
59.5932 MB /   1.00 sec =  499.9935 Mbps     0 /  7811 ~drop/pkt  0.00 ~%loss
59.5932 MB /   1.00 sec =  500.0000 Mbps     0 /  7811 ~drop/pkt  0.00 ~%loss
59.5932 MB /   1.00 sec =  499.9950 Mbps     0 /  7811 ~drop/pkt  0.00 ~%loss
59.5932 MB /   1.00 sec =  499.9905 Mbps     0 /  7811 ~drop/pkt  0.00 ~%loss
59.5932 MB /   1.00 sec =  499.9925 Mbps     0 /  7811 ~drop/pkt  0.00 ~%loss
59.5932 MB /   1.00 sec =  499.9950 Mbps     0 /  7811 ~drop/pkt  0.00 ~%loss
59.5932 MB /   1.00 sec =  499.9930 Mbps     0 /  7811 ~drop/pkt  0.00 ~%loss
59.5932 MB /   1.00 sec =  499.9920 Mbps     0 /  7811 ~drop/pkt  0.00 ~%loss

595.9778 MB /  10.01 sec =  499.5061 Mbps 99 %TX 3 %RX 0 / 78116 drop/pkt
   0.00 %loss
```

Reverse UDP Test

```
[phil@damp-ssc phil]$ nuttcp -r -u -l8000 -R500m -T10 -i1 –w8m damp-nrl
  56.8237 MB /   0.99 sec =  481.1408 Mbps     0 /  7448 ~drop/pkt  0.00 ~%loss
  59.5169 MB /   1.00 sec =  499.3279 Mbps     1 /  7802 ~drop/pkt  0.01 ~%loss
  59.2651 MB /   1.00 sec =  497.2097 Mbps     0 /  7768 ~drop/pkt  0.00 ~%loss
  59.0591 MB /   1.00 sec =  495.4825 Mbps     0 /  7741 ~drop/pkt  0.00 ~%loss
  58.9828 MB /   1.00 sec =  494.8439 Mbps     0 /  7731 ~drop/pkt  0.00 ~%loss
  60.7300 MB /   1.00 sec =  509.5001 Mbps     0 /  7960 ~drop/pkt  0.00 ~%loss
  59.4482 MB /   1.00 sec =  498.7424 Mbps     2 /  7794 ~drop/pkt  0.03 ~%loss
  59.7839 MB /   1.00 sec =  501.5677 Mbps     2 /  7838 ~drop/pkt  0.03 ~%loss
  58.8760 MB /   1.00 sec =  493.9463 Mbps     0 /  7717 ~drop/pkt  0.00 ~%loss
  59.8526 MB /   1.00 sec =  502.1347 Mbps     0 /  7845 ~drop/pkt  0.00 ~%loss

 595.8939 MB /  10.01 sec =  499.4702 Mbps 100 %TX 3 %RX 5 / 78110
    drop/pkt 0.01 %loss
```

Is Loss the Limiter?

bps = 0.7 * MSS / (rtt * sqrt(loss))

= 0.7 * 8948*8 / (0.072 * sqrt(5/78110))

= 87 Mbps

Conclusion: Too much loss on DC to CA path

Finding Packet Loss

- Set up firewall filters in the WAN routers to count nuttcp packets
- Example nuttcp
  - nuttcp -n10000 –u –l1400 –Ri50m –i1 *dest*
  - nuttcp -n10000 –u –l1400 –Ri50m –i1 –c1 *dest*
- Look at WAN counters for 10008 packets
- This usually localizes the problem to the source site, WAN, or destination site
- Then you start moving things around

Testing Notes

- When UDP testing
  - Be careful of fragmentation (path MTU)
  - Most systems are better at TCP than UDP
  - Setting large socket buffers helps
  - Raising process priority helps
  - Duplex problems don't show up with UDP!
- Having debugged test points is critical
  - Local and remote
- A performance history is valuable

Duplex Problem Symptoms

- Will never see it with pings or UDP tests
- TCP throughput is sometimes ½ of expected rate, sometimes under 1 Mbps
- Worse performance usually results when the mismatch is near the receiver of a TCP flow
- Sometimes reported as “late collisions”

Checking Systems for Errors

- Don't forget
  - **ifconfig**

    UP BROADCAST RUNNING MULTICAST MTU:9000 Metric:1

    RX packets:414639041 errors:0 dropped:0 overruns:0 frame:0

    TX packets:378926231 errors:0 dropped:0 overruns:0 carrier:0

    collisions:0 txqueuelen:1000
  - **netstat –s**

    Tcp: 59450 segments retransmited
  - **dmesg** or /var/log

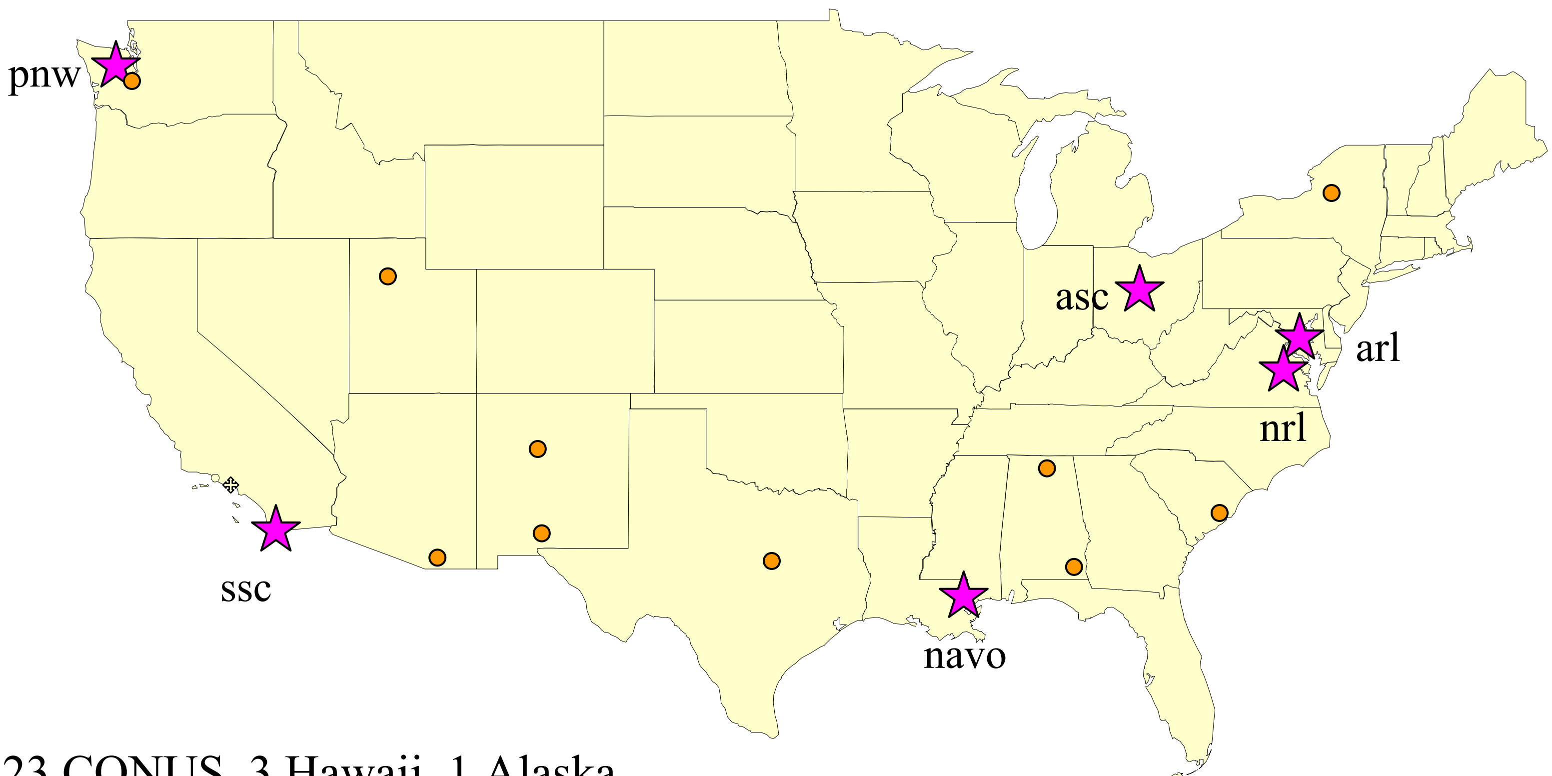
    ixgb: eth2 NIC Link is Up 10000 Mbps Full Duplex

Example Measurement Infrastructure

DREN Active Measurement Program (DAMP)

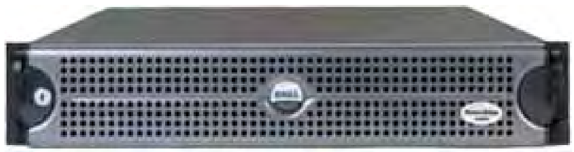
- Grew out of the NLANR AMP project
  - http://amp.nlanr.net/
- Long term active performance monitoring
  - Delay, loss, routes, throughput
  - Dedicated hardware; known performance
  - User accessible test points
- Invaluable at identifying and debugging problems

DREN AMP Systems



23 CONUS, 3 Hawaii, 1 Alaska

10GigE Test Systems



- Dell 2650, 3.0 GHz Xeon, 533 MHz FSB, 512 KB L2 cache, 1 MB L3 cache, PCI-X
  - Linux 2.4.26-web100
  - Two built-in BCM5703 10/100/1000 copper NICs, tg3 driver
  - Intel PXLA8590LR 10GigE NIC, ixgb 1.0.65 driver
- 4.6 Gbps TCP on LAN after tuning (7.7 Gbps loopback)
  - 2.2 Gbps TCP from MD to MS over OC48
- New NICs: Intel, Neterion X-Frame, Chelsio T110/N110

Precise Time

- 10 msec NTP accuracy w/o clock
  - asymmetric paths and clock hopping
- 10 usec NTP accuracy with attached clock
  - Serial port connection, tagged events
  - GPS: Trimble Acutime 2000 kit
  - CDMA: EndRun Technologies Praecis Ct

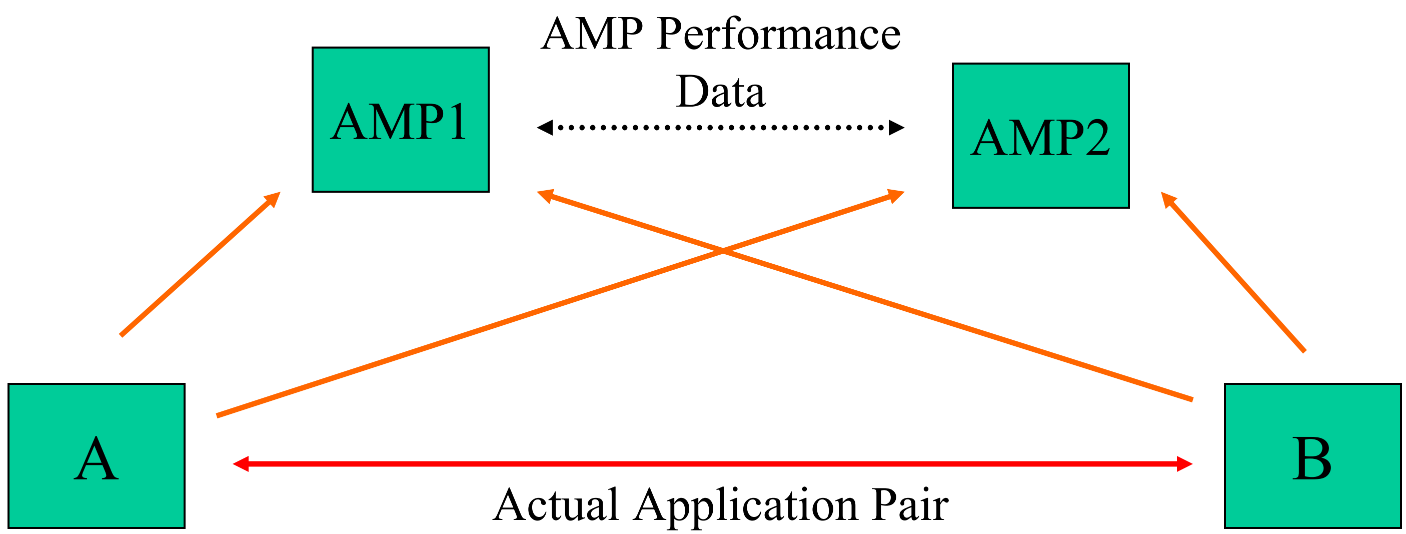
AMP Tests on Demand

- Web page interface
- 10 second TCP throughput test
  - Uses nuttcp third party support
- Can also access them directly

Need to test over a distance

- Round trip time amplifies most problems
- Many **local problems** won’t be seen over a short round trip time (i.e. within your site!)
  - Window size won’t matter
  - TCP error recovery is extremely quick
- Firewalls and screening routers (ACLs) could be part of the problem

Test Pairs



Problems We Have Found

- Small windows on end systems
- Duplex problems
- Bad cables (patch panels, fiber, cat5)
- Faulty switches, router modules, or NICs
- Slow firewalls or routers
  - Excessive logging
- Jumbo frame issues / mismatches
- Flow control issues
- WAN bit errors
- Routing issues
- Heat problems!

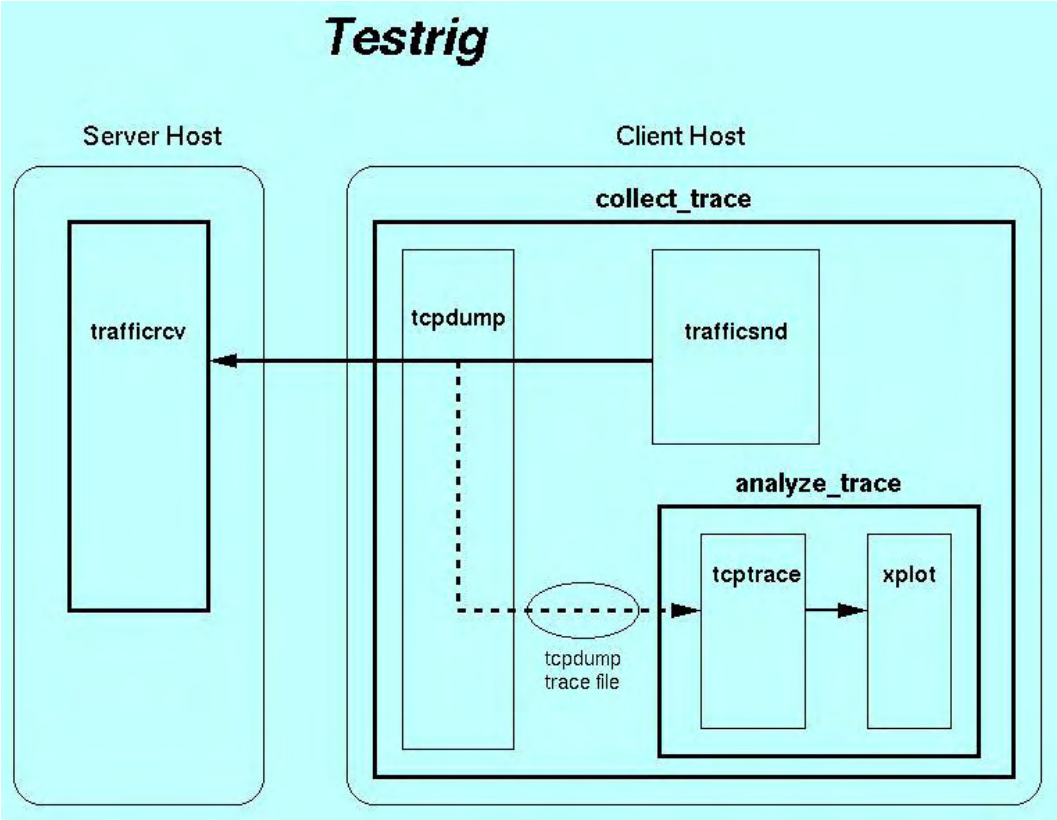
Advanced Debugging

TCP Traces and Tools

TCP/IP Analysis Tools

- tcpdump
  - www.tcpdump.org
- ethereal - GUI tcpdump (protocol analyzer)
  - www.ethereal.com
- tcptrace – stats/graphs of tcpdump data
  - www.tcptrace.org
- xplot – for displaying tcptrace graphs

"A Preconfigured TCP Test Rig"



- http://www.ncne.org/research/tcp/testrig/

Collecting A TCP Trace

tcpdump -p -w *trace.out* -s 100 host *$desthost* and port 5001 &

nuttcp -T10 -n200m -i1 -w10m *$desthost*

tcptrace -l -r *trace.out*

tcptrace -S -zxy *trace.out* (or –G for all graphs)

xplot *.xpl

tcptrace -l -r

```
TCP connection 1:
        host a:        damp-nrl-ge:58310
        host b:        damp-ssc-ge:5001
        complete conn: no       (SYNs: 2)  (FINs: 0)
        first packet:  Wed Oct 13 17:30:30.272579 2004
        last packet:   Wed Oct 13 17:30:39.289916 2004
        elapsed time:  0:00:09.017336
        total packets: 36063
        filename:      nrl-ssc.trace
   a->b:                              b->a:
     total packets:         23473           total packets:         12590
     ack pkts sent:         23472           ack pkts sent:         12590
     pure acks sent:            1           pure acks sent:        12589
     sack pkts sent:            0           sack pkts sent:         1774
     dsack pkts sent:           0           dsack pkts sent:           0
     max sack blks/ack:         0           max sack blks/ack:         1
     unique bytes sent: 209714276           unique bytes sent:         0
     actual data pkts:      23471           actual data pkts:          0
     actual data bytes: 210018508           actual data bytes:         0
     rexmt data pkts:          34           rexmt data pkts:           0
     rexmt data bytes:     304232           rexmt data bytes:          0
     zwnd probe pkts:           0           zwnd probe pkts:           0
     zwnd probe bytes:          0           zwnd probe bytes:          0
     outoforder pkts:           0           outoforder pkts:           0
     pushed data pkts:        555           pushed data pkts:          0
```

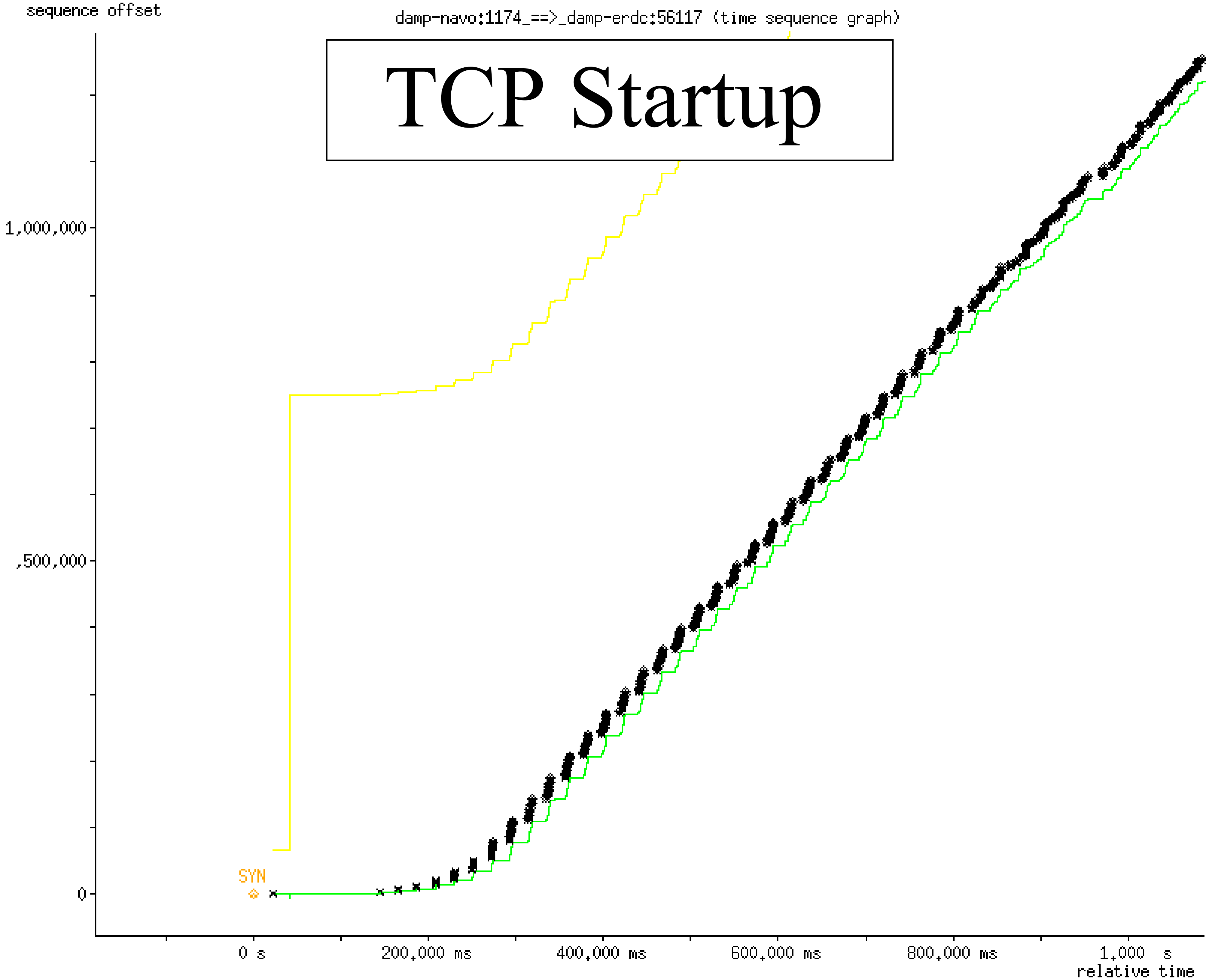
tcptrace -l -r (cont.)

```
SYN/FIN pkts sent:        1/0          SYN/FIN pkts sent:        1/0
req 1323 ws/ts:           Y/Y          req 1323 ws/ts:           Y/Y
adv wind scale:             7          adv wind scale:             7
req sack:                   Y          req sack:                   Y
sacks sent:                 0          sacks sent:              1774
urgent data pkts:           0 pkts     urgent data pkts:           0 pkts
urgent data bytes:          0 bytes    urgent data bytes:          0 bytes
mss requested:           8960 bytes    mss requested:           8960 bytes
max segm size:           8948 bytes    max segm size:              0 bytes
min segm size:           8948 bytes    min segm size:              0 bytes
avg segm size:           8947 bytes    avg segm size:              0 bytes
max win adv:            17920 bytes    max win adv:          8388480 bytes
min win adv:            17920 bytes    min win adv:            17792 bytes
zero win adv:               0 times    zero win adv:               0 times
avg win adv:            17920 bytes    avg win adv:          8044910 bytes
initial window:         17896 bytes    initial window:             0 bytes
initial window:             2 pkts     initial window:             0 pkts
ttl stream length:         NA          ttl stream length:         NA
missed data:               NA          missed data:               NA
truncated data:     209220494 bytes    truncated data:             0 bytes
truncated packets:      23471 pkts     truncated packets:          0 pkts
data xmit time:         8.945 secs     data xmit time:         0.000 secs
idletime max:           141.4 ms       idletime max:            73.8 ms
throughput:          23256786 Bps      throughput:                 0 Bps
```

tcptrace -l -r (cont.)

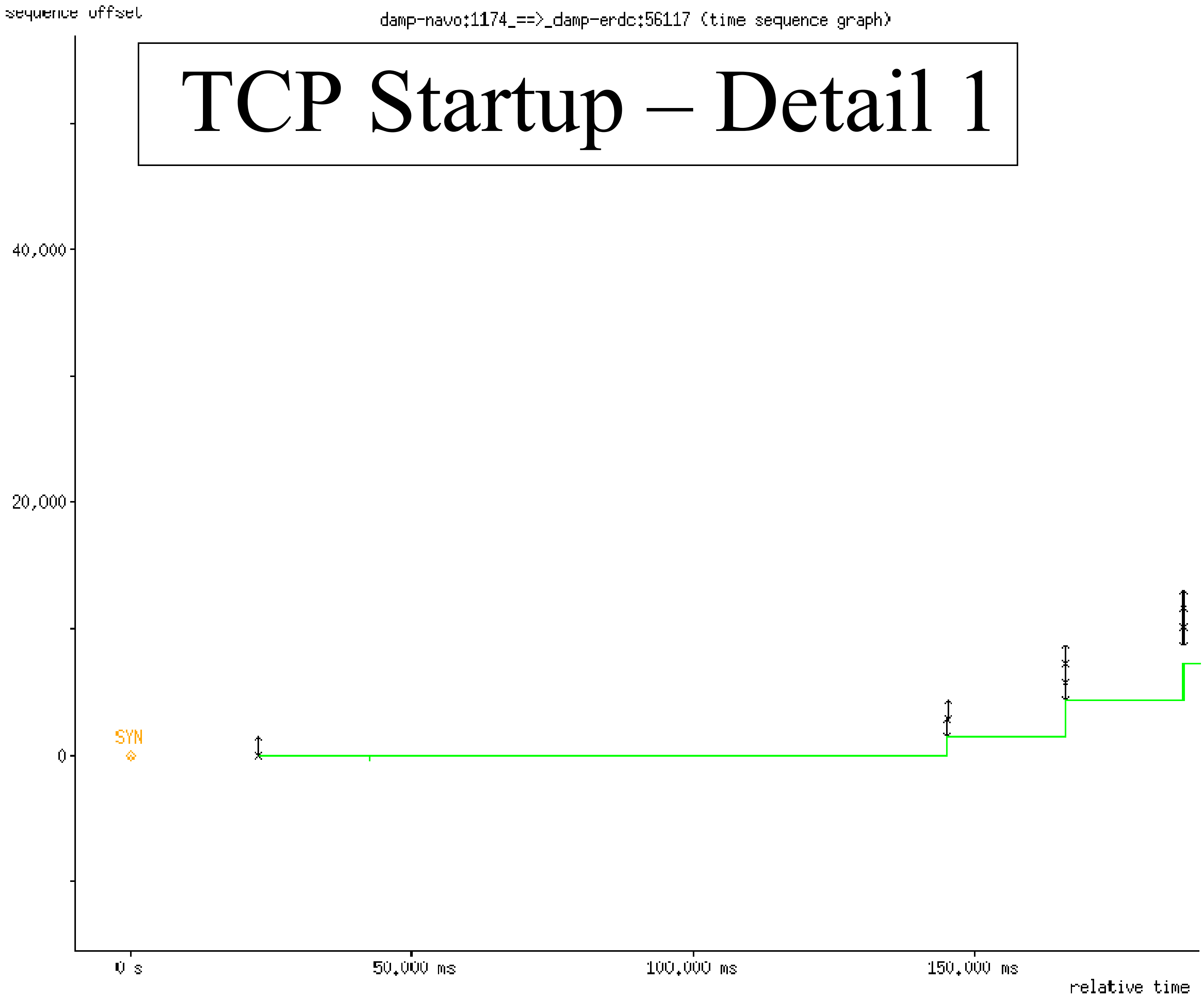
```
RTT samples:            10814          RTT samples:              1
RTT min:                  72.1 ms      RTT min:                0.0 ms
RTT max:                 144.2 ms      RTT max:                0.0 ms
RTT avg:                 102.1 ms      RTT avg:                0.0 ms
RTT stdev:                28.9 ms      RTT stdev:              0.0 ms

