TCP, Bandwidth Delay Product, and Web100 Net100 Projects

Peter O Neil
National Center for Atmospheric Research
April 25, 2002

NCAR

Education is not filling a bucket, but lighting a fire.

William Butler Yeats

NCAR

What is the Internet?

- ¥ The largest network of networks in the world
- ¥ Uses IP protocols and packet switching
- ¥ Runs on any communications substrate

NCAR

Topical List

- A Few Technology Trends/Problems
- TCP Protocol Overview
- Bandwidth Delay Product
- Impacts on HPNC
- Web100 problem space
- Net100 Problem space

NCAR

Technology Trends

- CPU performance increases 60% per year (Moore s Law)
- Optical Bandwidth Performance increases 300% per year
- Software algorithms improve 5-10% per year
- Large systems double the number of CPUs every 4 years
- Memory performance increases 10% per year

NCAR

Conclusions from Tech Trends

- Memory bandwidth remains the critical bottleneck for many years to come
  - New technology/ algorithms must evolve to conserve memory bandwidth
  - Applications must be able to adapt to take advantage of these technologies/algorithms
- Optical Bandwidth has now surpassed Server I/O Subsystem capabilities
  - I/O Subsystems must evolve to meet higher demands, new customer requirements

NCAR

Problems Posed by High Bandwidth Links

- Starting to see Internet have very high capacity links
- Regardless of congestion or queuing scheme used, as bandwidth delay product (BDP) increases, TCP becomes more oscillatory and prone to instabilities
- As BDP increases, throughput performance degrades
- Frustration with large BDP, yet to be felt
  - —1 Gb/s WAN * 100ms RTT = 100Mb/s throughput at best
- Static 1500B MTU doesn t help either

