

Bulk Data Transfer Techniques for High-Speed Wide-Area Networks

Brian L. Tierney

ESnet

Lawrence Berkeley National Laboratory

<http://fasterdata.es.net/>

(updated Sept 24, 2009)

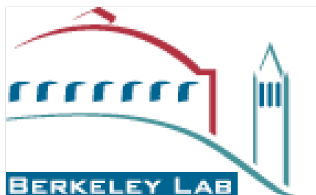


U.S. DEPARTMENT OF
ENERGY

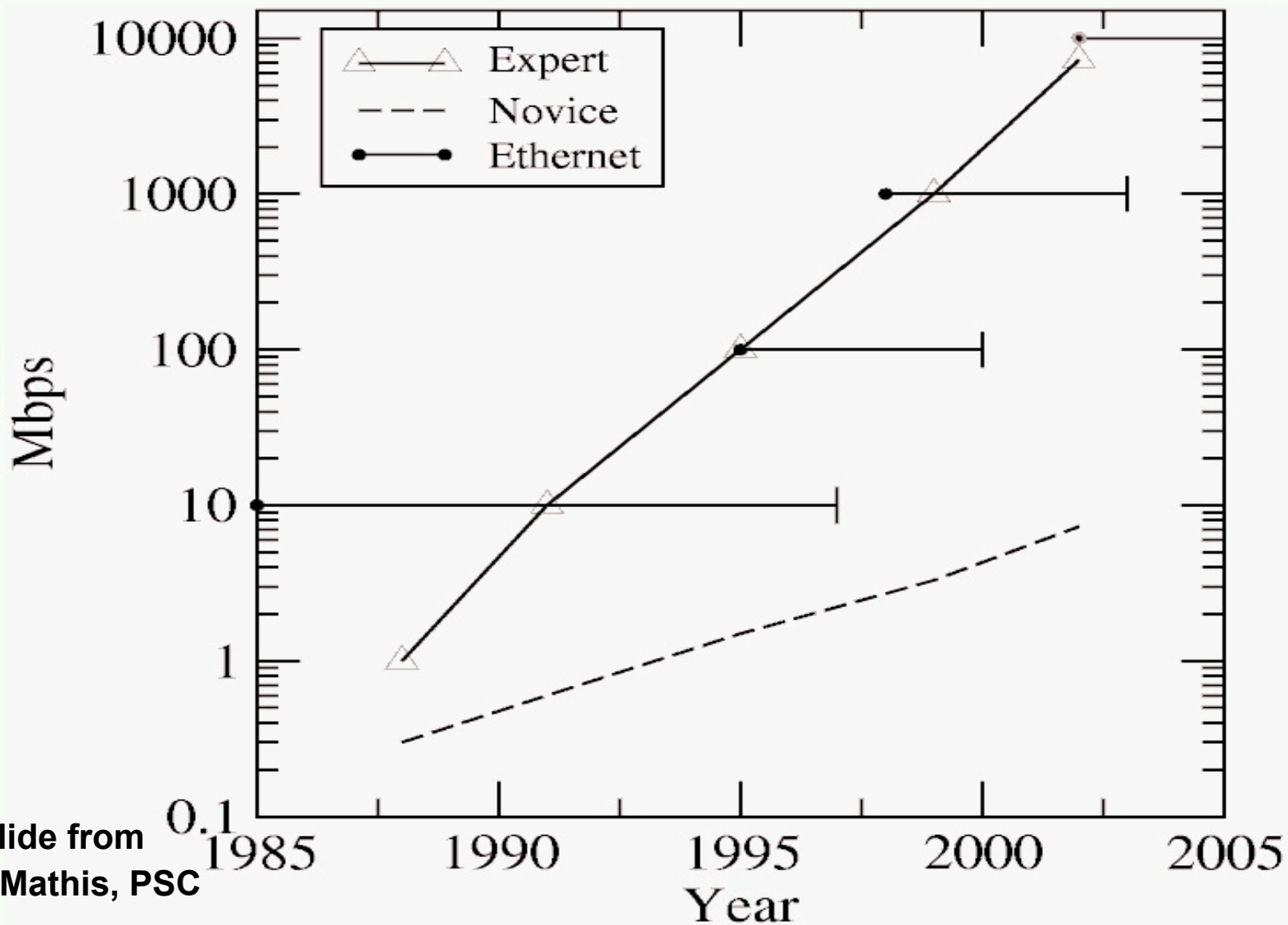
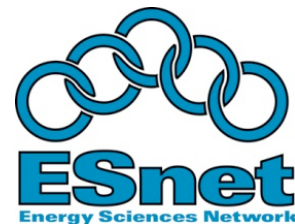
Office of
Science

**Why
does the
Network
seem so
slow?**

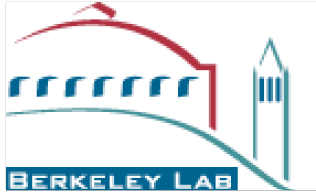




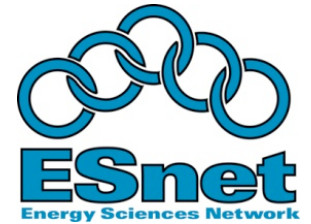
Wizard Gap



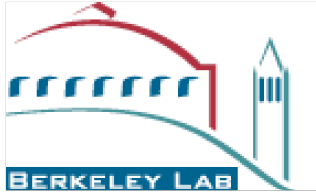
Slide from
Matt Mathis, PSC



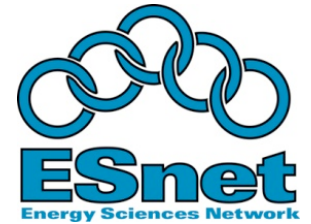
Today's Talk



- **This talk will cover:**
 - Some Information to help you become a “wizard”
 - Work being done so you don't have to be a wizard
- **Goal of this talk:**
 - Help you fully optimize wide area bulk data transfers
 - or help your users do this
- **Outline**
 - TCP Issues
 - Bulk Data Transfer Tools
 - Network Monitoring Tools
 - New TCP Stacks
 - how they help with, but not eliminate, the “wizard gap”

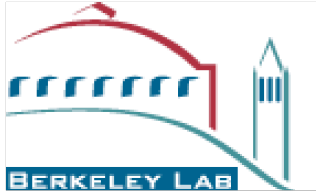


Time to Copy 1 Terabyte

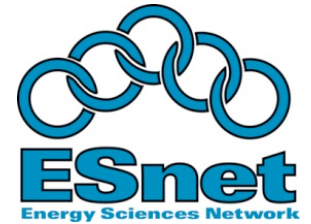


- **10 Mbps network : 300 hrs (12.5 days)**
- **100 Mbps network : 30 hrs**
- **1 Gbps network : 3 hrs**
- **10 Gbps network : 20 minutes**
 - need fast disk array for this

- **Compare these speeds to:**
 - **USB 2.0 portable disk**
 - 60 MB/sec (480 Mbps) peak
 - 20 MB/sec (160 Mbps) reported on line
 - 5-10 MB/sec reported by colleagues
 - 15-40 hours to load 1 Terabyte



Bandwidth Requirements

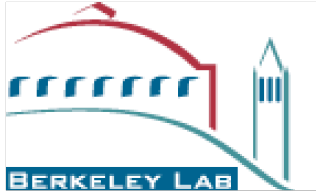


Bandwidth Requirements to move Y Bytes of data in Time X

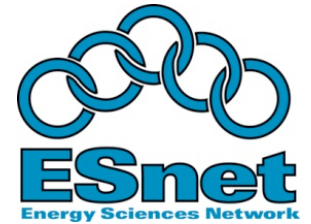
Bits per Second Requirements

10PB	25,020.0 Gbps	3,127.5 Gbps	1,042.5 Gbps	148.9 Gbps	34.7 Gbps
1PB	2,502.0 Gbps	312.7 Gbps	104.2 Gbps	14.9 Gbps	3.5 Gbps
100TB	244.3 Gbps	30.5 Gbps	10.2 Gbps	1.5 Gbps	339.4 Mbps
10TB	24.4 Gbps	3.1 Gbps	1.0 Gbps	145.4 Mbps	33.9 Mbps
1TB	2.4 Gbps	305.4 Mbps	101.8 Mbps	14.5 Mbps	3.4 Mbps
100GB	238.6 Mbps	29.8 Mbps	9.9 Mbps	1.4 Mbps	331.4 Kbps
10GB	23.9 Mbps	3.0 Mbps	994.2 Kbps	142.0 Kbps	33.1 Kbps
1GB	2.4 Mbps	298.3 Kbps	99.4 Kbps	14.2 Kbps	3.3 Kbps
100MB	233.0 Kbps	29.1 Kbps	9.7 Kbps	1.4 Kbps	0.3 Kbps
	1H	8H	24H	7Days	30Days

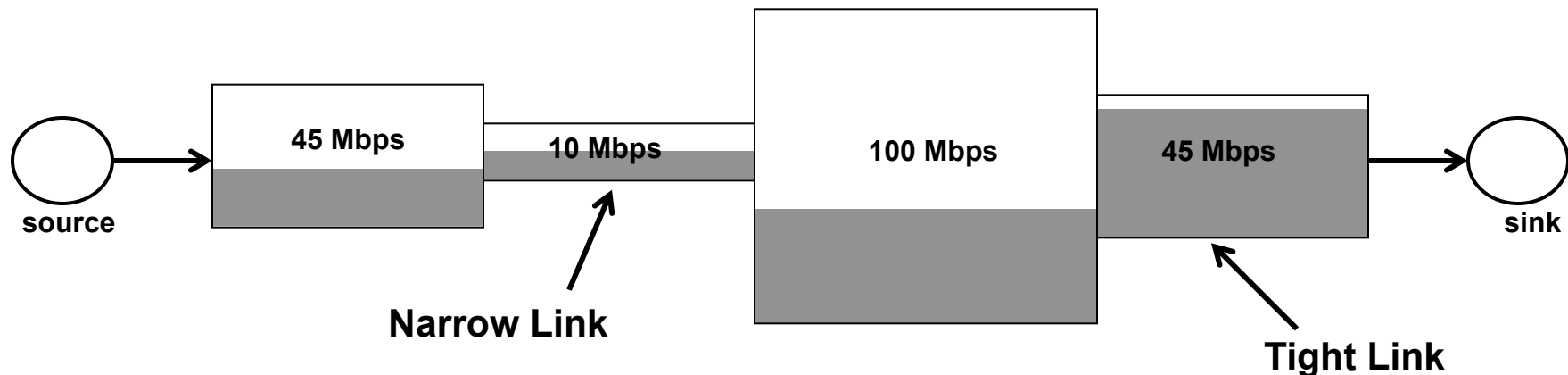
This table available at <http://fasterdata.es.net>

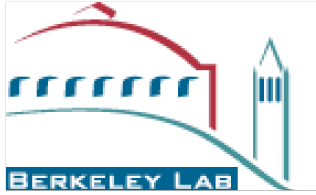


Terminology

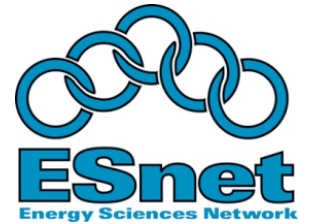


- The term “Network Throughput” is vague and should be avoided
 - **Capacity: link speed**
 - Narrow Link: link with the lowest capacity along a path
 - Capacity of the end-to-end path = capacity of the narrow link
 - **Utilized bandwidth: current traffic load**
 - **Available bandwidth: capacity – utilized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

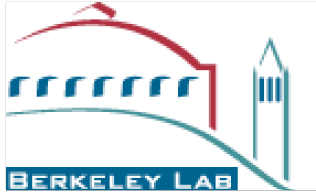




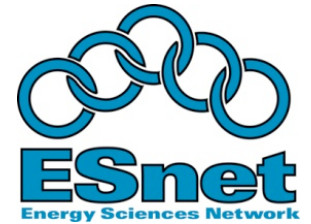
More Terminology



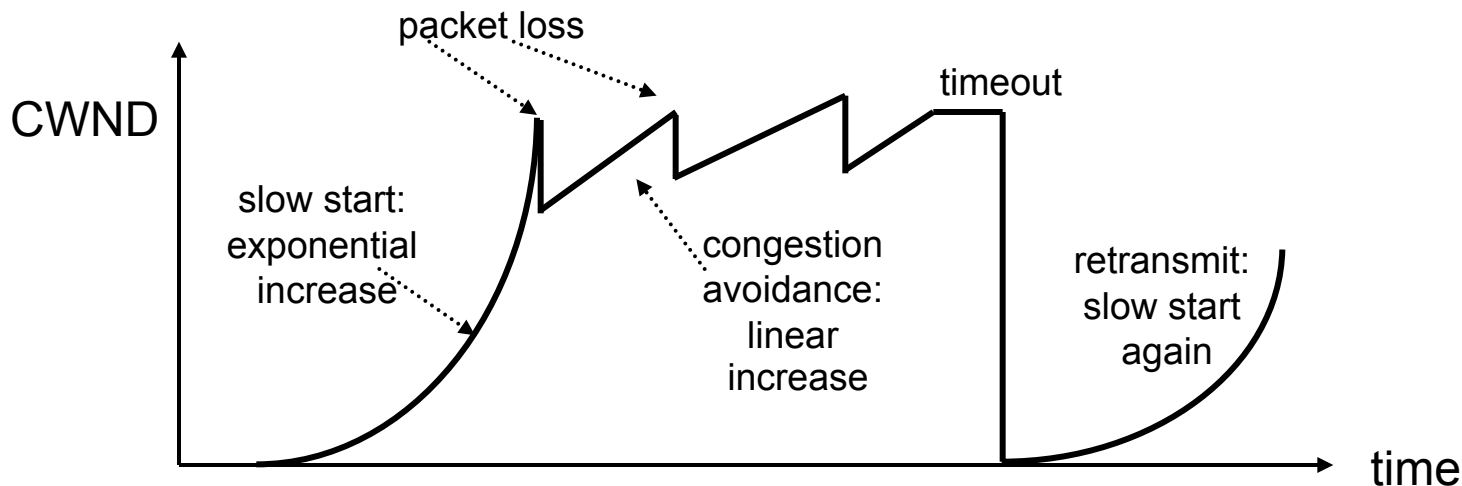
- **Latency: time to send 1 packet from the source to the destination**
- **RTT: Round-trip time**
- **Bandwidth*Delay Product = BDP**
 - The number of bytes in flight to fill the entire path
 - Example: 100 Mbps path; ping shows a 75 ms RTT
 - $BDP = 100 * 0.075 = 7.5 \text{ Mbits (940 KBytes)}$
- **LFN: Long Fat Networks**
 - A network with a large BDP