RTT from 3WHS:            72.1 ms      RTT from 3WHS:          0.0 ms

RTT full_sz smpls:      10813          RTT full_sz smpls:        1
RTT full_sz min:          72.7 ms      RTT full_sz min:        0.0 ms
RTT full_sz max:         144.2 ms      RTT full_sz max:        0.0 ms
RTT full_sz avg:         102.1 ms      RTT full_sz avg:        0.0 ms
RTT full_sz stdev:        28.9 ms      RTT full_sz stdev:      0.0 ms

post-loss acks:             4          post-loss acks:           0
     For the following 5 RTT statistics, only ACKs for
     multiply-transmitted segments (ambiguous ACKs) were
     considered.  Times are taken from the last instance
     of a segment.
ambiguous acks:            30          ambiguous acks:           0
RTT min (last):           72.7 ms      RTT min (last):         0.0 ms
RTT max (last):           74.9 ms      RTT max (last):         0.0 ms
RTT avg (last):           73.3 ms      RTT avg (last):         0.0 ms
RTT sdv (last):            0.8 ms      RTT sdv (last):         0.0 ms
segs cum acked:         12508          segs cum acked:           0
duplicate acks:          1742          duplicate acks:           0
triple dupacks:             4          triple dupacks:           0
max # retrans:              1          max # retrans:            0
min retr time:            73.7 ms      min retr time:          0.0 ms
max retr time:           140.2 ms      max retr time:          0.0 ms
avg retr time:            89.2 ms      avg retr time:          0.0 ms
sdv retr time:            10.3 ms      sdv retr time:          0.0 ms
```

TCP Startup

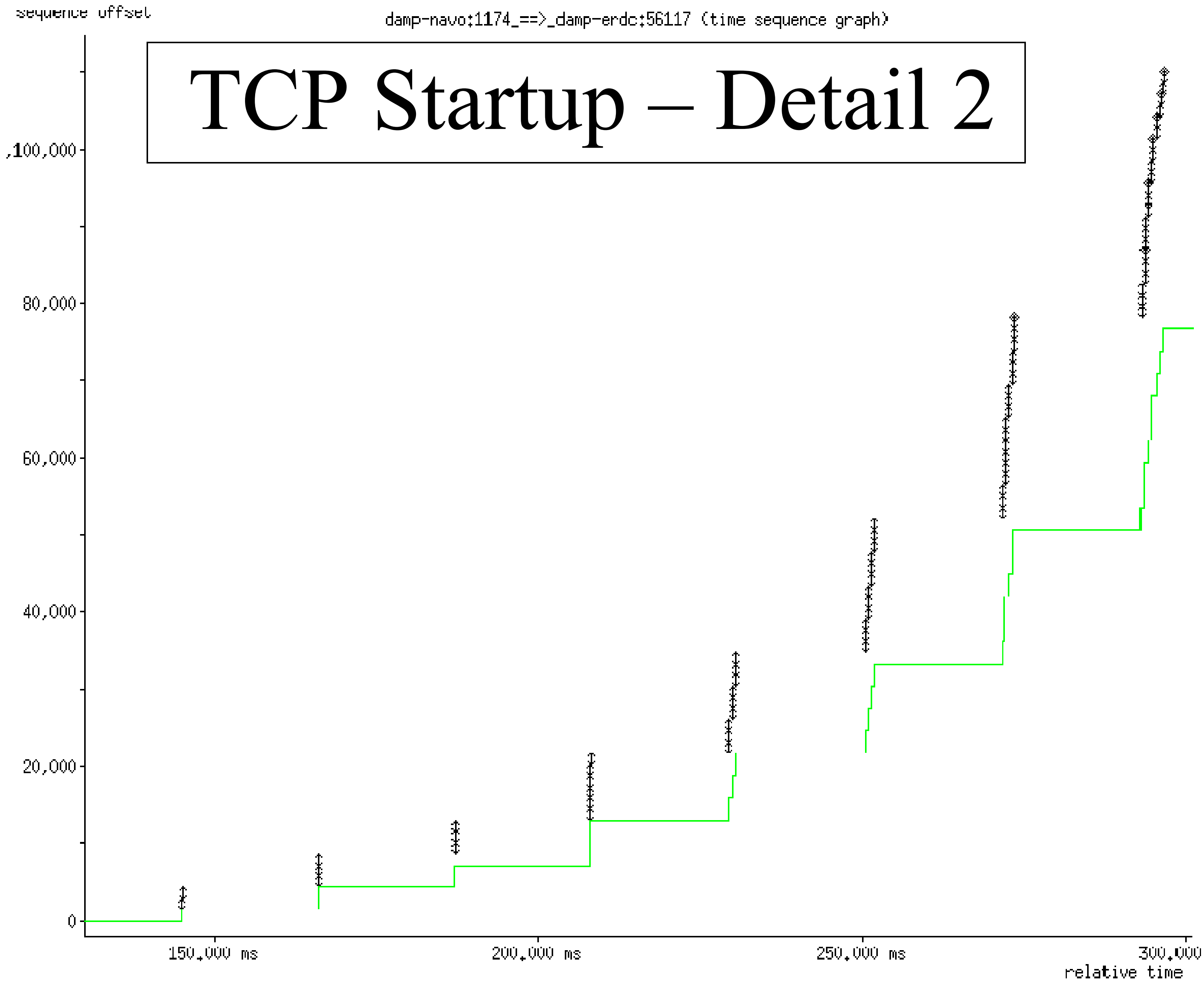


sequence offset
damp-navo:1174_==>_damp-erdc:56117 (time sequence graph)
1,000,000
,500,000
0
SYN
0 s
200,000 ms
400,000 ms
600,000 ms
800,000 ms
1,000 s
relative time

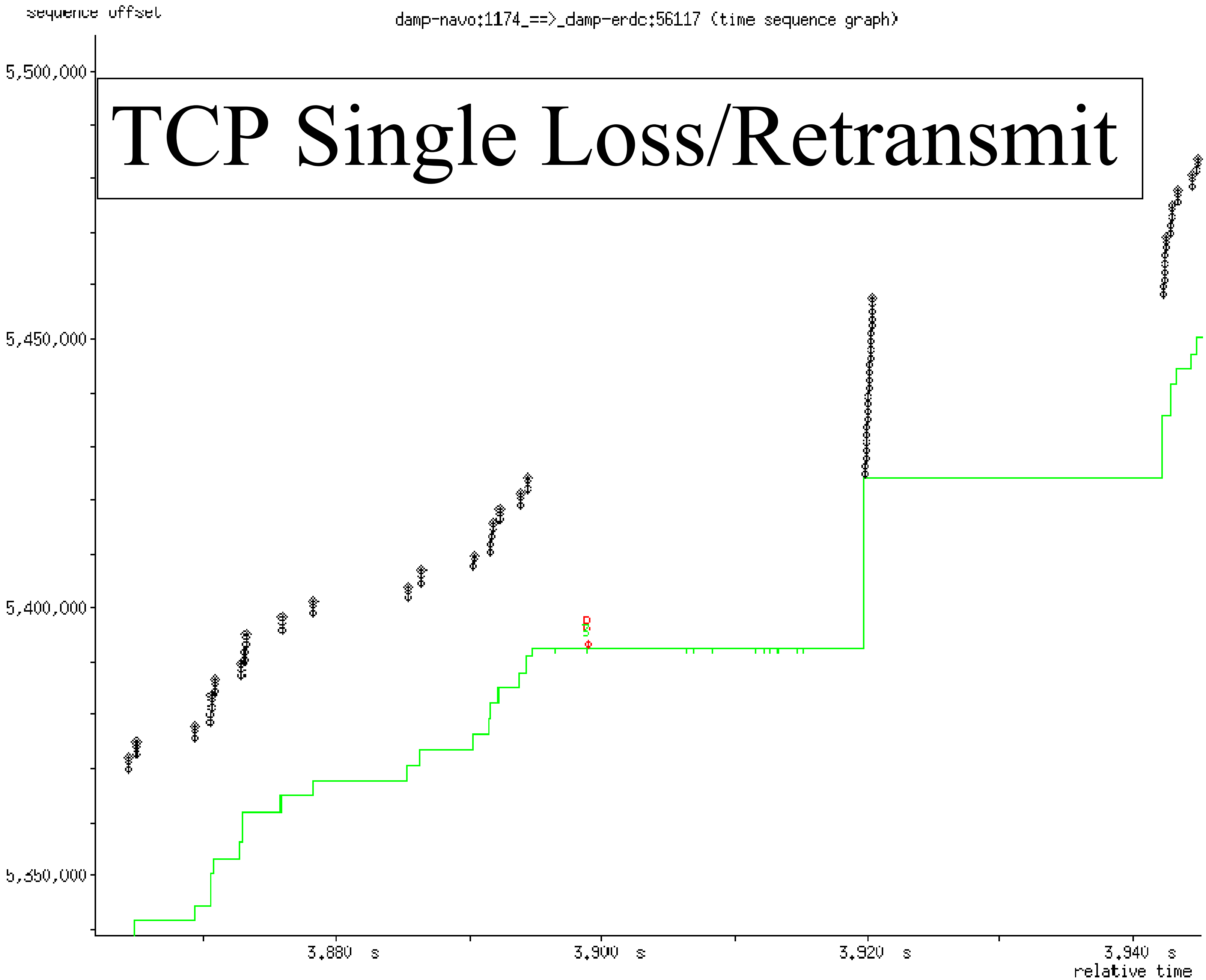
TCP Startup – Detail 1

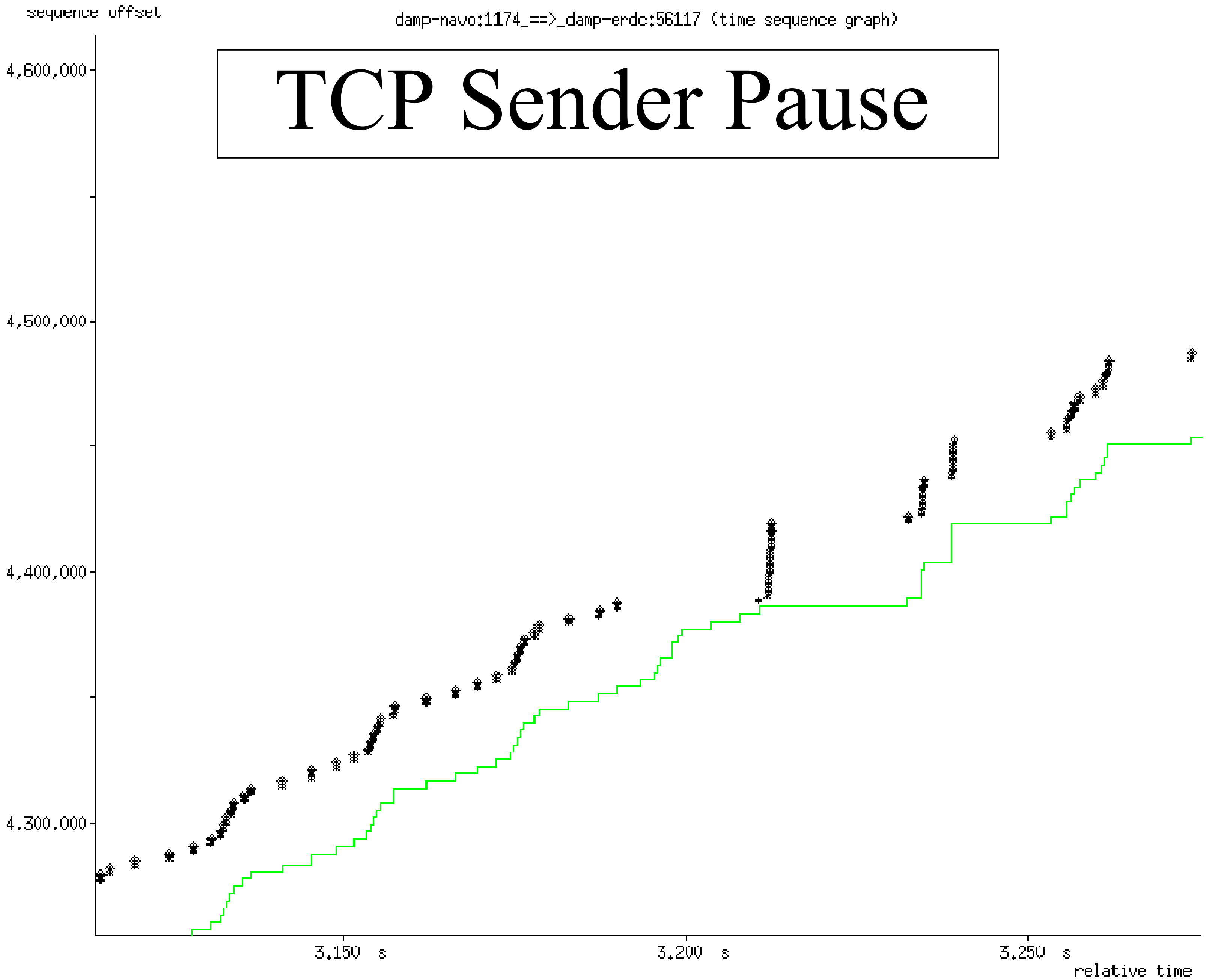


TCP Startup – Detail 2

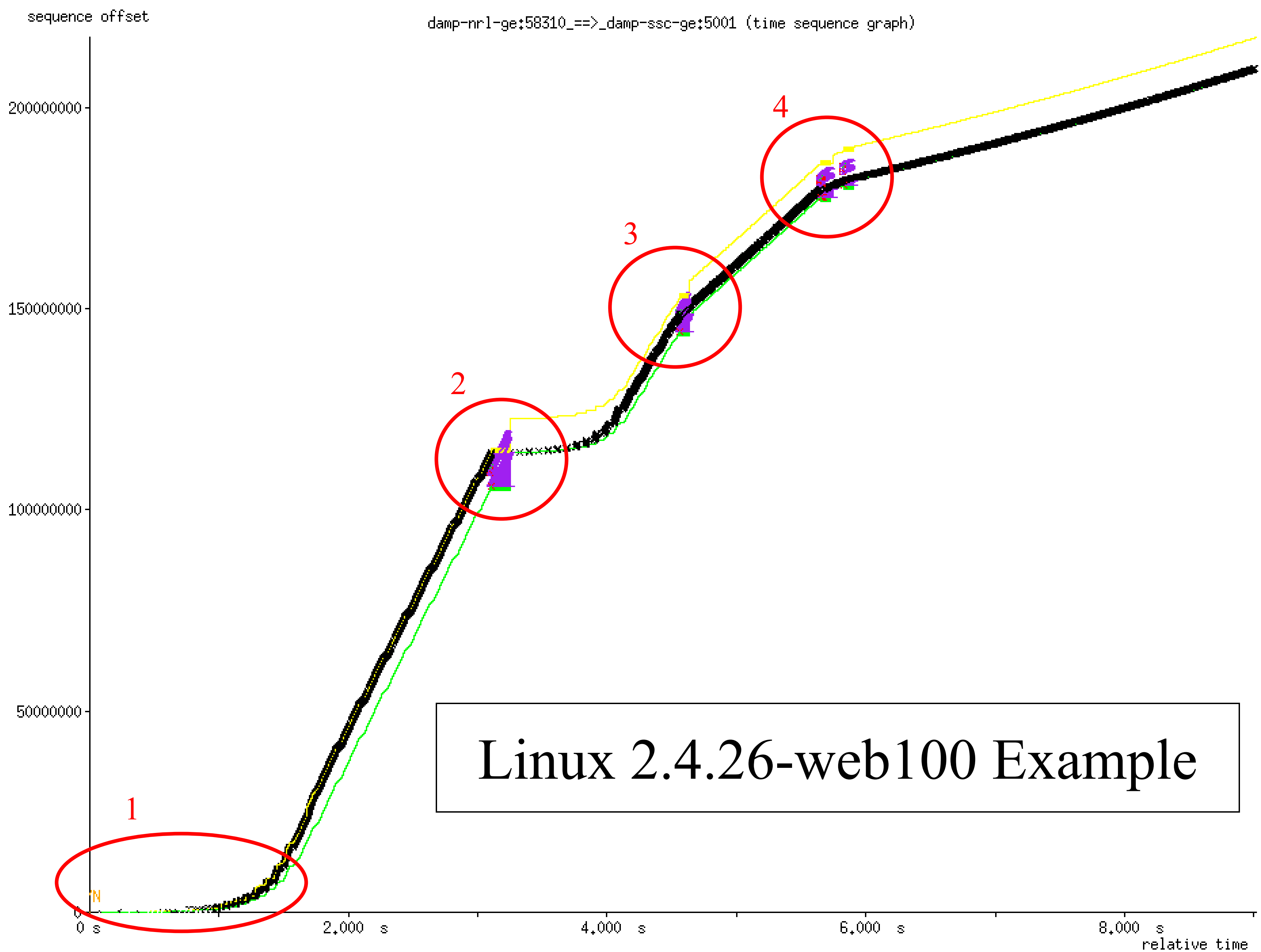


TCP Single Loss/Retransmit

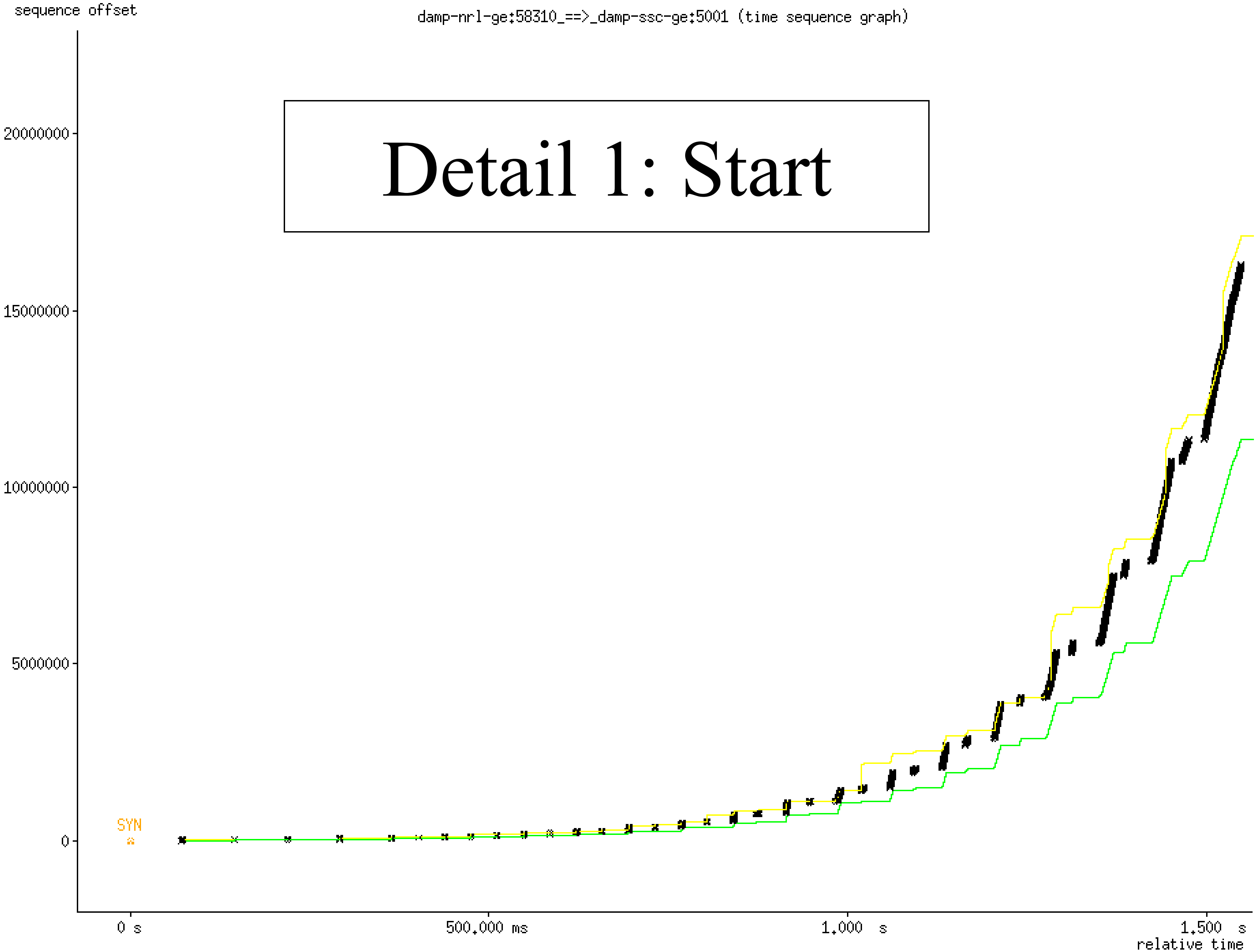




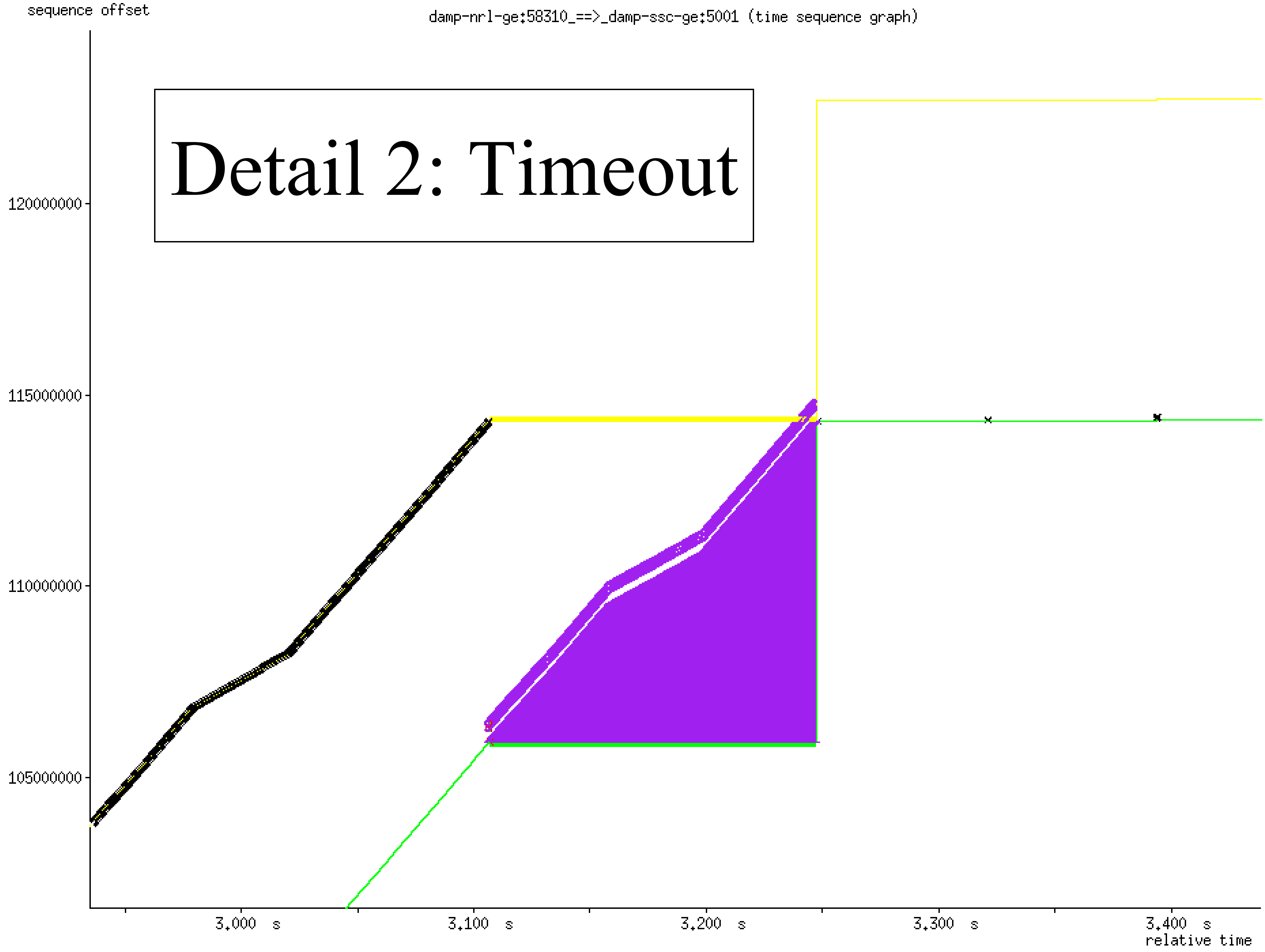
sequence offset
damp-navo:1174_==>_damp-erdc:56117 (time sequence graph)
TCP Sender Pause
4,600,000
4,500,000
4,400,000
4,300,000
3.150 s
3.200 s
3.250 s
relative time


sequence offset
damp-nrl-ge:58310_==>_damp-ssc-ge:5001 (time sequence graph)
200000000
150000000
100000000
50000000
0
0 s
2,000 s
4,000 s
6,000 s
8,000 s
relative time
1
2
3
4
Linux 2.4.26-web100 Example

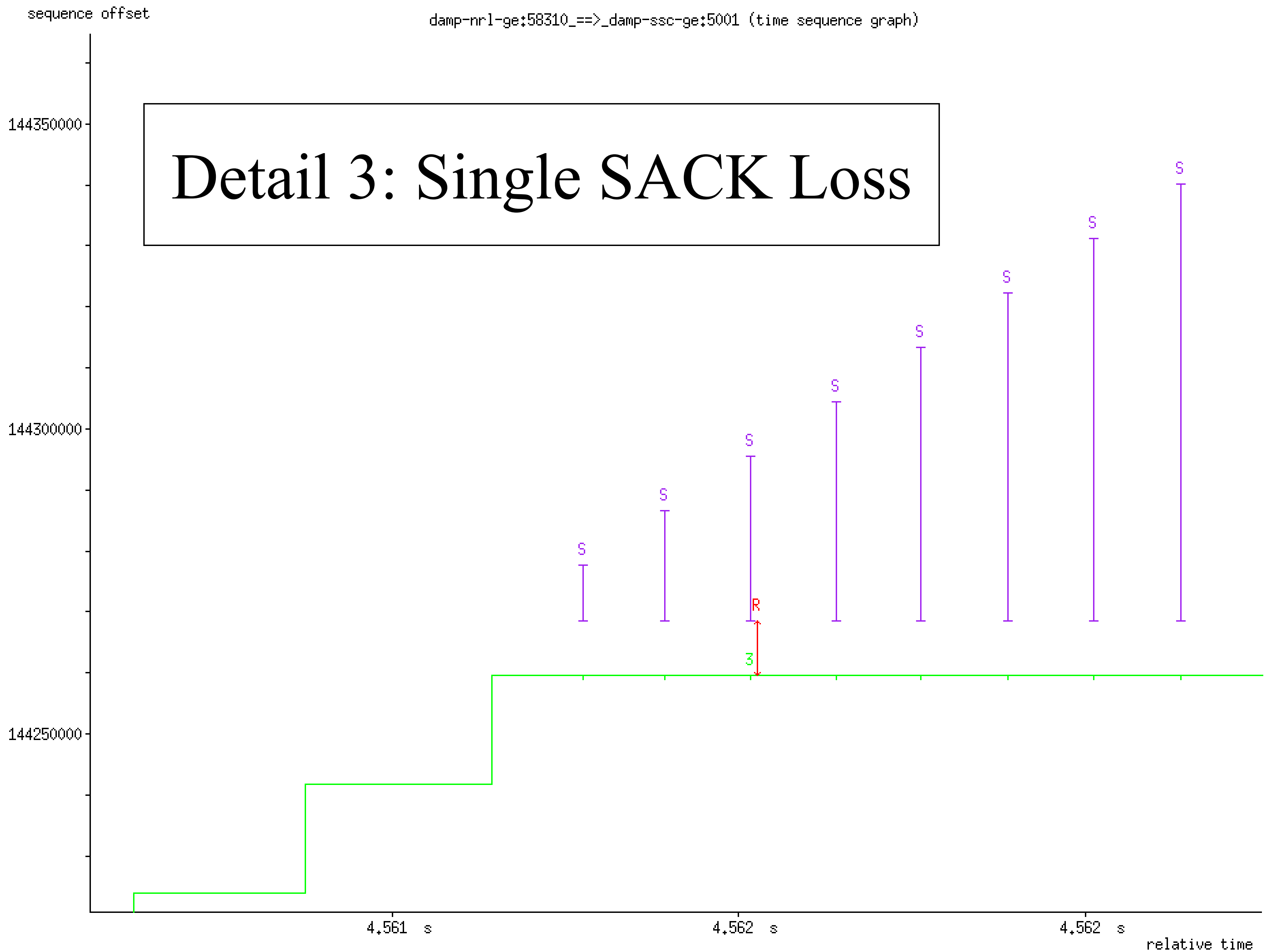