NCAR

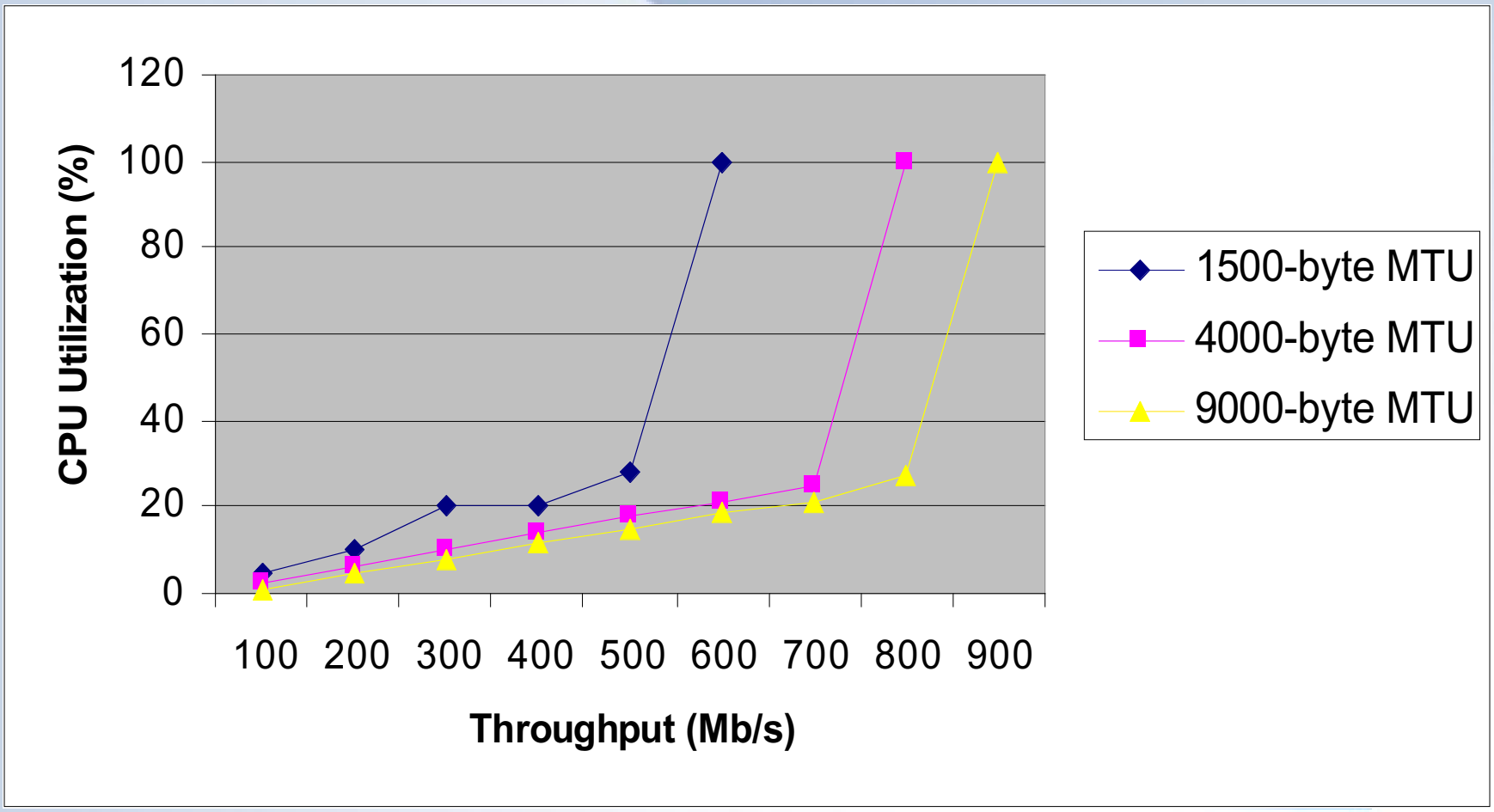
Path MTU

- Maximum Transfer Unit (MTU)
  - —Largest framing size of packets
  - —Various values
    - Minimum 536 Bytes
    - Ethernet 1500 Bytes (default)
    - FDDI 4352 Bytes
    - ATM 9180 Bytes
    - HIPPI 65280 Bytes
- At GigE, 1500 Byte packets order of magnitude too small
- At 10GigE, 1500 Byte packets 2-3 orders of magnitude too small

NCAR

666-MHz Alpha with Linux

(Courtesy: USC/ISI)



Even jumbograms suffer from high CPU utilization

NCAR

High ***Capacity*** Networks › High ***Speed*** Networks

NCAR

TCP Protocol Design

- Intended to operate reliably over almost any transmission medium
- Works at various transmission rates, delays, corruption, duplication, out-of-order segments
- Fiber optic transmission rates hitting limits of TCP
- How fast can TCP go?
  - —depends on balancing performance *and* reliability factors

NCAR

Network Information Flow Overview

- Senders maintain cwnd and RTT and communicate these to routers via congestion header in each packet
- Depending upon the difference between link bandwidth and input traffic rate, router tells flow on path link to increase or decrease their cwnds
- Flows converge to fairness based on cwnd and RTT values
- Routers along path continue this until bottleneck found
- When info reaches receiver, feedback goes back to sender via ACK s resulting in sender retuning cwnd

NCAR

TCP Window Size

- Perhaps, most important tuning factor
- Sender can only send more data after an acknowledgement received
- In theory, ideal window size keeps the pipe full — the Bandwidth Delay Product

NCAR

TCP Performance

- Dependent not upon bandwidth transfer rate itself
  - —product of transfer rate and round-trip time (RTT)
- Bandwidth*delay product
  - —measures amount of data that will fill the pipe
  - —buffer space at sender and receiver to gain maximum throughput on the TCP connection over the path
  - —amount of unacknowledged data TCP must handle to keep pipe full
  - —Problems arise when bandwidth*delay product is large

NCAR

Recovery from Losses

- ¥ Fast Retransmit and Fast Recovery algorithms help recovery of one packet loss per window without draining the pipeline
- ¥ More than one packet loss per window results in a retransmission timeout forcing pipeline drain and slow start algorithm to begin again
- ¥ Expanding window size to match pipe capacity increases probability of packet drops
- ¥ If congestion control uses random drops at routers, randomly dropped packets increases probability of dropping more than one packet per window
- ¥ SACK gives sender view of which segments queued at receiver and which in flight

NCAR

Round Trip Measurement

- TCP provides reliable data delivery by retransmitting segments that are not ack d within some RTO interval
- Determined by estimating mean and variance of RTT
  - —Time between the sending and receiving of ack