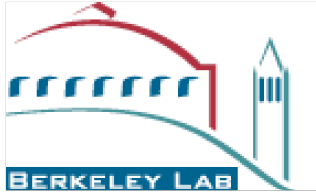


How TCP works: A very short overview

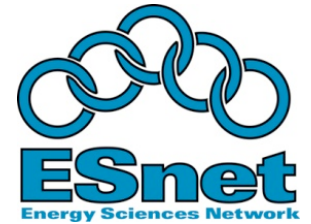


- **Congestion window (CWND) = the number of packets the sender is allowed to send**
 - **The larger the window size, the higher the throughput**
 - Throughput = Window size / Round-trip Time
- **TCP Slow start**
 - **exponentially increase the congestion window size until a packet is lost**
 - this gets a rough estimate of the optimal congestion window size

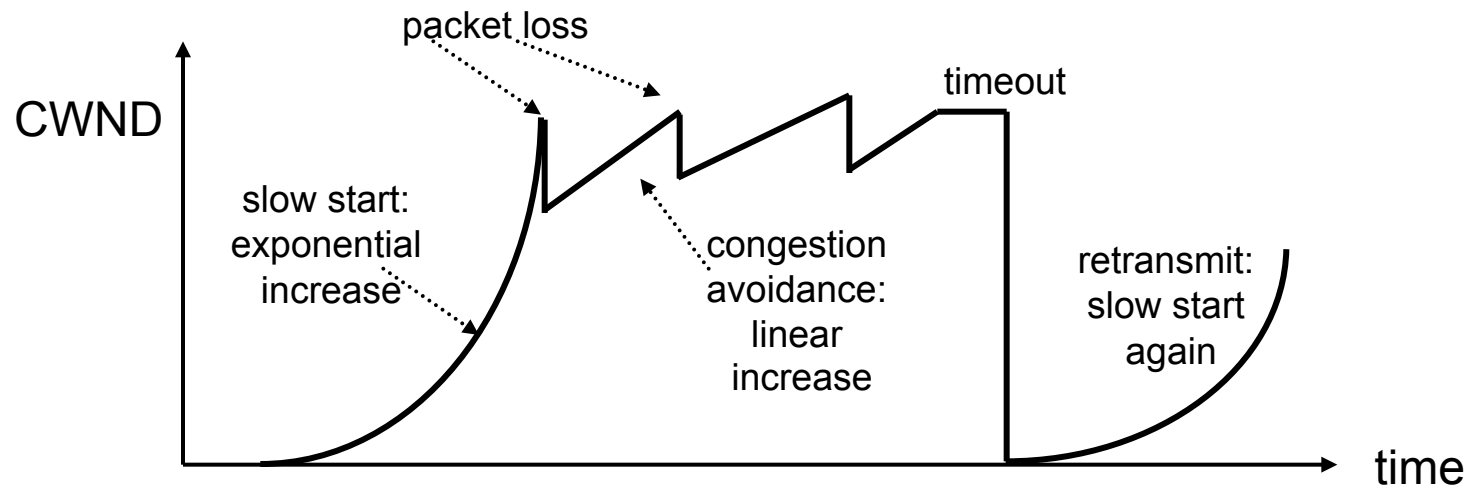


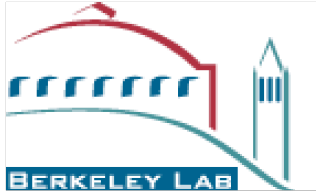


TCP Overview

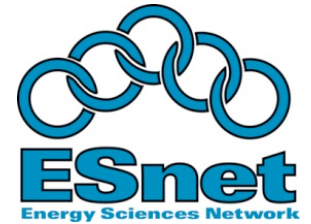


- Congestion avoidance
 - additive increase: starting from the rough estimate, linearly increase the congestion window size to probe for additional available bandwidth
 - multiplicative decrease: cut congestion window size aggressively if a timeout occurs

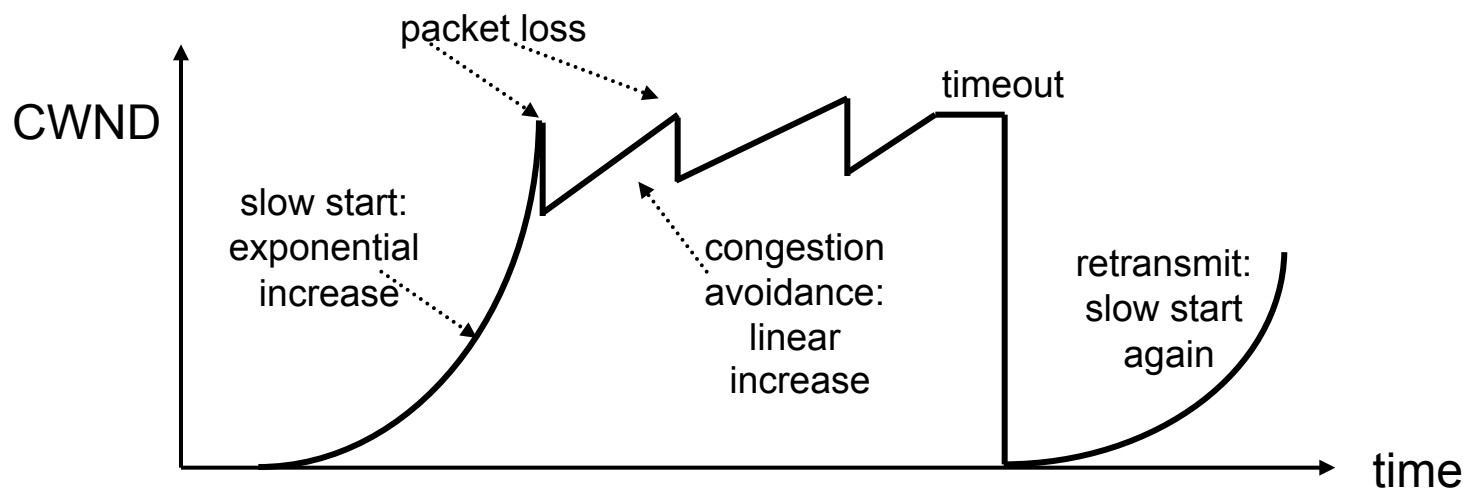


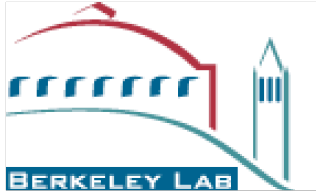


TCP Overview



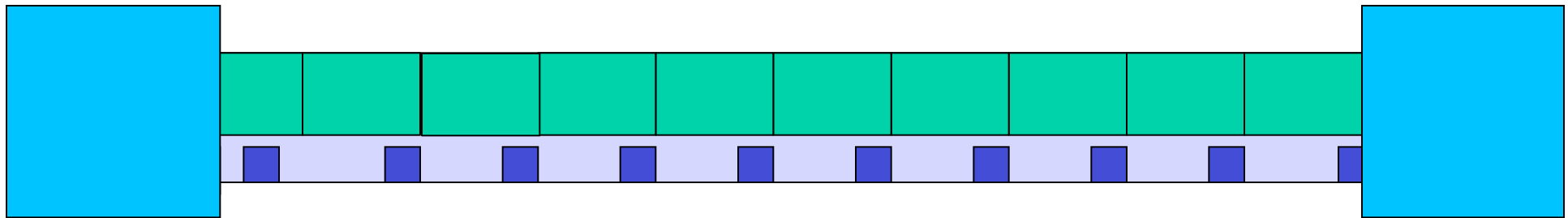
- **Fast Retransmit:** retransmit after 3 duplicate acks (got 3 additional packets without getting the one you are waiting for)
 - this prevents expensive timeouts
 - no need to go into “slow start” again
- **At steady state, CWND oscillates around the optimal window size**
- **With a retransmission timeout, slow start is triggered again**





TCP Window and ACKs

Small TCP Buffer Size



Data Packet



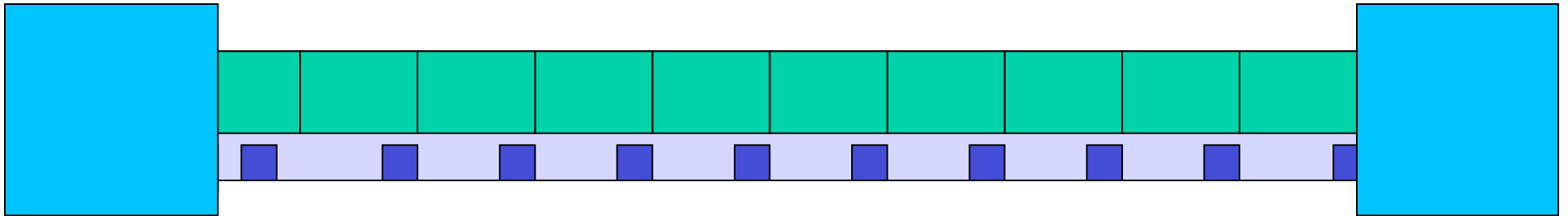
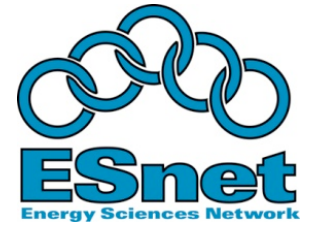
Acknowledgement

Animated Slide from Globus Project, ANL

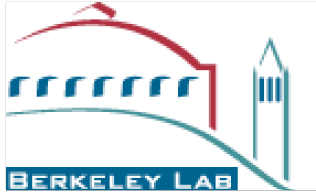


TCP Window and ACKs

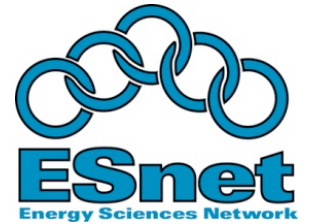
Optimized TCP Buffer Size



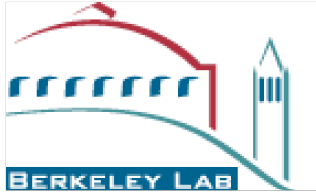
Animated Slide from Globus Project, ANL



TCP Performance Issues



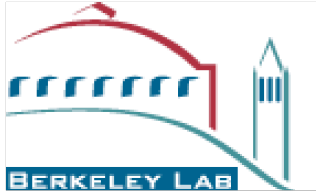
- **Getting good TCP performance over high-latency high-bandwidth networks is not easy!**
- **You must keep the TCP window full**
 - the size of the window is directly related to the network latency
- **Easy to compute max throughput:**
 - **Throughput = buffer size / latency**
 - **eg: 64 KB buffer / 40 ms path = 1.6 KBytes (12.8 Kbits) / sec**



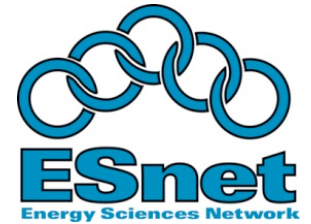
Setting the TCP buffer sizes



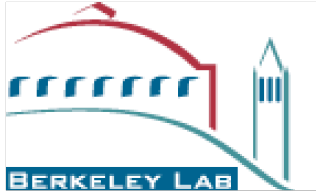
- **It is critical to use the optimal TCP send and receive socket buffer sizes for the link you are using.**
 - **Recommended size to fill the pipe**
 - 2 x Bandwidth Delay Product (BDP)
 - **Recommended size to leave some bandwidth for others**
 - around 20% of (2 x BDP) = .4 * BDP
- **Default TCP buffer sizes are way too small for today's high speed networks**
 - **Until recently, default TCP send/receive buffers were typically 64 KB**
 - **tuned buffer to fill LBL to BNL link: 10 MB**
 - 150X bigger than the default buffer size
 - **with default TCP buffers, you can only get a small % of the available bandwidth!**



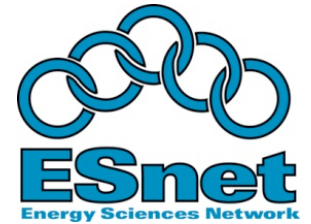
Buffer Size Example



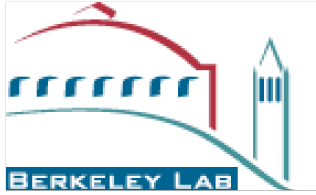
- **Optimal Buffer size formula:**
 - **buffer size = 20% * (2 * bandwidth * RTT)**
- **ping time (RTT) = 50 ms**
- **Narrow link = 500 Mbps (62 MBytes/sec)**
 - **e.g.: the end-to-end network consists of all 1000 BT ethernet and OC-12 (622 Mbps)**
- **TCP buffers should be:**
 - **.05 sec * 62 * 2 * 20% = 1.24 MBytes**



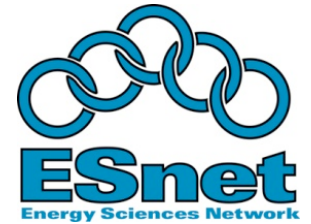
Socket Buffer Autotuning



- **To solve the buffer tuning problem, based on work at LANL and PSC, Linux OS added TCP Buffer autotuning**
 - Sender-side TCP buffer autotuning introduced in Linux 2.4
 - Receiver-side autotuning added in Linux 2.6
- **Most OS's now include TCP autotuning**
 - TCP send buffer starts at 64 KB
 - As the data transfer takes place, the buffer size is continuously re-adjusted up max autotune size
- **Current OS Autotuning default maximum buffers**
 - Linux 2.6: 256K to 4MB, depending on version
 - FreeBSD 7: 256K
 - Windows Vista: 16M
 - Mac OSX 10.5: 8M



Autotuning Settings



- **Linux 2.6**

```
net.core.rmem_max = 16777216
net.core.wmem_max = 16777216
# autotuning min, default, and max number of bytes to use
net.ipv4.tcp_rmem = 4096 87380 16777216
net.ipv4.tcp_wmem = 4096 65536 16777216
```

- **FreeBSD 7.0**

```
net.inet.tcp.sendbuf_auto=1
net.inet.tcp.recvbuf_auto=1
net.inet.tcp.sendbuf_max=16777216
net.inet.tcp.recvbuf_max=16777216
```

- **Windows Vista**

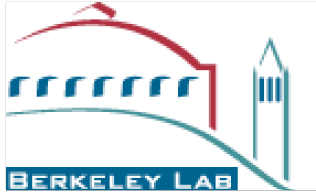
```
netsh interface tcp set global autotunninglevel=normal
```

- max buffer fixed at 16MB

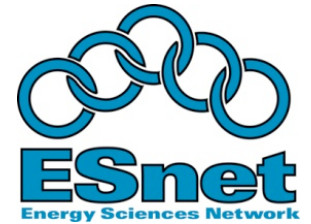
- **OSX 10.5 (“Self-Tuning TCP”)**

```
kern.ipc.maxsockbuf=16777216
```