Detail 1: Start



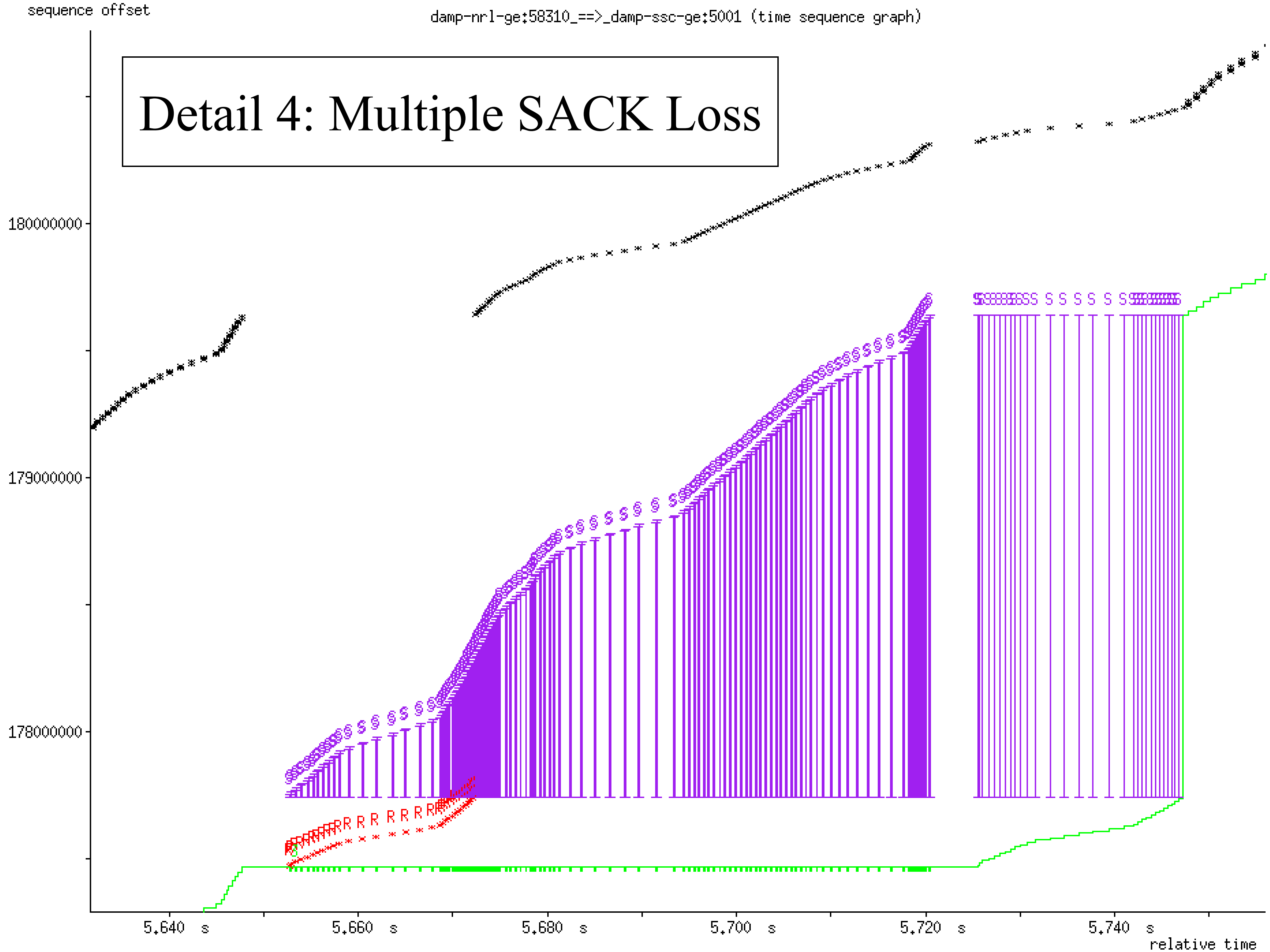
Detail 2: Timeout



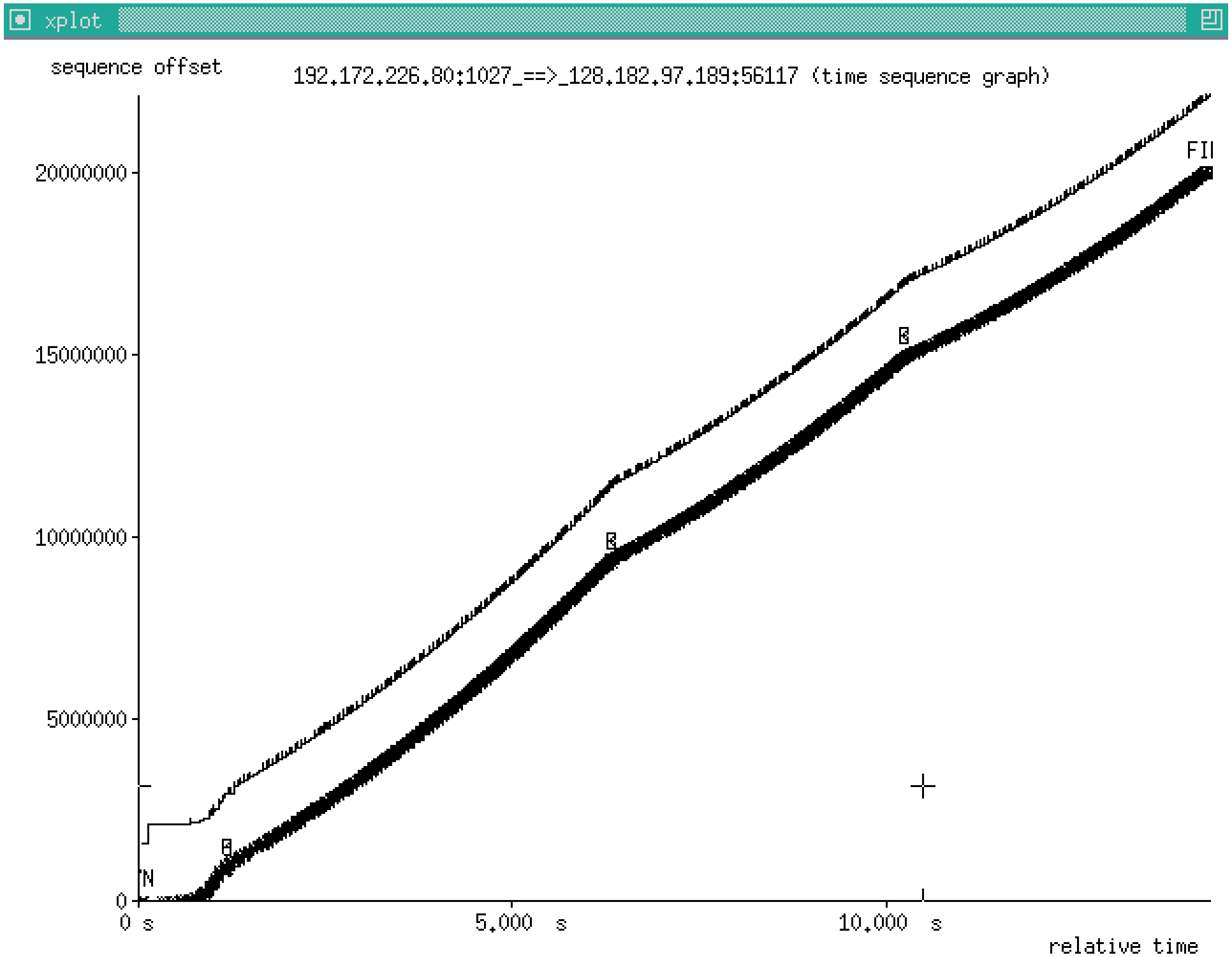
Detail 3: Single SACK Loss



Detail 4: Multiple SACK Loss

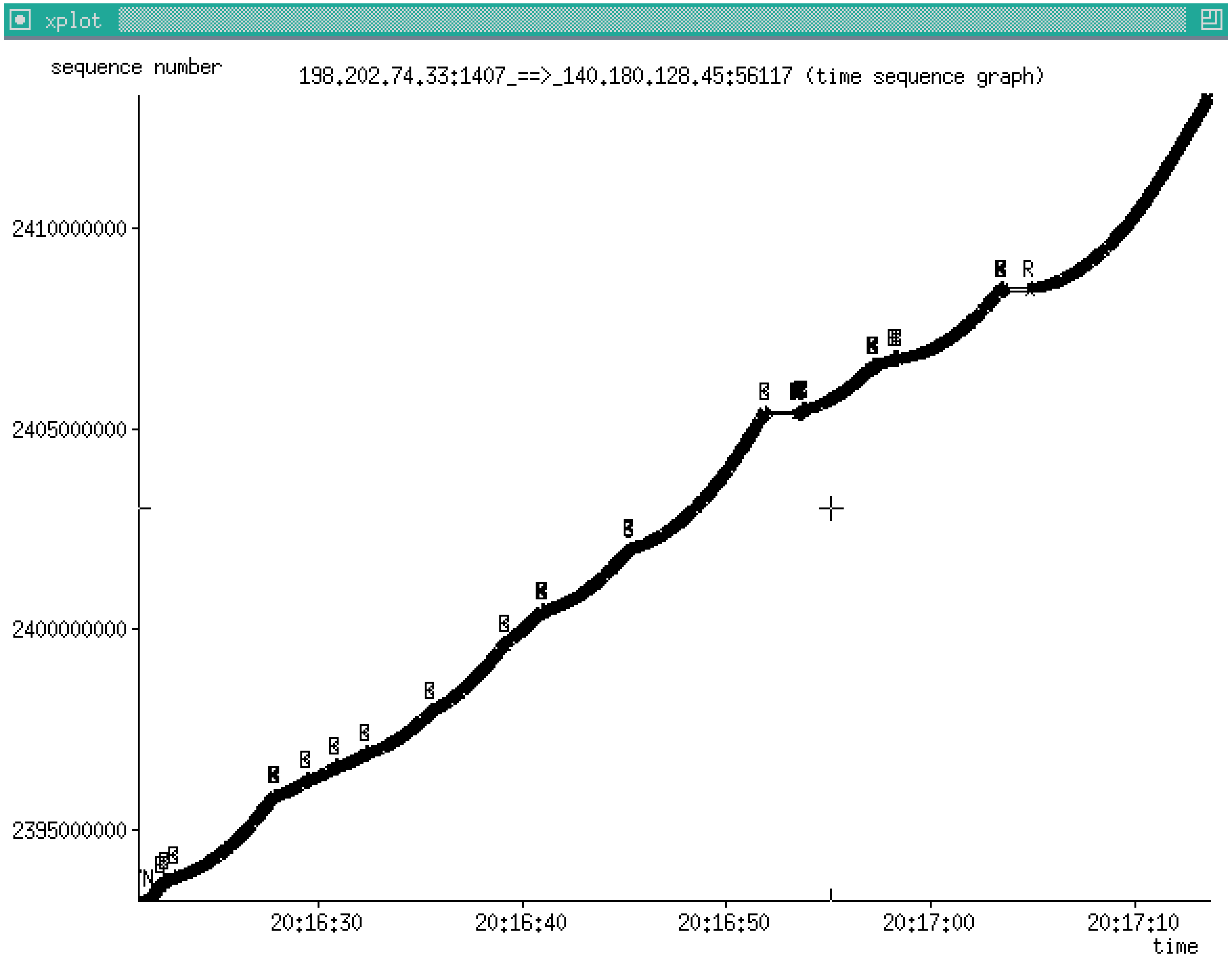


Normal TCP Scallops



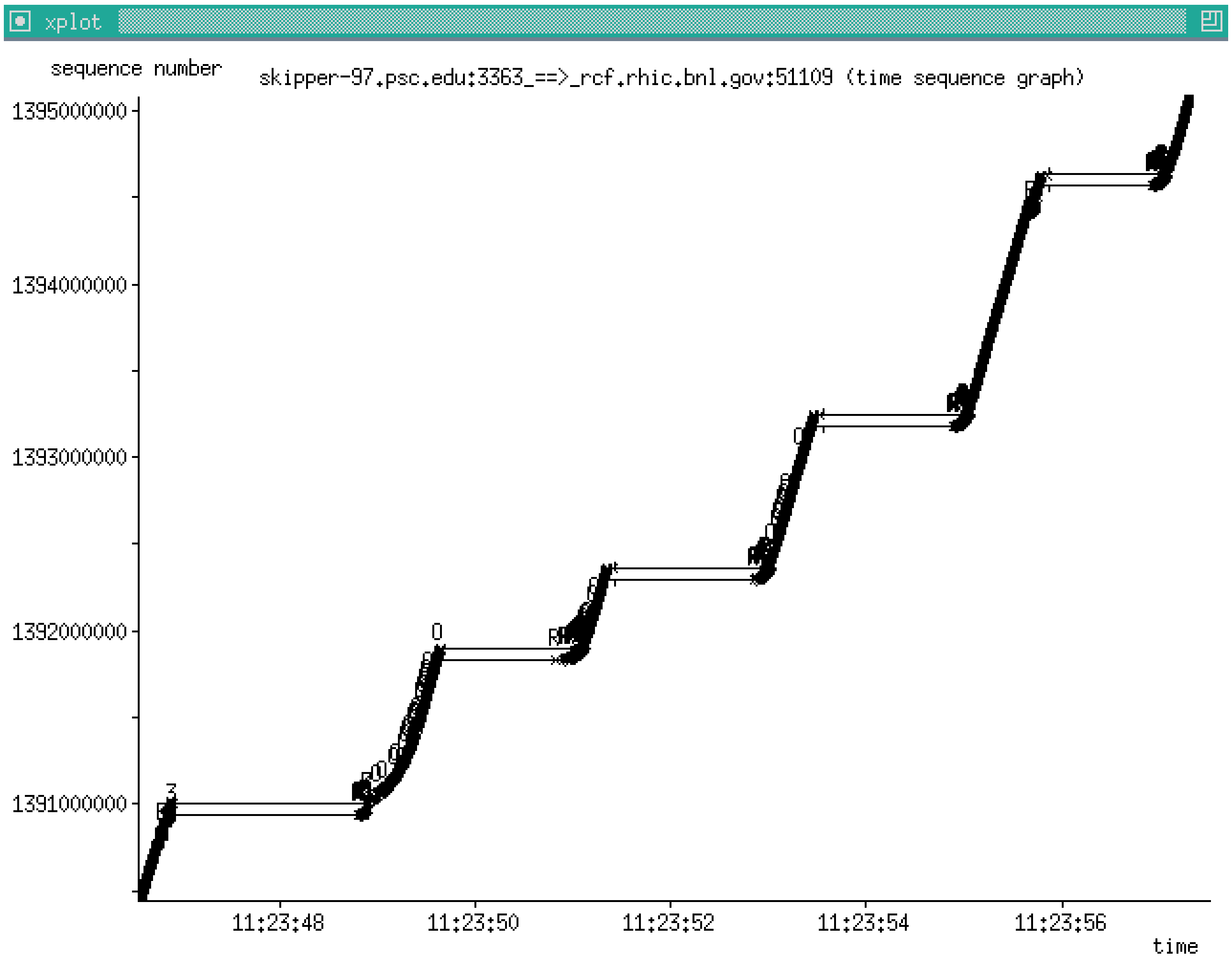
NLANR
NCNE

A Little More Loss



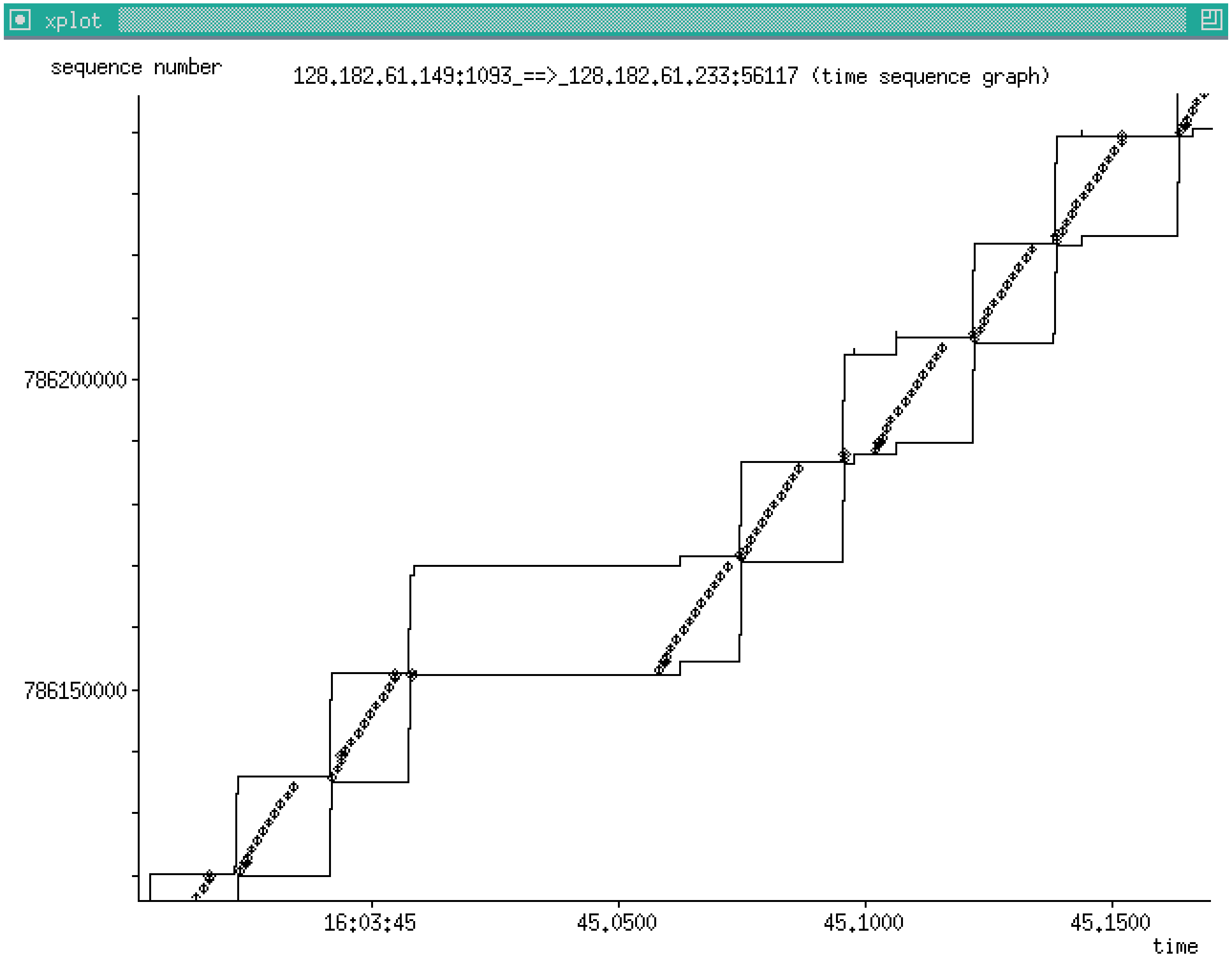
NLANR
NCNE

Excessive Timeouts



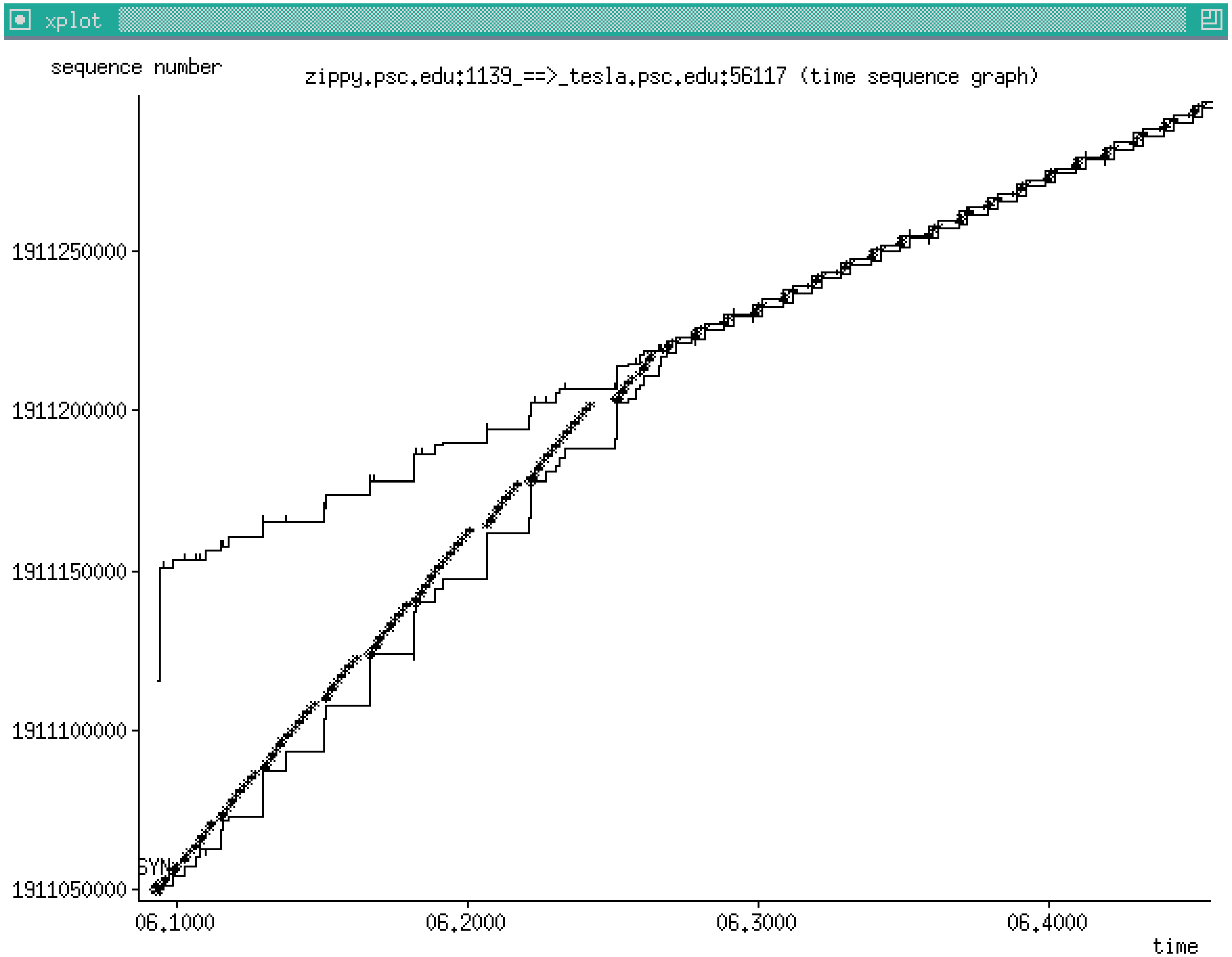
NLANR
NCNE

Bad Window Behavior



NLANR
NCNE

Receiving Host/App Too Slow



NLANR
NCNE

Web100

www.web100.org

- Set out to make 100 Mbps TCP common
- “TCP knows what’s wrong with the network”
  - The sender side knows the most
- Instruments the TCP stack for diagnostics
- Enhanced TCP MIB (IETF Draft)
- Linux 2.6 kernel patches + library and tools
- /proc/web100 file system

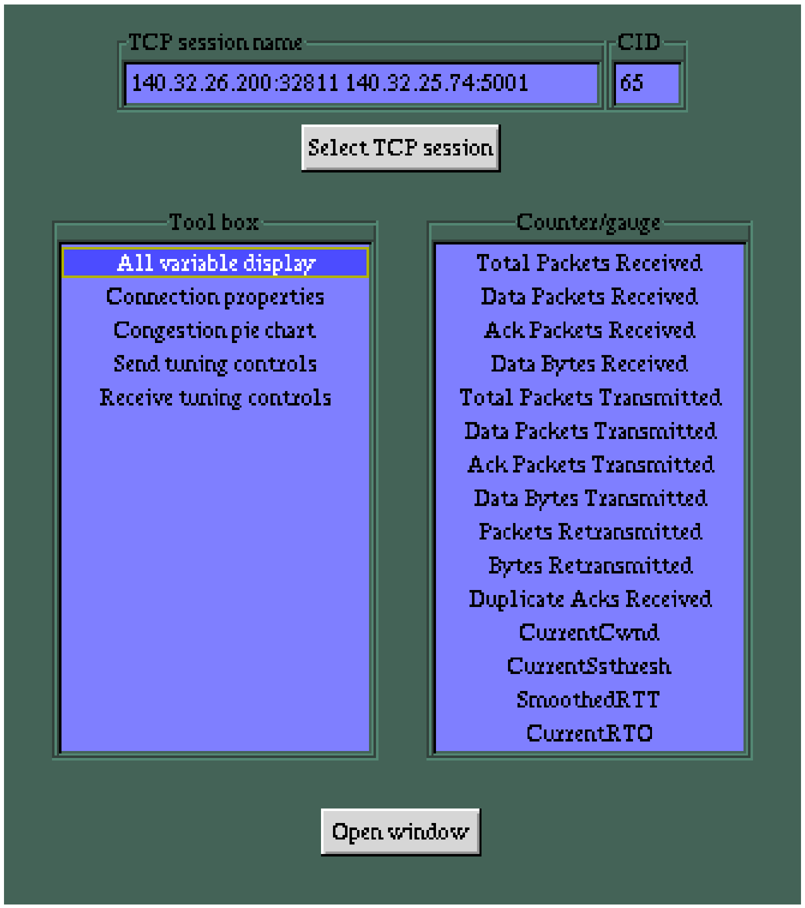
  e.g. /proc/web100/1010/{read,spec,spec-ascii,test,tune)

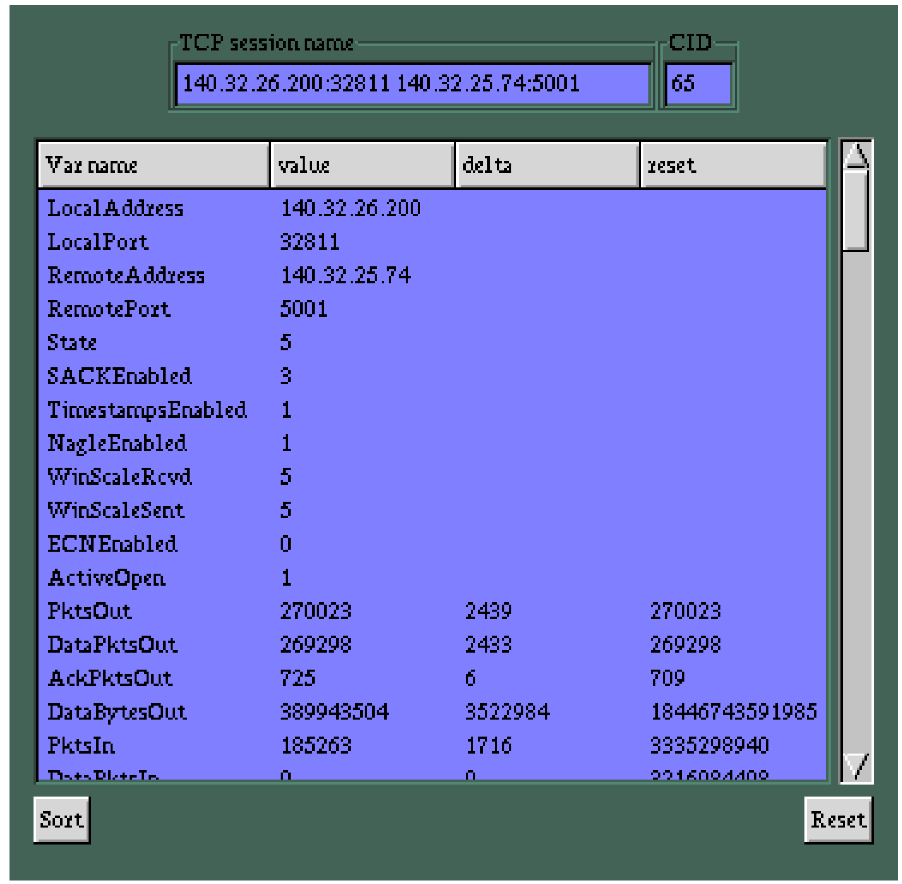
Web100 – Connection Selection

cmdline	pid	local add	local port	remote add	remote port	cid	state
	0	127.0.0.1	9753	127.0.0.1	32814	69	___
	0	127.0.0.1	32814	127.0.0.1	9753	68	___
nuttcp	2338	140.32.26.200	32811	140.32.25.74	5001	65	ESTBLSH
nuttcp	2338	140.32.26.200	32810	140.32.25.74	5000	64	ESTBLSH
nuttcp	2338	140.32.26.200	32810	140.32.25.74	5000	64	ESTBLSH
	0	127.0.0.1	6010	127.0.0.1	32796	49	ESTBLSH
gutil	2173	127.0.0.1	32796	127.0.0.1	6010	48	ESTBLSH
	0	192.168.0.1	22	192.168.0.100	60642	47	___
	0	140.32.26.200	1090	198.253.246.7	4635	33	ESTBLSH
	0	140.32.26.200	1090	63.196.71.245	1834	32	ESTBLSH
	0	140.32.26.200	1090	198.97.151.50	1034	31	ESTBLSH
	0	140.32.26.200	1090	140.32.26.61	34694	30	ESTBLSH
	0	140.32.26.200	1090	132.250.100.154	38024	29	ESTBLSH
	0	140.32.26.200	1090	199.165.80.6	39728	24	ESTBLSH
	0	140.32.26.200	1090	204.222.178.138	1067	23	ESTBLSH
	0	140.32.26.200	1090	140.31.150.2	1041	20	ESTBLSH

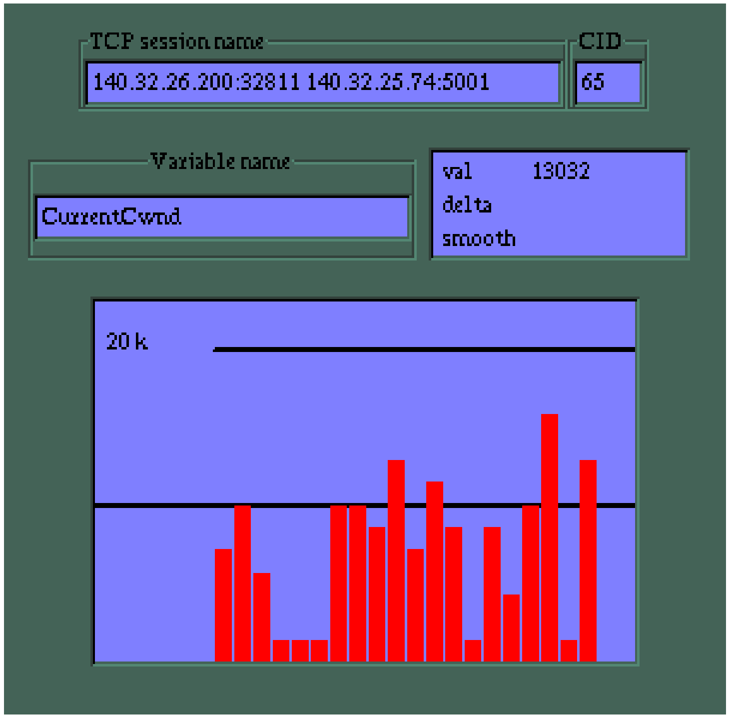
Sort entries by: ◇ cmdline ◇ local port ◇ remote add ◇ remote port ◇ cid

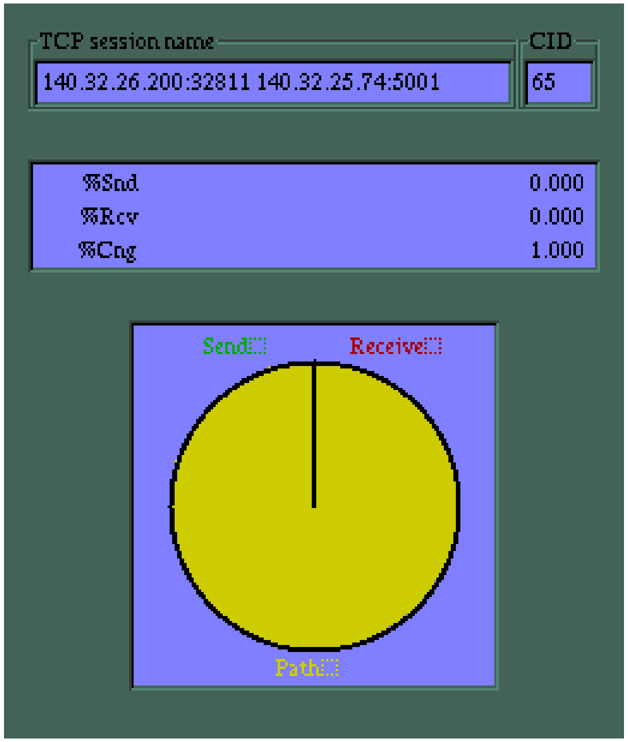
Web100 - Tool/Variable Selection





Web100 – Variable Display, Triage Chart





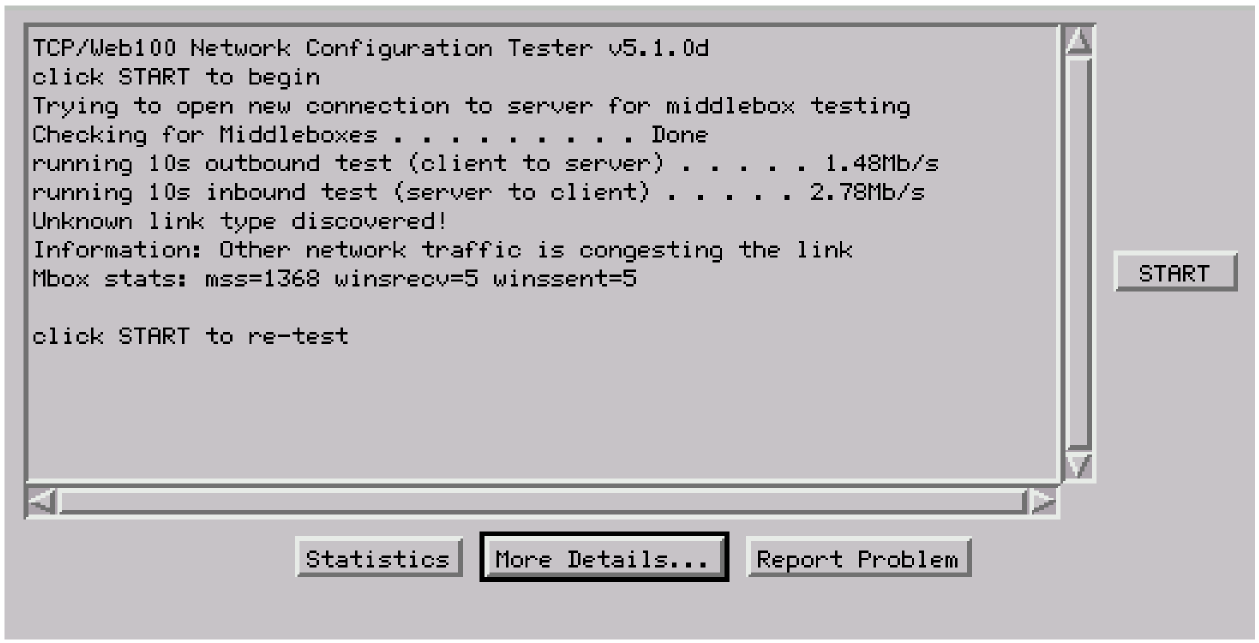
See also www.net100.org for more work based on Web100

Network Diagnostic Tool (NDT)

- http://e2epi.internet2.edu/ndt/
- Java applet runs a test and uses Web100 stats to diagnose the end to end path
- Developed by Richard Carlson, ANL

NDT Example

http://miranda.ctd.anl.gov:7123/



NDT Statistics