NCAR

TCP Reliability

- High transfer rate alone can threaten reliability by violating TCP design assumptions for duplicate packet detection and sequence number use
- TCP reliability depends upon existence of a bound on the lifetime of a segment (maximum segment lifetime) enforced by TTL field at IP layer

NCAR

Bandwidth Delay Product

- Bandwidth * Delay = number of bytes in flight to fill the path
- TCP needs rwin ‡ BW*Delay product to achieve maximum throughput
- TCP often needs sender size buffers of 2*BW*Delay to recover from errors
- You need to send about 3*BW*Delay bytes for TCP to achieve maximum speed (without loss or congestion)

NCAR

TCP Throughput Issues

- Bytes/sec † **available** bandwidth along path
- Bytes/sec † rwin size / RTT
  - 8kB window, 87 msec RTT = 750kb/s
  - 64kB window, 14 msec RTT = 37Mb/s
- Doubling MTU, doubles your throughput
- Halving RTT latency, doubles throughput
- Decreasing packet loss rate, improves throughput
- Look for sources of loss and path slowdown

NCAR

Things to Keep in Mind About TCP

- TCP is an adaptive protocol
- Tries to keep going faster until it hits bump in path
- Always slows down (dramatically) when a loss in the flow stream is detected (AIMD)
- How much TCP sends at any one time is a function of the size of send and receive buffers
- Sends, controlled by timing of received ACK s
  - —Ack packets must also compete for bandwidth, timing delayed, and they may be dropped from router queues

NCAR

TCP Congestion Window (cwnd)

- ¥ cwnd controls the startup and limits the throughput in the face of packet loss
- ¥ cwnd gets larger after every new ACK
- ¥ cwnd gets smaller when loss is detected
- ¥ Usable window = min (rwin, cwnd)
- ¥ cwnd increased by one for every new ACK
- ¥ cwnd doubles every round trip time
- ¥ cwnd resets to zero after a loss and slow starts again

TCP Algorithms

- Various TCP algorithms incorporate different congestion control mechanisms and implementations may or may not use features such as SACK or window scaling.

NCAR

TCP Reno

- Reno RFC defined four central mechanisms
  - —Slow Start
  - —Congestion Avoidance
  - —Fast Retransmit
  - —Fast Recovery
- Uses Additive Increase Multiplicative Decrease (AIMD) for congestion control
  - —Overly aggressive; probes network by inducing packet loss
  - —Will use up any available buffer space
- Most widely deployed stack but AIMD induces chaotic, oscillatory behavior in the network

NCAR

TCP Vegas

- Extends TCP Reno by trying to avoid rather than reaching and reacting to congestion
- When cwnd increases in size, the expected sending rate (ESR) increases
- If actual sending rate stays roughly same, then not enough bandwidth to send at ESR
  - —Increasing cwnd will only fill buffer space in network
- Vegas detects this and avoids congestion by adjusting cwnd and ESR to available bandwidth

NCAR

Vegas Incremental Deployment Ok

- Number of comparative studies done at ISI, LBL, and LANL using Network Simulator (NS) with Reno and Vegas
  - http://www.isi.edu/nsnam/ns/
  - http://public.lanl.gov/radiant/website/pubs.html
  - http://www.csm.ornl.gov/~dunigan/net100/netlinks.html
- Vegas may provide better throughput performance due to lower loss rates as compared to Reno

NCAR

Delayed ACKs

- TCP receivers send ACKs
  - —After every second MSS received
  - —After a delayed ACK timeout
  - —On every segment after a loss (missing segment)
- New segment sets the ACK timer (0-200msec)
- Second segment (or timeout) triggers an ACK and zeros the delayed ACK timer

NCAR

Detecting Loss

- Packets are discarded when queues become full or nearly full depending on congestion control algorithm used
- Duplicate ACKs get sent after missing or out of order packets received
- Most TCP s retransmit after third duplicate ACK