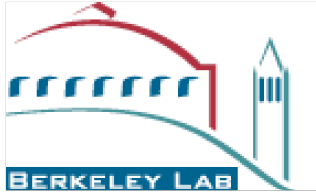
- **For more info, see: <http://fasterdata.es.net/TCP-Tuning/>**



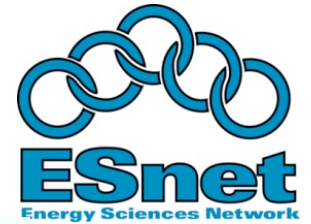
Sample Buffer Sizes



- **LBL to (50% of pipe)**
 - **SLAC (RTT = 2 ms, narrow link = 1000 Mbps) : 256 KB**
 - **BNL: (RTT = 80 ms, narrow link = 1000 Mbps): 10 MB**
 - **CERN: (RTT = 165 ms, narrow link = 1000 Mbps): 20.6 MB**
- **Note: default buffer size is usually only 64 KB, and default maximum autotuning buffer size for is often only 256KB**
 - **e.g.: FreeBSD 7.2 Autotuning default max = 256 KB**
 - **10-150 times too small!**
- **Home DSL, US to Europe (RTT = 150, narrow link = 2 Mbps): 38 KB**
 - **Default buffers are OK.**

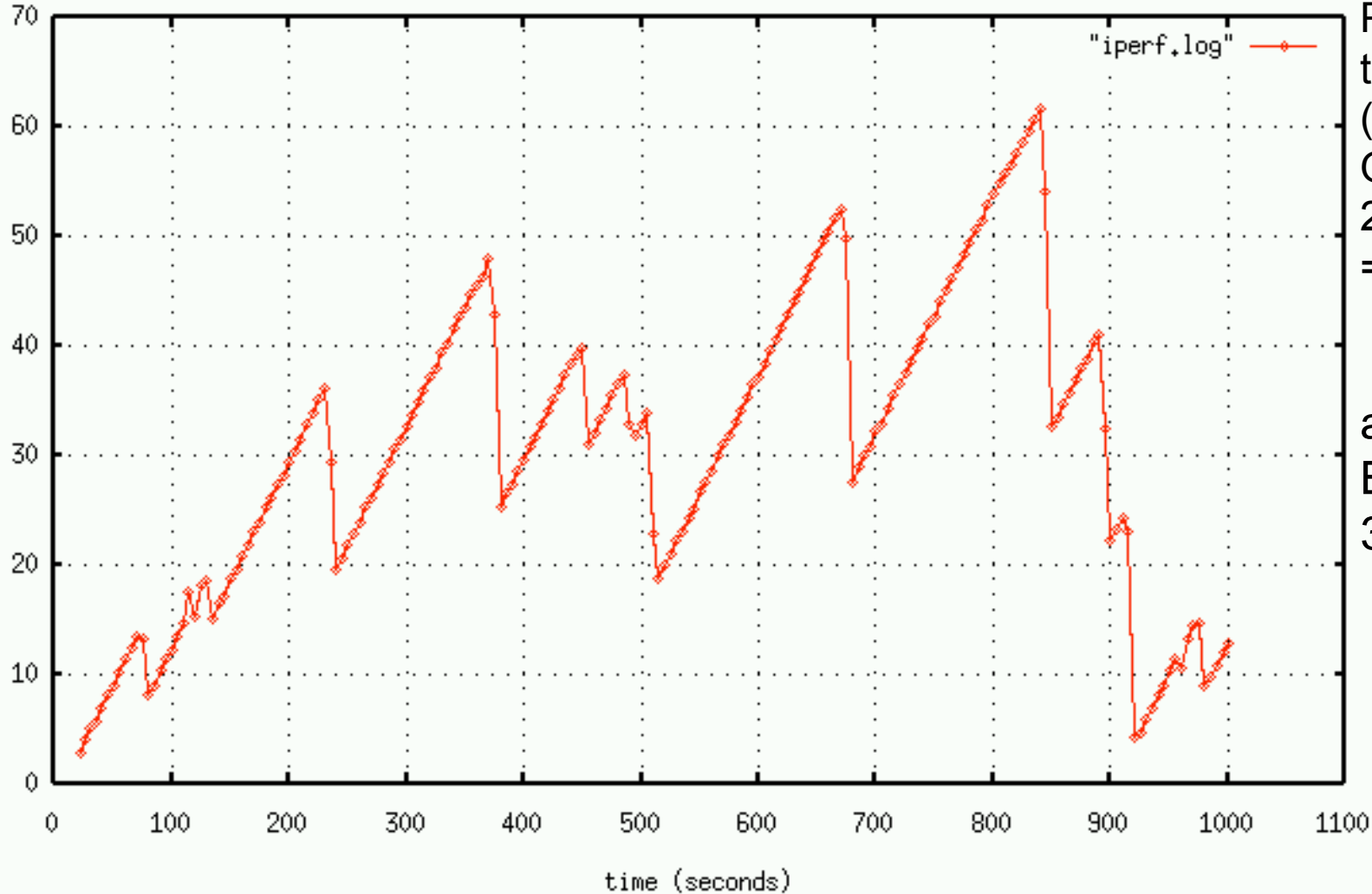


More Problems: TCP congestion control



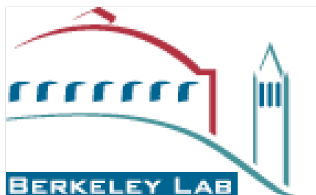
throughput
Mbits/second

iperf results

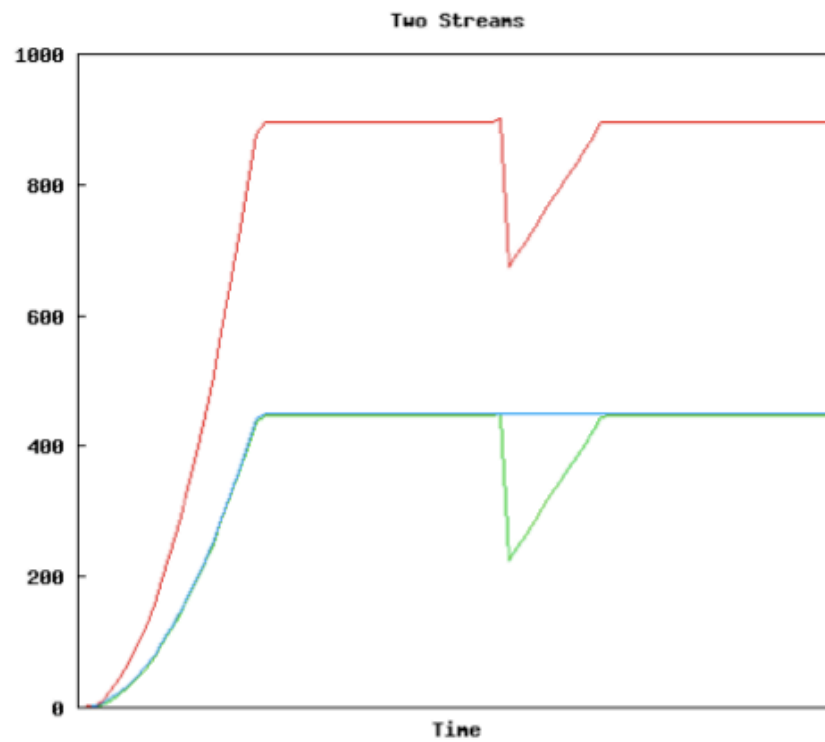
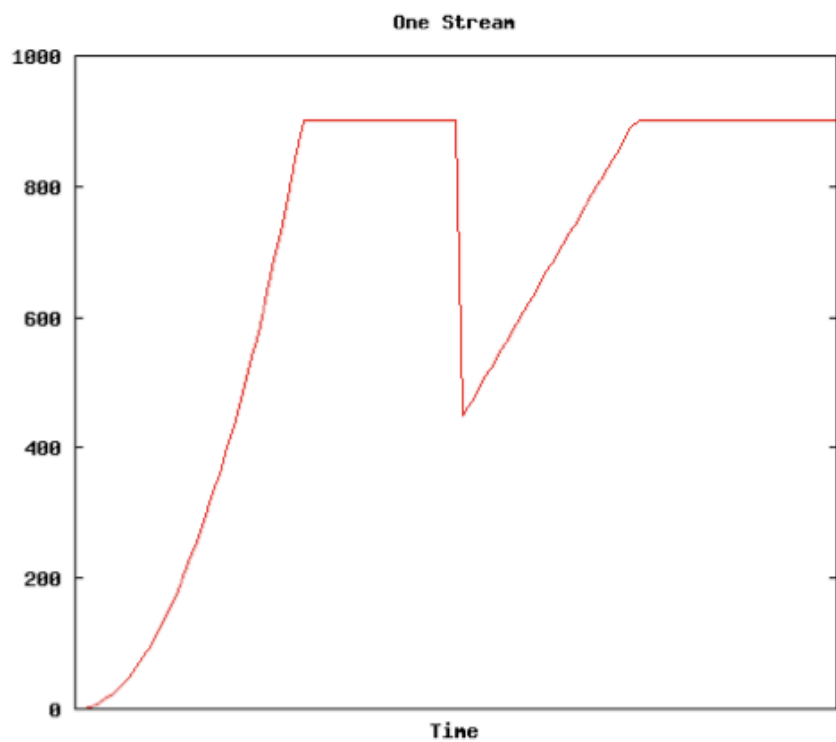
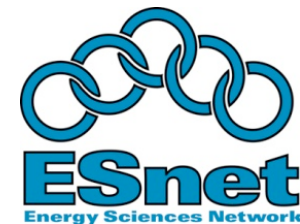


Path = LBL
to CERN
(Geneva)
OC-3, (in
2000), RTT
= 150 ms

average
BW =
30 Mbps

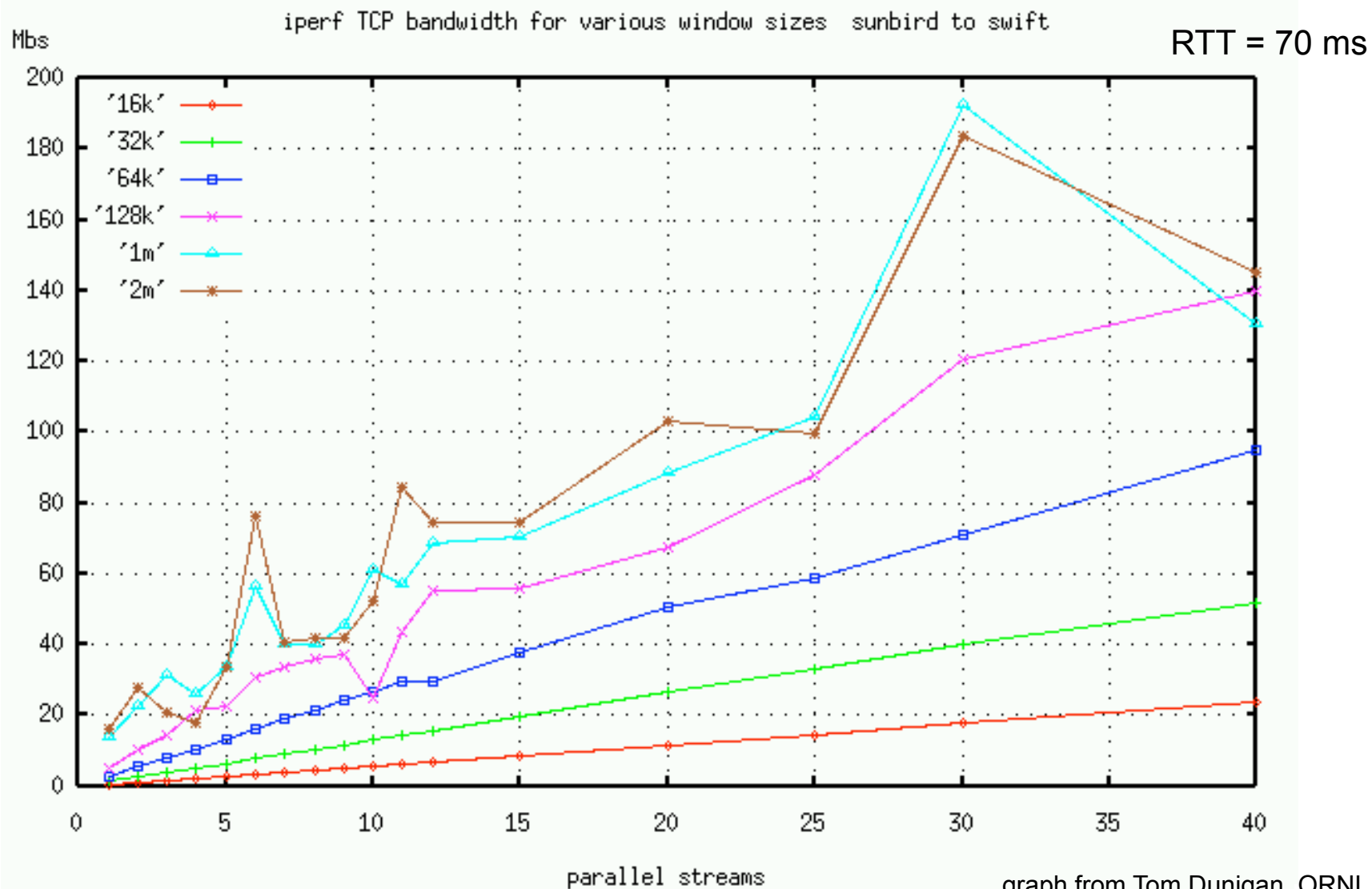


Parallel Streams Can Help

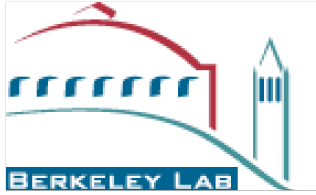




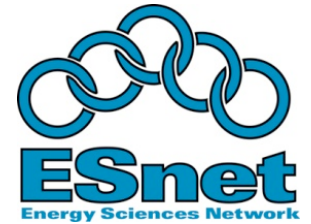
Parallel Streams Results with various TCP window sizes