```
WEB100 Enabled Statistics:
Checking for Middleboxes . . . . . . . . . . Done
running 10s outbound test (client to server) . . . . . 1.48Mb/s
running 10s inbound test (server to client) . . . . . 2.78Mb/s

        ------  Web100 Detailed Analysis  ------
Unknown network link discovered.
Link set to Half Duplex mode
Information: throughput is limited by other network traffic.
Good network cable(s) found
Normal duplex operation found.

Web100 reports the Round trip time = 72.26 msec; the Packet size = 1368 Bytes; and
There were 8 packets retransmitted, 79 duplicate acks received, and 82 SACK blocks received
The connection stalled 3 times due to packet loss
The connection was idle 0.93 seconds (9.3%) of the time
This connection is network limited 99.23% of the time.
  Contact your local network administrator to report a network problem

     web100 reports TCP Performance Enhancement Settings are:
RFC 2018 Selective Acknowledgment: ON
RFC xxxx Nagle Algorithm: ON
RFC xxxx Explicit Congestion Notification: OFF
RFC 1323 Time Stamping: ON
RFC 1323 Window Scaling: ON
```

CLOSE

Unsigned Java Applet Window

Iperf with Web100, Clean Link

```
wcisd$ iperf-web100 -e -w400k -p56117 -c damp-wcisd
------------------------------------------------------------
Client connecting to damp-wcisd, TCP port 56117
TCP window size:  800 KByte (WARNING: requested  400 KByte)
------------------------------------------------------------
[  3] local 192.168.26.200 port 33185 connected with 192.168.26.61 port 56117
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0-10.0 sec   113 MBytes  94.1 Mbits/sec
        ------  Web100 Analysis  ------

        100 Mbps FastEthernet link found
        Good network cable(s) found
        Duplex mismatch condition NOT found
        Link configured for Full Duplex operation
        Information: This link is congested with traffic

Web100 reports the Round trip time = 14.0 msec; the Packet size = 1448 Bytes; and
There were 1 packets retransmitted, 0 duplicate acks received, and 0 SACK blocks received
This connection is network limited 99.99% of the time.
Contact your local network administrator to report a network problem

        Web100 reports the Tweakable Settings are:
RFC-1323 Time Stamping: On
RFC-1323 Window Scaling Option: On
RFC-2018 Selective Acknowledgment (SACK): On
```

Iperf with Web100, Lossy Link

```
wcisd$ iperf-web100 -e -w400k -p56117 -c damp-ssc2
------------------------------------------------------------
Client connecting to damp-ssc2, TCP port 56117
TCP window size:  800 KByte (WARNING: requested  400 KByte)
------------------------------------------------------------
[  3] local 192.168.26.200 port 33198 connected with 192.168.25.74 port 56117
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0-10.2 sec  35.0 MBytes  28.9 Mbits/sec
        ------  Web100 Analysis  ------

        Unknown link type found
        Good network cable(s) found
        Warning: Duplex mismatch condition exists: Host HD and Switch FD
        Information: link configured for Half Duplex operation
        No congestion found on this link