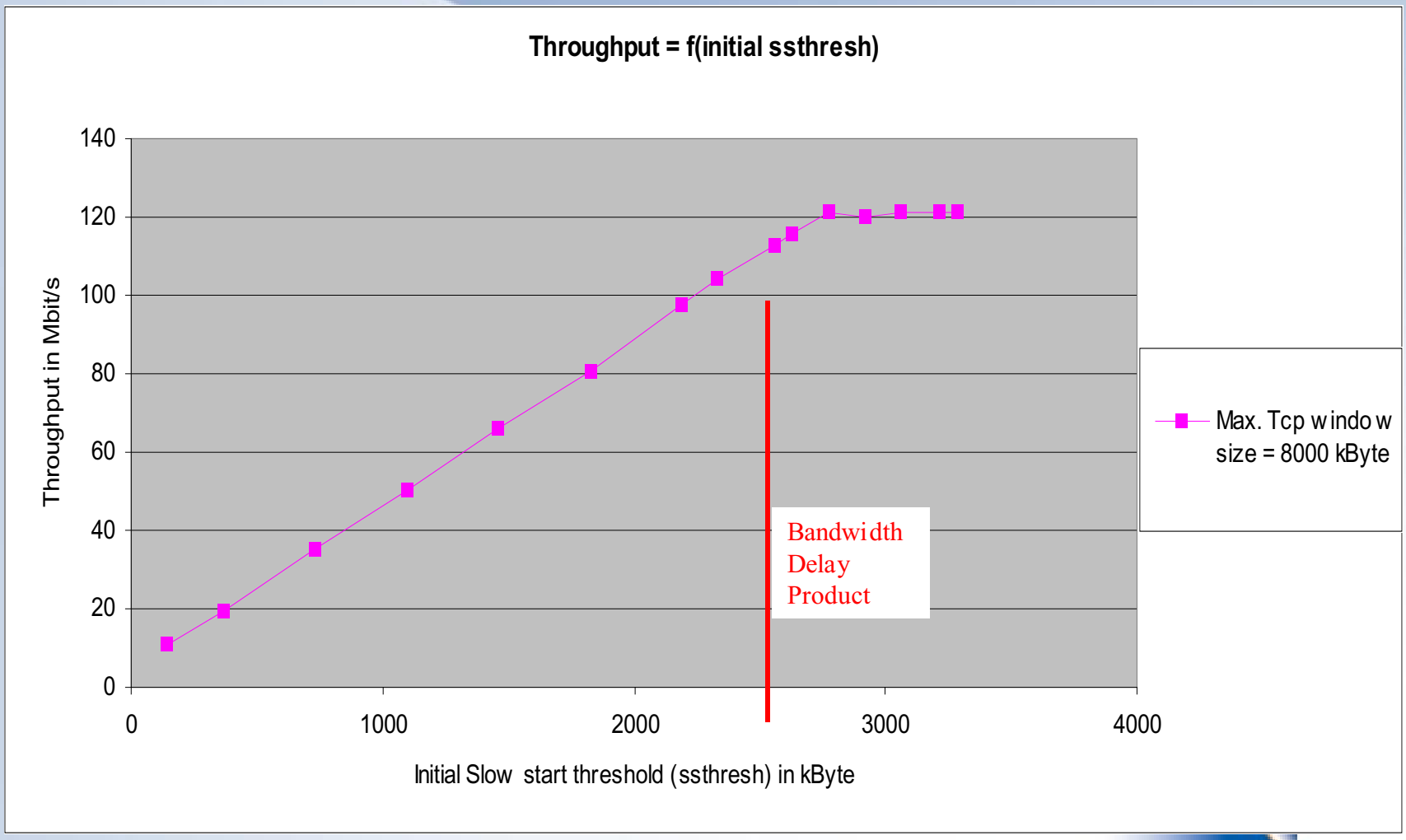
NCAR

Selective Acknowledgement (SACK)

- Specifies to sender exactly which Bytes were missing
- Better measures the right edge of cwnd
- Very good at keeping queues full (more efficient use of available bandwidth)
- However, will increase latencies
- Most OS s come with SACK including Linux and windoze