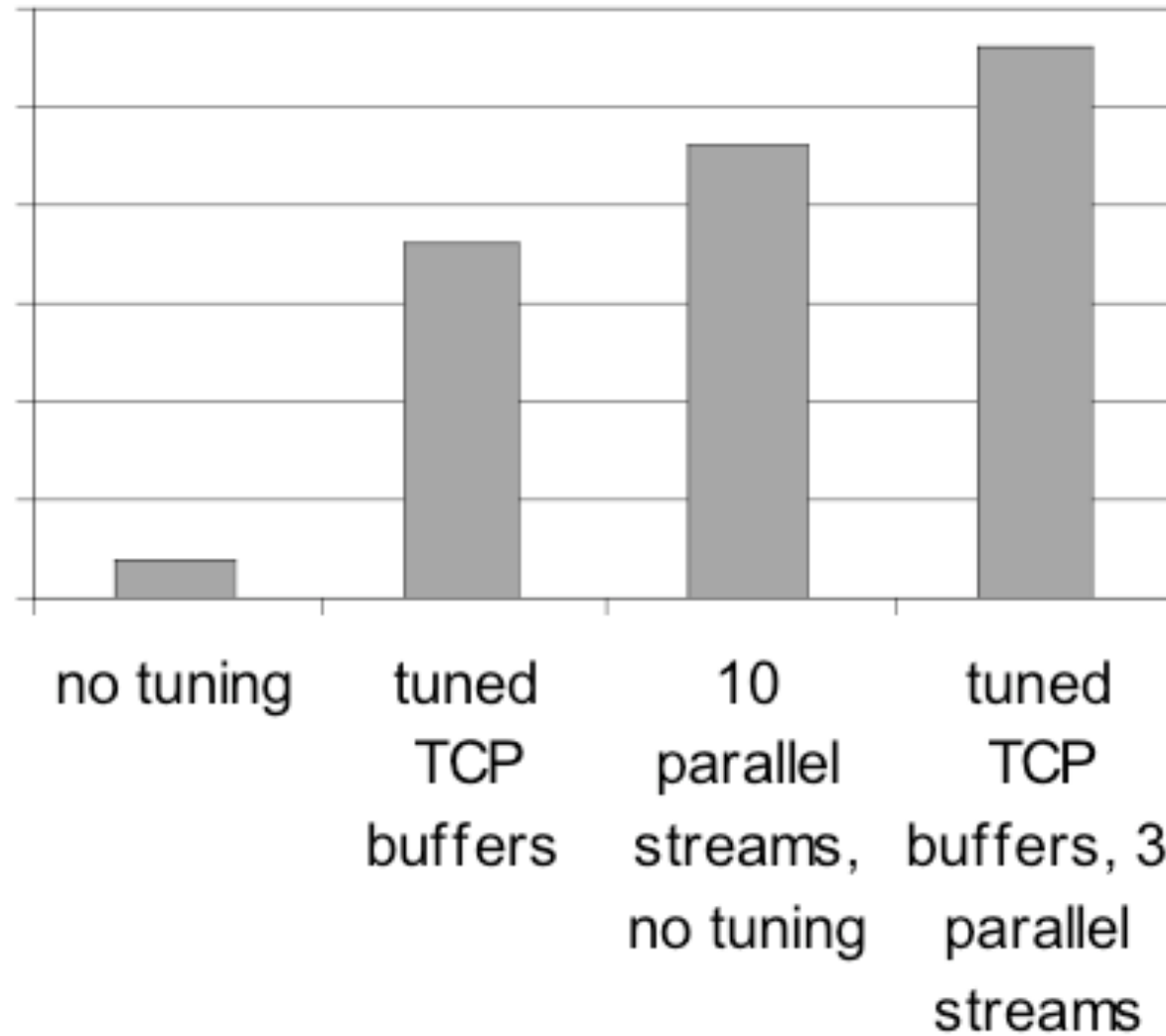
graph from Tom Dunigan, ORNL

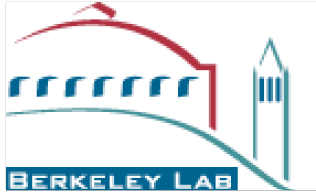


Tuned Buffers vs. Parallel Steams

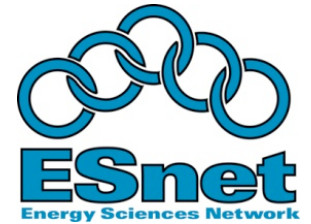


Throughput (Mbits/sec)

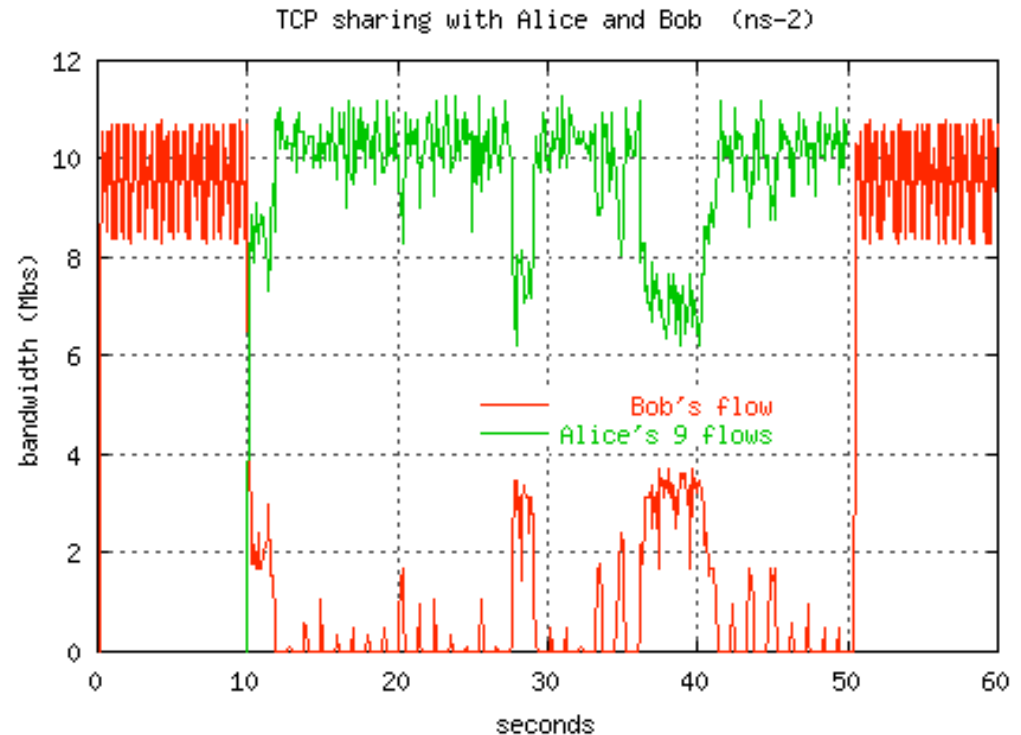




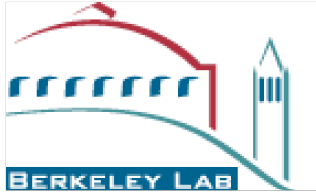
Parallel Streams Issues



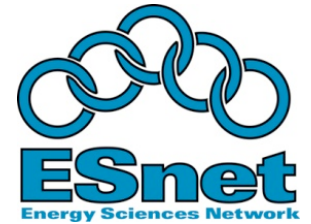
- **Potentially unfair**
- **Places more load on the end hosts**
- **But they are necessary when you don't have root access, and can't convince the sysadmin to increase the max TCP buffers**



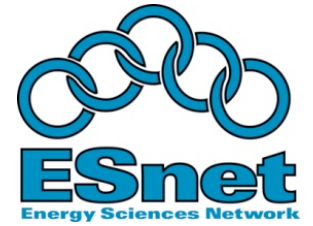
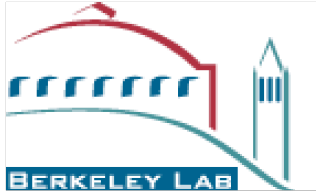
graph from Tom Dunigan, ORNL



Summary so far



- **To optimize TCP throughput, do the following:**
 - Use a newer OS that supports TCP buffer autotuning
 - increase the maximum TCP autotuning buffer size
 - use a few parallel streams if possible
- **But also, try to ‘play nice’:**
 - Leave some bandwidth for others.
 - e.g.: Don’t try to completely fill your networks
1000BT uplink
 - **Good ‘rule of thumb’:**
 - Try for 100-200 Mbps per stream, and use around 4 streams

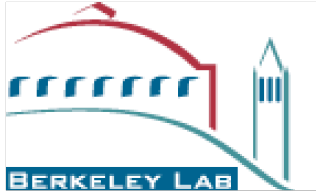


Bulk Transfer Tools

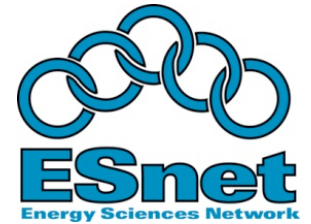


U.S. DEPARTMENT OF
ENERGY

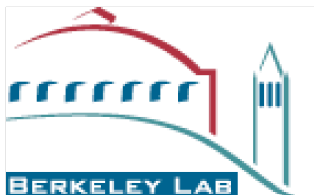
Office of
Science



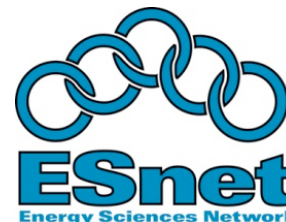
Sample Data Transfer Results: LBL to BNL



- **Using the right tool is very important**
 - **scp / sftp : 10 Mbps**
 - standard Unix file copy tools
 - fixed 1 MB TCP window in openSSH
 - ⇒ only 64 KB in openssh versions < 4.7
 - **ftp : 400-500 Mbps**
 - assumes TCP buffer autotuning
 - **parallel stream FTP: 800-900 Mbps**

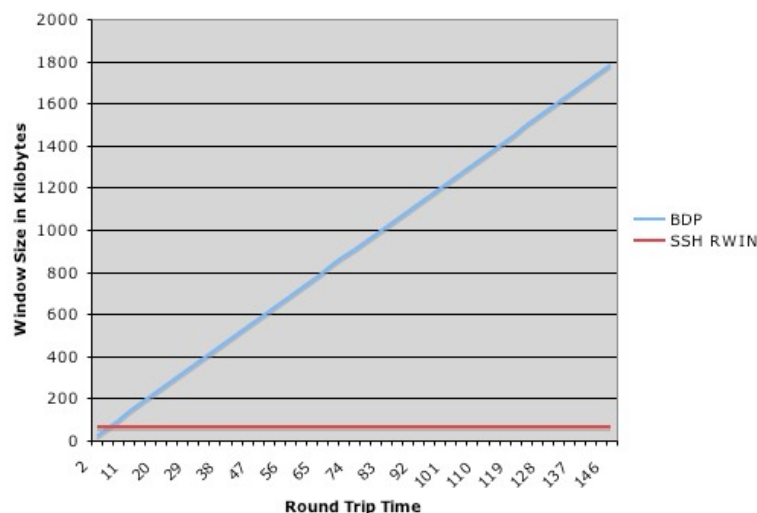


scp

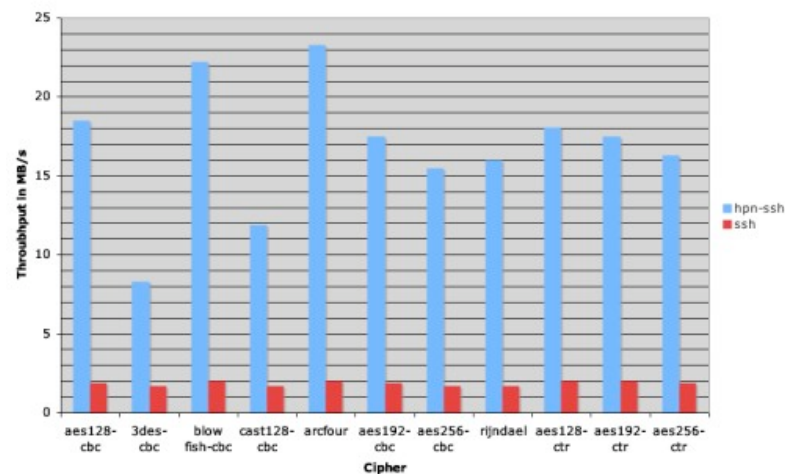


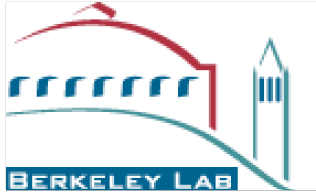
- Don't use scp to copy large files across a WAN!
 - scp has its own internal 1MB buffering/windowing that prevents it from ever being able to fill LFNs!
- Explanation of problem and openssh patch solution from PSC:
 - <http://www.psc.edu/networking/projects/hpn-ssh/>

BDP versus SSH Receive Window for a 100Mbps Path

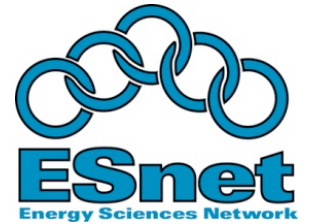


Throughput Speeds of HPN-SSH Versus SSH

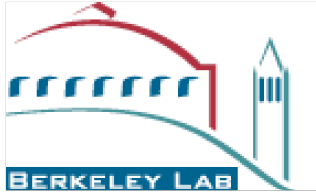




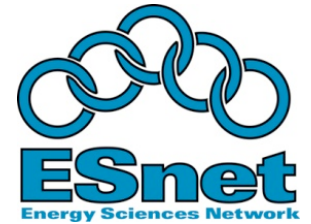
sftp



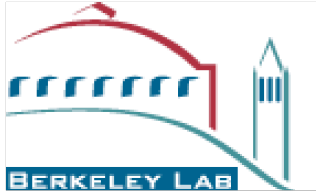
- **Uses same code as scp, so don't use sftp WAN transfers unless you have installed the HPN patch from PSC**
- **But even with the patch, SFTP has yet another flow control mechanism**
 - **By default, sftp limits the total number of outstanding messages to 16 32KB messages.**
 - **Since each datagram is a distinct message you end up with a 512KB outstanding data limit.**
 - **You can increase both the number of outstanding messages ('-R') and the size of the message ('-B') from the command line though.**
- **Sample command for a 128MB window:**
 - **sftp -R 512 -B 262144 user@host:/path/to/file outfile**