Web100 reports the Round trip time = 2.0 msec; the Packet size = 1448 Bytes; and
There were 617 packets retransmitted, 4072 duplicate acks received, and 4370 SACK blocks received
The connection stalled 1 times due to packet loss
The connection was idle for 0.21 seconds (2.06%) of the time
This connection is network limited 99.99% of the time.
Contact your local network administrator to report a network problem
```

NPAD / pathdiag

- Network Path and Application Diagnostics
- A new NDT like system from the people that brought you Web100
- http://www.psc.edu/networking/projects/pathdiag/
- Example server:
  - http://kirana.psc.edu/NPAD/

Going Faster

Cheating Today, Improving TCP Tomorrow

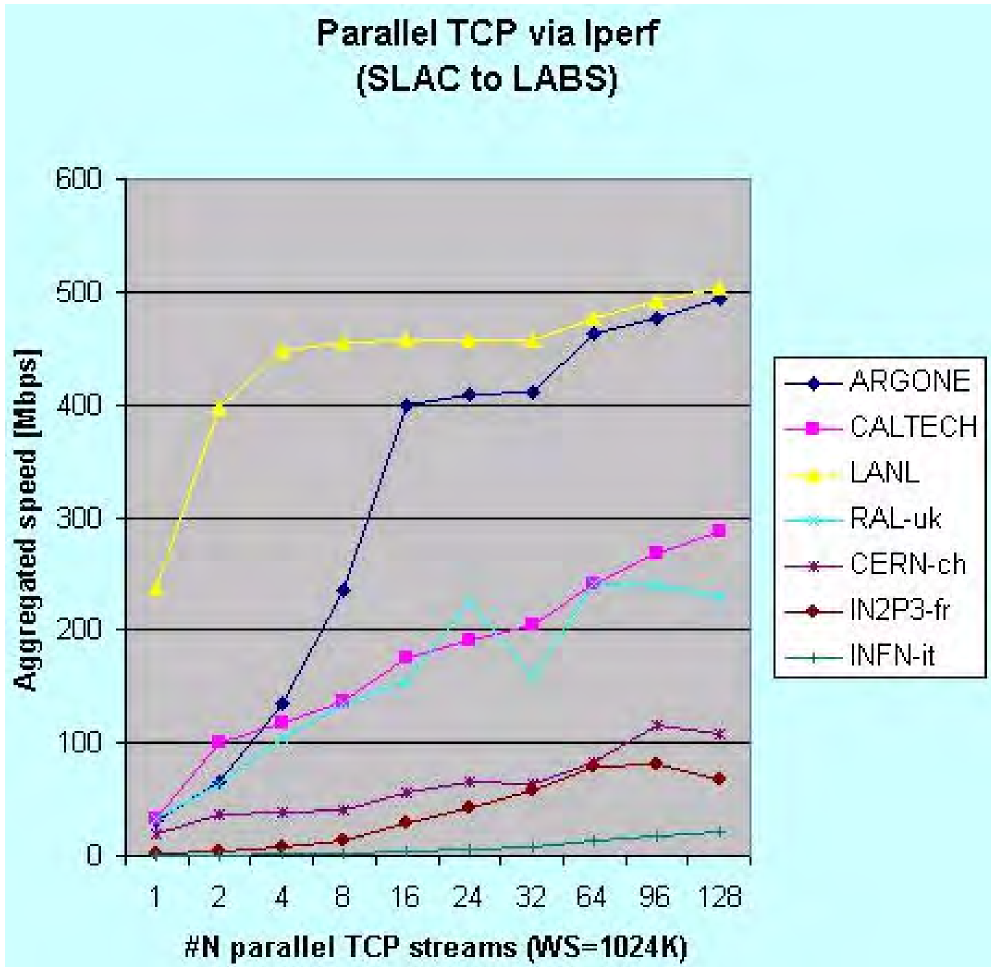
Multiple Streams

- Often N streams provide more total throughput than one stream
- It's "cheating" however because it gives you more than your fair share
- There may however be bandwidth going to waste, so go for it
- *One stream should be enough however for a good / modern / tuned TCP stack*

Parallel TCP Streams

- nuttcp and iperf can test parallel streams
  - Lets you see if this will help
- PSockets (Parallel Sockets library), SC2000
  - http://citeseer.ist.psu.edu/386275.html
- Several parallel client applications exist
  - MPSCP, bbFTP, UberFTP

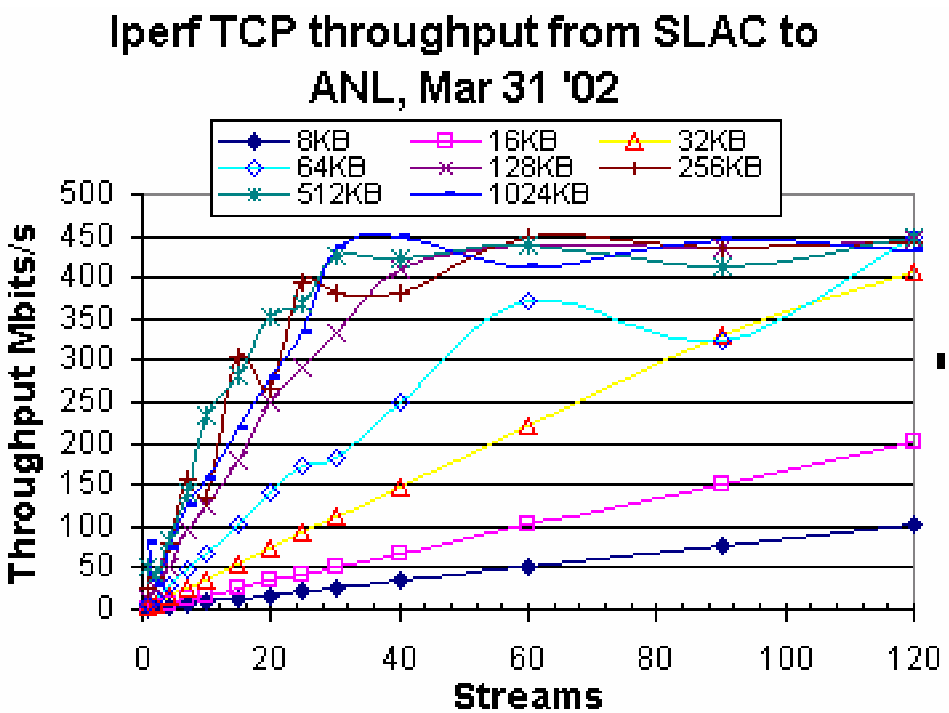
Parallel TCP Stream Performance

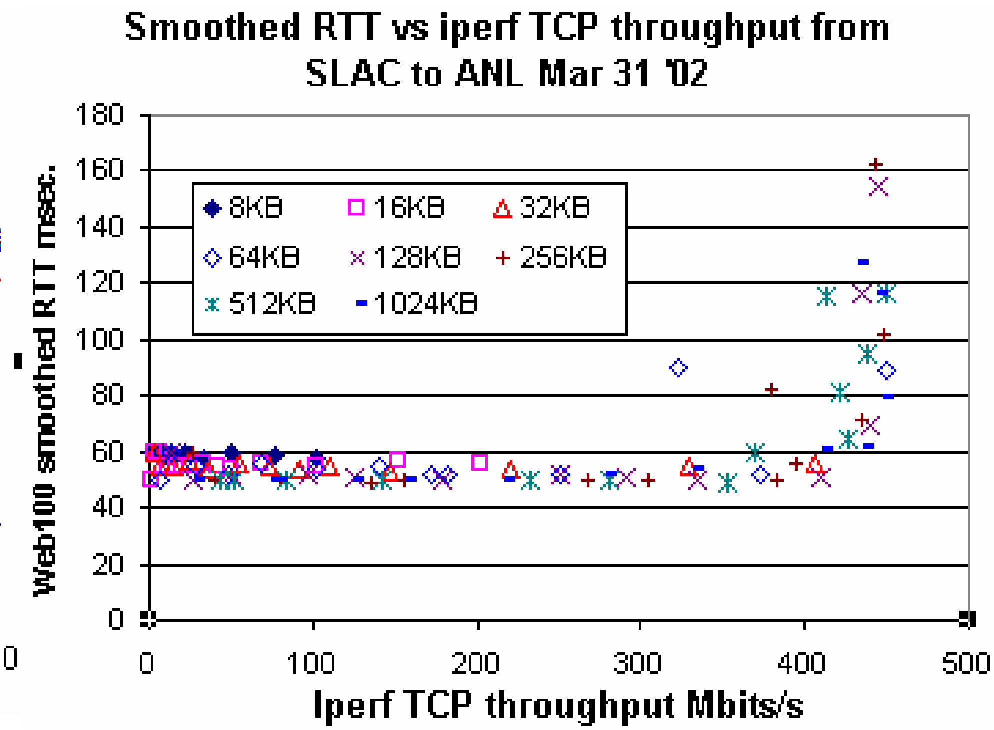


Les Cottrell, SLAC

Parallel TCP Streams

Throughput and RTT by Window Size





Les Cottrell, SLAC

MPSCP

- Multi-Path Secure Copy (MPSCP)
- Marty Barnaby, Sandia National Laboratory (DoE), mlbarna@sandia.gov
- Open source, BSD license
- http://www.sandia.gov/MPSCP/mpscp_design.htm

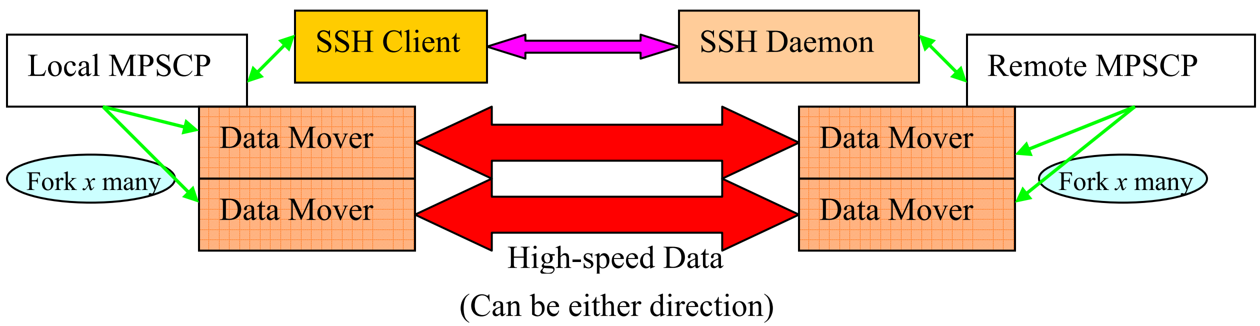


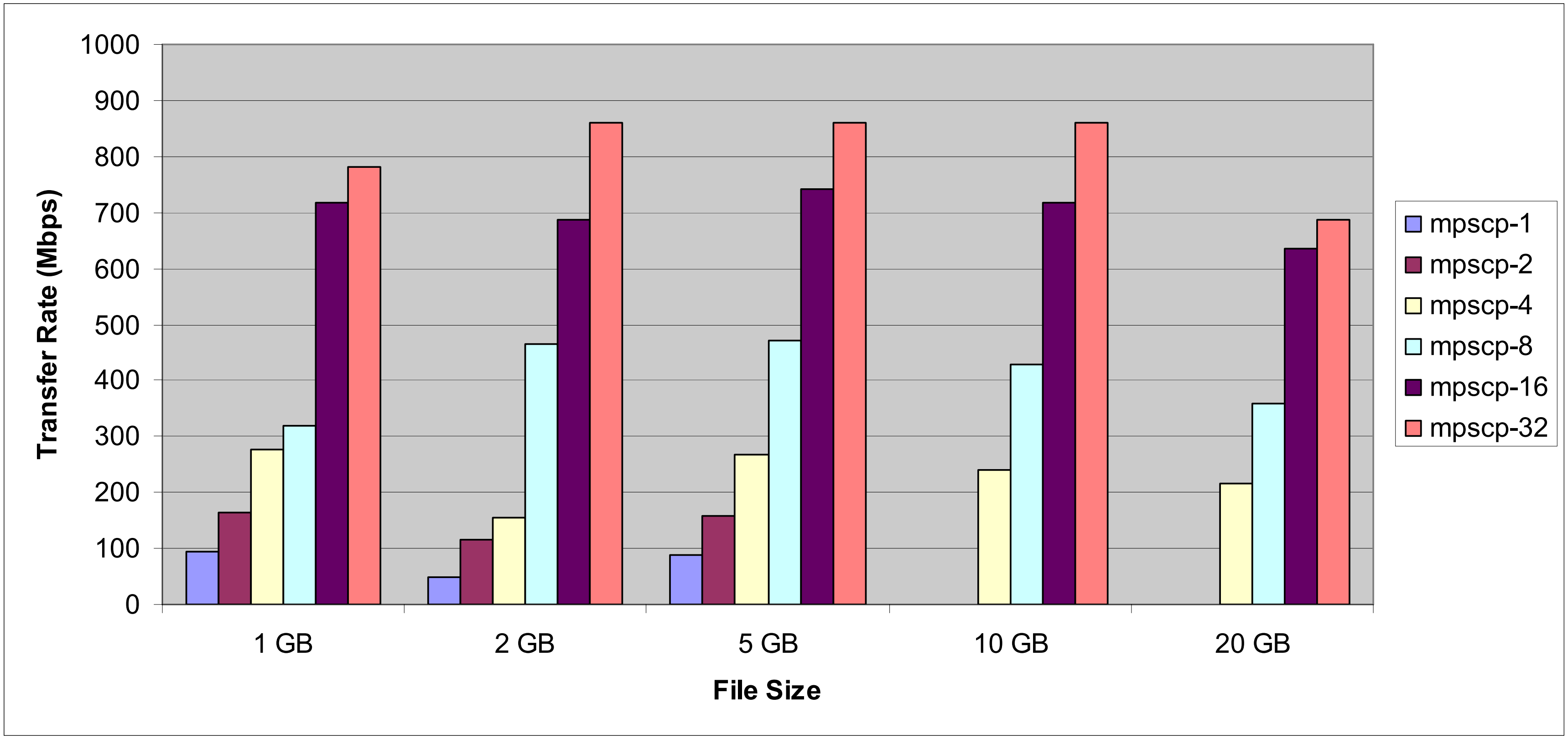
Diagram from the MPSCP Website

Characteristics of MPSCP

- Host-to-host file copy utility intended to enable greater transfer rates over high bandwidth networks
- Uses Secure Shell (ssh) for user authentication and control stream
  - Control stream is encrypted
- Uses multiple TCP data streams (1 to 64)
  - Data streams are not encrypted
- Can use multiple interfaces on end systems
- User interface is command line based and very easy to use
  - Similar to Remote Copy (rcp) and Secure Copy (scp)
- Same application binary on both sending and receiving hosts
  - No daemon required
- DoD HPCMP added an MD5 hash function to MPSCP

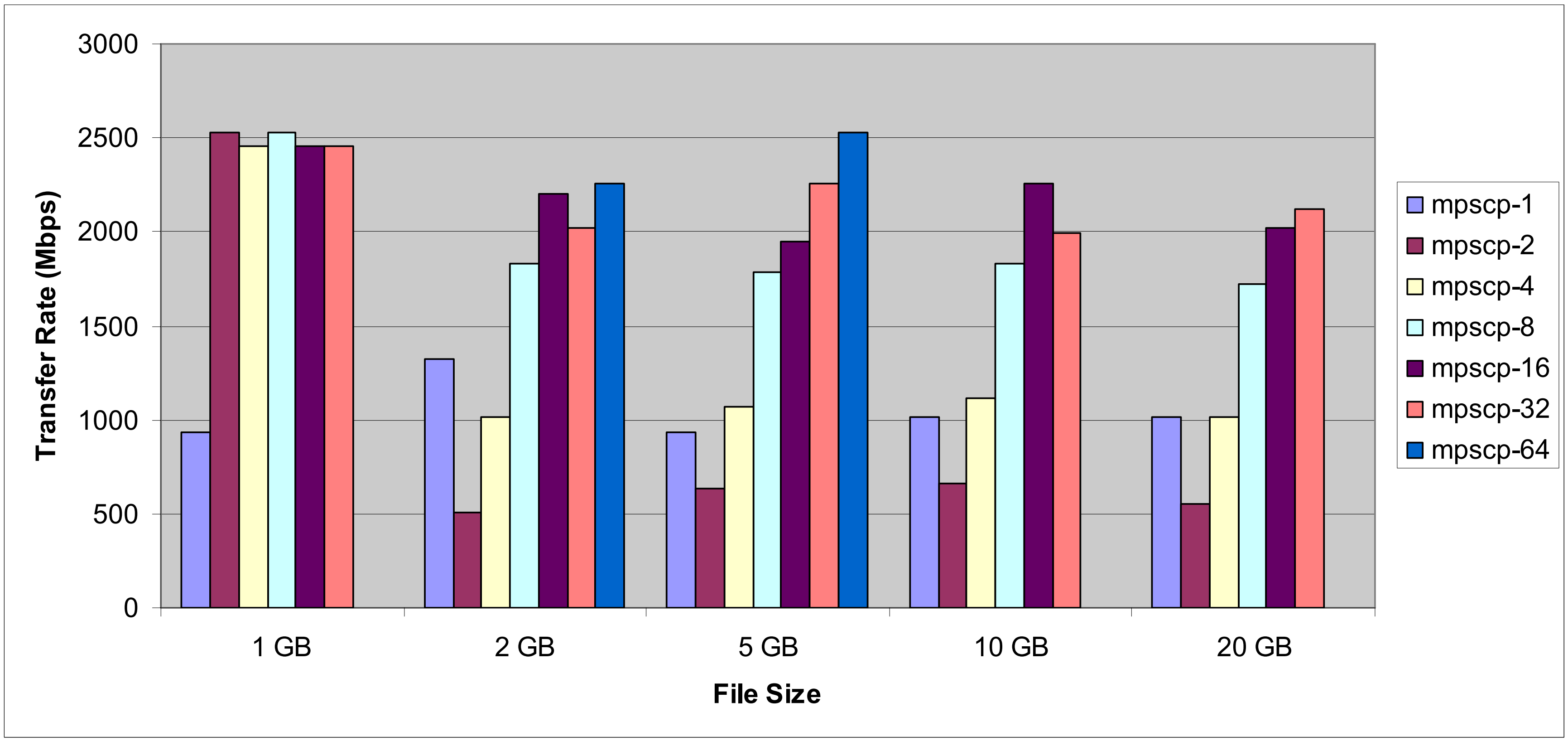
2006 MPSCP Test Results

From SGI Altix in OH to SGI Origin in MS



2006 MPSCP Test Results

From HP XC to SGI Altix on a LAN



Ralph McEldowney, ASC

bbFTP

- Developer: Gilles Farrache, IN2P3 Computing Center, Lyon, France
  - Current version: 3.2.0 released on 30 May 2005
  - Website: http://doc.in2p3.fr/bbftp/
- Description:
  - Contact Info: bbftp@in2p3.fr
  - Open source software released under the GNU GPL
  - Multiple stream data transfers
  - SSH and Certificate user authentication module
  - Compiled and tested on Linux, Solaris, and AIX
- Used by NASA, DOE, and other HPC centers

UberFTP

- Developer: Storage Enabling Technologies, National Center for Supercomputing Applications (NCSA), University of Illinois
  - Current version: 1.20 released on 21 July 2006
  - Website: http://dims.ncsa.uiuc.edu/set/uberftp/
  - Contact Info: gridftp@ncsa.uiuc.edu
- Description:
  - Open source software license
  - Interactive GridFTP-enabled FTP client
  - Supports GSI authentication, parallel data channels, and third party transfers
- Used by NSF TeraGrid sites and other HPC centers

Improvements to TCP

- Different congestion control schemes
  - Some based on delay / rtt variation
- Pacing - removing burstiness by spreading the packets over a round trip time
  - Vegas, FAST, BLUE
- Autotuning windows and buffer space
- Modifications to prevent "cheating"

Increased Initial Windows

- Allows ~4KB initial window rather than one or two segments
  - min(4*MSS, max(2*MSS, 4380 bytes))
- RFC 3390, Oct 2002, Proposed Standard

Appropriate Byte Counting

- When an ACK is received, increase cwin based on the *number of new bytes ACK'd*
- Prevents receiver from "cheating" and making the sender open cwin too quickly
  - e.g. receiver ACKs every byte
- Increases by at most 2*MSS bytes per ACK
  - To avoid bursts when one ACK covers a huge number of bytes
- RFC 3465, Feb 2003

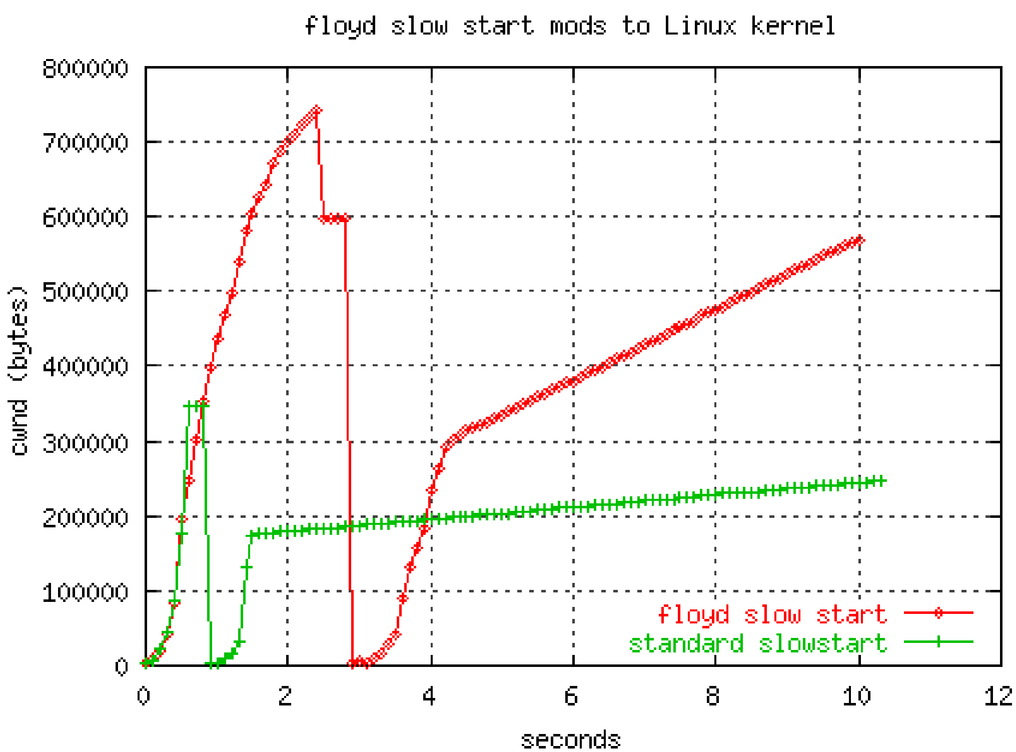
Limited Slow-Start

- In slow-start, the congestions window (cwin) doubles each round trip time
- For large cwins, this doubling can cause massive packet loss (and network load)
- Limited slow-start adds *max_ssthresh* (proposed value of 100 MSS)
- Above *max_ssthresh* cwin opens slower, never bursts more than 100 MSS

    cwin += (0.5*max_ssthresh/cwin) * MSS

- RFC 3742, Apr 2004

Limit Slow-Start Example



Tom Dunigan, ORNL

Quick-Start

draft-amit-quick-start-01.txt

- IP option in the TCP SYN specifies desired initial sending rate
  - Routers on the path decrement a TTL counter and decrease initial sending rate if necessary
- If all routers participated, receiver tells the sender the initial rate in the SYN+ACK pkt
- The sender can set cwin based on the rtt of the SYN and SYN+ACK packets

TCP Congestion Control

- The past few years have seen an explosion of work on modifications / alternatives to the "Reno" algorithm
- Many of them are now available on Linux
  - Reno, BIC, Cubic, Westwood, H-TCP, High Speed TCP, Hybla, Scalable TCP
  - Veno, TCP Low-Priority

Pluggable Congestion Control

- Introduced in Linux 2.6.13 (better in 2.6.14)
- Allows congestion control algorithms to be changed
  - System wide, via **`sysctl`**
  - Per socket, via socket options
- Allows them to be used by non-TCP protocols

TCP Reno

- Most modern TCP's are "Reno" based, from the 1990 BSD Reno Unix release
- Reno defined (refined) four key mechanisms
  - Slow Start
  - Congestion Avoidance
  - Fast Retransmit
  - Fast Recovery
- NewReno refined fast retransmit/recovery when non-SACK partial acknowledgements are available
  - Proposed Standard, RFC3782, Apr 2004

What’s Wrong With Reno?

- Poor performance on high bandwidth delay paths
  - Adapts too slowly
  - Requires an unreasonably low loss rate
- Round trip time unfairness
  - Short rtt flows get more bandwidth
- Loss isn’t always congestion

BIC TCP

- Binary Increase Congestion Control (BIC)
- More aggressive rate adaptation
- Better round trip time fairness
- BIC is now the Linux default!
  - since 2.6.8, but had issues until 2.6.11
- http://www.csc.ncsu.edu/faculty/rhee/export/bitcp/

CUBIC

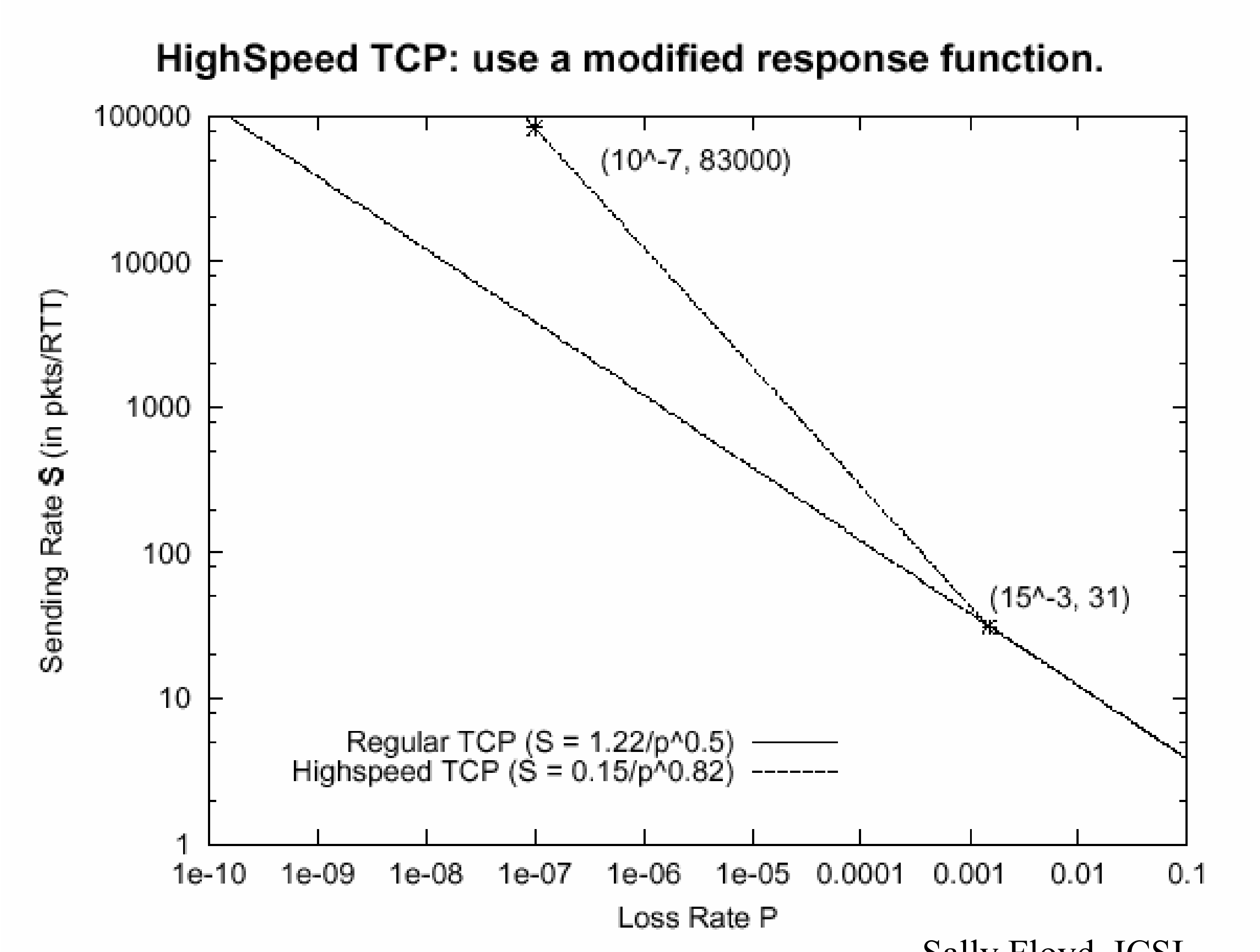
- Derivative of BIC-TCP
- Uses a cubic window growth function
  - More TCP friendly
- Low utilization detection
  - Allows CUBIC to be more aggressive

TCP Westwood (TCPW)

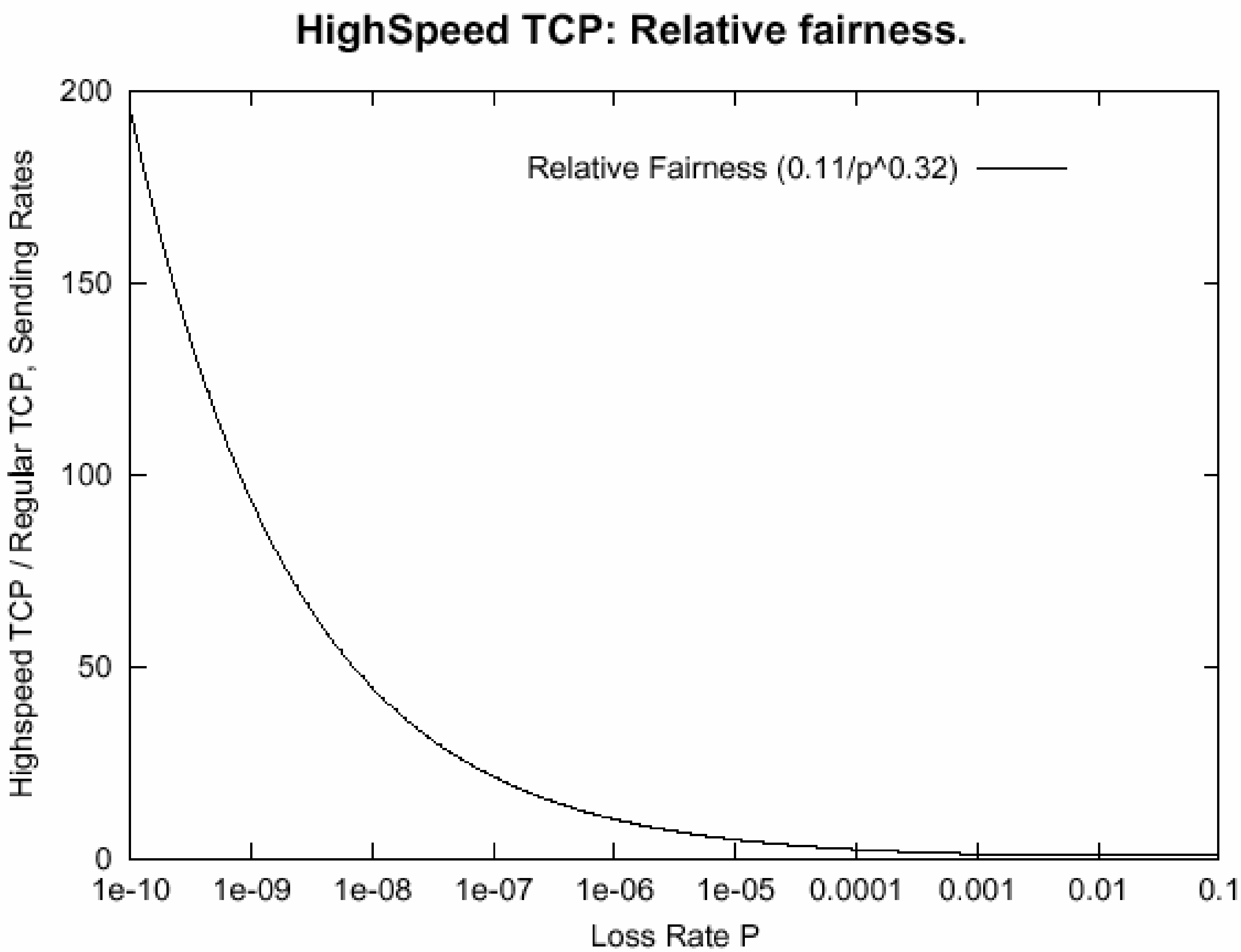
- For large bandwidth*delay pipes
- Rate Estimation from ACK stream
  - cwnd = RE x RTT_{min}
  - ssthresh
- Agile Probing phase on Persistent Non Congestion Detection (PNCD)
- http://www.cs.ucla.edu/NRL/hpi/tcpw/

HighSpeed TCP (HSTCP)

- Changes the AIMD parameters
- Identical to standard TCP for loss rates above 10^{-3} for fairness (cwin <= 38)
- Allows cwin to reach 83000 segments for 10^{-7} loss rates
  - Good for 10 Gbps over 100 msec rtt
  - Std TCP would be limited to ~440 Mbps
- RFC 3649, Dec 2003
- www.icir.org/floyd/hstcp.html

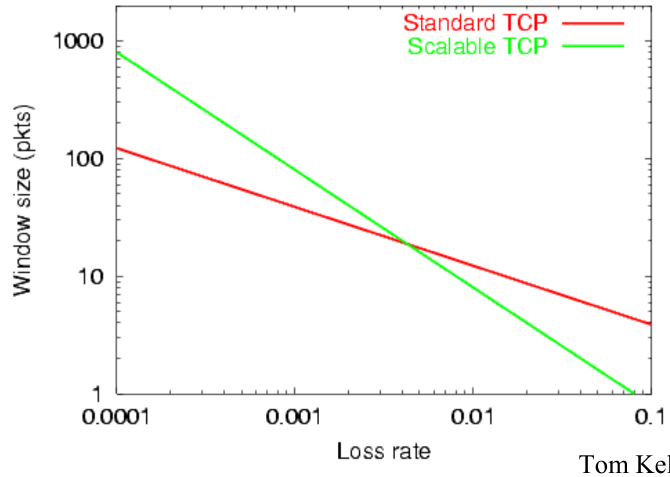


Sally Floyd, ICSI



Sally Floyd, ICSI

Scalable TCP (STCP)



Tom Kelly

- Loss recovery time is independent of sending rate
  - cwnd := cwnd + 0.01, for each ack while not in loss recovery
  - cwnd := 0.875 * cwnd, on each loss event
- http://www.deneholme.net/tom/scalable/

TCP Low Priority

- Uses unused bandwidth only
  - Similar to TCP-Nice
  - Similar to the QBone Scavenger Service (QoS)
- Showed that this could be done via congestion control changes only
- Both standard and High Speed TCP versions
- In Linux 2.6.18
- http://www-ece.rice.edu/networks/TCP-LP/

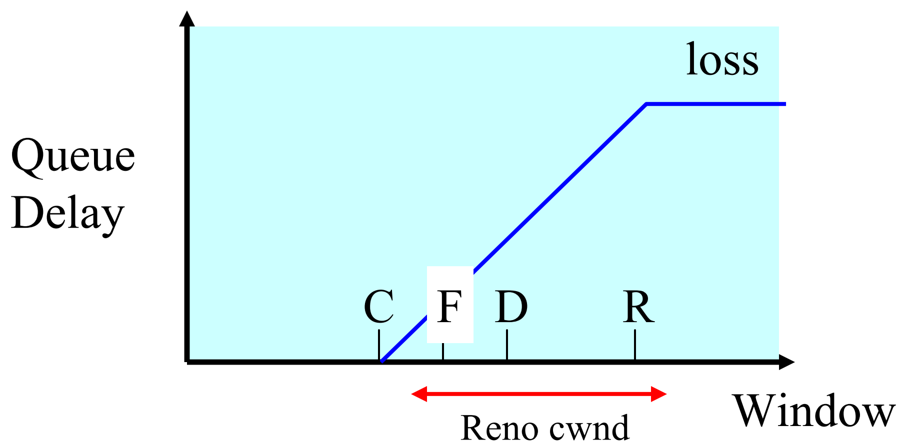
Hybla

- For long delay pipes, e.g. satellite
- Increases adaptation rate with increased round trip time
- Mostly Reno without the round trip time unfairness
- http://www3.interscience.wiley.com/cgi-bin/abstract/109604907/ABSTRACT

Veno

- For wireless networks
  - Where loss is often not congestion
- Uses a TCP Vegas like estimate of the connection speed to better set parameters
  - Reno + Vegas
- http://www.ntu.edu.sg/home/ascpfu/veno/veno.html

FAST TCP



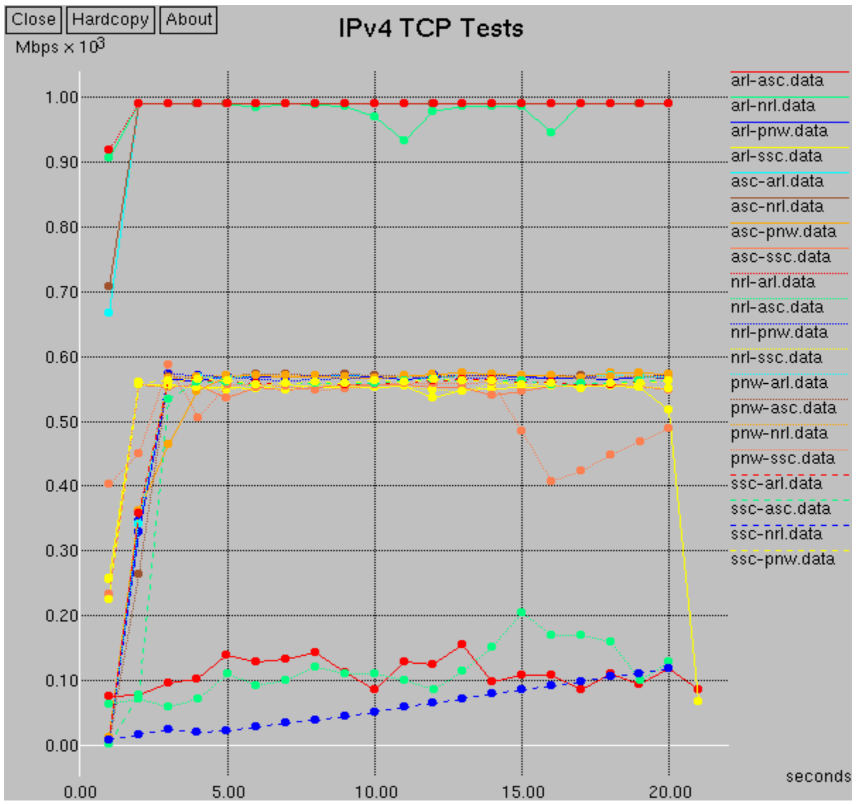
TCP Operating Points
C: CARD
F: FAST, Vegas
D: DUAL
R: Reno, HSTCP, STCP

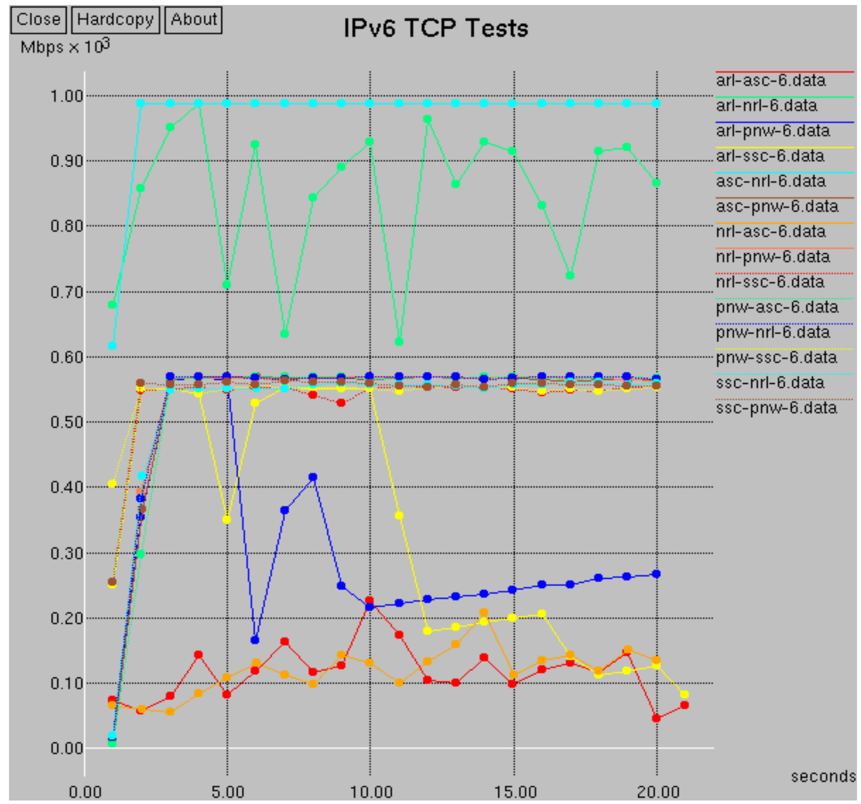
- Delay based congestion control
  - “multi-bit” feedback vs. binary loss signal
  - Avoids oscillations of cwnd and unnecessary loss
- Won SC05 bandwidth challenge
- http://netlab.caltech.edu/FAST/

Compound TCP

- Combines loss (NewReno) and delay based congestion control
- Default in Windows Vista!
  - *Vista also autotunes the TCP receive window*
- Option in Linux since 2.6.18
- MSR-TR-2005-86, July 2005

What About IPv6?





IPv4/IPv6 Observations

- Round trip times on all paths were within one millisecond of each other
- One or more paths achieved line rate performance for both IPv4 and IPv6 at OC12 and 1GigE rates