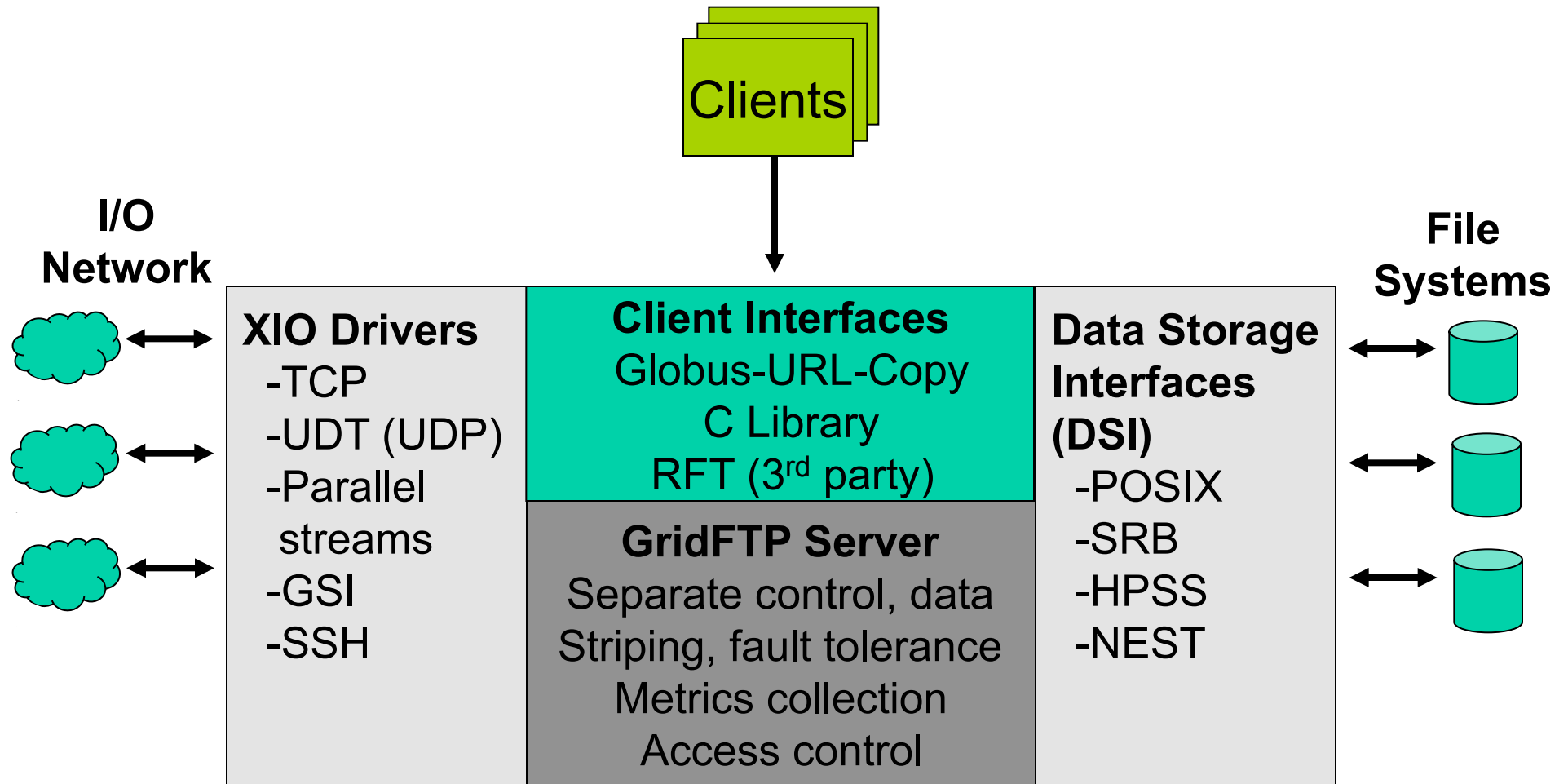
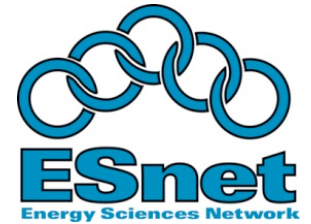
GridFTP

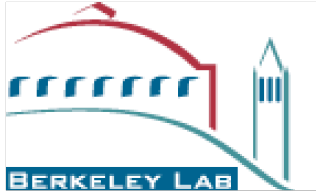


- **GridFTP from ANL has everything needed to fill the network pipe**
 - Buffer Tuning
 - Parallel Streams
- **Supports multiple authentication options**
 - anonymous
 - ssh (available in starting with Globus Toolkit version 4.2)
 - X509
- **Ability to define a range of data ports**
 - helpful to get through firewalls
- **Sample Use:**
 - `globus-url-copy -p 4`
`sshftp://data.lbl.gov/home/mydata/myfile`
`file://home/mydir/myfile`
- **Available from: <http://www.globus.org/toolkit/downloads/>**

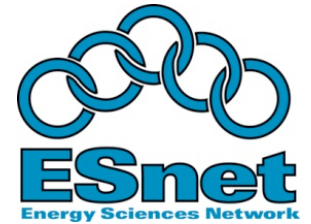


GridFTP

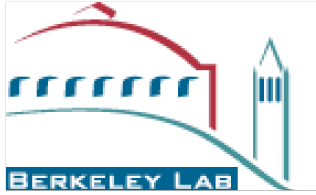




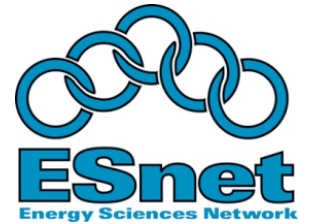
newer GridFTP Features



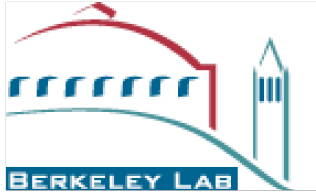
- **ssh authentication option**
 - Not all users need or want to deal with X.509 certificates
 - **Solution: Use SSH for Control Channel**
 - Data channel remains as is, so performance is the same
 - see <http://fasterdata.es.net/gridftp.html> for a quick start guide
- **Optimizations for small files**
 - **Concurrency option (-cc)**
 - establishes multiple control channel connections and transfer multiple files simultaneously.
 - **Pipelining option:**
 - Client sends next request before the current completes
 - **Cached Data channel connections**
 - Reuse established data channels (Mode E)
 - No additional TCP or GSI connect overhead
- **Support for UDT protocol**



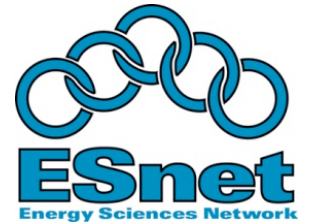
new GridFTP “bottleneck detector”



- **new command line option for globus-url-copy, “-nlb”**
 - nlb = NetLogger bottleneck
 - Uses NetLogger libraries for analysis of network and disk I/O
 - <http://acs.lbl.gov/NetLogger>
- **Possible “Bottleneck:” results are:**
 - network: somewhere in the network
 - disk read: sender's disk
 - disk write: receiver's disk
 - unknown: disk/network are about the same and/or highly variable



new GridFTP “bottleneck detector”



- **Sample Output:**

Total instantaneous throughput:

disk read = 1235.7 Mbits/s

disk write = 2773.0 Mbits/s

net read = 836.3 Mbits/s

net write = 1011.7 Mbits/s

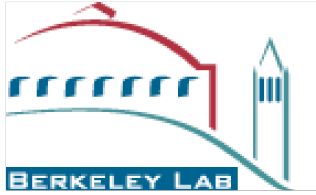
Bottleneck: network

- **Ignore the "net write" value**

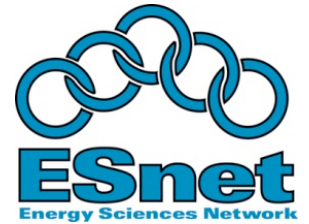
- the time to write to the network is strongly influenced by system and TCP buffer artifacts.

- ***instantaneous throughput* is the average # of bytes divided by the time spent blocking on the system call.**

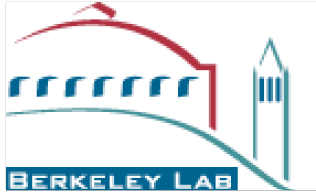
- ***instantaneous throughputs* are higher than the overall throughput of the transfer:**
 - does not include the time waiting for data to be available
 - primarily useful for comparison and not as absolute numbers



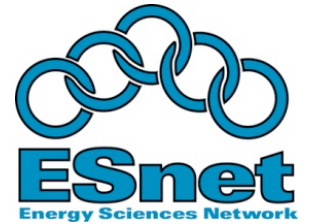
new GridFTP “bottleneck detector”



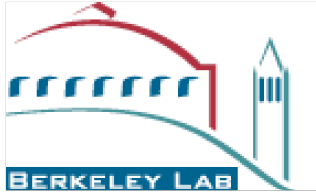
- **-nlb not enabled by default**
 - use `./configure --enable-netlogger`
 - additional server configuration flags needed
 - instructions at:
 - <http://www.cedps.net/index.php/Gridftp-netlogger>



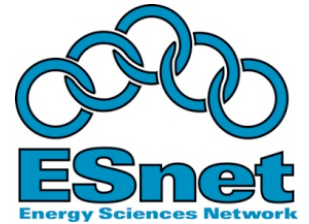
bbftp



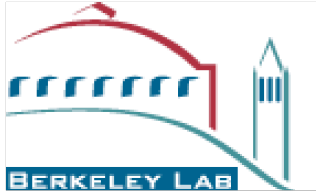
- **bbftp (from the HEP “Babar”) project also everything needed to fill the network pipe**
 - Buffer Tuning
 - Parallel Streams
- **Supports ssh authentication options**
- **Supports on-the-fly compression**
- **Sample Use:**
 - `bbftp -p 4`
`bbftp://data.lbl.gov/home/mydata/myfile`
`file://home/mydir/myfile`
- **Available from: <http://doc.in2p3.fr/bbftp/>**



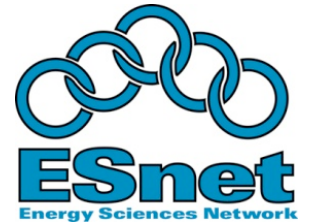
Other Tools



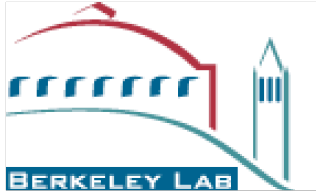
- **bbcp:** <http://www.slac.stanford.edu/~abh/bbcp/>
 - supports parallel transfers and socket tuning
 - `bbcp -P 4 -v -w 2M myfile remotehost:filename`
- **lftp:** <http://lftp.yar.ru/>
 - parallel file transfer, socket tuning, HTTP transfers, and more.
 - `lftp -e 'set net:socket-buffer 4000000; pget -n 4 [http|ftp]://site/path/file; quit'`
- **axel:** <http://wilmer.gaast.net/main.php/axel.html>
 - simple parallel accelerator for HTTP and FTP.
 - `axel -n 4 [http|ftp]://site/file`



Download Managers



- There are a number of nice browser plugins that can be used to speed up web-initiated data transfers
 - all support parallel transfers
- Firefox add-on (All OSes):
 - DownThemAll : <http://www.downthemall.net>
 - this is my favorite: probably the best/simplest solution
- For Linux:
 - aria: <http://aria-rpm.sourceforge.net>
- For Windows:
 - FDM: <http://www.freedownloadmanager.org>
 - Stardownloader: <http://www.stardownloader.com/>
- For OSX:
 - Speed Download: <http://www.yazsoft.com/> (\$25)



MS Windows Tool: Filezilla



- **Open Source:**
 - <http://filezilla-project.org/>
 - includes client and server
- **Features:**
 - ability to transfer multiple files in parallel

FileZilla

File Edit Transfer Server Help

Host: 127.0.0.1 Username: filezilla Password: Port: Quickconnect

Response: 226 Transfer OK
 Status: File transfer successful
 Status: Starting upload of C:\dev\svn\FileZilla3\src\bin\fszftp.exe
 Command: PORT 127,0,0,1,16,14
 Response: 200 Port command successful
 Command: STOR fszftp.exe
 Response: 150 Opening data channel for file transfer.

Local site: C:\dev\svn\ Remote site: /c:/ftproot/filezilla/FileZilla3/src/

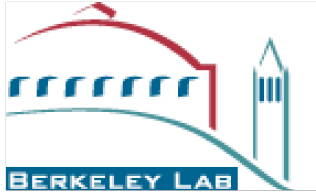
Filename	Filesize	Filetype	Last modified	Permissions	Owner / Gr
..					
FileZilla3		Dateiordner	01/09/2007 18:00:48		
Kopie von putty		Dateiordner	09/08/2007 16:45:25		
putty		Dateiordner	09/08/2007 17:41:58		
Seminar		Dateiordner	13/05/2007 19:00:55		
wxWidgets		Dateiordner	26/08/2007 10:14:11		
wxWidgets_trunk		Dateiordner	30/08/2007 00:20:17		
XRCed		Dateiordner	27/07/2007 21:22:11		
todo.txt	263	Textdokument	03/08/2007 14:17:32		

Filename	Filesize	Filetype	Last modified	Permissions	Owner / Gr
..					
.svn		Dateiordner			
bin		Dateiordner			
Makefile.am	160	AM-Datei	02/09/2007 15:11:00	-rw-r--r--	ftp ftp
Makefile.in	16201	IN-Datei	02/09/2007 15:11:00	-rw-r--r--	ftp ftp
FileZilla.sln	2351		11:00	-rw-r--r--	ftp ftp
FileZilla.suo	24064		11:00	-rw-r--r--	ftp ftp

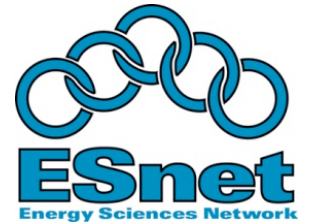
Server / Local file	Direction	Remote file	Size	Pri
filezilla@127.0.0.1				
C:\dev\svn\FileZilla3\src\bin\FileZilla_unicode_d...	-->	/c:/ftproot/filezilla/FileZilla3/src/bi...	15477996	No
00:00:23 elapsed 00:00:07 left				
C:\dev\svn\FileZilla3\src\bin\fszftp.exe	-->	/c:/ftproot/filezilla/FileZilla3/src/bi...	2359963	Normal
00:00:03 elapsed 00:00:01 left				
C:\dev\svn\FileZilla3\src\engine\asynchostreso...	-->	/c:/ftproot/filezilla/FileZilla3/src/e...	1195	Normal

Queued files (2302) Failed transfers (1) Successful transfers

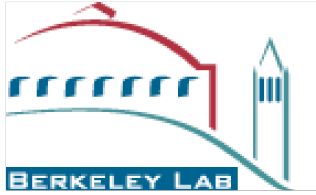
Queue: 664 MB



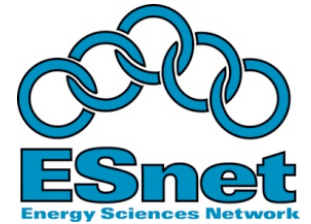
Special Purpose Data Transfer Tools



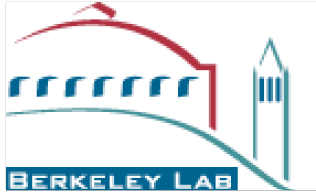
- **HPSS Tools: HSI and pftp**
 - both support buffer tuning and parallel transfers
 - per destination buffer tuning must be done by HPSS admin
- **Berkeley Storage Manager (BestMan) from LBNL**
 - supports buffer tuning and parallel transfers
 - <https://sdm.lbl.gov/bestman/>
- **Irods from UNC**
 - <https://www.irods.org/index>



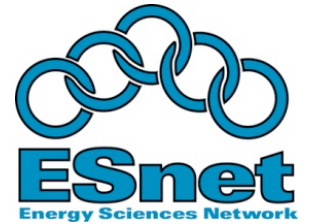
Other Bulk Data Transfer Issues



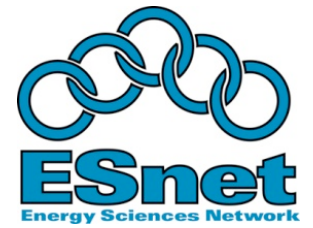
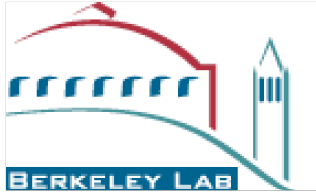
- **Firewalls**
 - **many firewalls can't handle 1 Gbps flows**
 - designed for large number of low bandwidth flow
 - some firewalls even strip out TCP options that allow for TCP buffers > 64 KB
- **Disk Performance**
 - In general need a RAID array to get more than around 500 Mbps
- **Other subtle issues start to come up at speeds above 500 Mbps**
 - Router buffering: <http://fasterdata.es.net/cisco.html>
 - TCP congestion control, disk tuning, etc.