- One some paths IPv4 appears more robust, reasons unknown

UDP Transfer Protocols

NETBLT

- RFC969, 1985
- Block transfer protocol (not streaming)
  - Send a block at predetermined rate
  - Wait for lost packet list
  - Resend those, etc.

RBUDP

- Reliable Blast UDP
- Similar to NETBLT but in active development
- Includes real time and FEC support
- http://www.evl.uic.edu/cavern/quanta

Tsunami

- UDP data, TCP control, rate adaptive
  - Loss rate controls sending rate
- File transfer protocol, no API
- Transferred 1 TB of data at ~1 Gbps over a 12000 km "light path" (Vancouver to Geneva), Sep 2002
- Was created because TCP over that path was getting only 10's to 100's of Mbps!
- Last release, Dec 2002
- http://anml.iu.edu/research.shtml?prim=lab_research
- http://www-iepm.slac.stanford.edu/bw/Tsunami.htm

SABUL

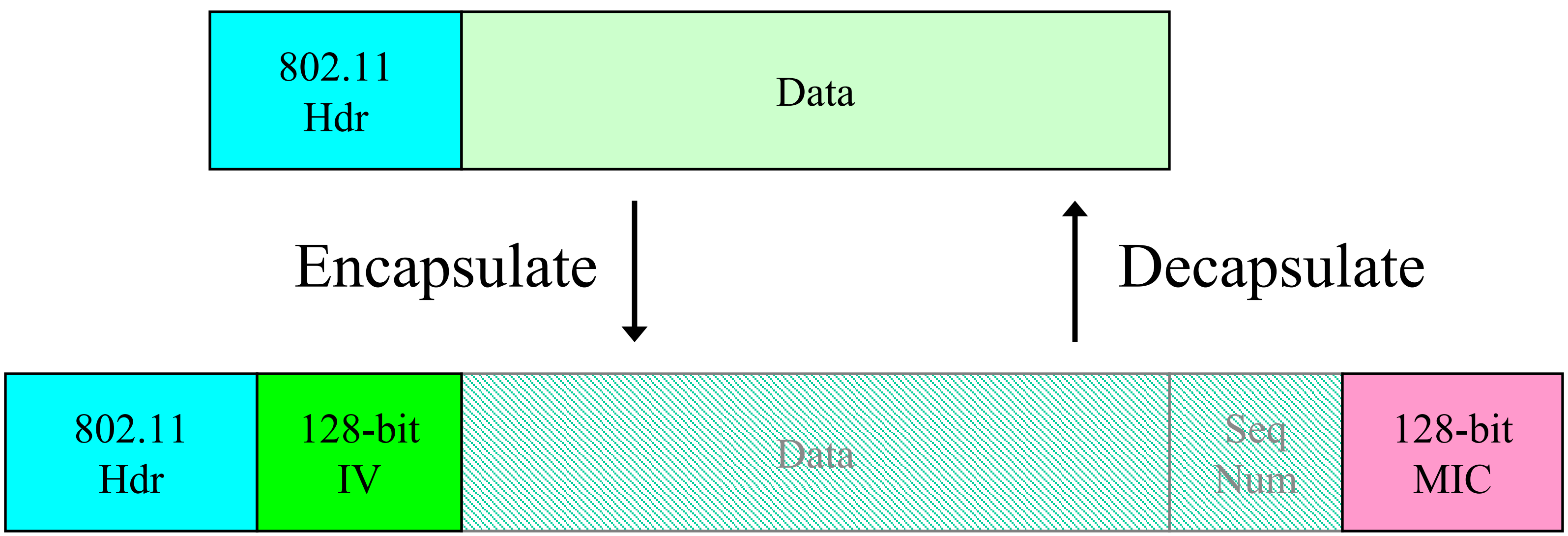
- Simple Available Bandwidth Utilization Library
- National Center for Data Mining (NCDM) at UIC (University of Illinois at Chicago)
- 2000-2003, now in third generation
- UDP data, TCP control, rate adaptive
- Streaming protocol, window + AIMD rate control (not rtt dependent)
- Includes FTP like application, API
- http://www.dataspaceweb.net/sabul.htm

UDT

- UDP-based Data Transport
- UDP for data and control
- Grew out of SABUL work, 2003+
- http://sourceforge.net/projects/dataspace

UDP Protocol Security!

- Session hijacking, corruption, encryption
  - Learn something from 802.11 wireless?
  - See also: DTLS (Datagram Transport Layer Security)



TCP Revisited

- “Anything you can do I can do too”
  - It’s just algorithms (e.g. congestion control)
- The real UDP advantage:
  - **User space implementations**, i.e. easier experimentation and deployment!
  - But user space is not as efficient
    - Until channels…

Beyond TCP and UDP

New Protocols

What’s Wrong with TCP?

- Once size doesn’t fit all
  - Bulk transfer, transactions, real-time, etc.
- Byte stream semantics vs. datagrams
  - No message boundaries, head-of-line blocking
- TCP congestion control (“Reno”) can introduce large rate changes and delays
- The Reno AIMD algorithm doesn’t scale well for high bandwidth uncongested networks

What’s Wrong with UDP?

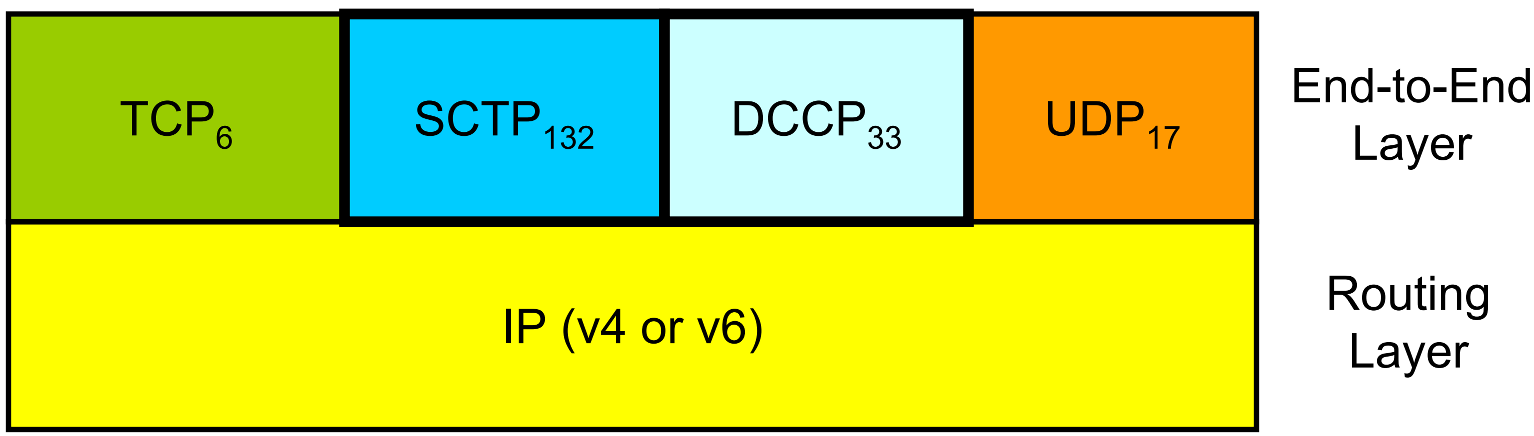
- Unreliable
- No congestion control (or rate control)

TCP and UDP together

- UDP can muscle TCP out
  - Since TCP backs off, but UDP doesn’t
- TCP often builds large queues
  - Which causes UDP delays and loss

- To make them co-exist, we often resort to Class of Service (CoS) solutions

New Protocols



SCTP

- Stream Control Transmission Protocol
- RFC2960, Oct 2000
  - RFC3257 SCTP Applicability Statement
  - RFC3286 Introduction to SCTP
  - RFC3309 Checksum Change, Sep 2002
  - RFC4460 SCTP Specification Errata and Issues

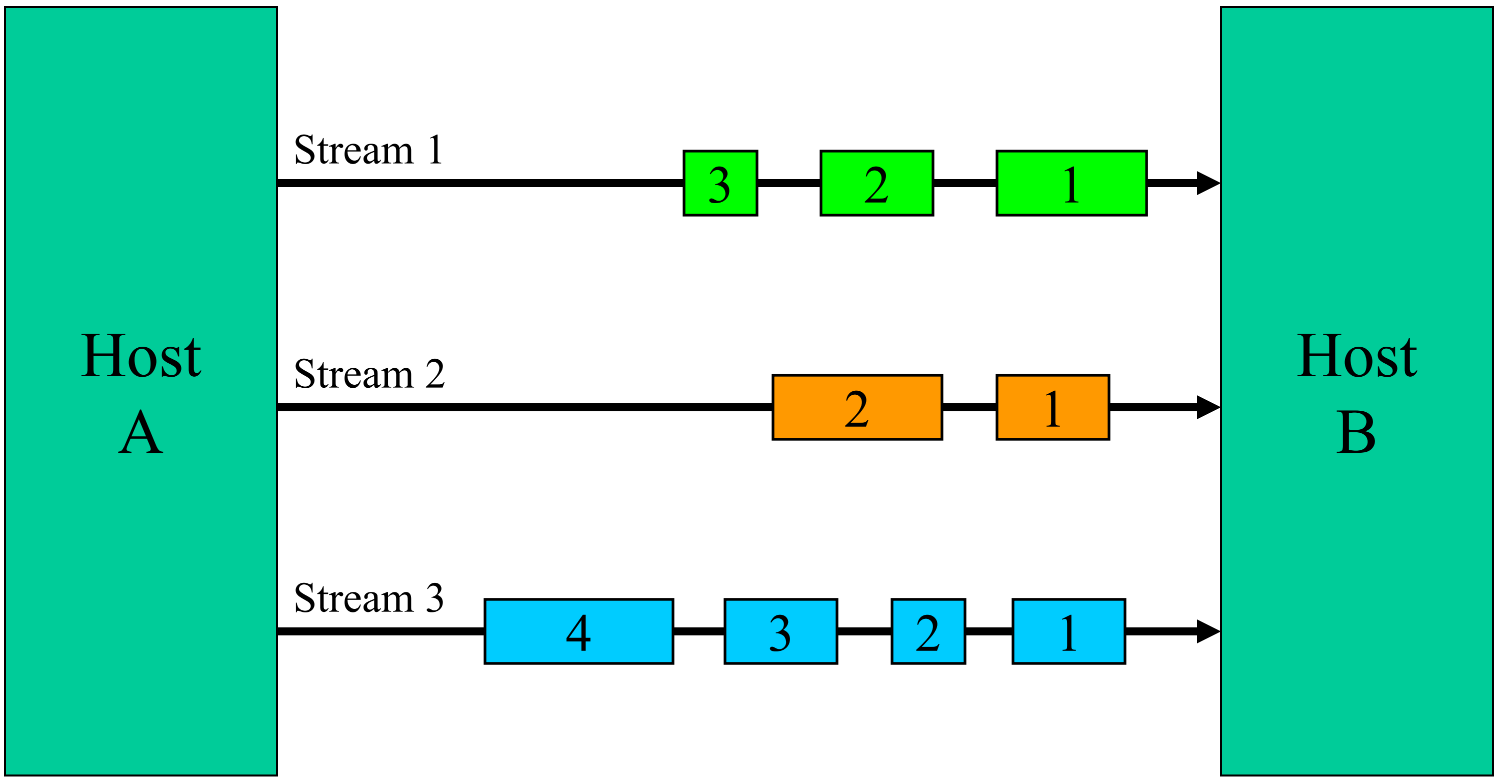
SCTP History

- Begun for SS7 transport over IP (IETF sigtran working group)
- Became the third general purpose IETF transport (after UDP and TCP)

SCTP Features

- Multi-streams support
  - Avoids head-of-line blocking
- Multi-homing support
  - Transparent to application
- Preservation of message boundaries
- Unordered reliable message delivery
  - Ordered delivery is also available

SCTP Multi-Streams



SCTP Features Continued

- Improved security
  - 32 bit Verification Tag
  - Cookie based DoS protection
- Improved data integrity (CRC32)

SCTP Implementations

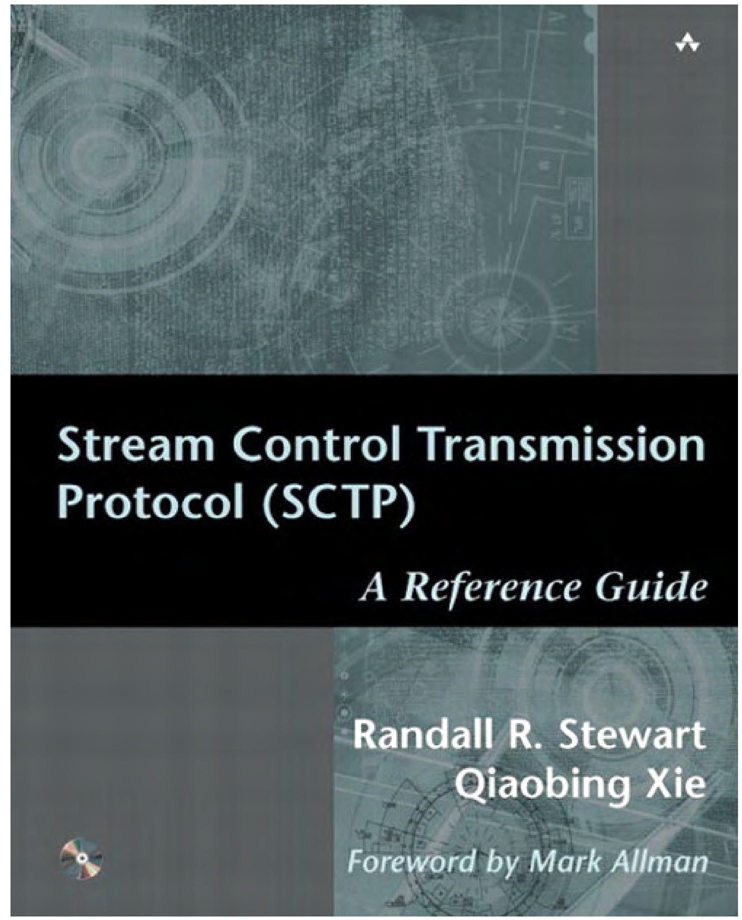
- Linux, “LKSCTP”, http://lksctp.sourceforge.net/
  – First released Jan 2001
  – In 2.4.23 and 2.6
- In Sun Solaris 10
- BSD via the KAME project (IPv6 stack)
- User space implementations – multiple OS
  – http://www.sctp.de/sctp-download.html
- Numerous commercial implementations

SCTP Performance

- Some tests indicate that the BSD/KAME and Solaris 10 SCTP implementations are better than Linux (2.6.15)
  - http://sctp.fh-muenster.de/Performance/index.html
- The CRC32 does have a performance impact

SCTP Resources

- www.sctp.be
  - Excellent, up to date
- www.sctp.org
  - Good implementation list
- *SCTP, A Reference Guide*, Randall Stewart, Oct 2001



DCCP

- Datagram Congestion Control Protocol
- RFC4340, Proposed Standard, Mar 2006
- Congestion Control ID Profiles

  CCID 2, RFC4341, TCP-like Congestion Control

  CCID 3, RFC4342, TCP Friendly Rate Control (TFRC)

DCCP Design Rationale

- Minimal overhead and complexity
  - Wanted existing streaming UDP applications to have little reason not to switch
- Reliable connection setup, teardown, options, feedback
- Choice of congestion/rate control algorithms
- Use of Explicit Congestion Notification (ECN) and the ECN Nonce

UDP and DCCP Headers

8 Byte UDP Header

Src Port	Dst Port
Length	Checksum

12+ Byte DCCP Header

<table>
<tr><td colspan="5">Src Port</td><td>Dst Port</td></tr>
<tr><td colspan="3">Data Offset</td><td>CCVal</td><td>CsCov</td><td>Checksum</td></tr>
<tr><td>0</td><td>Type</td><td>X</td><td colspan="2">0</td><td>Sequence #</td></tr>
<tr><td colspan="6">Options</td></tr>
</table>

- Seq # can be 16 bits (X=0) or 48 bits (X=1)
- CCVal = 4 bit Congestion Control (CCID specific)
- CsCov = Checksum Coverage
- Type = 1 of 10 packet types

DCCP Differences from UDP

- Congestion Controlled
- Session Oriented
  - Nicer on firewalls, NAT
- "UDP + congestion control, handshakes, and acknowledgements"

DCCP Differences from TCP

- Unreliable datagrams
- No Flow Control
  - No Receive Window, just congestion control
- Can distinguish different types of loss
  - Corruption, buffer overflow, etc.
- Denial of Service (DoS) protection
- No simultaneous open or half-closed states
- “TCP – byte stream semantics and reliability”

DCCP Implementations

- In Linux since 2.6.14, Oct 2005
  - http://linux-net.osdl.org/index.php/DCCP
- Preliminary FreeBSD support available
  - http://www.jp.nishida.org/dccp/
- User space DCCP from Berkeley
  - http://www.cs.ucsd.edu/~tsohn/projects/dccp/
  - http://inesc-0.tagus.ist.utl.pt/~pmsrve/dccp/

DCCP Support

- tcpdump 3.9.4 and later
- Patches available for
  - Ethereal, Iperf, netcat
- Python, Ruby

DCCP Resources

- http://www.read.cs.ucla.edu/dccp/
- The IEFT working group
  - http://www.ietf.org/html.charters/dccp-charter.html
- RFC’s mentioned earlier

Protocol Comparison

	Type	Cong Cont	Reliable	Order
TCP	Byte stream	Yes	Yes	Yes
SCTP	Byte stream or Datagram	Yes	Yes	Yes or No
DCCP	Datagram	Yes	Session	Sequenced
UDP	Datagram	No	No	No

What Goes Where?

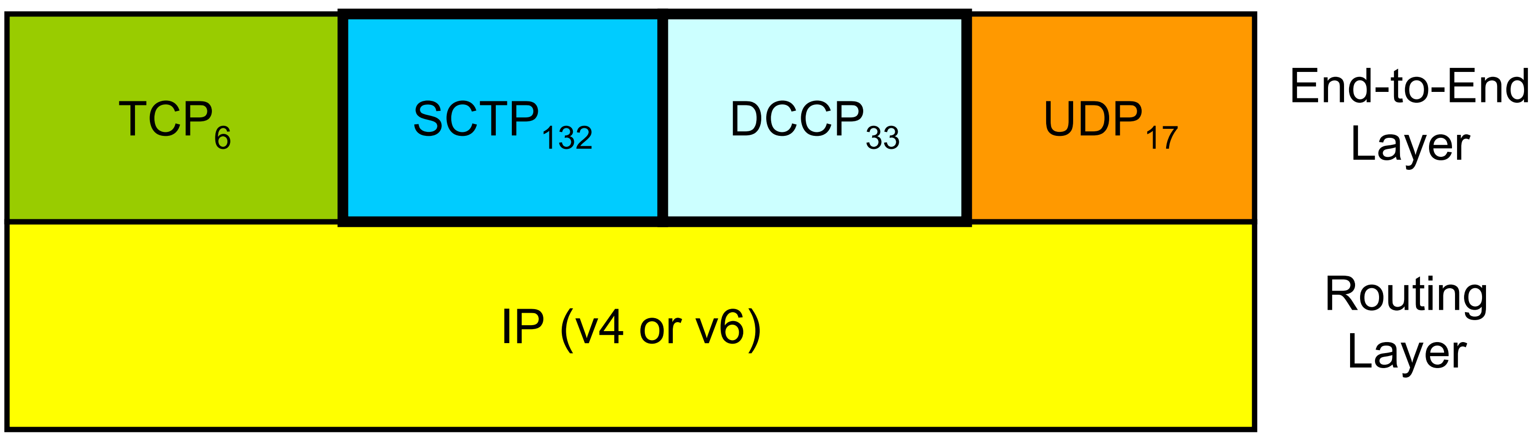
Email
SSH
FTP
P2P

Web
RPC
Transactions

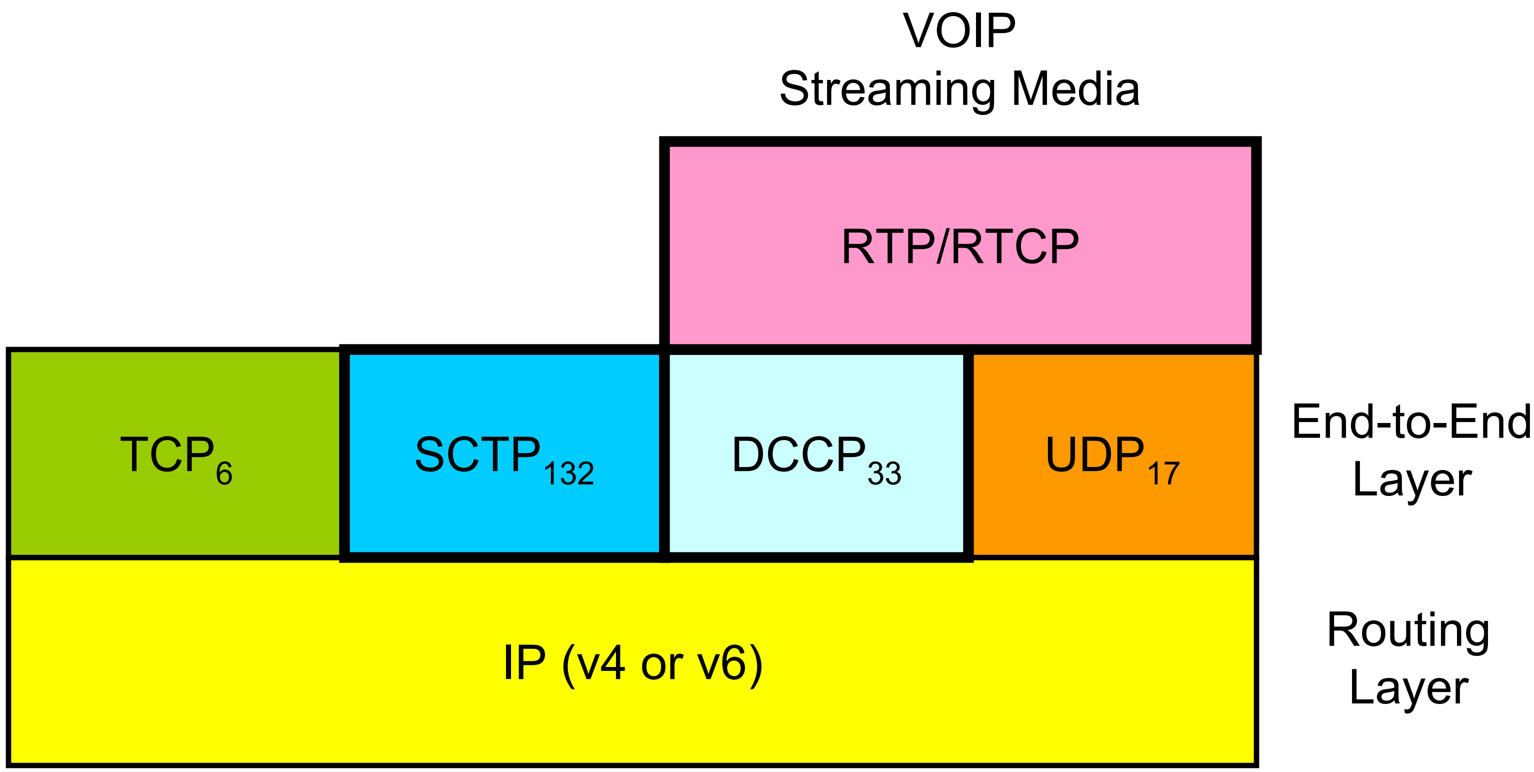
VOIP
Streaming Media

DNS
NTP
Multicast

Simulations?



Real-time Transport Protocol



New Protocol Summary

- There are now more transport choices than just "TCP or UDP"
- Most UDP streaming should move to DCCP
- Some applications could benefit from SCTP
- Multicast is still a tough case (UDP only)

Pluggable Protocols?

- If we can choose a transport protocol, and plug custom congestion control, why not plug the whole networking stack?
- Introducing…

Network Channels

- The next big thing in network performance from Van Jacobson
- Cache friendly ring buffers linking packet streams to user processes
- Minimal interrupt / softint processing
- Almost all protocol processing happens in user space
  - SMP and cache friendly
  - Allows application specific customization
  - More stable than the current layered kernel/user approach

Network Channels

- Demonstrated 6x improvement in bits per CPU cycle!
- Reduced latency
- Nearly linear speed up with multiple CPUs and cores
- Protocol libraries makes almost anything possible
- See “Speeding up Networking” Jan 2006
  - http://www.lemis.com/grog/Documentation/vj/lca06vj.pdf

What’s Happening Down Below

- Generic Framing Procedures (GFP)
  - Framed (GFP-F)
  - Transparent (GFP-T)
- User Controlled Light Paths (UCLP)
  - With GMPLS control

Data Integrity

Corrupted Packets

- V. Paxson
  - Claims most corruption caused by T1's
  - 1 in 5000 packets corrupted
  - TCP checksum would then let 1 in 300M corrupted packets pass
    - Roughly two errors per terabyte

More Corrupted Packets

- J. Stone, C. Partridge, SIGCOMM 2000
  - *"Traces of Internet packets over the last two years show that 1 in 30,000 packets fails the TCP checksum, even on links where link-level CRCs should catch all but 1 in 4 billion errors."*
  - DMA transfers account for many of these errors
  - Conclude that 1 in 200M TCP packets pass with undetected errors

Data Integrity

- How do you know that your transferred copy is correct?
- A simple approach would be copy it two (or more) times and compare the results
- But you would like to check it with something smaller than the entire dataset

Checksums, CRC’s, Hashes

- All map large blocks of data into smaller keys
- Checksums are simple algebraic sums
- Cyclic Redundancy Checks (CRC’s) are good at detecting multiple bit errors
- Hashes are good at randomizing the key space
  - Good for database lookups, sorting, etc.
  - Cryptographic hashes are good for data integrity checks

Simple Checksums

- Can not detect several kinds of errors
  - Reordering of bytes
  - Inserting or deleting zero valued bytes
  - Multiple errors that cancel (same sum)

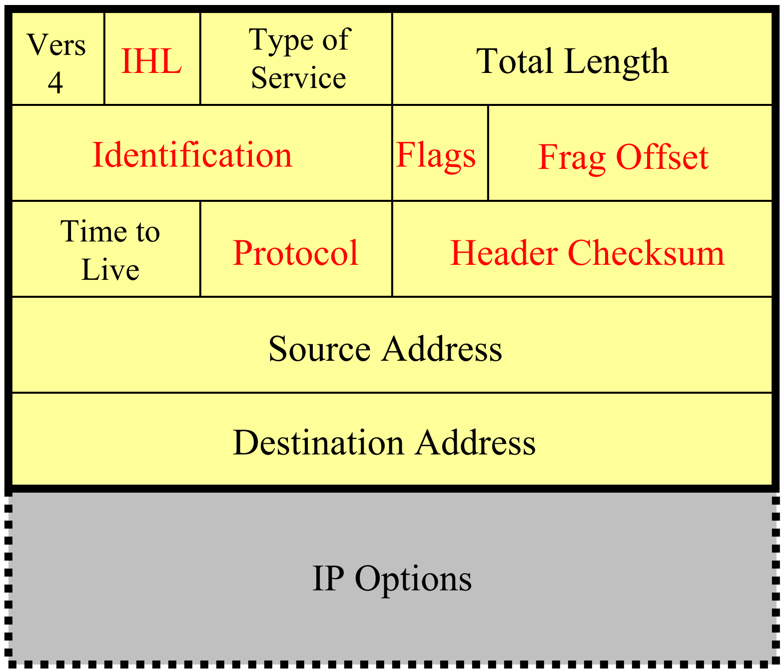