Selecting a bulk data transfer tool



- **First, determine which security model you require**
 - anonymous: (e.g.: FTP, HTTP) anyone can access the data
 - simple password: (e.g.: FTP, HTTP) most sites no longer allow this method since the password can be easily captured
 - password encrypted: (e.g.: bbcp, bbftp, GridFTP) control channel is encrypted, but data is unencrypted
 - everything encrypted: (e.g.: scp, sftp, GridFTP, HTTPS-based web server) both control and data channels are encrypted
- **Most open science projects seem to prefer option #3.**
- **If you require option #4, tools that perform well over a WAN are limited to:**
 - GridFTP with X509 keys,
 - HPN-patched versions of scp/sftp.

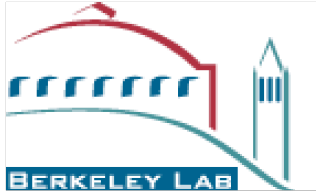


Network Monitoring Tools

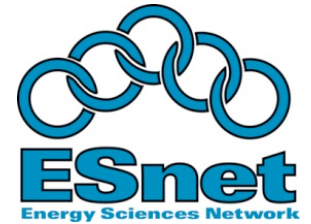


U.S. DEPARTMENT OF
ENERGY

Office of
Science



traceroute



```
>traceroute pcgiga.cern.ch
```

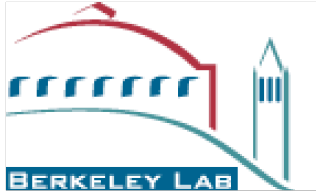
```
traceroute to pcgiga.cern.ch (192.91.245.29), 30 hops max, 40 byte packets
```

```
 1  ir100gw-r2.lbl.gov (131.243.2.1)  0.49 ms   0.26 ms   0.23 ms
 2  er100gw.lbl.gov (131.243.128.5)  0.68 ms   0.54 ms   0.54 ms
 3  198.129.224.5 (198.129.224.5)  1.00 ms   *d9*     1.29 ms
 4  lbl2-ge-lbnl.es.net (198.129.224.2)  0.47 ms   0.59 ms   0.53 ms
 5  snv-lbl-oc48.es.net (134.55.209.5)  57.88 ms  56.62 ms  61.33 ms
 6  chi-s-snv.es.net (134.55.205.102)  50.57 ms  49.96 ms  49.84 ms
 7  ar1-chicago-esnet.cern.ch (198.124.216.73)  50.74 ms  51.15 ms  50.96 ms
 8  cernh9-pos100.cern.ch (192.65.184.34)  175.63 ms 176.05 ms 176.05 ms
 9  cernh4.cern.ch (192.65.185.4)  175.92 ms 175.72 ms 176.09 ms
10  pcgiga.cern.ch (192.91.245.29)  175.58 ms 175.44 ms 175.96 ms
```

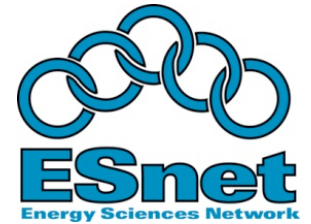
Can often learn about the network from the router names:

ge = Gigabit Ethernet

oc48 = 2.4 Gbps (oc3 = 155 Mbps, oc12=622 Mbps)



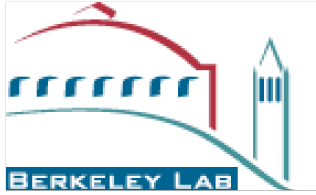
Iperf



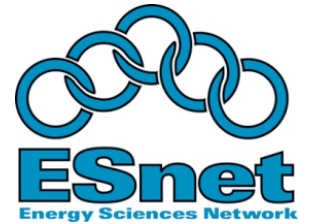
- **iperf** : nice tool for measuring end-to-end TCP/UDP performance
 - <http://sourceforge.net/projects/iperf>
 - Can be quite intrusive to the network
- **Example:**
 - **Server:** `iperf -s -w 2M`
 - **Client:** `iperf -c hostname -i 2 -t 20 -l 128K -w 2M`

Client connecting to hostname

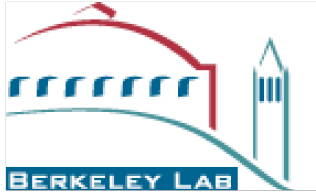
[ID]	Interval	Transfer	Bandwidth
[3]	0.0- 2.0 sec	66.0 MBytes	275 Mbits/sec
[3]	2.0- 4.0 sec	107 MBytes	451 Mbits/sec
[3]	4.0- 6.0 sec	106 MBytes	446 Mbits/sec
[3]	6.0- 8.0 sec	107 MBytes	443 Mbits/sec
[3]	8.0-10.0 sec	106 MBytes	447 Mbits/sec
[3]	10.0-12.0 sec	106 MBytes	446 Mbits/sec
[3]	12.0-14.0 sec	107 MBytes	450 Mbits/sec
[3]	14.0-16.0 sec	106 MBytes	445 Mbits/sec
[3]	16.0-24.3 sec	58.8 MBytes	59.1 Mbits/sec
[3]	0.0-24.6 sec	871 MBytes	297 Mbits/sec



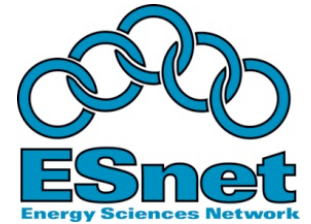
pathrate / pathload



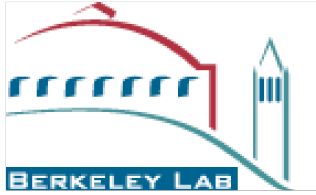
- **Nice tools from Georgia Tech:**
 - pathrate: measures the capacity of the narrow link
 - pathload: measures the available bandwidth
- **Both work pretty well.**
 - pathrate can take a long time (up to 20 minutes)
 - These tools attempt to be non-intrusive
- **Open Source; available from:**
 - <http://www.pathrate.org/>



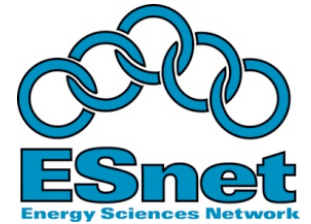
Duplex Mismatch Issues



- **A common source of trouble with Ethernet networks is that the host is set to full duplex, but the Ethernet switch is set to half-duplex, or visa versa.**
- **Most newer hardware will auto-negotiate this, but with some older hardware, auto-negotiation sometimes fails**
 - **result is a working but very slow network (typically only 1-2 Mbps)**
 - **best for both to be in full duplex if possible, but some older 100BT equipment only supports half-duplex**
- **NDT is a good tool for finding duplex issues:**
 - **<http://e2epi.internet2.edu/ndt/>**

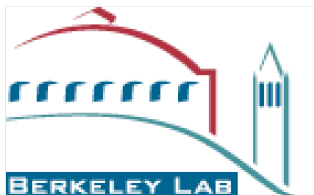


tcpdump / tcptrace



- **tcpdump: dump all TCP header information for a specified source/destination**
 - <http://www.tcpdump.org/>
- **tcptrace: format tcpdump output for analysis using xplot**
 - <http://www.tcptrace.org/>
- **Sample use:**

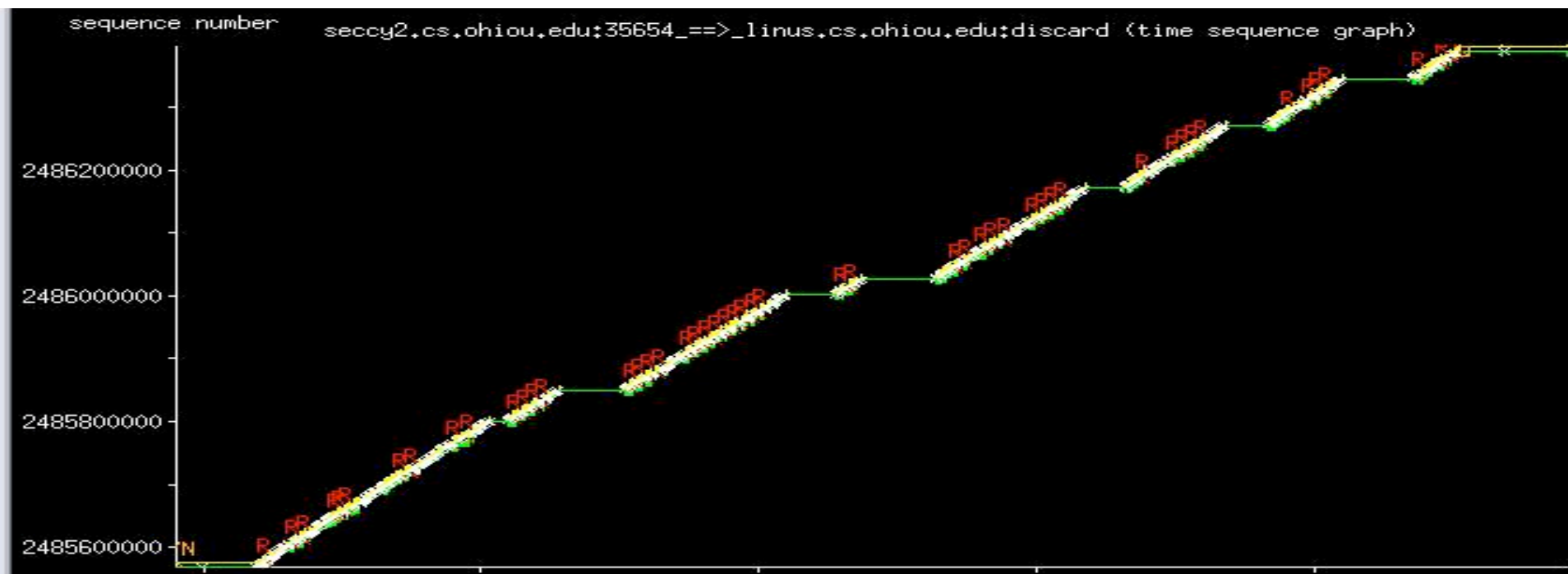
```
tcpdump -s 100 -w /tmp/tcpdump.out host hostname
tcptrace -S1 /tmp/tcpdump.out
xplot /tmp/a2b_tsg.xpl
```



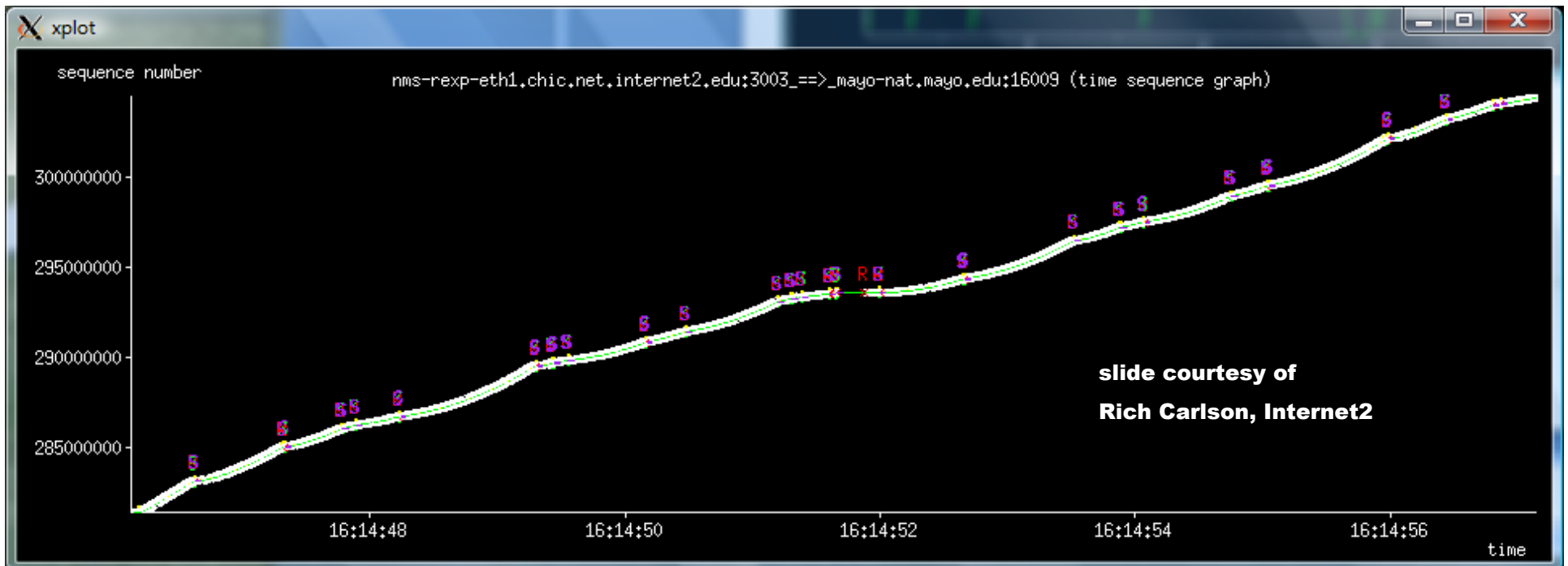
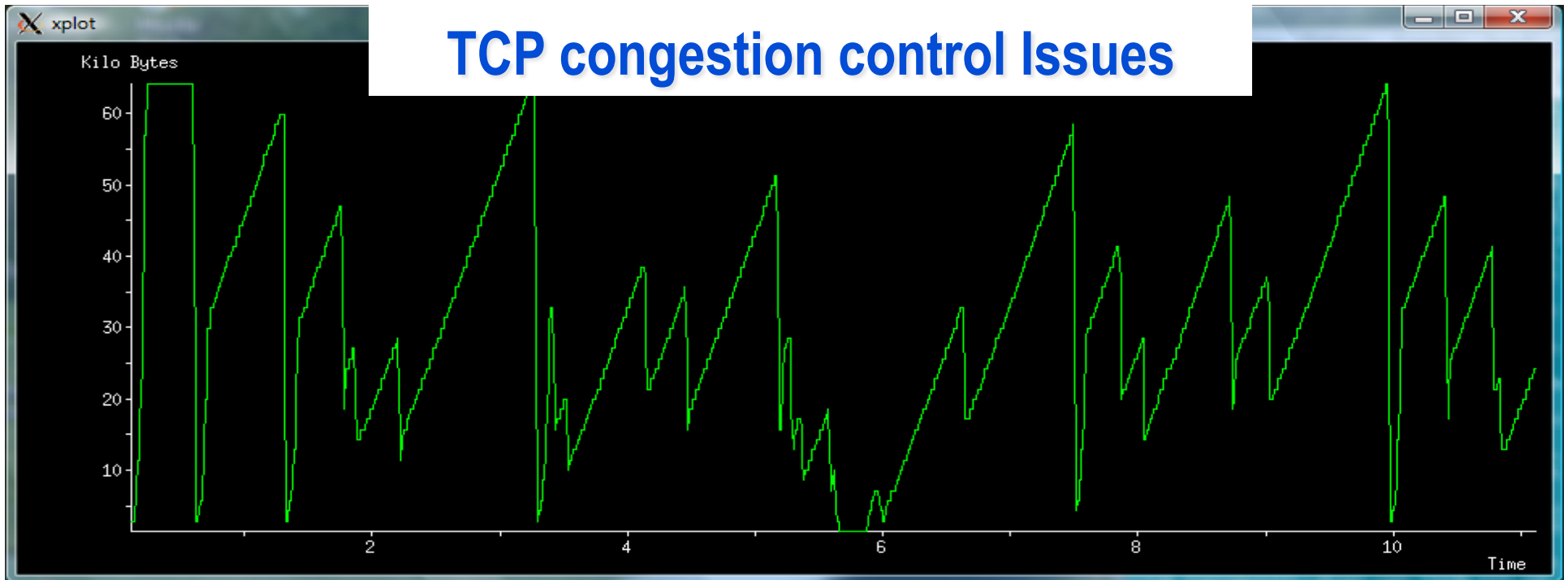
tcptrace and xplot

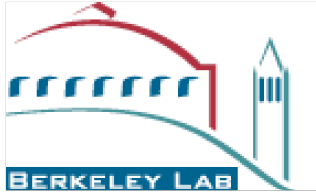


- X axis is time
- Y axis is sequence number
- the slope of this curve gives the throughput over time.
- xplot tool make it easy to zoom in

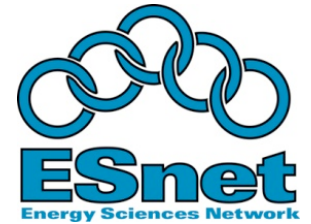


TCP congestion control Issues

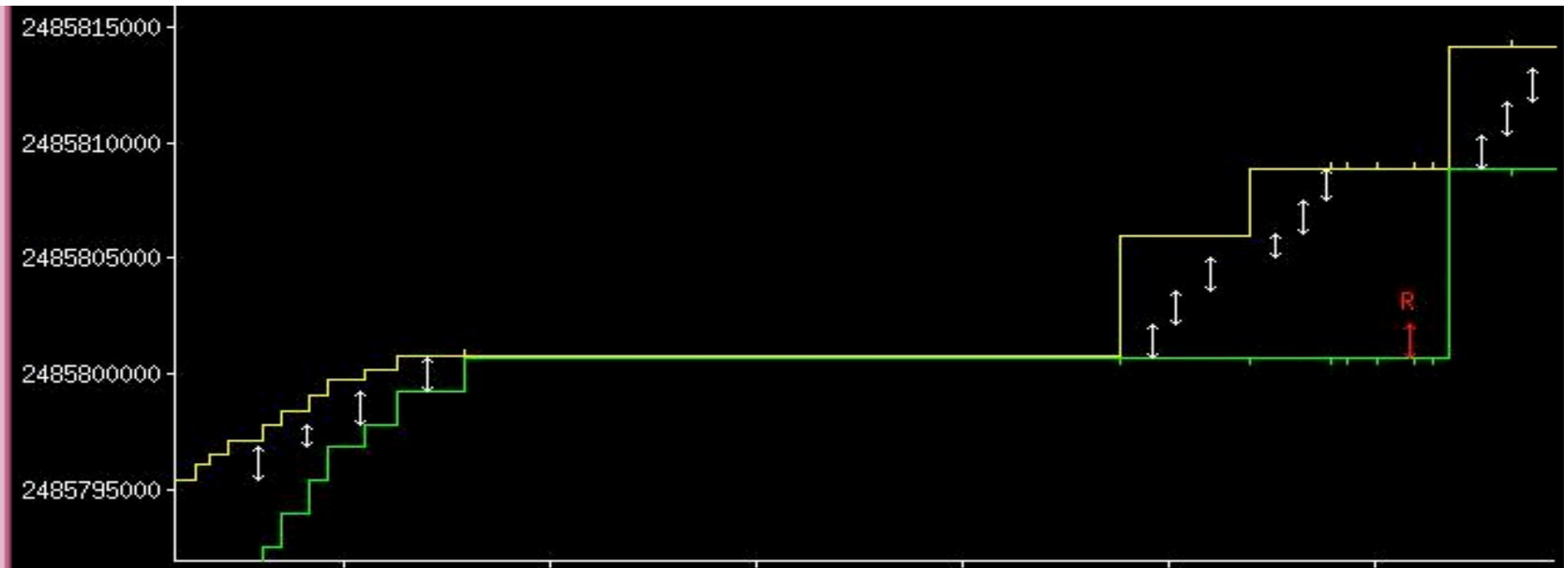


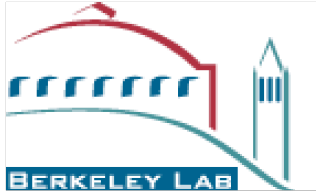


Zoomed In View

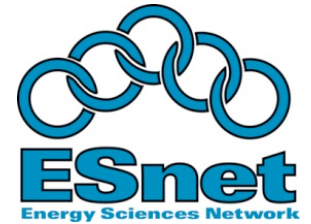


- Green Line: **ACK values received from the receiver**
- Yellow Line **tracks the receive window advertised from the receiver**
- Green Ticks **track the duplicate ACKs received.**
- Yellow Ticks **track the window advertisements that were the same as the last advertisement.**
- White Arrows **represent segments sent.**
- Red Arrows (R) **represent retransmitted segments**

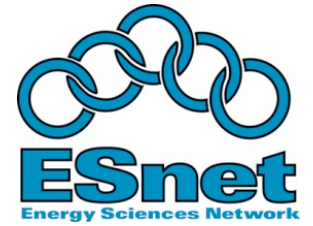
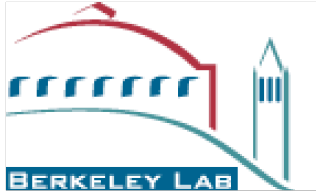




Jumbo Frames



- **Standard Ethernet packet is 1500 bytes (aka: MTU)**
- **Some gigabit Ethernet hardware supports “jumbo frames” (jumbo packet) up to 9 KBytes**
 - **Helps performance by reducing the number of host interrupts**
 - **Esnet, Internet2, and GEANT are all 9KB “jumbo-clean”**
 - But many sites have not yet enabled Jumbo Frames
 - Some jumbo frame implementations do not interoperate
- **First Ethernet was 3 Mbps (1972)**
- **First 10 Gbit/sec Ethernet hardware: 2001**
 - **Ethernet speeds increased 3000x since the 1500 byte frame was defined**
 - **Computers now have to work 3000x harder to fill the network**
- **To see if your end-to-end path is jumbo clean:**
 - **ping -M do -s 8972 192.0.2.254 (FreeBSD)**
 - **ping -D -s 8972 192.0.2.254**
 - **header math: 20 bytes IP + 8 bytes ICMP + 8972 bytes payload = 9000**

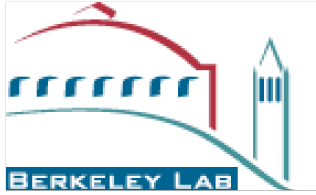


Recent TCP Work

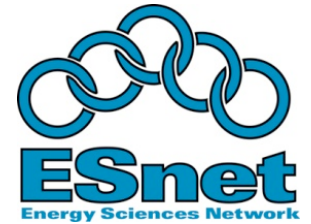


U.S. DEPARTMENT OF
ENERGY

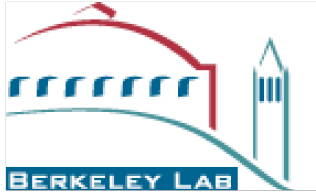
Office of
Science



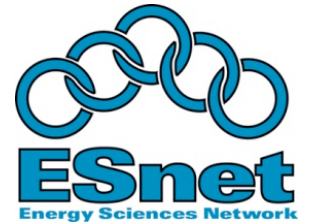
TCP Response Function



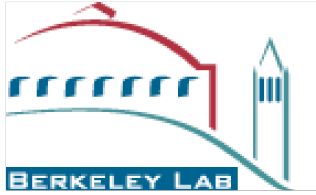
- Well known fact that TCP reno does not scale to high-speed networks
- Average TCP congestion window = $1.2/\sqrt{p}$ segments
 - p = packet loss rate
 - see: <http://www.icir.org/floyd/hstcp.html>
- What this means:
 - For a TCP connection with 1500-byte packets and a 100 ms round-trip time
 - filling a 10 Gbps pipe would require a congestion window of 83,333 packets,
 - a packet drop rate of at most one drop every 5,000,000,000 packets.
 - requires at most one packet loss every 6000s, or 1h:40m to keep the pipe full



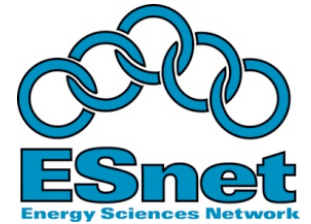
Proposed TCP Modifications



- **Many proposed alternate congestion control algorithms:**
 - **BIC/CUBIC**
 - **HTCP: (Hamilton TCP)**
 - **Scalable TCP**
 - **Compound TCP**
 - **And several more**



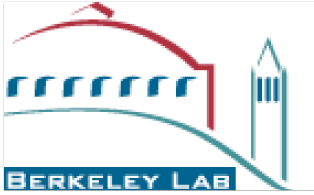
Linux 2.6.12 Results



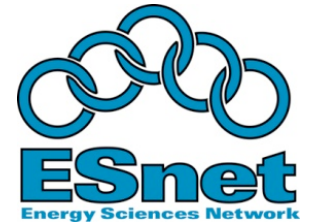
- BIC-TCP is included in Linux 2.6
 - results up to 100x faster than un-tuned!

Path	Linux 2.4 Un-tuned	Linux 2.4 Hand-tuned	Linux 2.6 with BIC	Linux 2.6, no BIC
LBL to ORNL RTT = 67 ms	10 Mbps	300 Mbps	700 Mbps	500 Mbps
LBL to PSC RTT = 83 ms	8 Mbps	300 Mbps	830 Mbps	625 Mbps
LBL to IE (Ireland) RTT = 153 ms	4 Mbps	70 Mbps	560 Mbps	140 Mbps

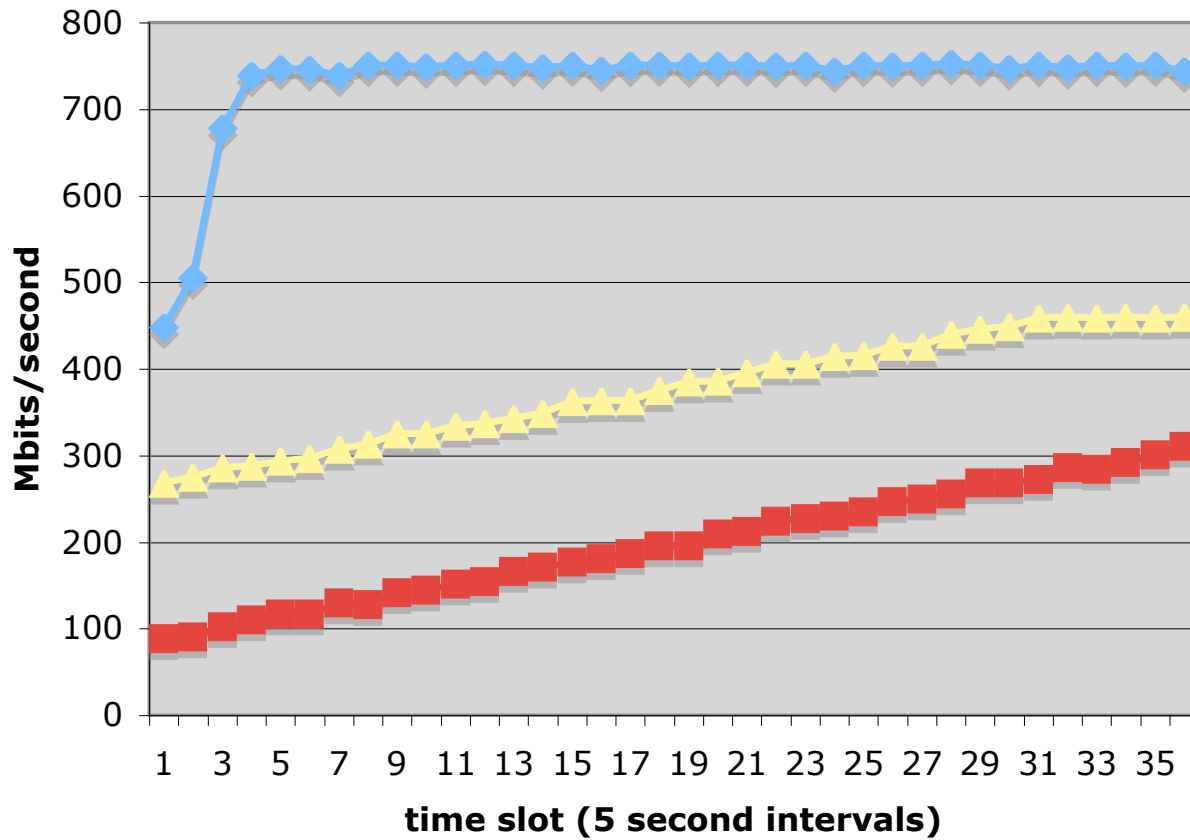
- Results = Peak Speed during 3 minute test
- Must increase default max TCP send/receive buffers
- Sending host = 2.8 GHz Intel Xeon with Intel e1000 NIC