The Internet Checksum

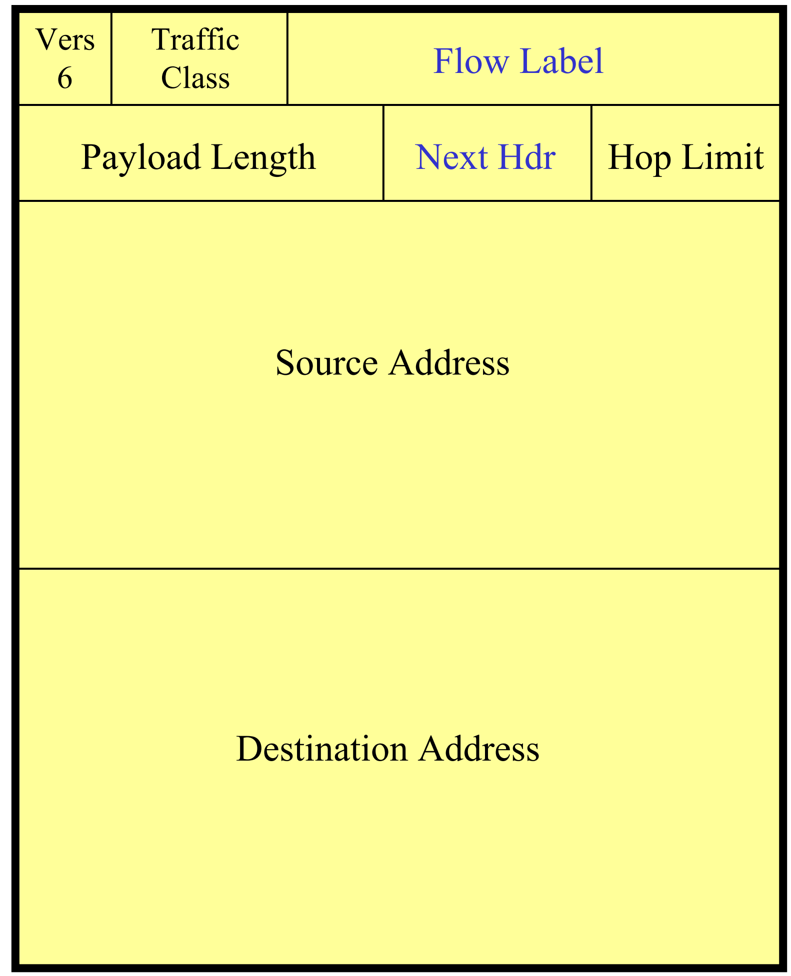
- 16 bits long, summed 16 bits at a time
- For UDP/TCP, includes a 12 byte “pseudo header” from the IP layer with the network addresses and payload length
- RFC 1071 – Computing the Internet Checksum

IP Data Protection

- IPv4 uses the 16-bit Internet Checksum over the header information only
  - *This checksum is recalculated/rewritten hop by hop, since e.g. the TTL changes*
- IPv6 has none, not even over the header
  - Assumes layer 2 will protect it

IPv4 and IPv6 Headers





v4 Header = 20 Bytes + Options
v6 Header = 40 Bytes

UDP Data Protection

- 16-bit Internet Checksum covers the UDP header and payload
- Optional for IPv4 UDP
- Required for IPv6 UDP

TCP Data Protection

- 16-bit Internet Checksum over the entire segment
- Roughly every 65536th corrupted segment goes undetected

The Danger Of Offloading

- Many NICs today support TCP Checksum offloading
- Studies have shown many errors occur during DMA transfers
  - Offloading would miss these
- Nothing beats reading the data back from disk, e.g. md5sum

Ethernet Data Protection

- IEEE 802 CRC32
- Strong error detection for 1500 byte packets
- Detects all triple bit errors up to 11455 bytes
- Roughly 1 in 4 billion errors go undetected, but strength is non-uniform
- http://www.cse.ohio-state.edu/%7Ejain/papers/xie1.htm

Cryptographic Hashes

- One-way functions
  - Easy to compute, nearly impossible to invert
- Collision free
  - Infeasible to find two inputs that result in the same output
- The workhorses of modern cryptography

Cryptographic Hash History

- 1990, MD4, Ron Rivest
- 1992, MD5
- 1993, SHA, NSA (a.k.a. SHA-0)
- 1995, SHA-1
- 2002, SHA-256, SHA-384, SHA-512
- 2004, SHA-224
- 2010, NIST no longer endorses SHA-1

Common Hashes

- MD5 – Message Digest algorithm #5
  - 128 bit hash, RFC1321 (April 1992)
  - `md5sum --check md5sums`
- SHA-1 – Secure Hash Algorithm #1
  - 160 bit hash

Stronger Hashes

- Recent work has shown some weakness in MD5 and SHA-0 (Crypto2004 conference)
- NIST plans to phase out SHA-1 by 2010
- NIST FIPS 180-2, Secure Hash Standard, Aug 2002, also includes:
  - SHA-224, SHA-256, SHA-384, SHA-512
  - http://csrc.nist.gov/CryptoToolkit/tkhash.html

HMAC

- Keyed-Hash Message Authentication Codes
- Combines a secret key and a cryptographic hash
- HMAC-SHA1, HMAC-MD5, HMAC-RIPEMD
- RFC2104, Feb 1997, ANS X9.71

Advanced Encryption Standard (AES)

- AES (Rijndael) data encryption/decryption
- Federal Information Processing Standards Publication, FIPS-197
- 128, 192, 256 bit keys, 128 bit data blocks
- Can be used as a Message Authentication Code (MAC)
  - AES-XCBC-MAC-96, RFC 3566, Sep 2003
- scp defaults to aes128-cbc and hmac-md5

Encryption/Hash Speeds

"openssl speed" OpenSSL 0.9.7d, 2.6 GHz Xeon, 400 MHz FSB, 512KB L2 cache

type	16 bytes	64 bytes	256 bytes	1024 bytes	8192 bytes
md2	2335.38k	5298.12k	7781.61k	8827.85k	9161.67k
mdc2	6480.09k	7374.90k	7717.70k	7677.92k	7790.82k
md4	19268.23k	63363.31k	173880.13k	303676.32k	391984.43k
md5	**16325.52k**	**54524.02k**	**143885.24k**	**246730.38k**	**311775.74k**
hmac(md5)	17483.15k	58603.05k	151600.95k	252513.14k	313778.51k
sha1	**15804.80k**	**51292.97k**	**126032.04k**	**199600.09k**	**240482.89k**
rmd160	11629.85k	31237.76k	62984.82k	84785.10k	93663.74k
rc4	84407.69k	93179.20k	95925.54k	96411.00k	96438.66k
des cbc	**43456.93k**	**43846.81k**	**43784.50k**	**43868.37k**	**43858.06k**
des ede3	**17158.70k**	**17168.26k**	**17128.48k**	**17191.63k**	**17186.59k**
idea cbc	24765.82k	25599.68k	25818.65k	25979.85k	25950.68k
rc2 cbc	24287.21k	24957.35k	25301.88k	25439.74k	25317.71k
rc5-32/12 cbc	91339.13k	89333.21k	90386.24k	90881.38k	90675.90k
blowfish cbc	77119.43k	78330.17k	78255.31k	78479.08k	77552.78k
cast cbc	48512.65k	50893.86k	51452.11k	51747.77k	51385.43k
aes-128 cbc	**70686.37k**	**65725.38k**	**65752.90k**	**65180.02k**	**64708.38k**
aes-192 cbc	55336.89k	57739.22k	58446.44k	57441.14k	57590.31k
aes-256 cbc	56045.36k	52542.96k	52584.07k	51628.97k	51985.99k

Results are in Bytes per second

AES-128 CBC ran ~0.5 Gbps

Local Transfer Examples

- 744 MB file, 2.6 GHz Xeon
  - Disk read, loopback interface, no disk write
  - both encrypt and decrypt for scp
- **`wget -O /dev/null ftp://localhost/tmp/file`**
  - 0:13 sec, 455 Mbps
- **`scp localhost:/tmp/file /dev/null`**
  - AES-128 (default), 0:49 sec, 121 Mbps
  - 3DES, 1:53, 53 Mbps
  - Blowfish, 0:39, 153 Mbps

Remote Transfer Examples

- Aberdeen, MD to San Diego, CA, 744 MB
- scp 14 minutes
  - compared to 49 seconds on localhost
- wget v1.9.1, 4 minutes
  - from vsftpd 1.2.2 server at Aberdeen
- wget –passive-ftp, 15 seconds
  - wget with large window modification

Summary: Layers of Protection

- Application: Hashes
- Transport: TCP/UDP checksum
  - Optional: SSL/TLS/DTLS
- Network: IP header checksum
  - Optional: IPsec
- Link: Ethernet CRC32 or POS FCS-32
- Physical: symbols, ECC, FEC

Storage Area Networks

SANs and IP Storage

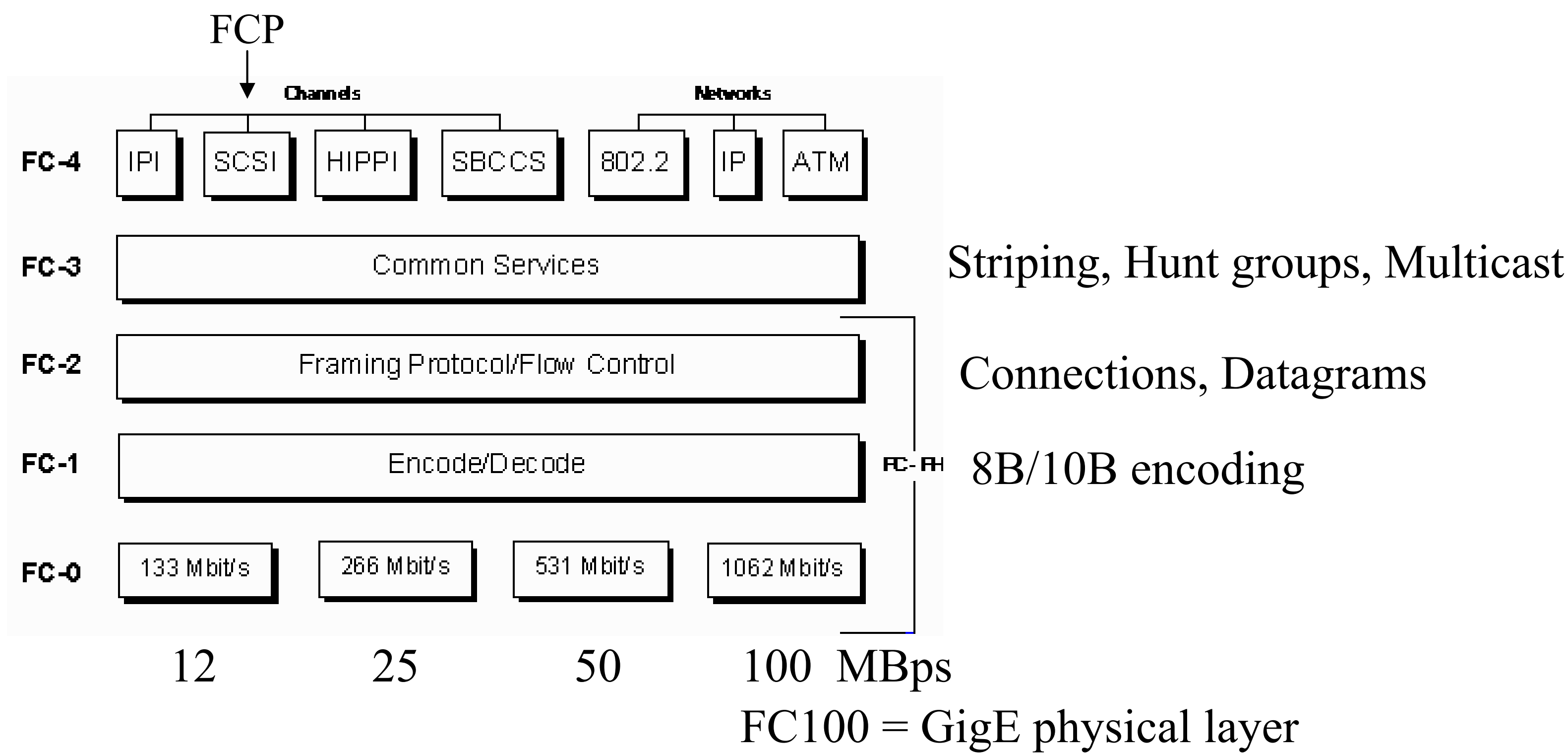
Storage Area Network (SAN)

- Dedicated network for accessing Network Attached Storage (NAS)
- Usually Fibre Channel
  - Fibre Channel Protocol (FCP) = SCSI Commands over FC
- IP Storage is coming

Fibre Channel

- Began in 1988 to improve HIPPI connections
  - First ANSI standard in 1994
  - 1 Gbps (100 MBps) standard in 1996
- Runs on both fiber and copper
- Ring or mesh (switch) topology
- FC disks have dual / redundant connections

Fibre Channel Architecture



- 2 Gbps 2001, 4 Gbps 2004
- 10 Gbps available, but not backward compatible

FCIP

- Fibre Channel over TCP/IP
  - Encapsulates FC frames in TCP/IP
- Provides Inter-Switch Links (ISLs)
- Allows multiple FC SAN's to be connected via the internet
- RFC 3821, July 2004

iFCP

- Internet Fiber Channel Protocol
- FC-4 layer interface with a TCP/IP sublayer
  - Replaces the FC SAN with an IP network
- Gateways interface between FC and IP devices
- Emulates fabric services
- RFC4172, Sep 2005

iSCSI

- Internet Small Computer Systems Interface
  - Serialized SCSI over TCP
  - Initiator to target (tcp port 3260)
- Removes 25 meter distance limit
- Includes initiator/target authentication via iSCSI Login PDUs
  - Challenge Response Authentication Protocol (CHAP)
  - Secure Remote Password (SRP)
- Uses CRC32c (but optional!)
- RFC 3720, Apr 2004

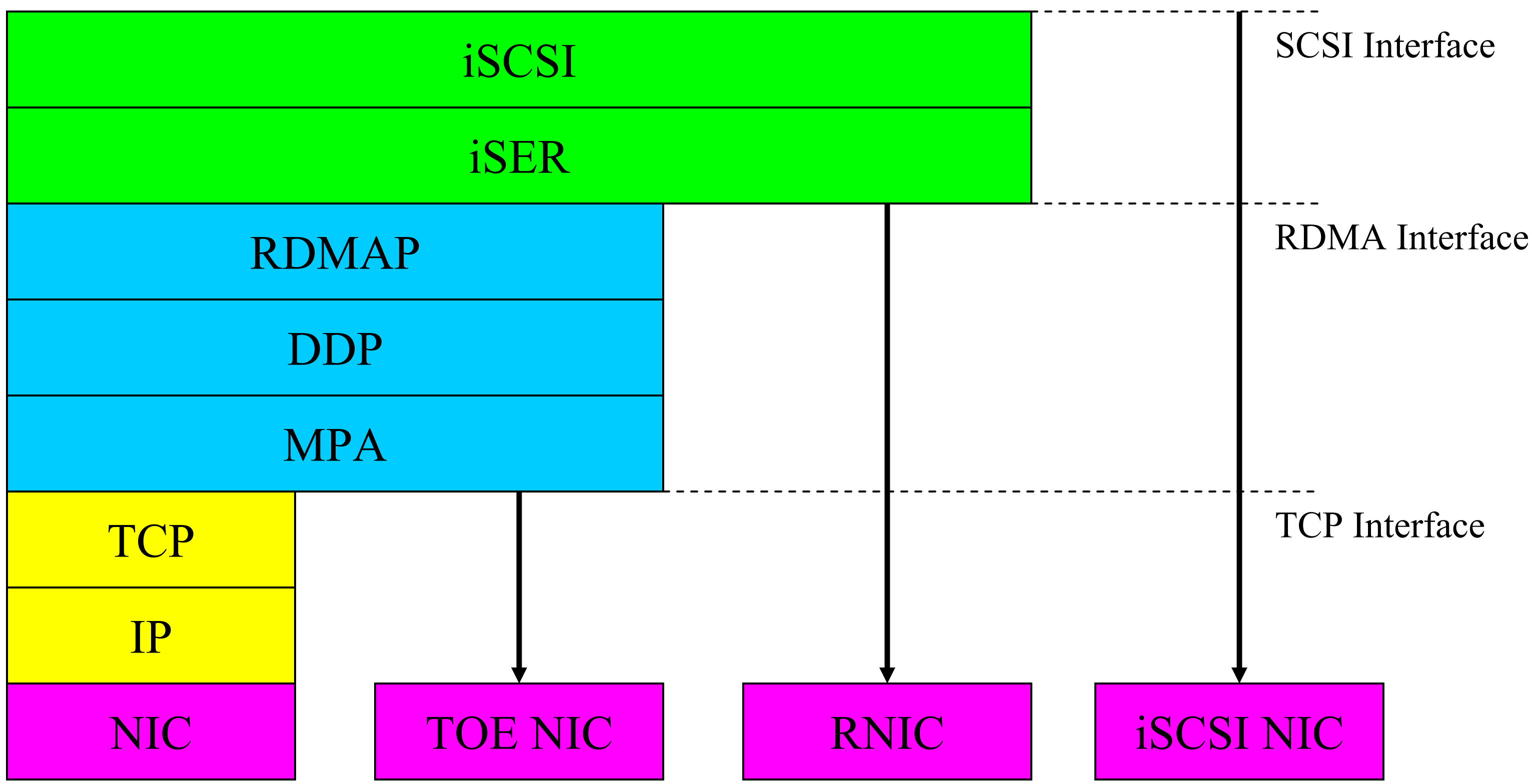
Remote Direct Memory Access (RDMA)

- Zero-copy end-to-end data movement
- Useful to things like MPI
- Available over InfiniBand
- Available over TCP/IP via iWarp
  - *Would be easier to implement with SCTP*
- Sockets Direct Protocol (SDP) can run over iWarp or InfiniBand RDMA

iWarp

- Collective name given to three protocols
  - MPA – Marker PDU Alignment for TCP
  - DDP – Direct Data Placement
  - RDMAP – RDMA Protocol
- Allows RDMA over TCP/IP
- Defined by the RDMA Consortium
  - www.rdmaconsortium.org

Enhanced Network Interface Cards



IP SAN Security

- FC SANs provide little to no security
- FCIP, iFCP, iSCSI all depend on IPsec for per-packet
  - Authentication
  - Confidentiality
  - Integrity
  - Replay protection
- RFC 3723, Securing Block Storage Protocols over IP, Apr 2004

SANs in Perspective

- Came out in the 10/100 Mbps Ethernet era
  - IP networks simply weren't fast enough, nor did they have QoS controls
- Held on through the introduction of 1Gbps Ethernet
  - Claimed to still be faster and cheaper
- Today: Can no longer compete with 1 GigE and soon 10 GigE pricing
- IP attached iSCSI storage is coming

Peer to Peer (P2P) File Transfer

Peer to Peer Basics

- Remove the central server bottleneck
  - Like automatic mirroring
- Can **download** data from anyone (peers) with a copy, or partial copy
- Can **find** data from central servers or peers

P2P File Sharing Networks

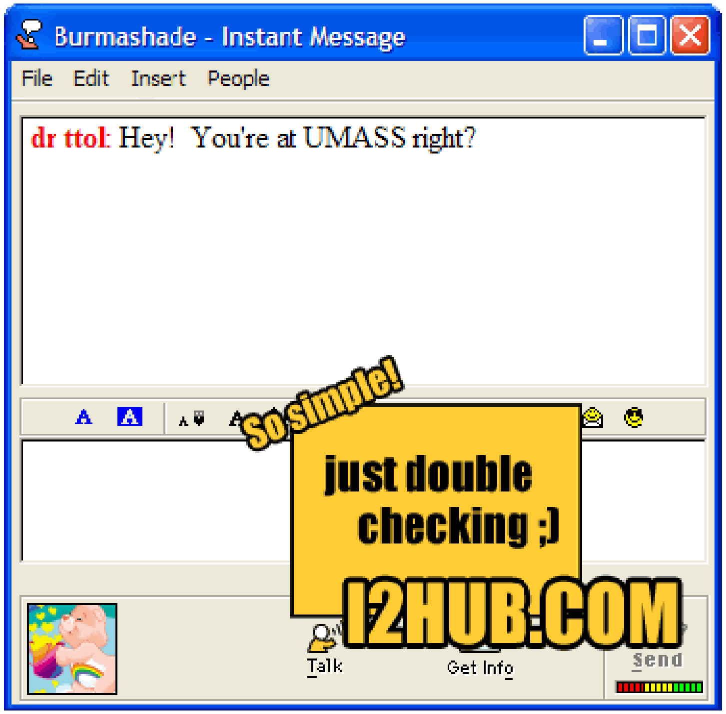
- Napster
  - Began June 1999, over 26 million users by Feb 2001
  - Primarily served music (mp3) files
  - Central server to find peers, download from peer(s)
- FastTrack
  - No central database, flooding, order N lookups
  - Clients: KaZaA, Grokster, iMesh, (Morpheus)
  - Closed source network, began March 2001
- Gnutella (G1)
  - Open source FastTrack like network
  - Clients: LimeWire, Morpheus / Gnucleus

P2P File Sharing Networks

- eDonkey2000
  - Did for movies what Napster did for music
  - Allows partial uploads
  - Clients: eDonkey, eMule
- Direct Connect
  - Began Nov 1999
  - Over 12 PB of data on 250k hosts in 2004!
  - “Hubs” keep track of network members

i2Hub

- Direct Connect P2P network on Internet2
- "faster because it uses Internet2"
- Reality check:
  - "at UCLA, i2hub downloads run 30 kBps to 200 kBps, vs. 600 kBps for a good server"
- Mar 2004 – Nov 2005



Distributed Hash Tables (DHT)

- Hot topic since 2001
  - CAN, Chord, Pastry, Tapestry
  - Could be used to replace DNS, web caches, etc.
- O(log N) vs. O(N) lookups
- Provable properties
- Many research projects in DHT's
  - FreeNet, OceanStore
  - Automated P2P backup projects
    - Pastiche and PAST built on Pastry

Newer P2P Networks

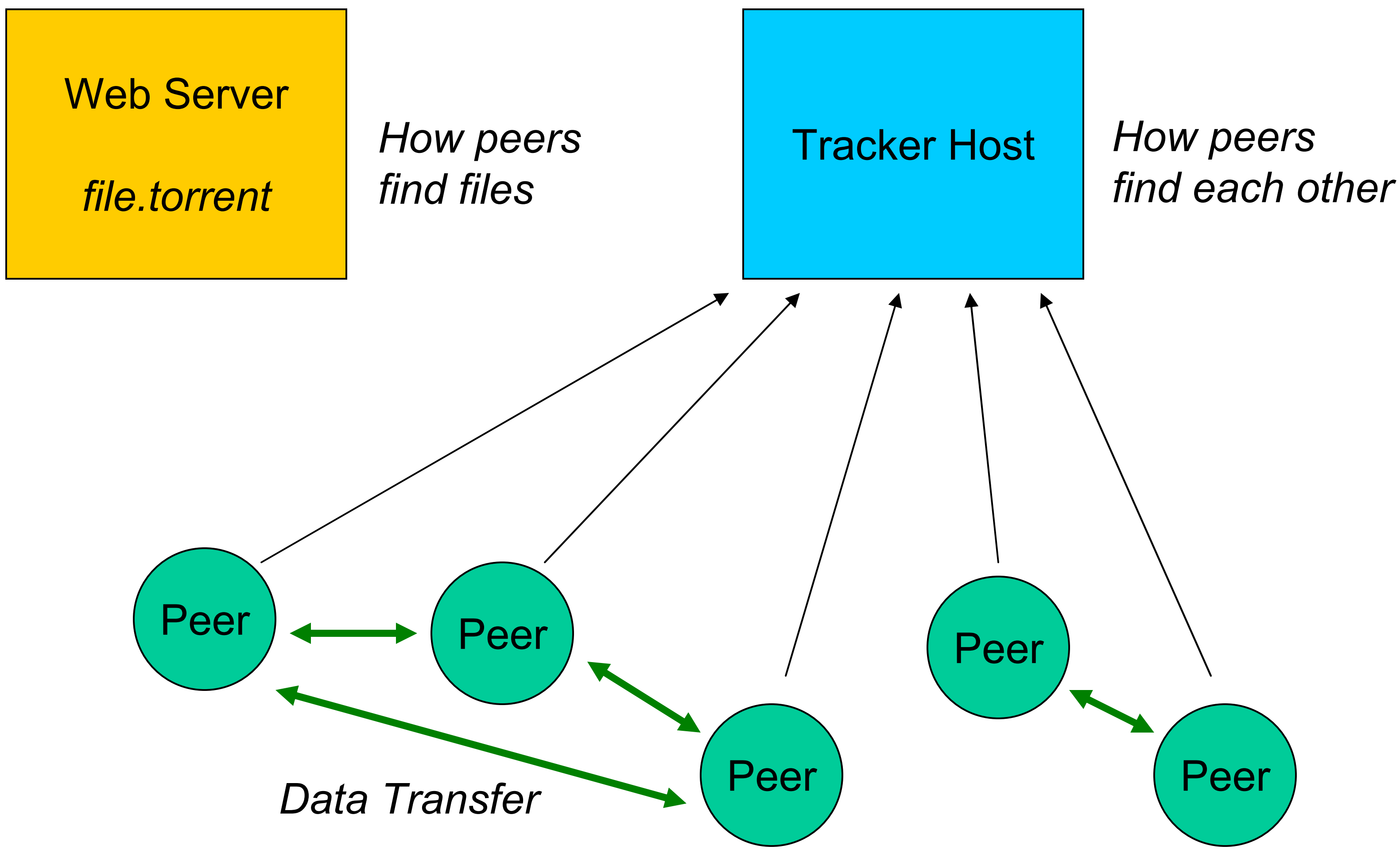
- Gnutella2 (G2)
  - DHT network from Shareaza
- Overnet
  - DHT network from eDonkey2000
- NEONet
  - DHT network from Morpheus

BitTorrent