Linux 2.6.12 Results



TCP Results



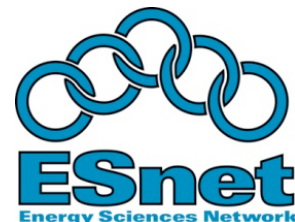
Note that BIC reaches Max throughput MUCH faster

- Linux 2.6, BIC TCP
- Linux 2.4
- Linux 2.6, BIC off

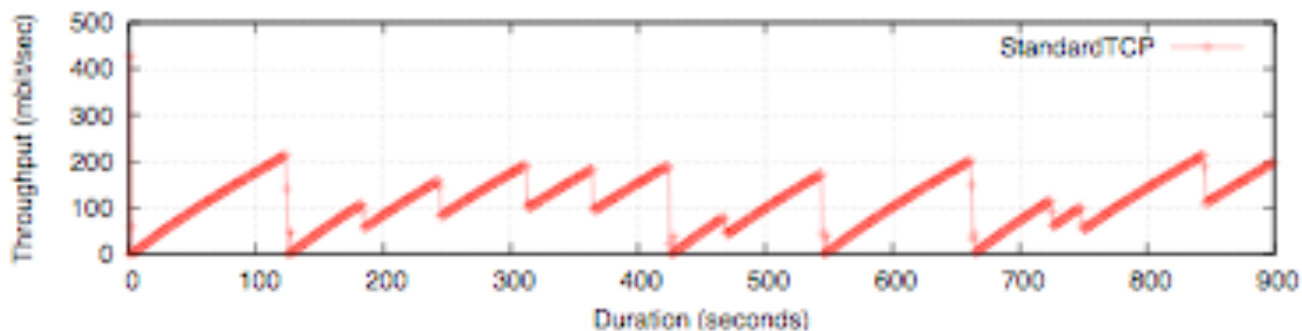
RTT = 67 ms



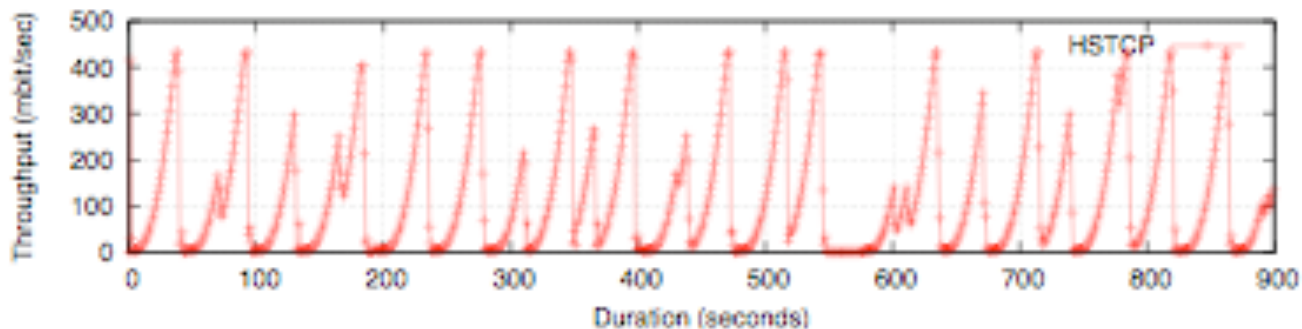
Comparison of Various TCP Congestion Control Algorithms



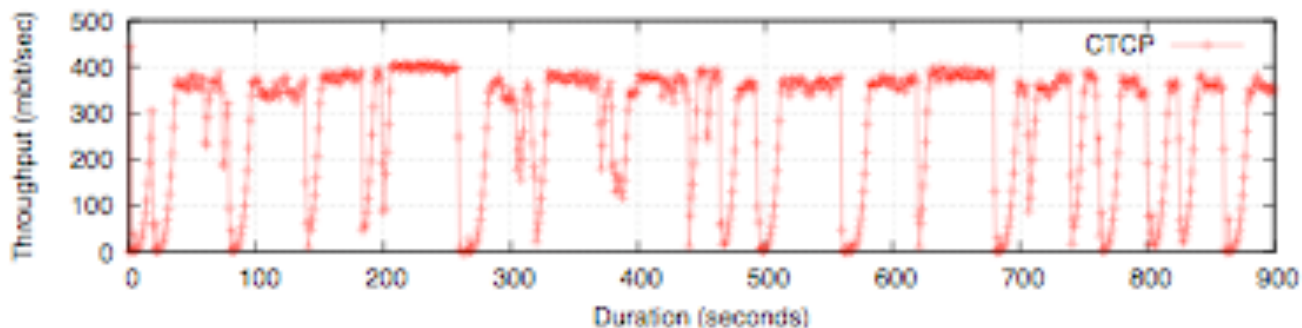
StandardTCP from SLAC to Florida

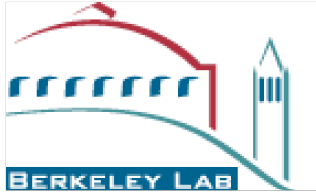


HSTCP from SLAC to Florida

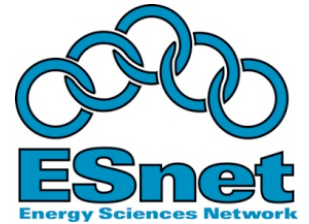


CTCP from SLAC to Florida





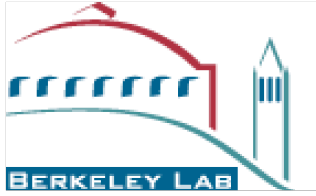
Selecting TCP Congestion Control in Linux



- To determine current configuration:

```
sysctl -a | grep congestion
```

```
net.ipv4.tcp_congestion_control = cubic  
net.ipv4.tcp_available_congestion_control = cubic  
reno
```
- Use `/etc/sysctl.conf` to set to any available congested congestion control.
- Supported options (may need to be enabled by default in your kernel):
 - CUBIC, BIC, HTCP, HSTCP, STCP, LTCP, more..

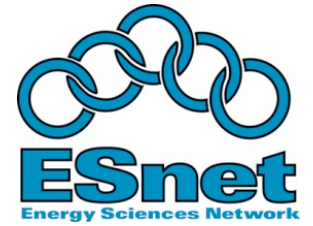
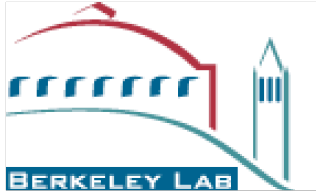


Selecting TCP Congestion Control in Windows Vista



- Vista includes "Compound TCP (CTCP)", which is similar to cubic on Linux.
- To enable this, set the following:

```
netsh interface tcp set global congestionprovider=ctcp"
```

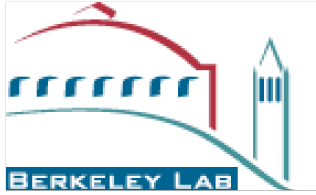


Application Performance Issues

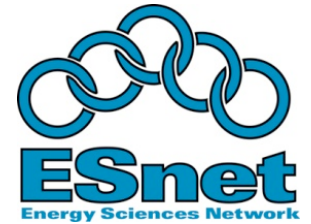


U.S. DEPARTMENT OF
ENERGY

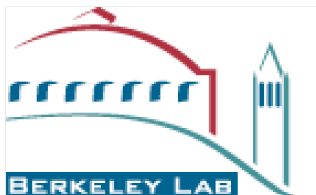
Office of
Science



Techniques to Achieve High Throughput over a WAN



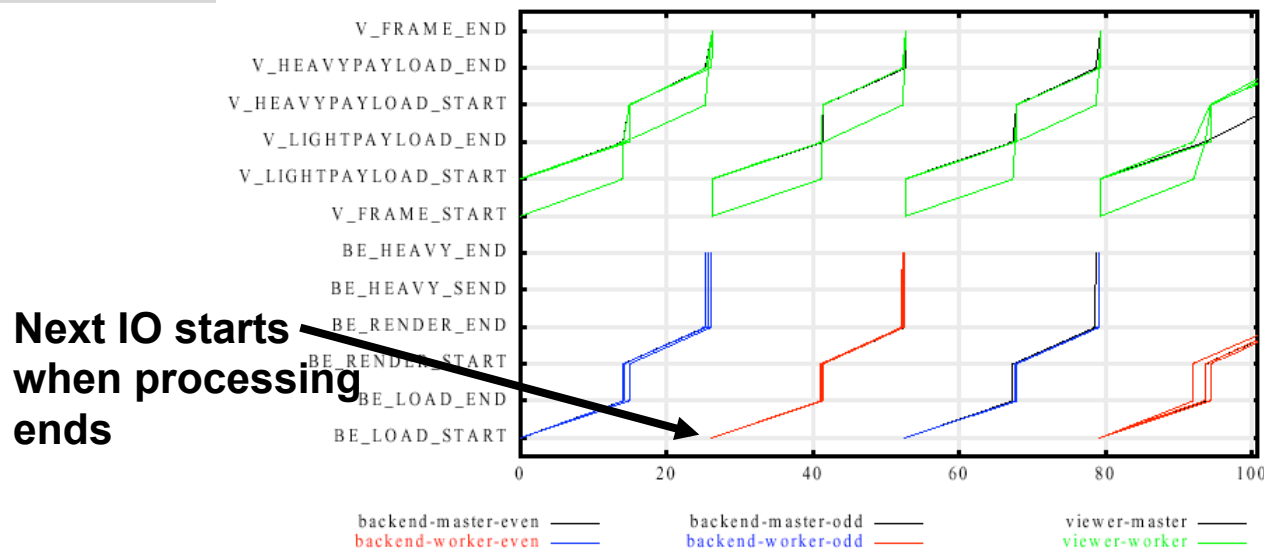
- **Consider using multiple TCP sockets for the data stream**
- **Use a separate thread for each socket**
- **Keep the data pipeline full**
 - use asynchronous I/O
 - overlap I/O and computation
 - read and write large amounts of data (> 1MB) at a time whenever possible
 - pre-fetch data whenever possible
- **Avoid unnecessary data copies**
 - manipulate pointers to data blocks instead



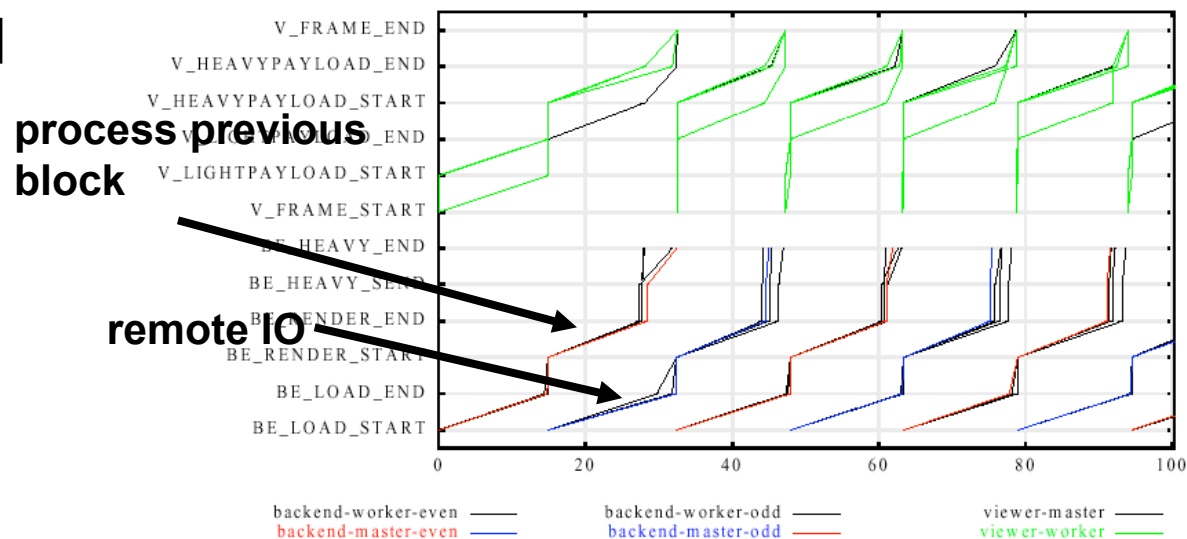
Use Asynchronous I/O



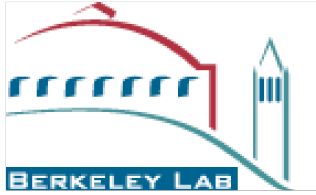
- I/O followed by processing



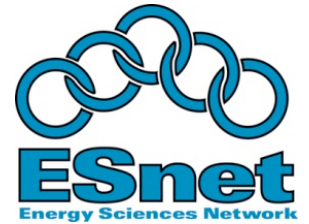
- overlapped I/O and processing



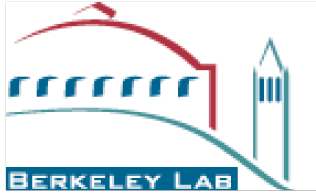
almost a 2:1 speedup



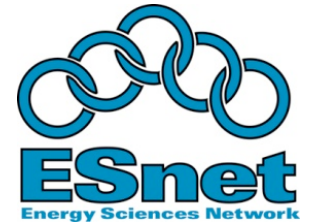
Conclusions



- **The wizard gap is starting to close (slowly)**
 - If max TCP autotuning buffers are increased
- **Tuning TCP is not easy!**
 - **no single solution fits all situations**
 - need to be careful to set TCP buffers properly
 - sometimes parallel streams help throughput, sometimes they hurt
 - **Autotuning helps a lot**
- **Lots of options for bulk data tools**
 - **Choose the one that fills your requirements**
 - **Don't use unpatched scp!**



More Information



- <http://fasterdata.es.net/>
- email: BLTierney@es.net