```
 -------------------------------------------------------------------------------
| file:     tettnang-binary-i386-iso
| size:     2,285,617,943 (2.1 GB)
| dest:     /usr/local/src/BitTorrent-3.4.2/bt/tettnang-binary-i386-iso
| progress: ####################################_______________________________
| status:   finishing in 0:31:53 (53.6%)
| speed:    530.9 KB/s down - 112.9 KB/s up
| totals:     1.1 GB   down - 212.8 MB   up
| error(s): 
|
```

- 35% of all internet traffic?
  - Maybe in some places
- http://www.bittorrent.com/
- http://dessent.net/btfaq/
- http://en.wikipedia.org/wiki/Bittorrent

BitTorrent


Web Server
file.torrent
How peers
find files
Tracker Host
How peers
find each other
Peer
Peer
Peer
Peer
Peer
Data Transfer

BitTorrent Terminology

- Peer – a client with a file
  - "Seed" if complete file
  - "Leach" if partial file
- Swarm – the collection of all peers with parts or all of a given file
- "Swarming" – uploading partial files

BitTorrent – How it Works

- .torrent file
  - Tracker location(s)
  - 256 KB file pieces, SHA1 hash of each piece
- Downloads
  - 16 KB sub-piece transfers
  - 5 pipelined requests (~80 KB window)
  - 4 active peers, reselected every 10 seconds
    - New opportunistic unchoke every 30 seconds

BitTorrent – Reality Check

- One seed, One leach, Local LAN
  - 14 Mbps to 40 Mbps
- Two seeds, One leach, Local LAN
  - 14 Mbps + 14 Mbps = 30 Mbps
- One seed, One leach, CA to MD
  - 3.4 Mbps

mod_bt – Integrated Web/BitTorrent

- Combines tracker and seed functions in an Apache module
- Worse case, downloads like a web server
- BitTorrent function can only help
- www.crackerjack.net/mod_bt/

New BitTorrent Features

- Multiple trackers per torrent
- Trackerless torrents using a DHT
- Peer exchange (PEX)
- Encryption
- Web seeding

Abstract Storage Layers

- Internet caches
- I2 Logistical Networking
- Decouple LAN/WAN tuning issues
- Should storage be a network resource?

Internet Web Caches

- IRCache project 1995-2000
- Internet Cache Protocol (ICP)
  - RFC2186, RFC2187, Sep 1997
  - Used in Squid and several web cache products
  - http://icp.ircache.net/
- Hyper Text Caching Protocol (HTCP)
  - RFC2756, Jan 2000

Logistical Networking

- http://loci.cs.utk.edu/
- Internet Backplane Protocol (IBP)
- Logistical Backbone (L-Bone)
  - Directory of IBP depots
- exNodes
  - Collection of IBP allocations
- Logistical Runtime System (LoRS)
  - Upload and Download services

IBP

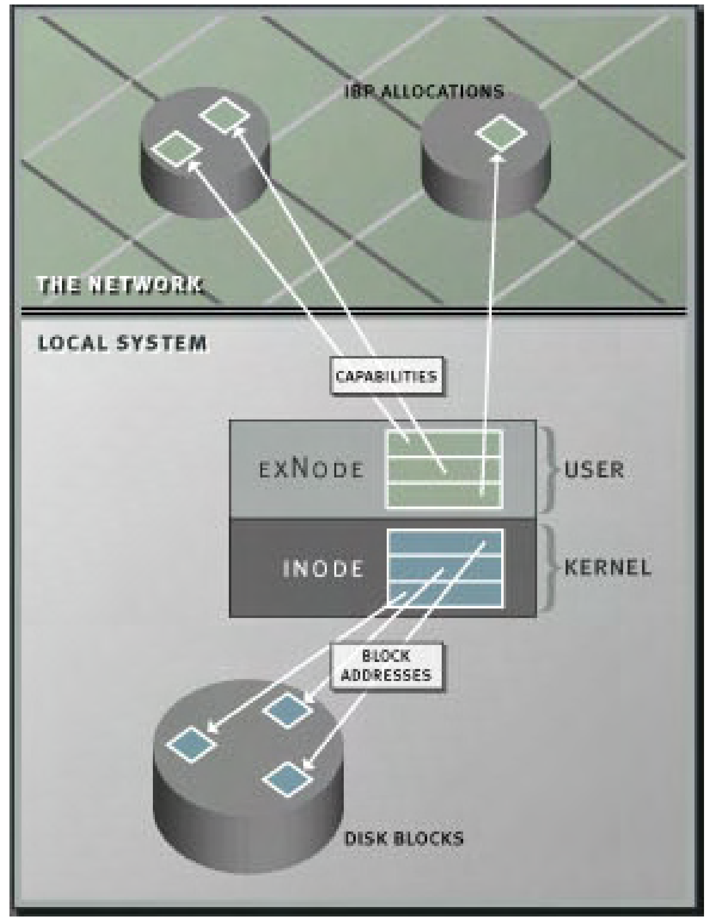
- Implements append-only byte arrays
- Often anonymous creation, limited time storage
- Need **capabilities** to access data
  – Crypto secure URL’s: read/write/manage
- Supports Data Mover plugins
- 4 GB allocation limit?

IBP Commands

- Storage Management
  - `IBP_allocate, IBP_manage`
- Data Transfer
  - `IBP_store, IBP_load, IBP_copy, IBP_mcopy`
- Depot Management
  - `IBP_status`

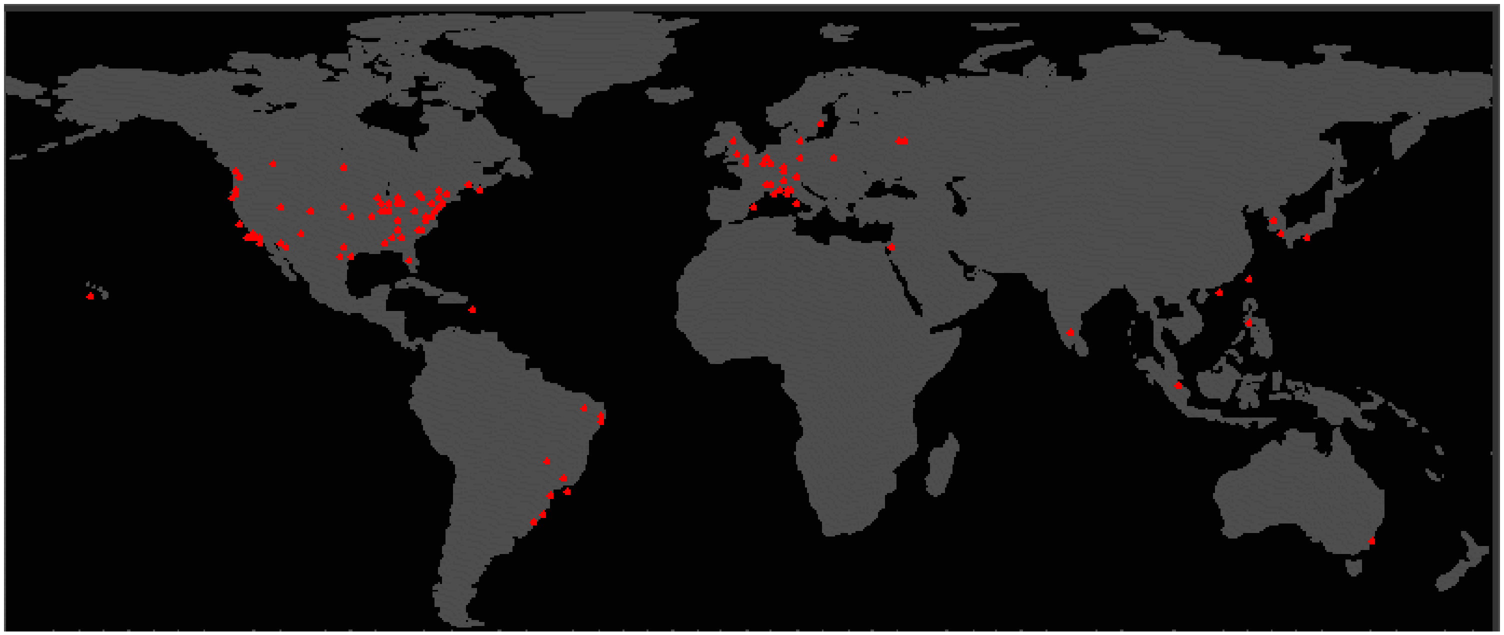
exNodes

- Holds information and capabilities about storage on one or more IBP depots
  - *think inode for L-Bone storage*
- XML representation
  - Can be emailed, etc.



L-Bone

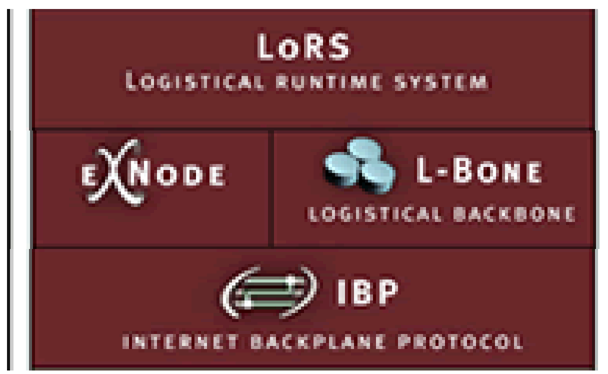
L-BONE LOGISTICAL BACKBONE



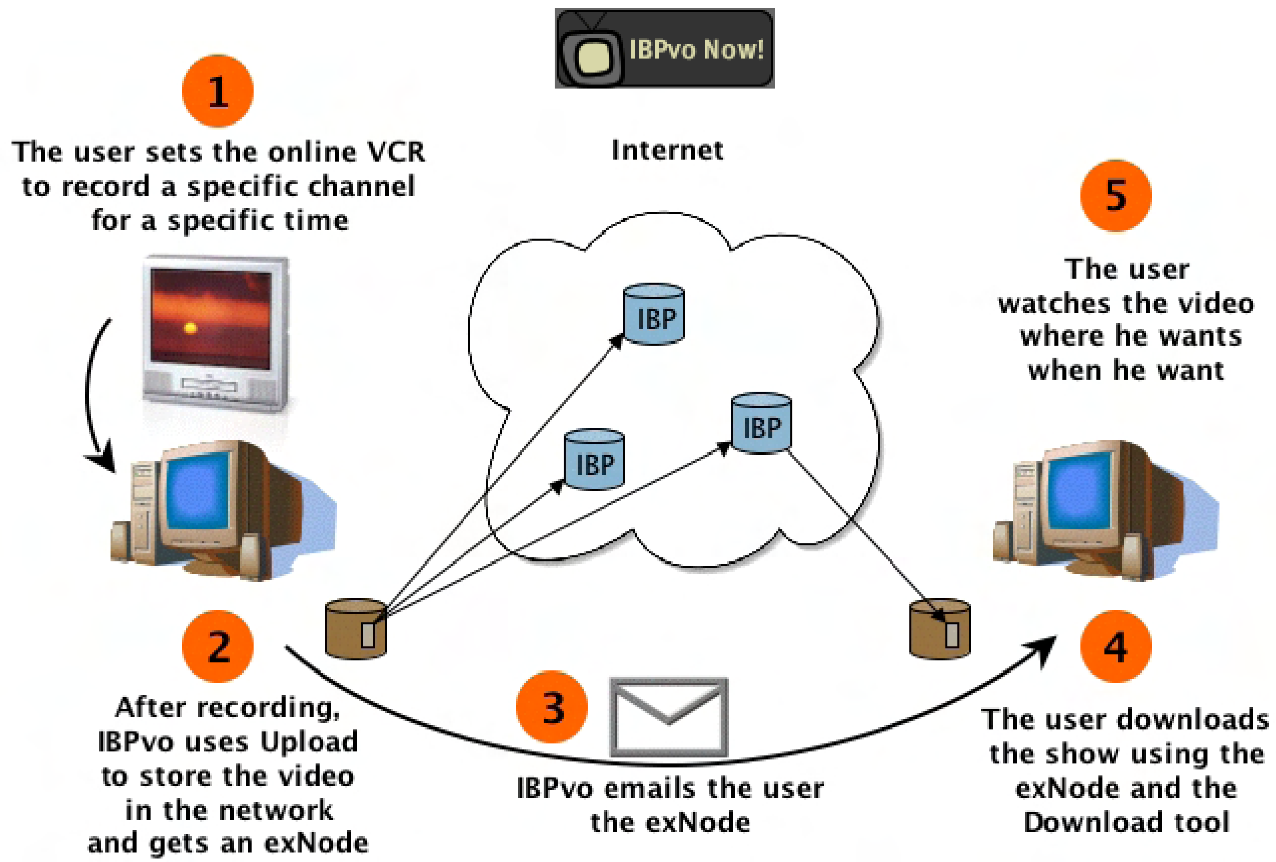
557 public IBP servers in 30 countries and 38 American states.
5.6 TB online (21 TB listed), 99% free - Oct 2006

Logistical Run Time System

- LoRS tools for using the L-Bone
- Finds allocations, creates exNodes, etc.
- Command line, visual, and java versions



Example Application: IBPvo



http://promise.sinrg.cs.utk.edu/ibpvo/about.html

PLANETLAB

- Think world-wide shared Linux cluster
- Over 100 universities doing research in overlay networks, routing, P2P, security, performance, etc.
- Over half of the L-Bone servers are PlanetLab hosts
- Internet2 has PlanetLab hosts on the backbone

Review

- High performance comes from all levels
  - They complement each other
- Network capacity vs. speed
- How TCP throughput depends on delay, loss, packet size
- Importance of window and buffer sizes and how to tune them

Review

- How to test and debug network performance
- Protect your data, the network isn't perfect
- TCP continues to improve, most notably in congestion control
- Alternatives exist: SCTP, DCCP, and UDP transport experiments
- Storage Area Networks and IP storage

Review

- Peer to Peer is where the file transfer action is
  - But most P2P assumes poor network performance
  - High performance BitTorrent hasn't been written yet
- There is a lot of work today in DHT and abstract storage layers
- Network Channels are coming!
  - The future will be interesting

Recommended Resources

- System tuning details
  - http://www.psc.edu/networking/projects/tcptune/
- Tom Dunigan's Network Performance Links
  - http://www.csm.ornl.gov/~dunigan/netperf/netlinks.html
- SLAC's Network Monitoring pages
  - http://www-iepm.slac.stanford.edu/
- CAIDA Internet Measurement Tool Taxonomy
  - http://www.caida.org/tools/

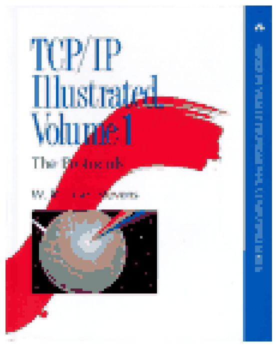
Recommended Resources

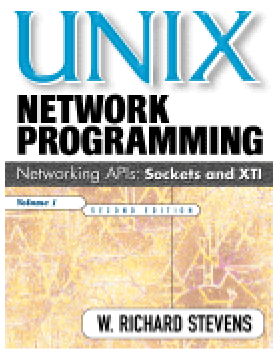
- nuttcp and Iperf for TCP and UDP testing
  - ftp://ftp.lcp.nrl.navy.mil/pub/nuttcp/
  - http://dast.nlanr.net/Projects/Iperf/
- tcptrace and xplot for TCP traces
  - http://www.tcptrace.org/
- Web100 / Net100
  - http://www.web100.org/
  - http://www.csm.ornl.gov/~dunigan/net100/

Resources

- Request For Comments (RFC)
  - http://www.rfc-editor.org/
  - ftp://ftp.rfc-editor.org/in-notes/rfc968.txt
- Internet Engineering Task Force (IETF)
  - http://www.ietf.org/
  - http://www.ietf.org/html.charters/wg-dir.html

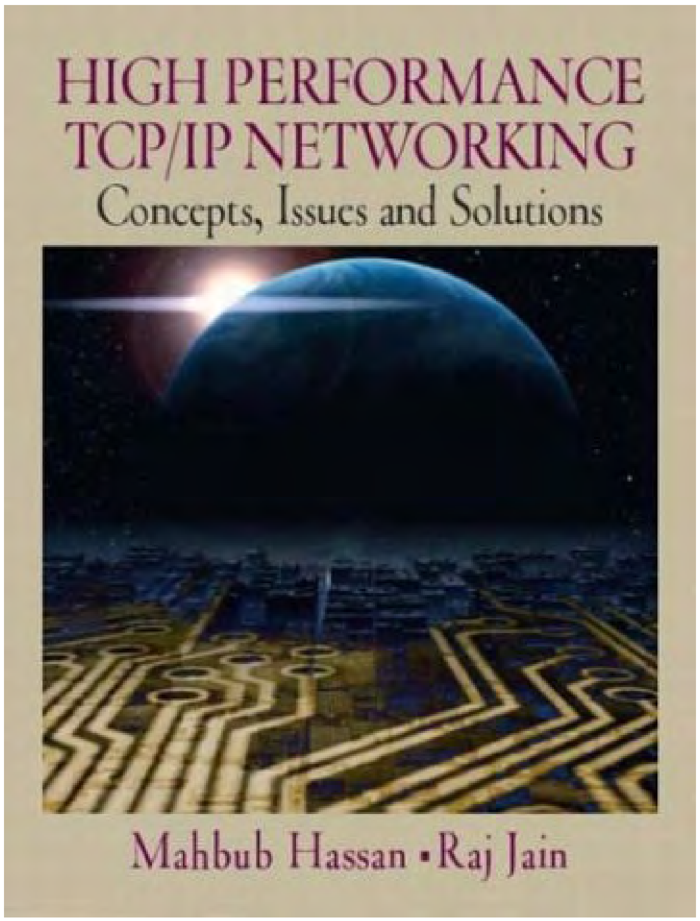
W. Richard Stevens' Books



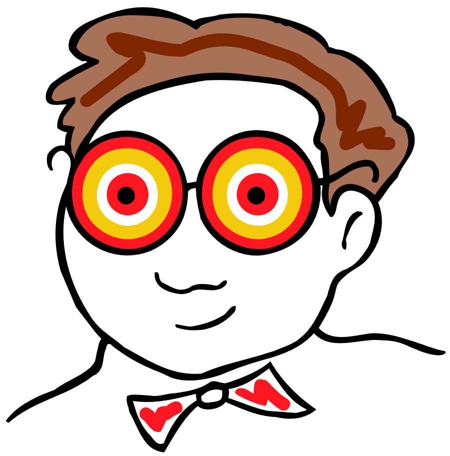


- TCP/IP Illustrated, 3 volumes
- Unix Network Programming, 2 volumes
- http://www.kohala.com/start/

Resources



- High Performance TCP/IP Networking
- M. Hassan, R. Jain
- October, 2003
- ISBN: 0130646342



Thank You!

Phillip Dykstra
WareOnEarth Communications Inc.
2109 Mergho Impasse
San Diego, CA 92110
phil@sd.wareonearth.com
619-574-7796