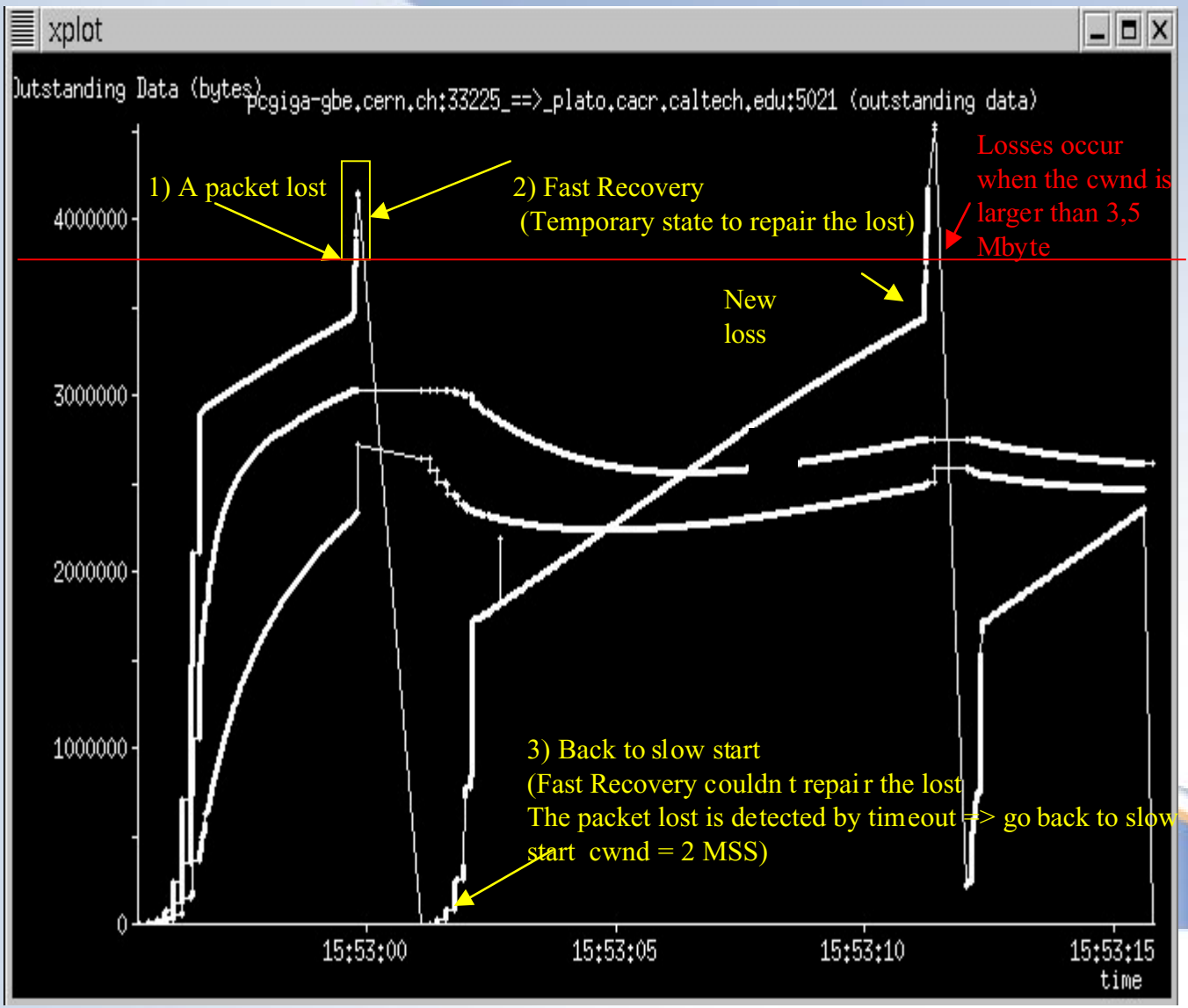
NCAR

Influence of SSTTHRESH on TCP Performance



NCAR

Cwnd too large for Available Bandwidth



NCAR

Web100

NSF Project — NCSA, PSC, NCAR

www.web100.org

NCAR

Motivations: What s the Problem?

- High performance flows slower than line rate
  - —Delays continue/increase even with higher bandwidth
- TCP tuning issues are non-trivial
- Poorly conceived stacks
- Router/switch buffer queues inadequate
- Slow start and AIMD algorithm
- Eliminate/dramatically reduce the wizard g ap
- Need for kernel instrumentation set for TCP variables

NCAR

The Wizard Gap

TCP over a long haul path

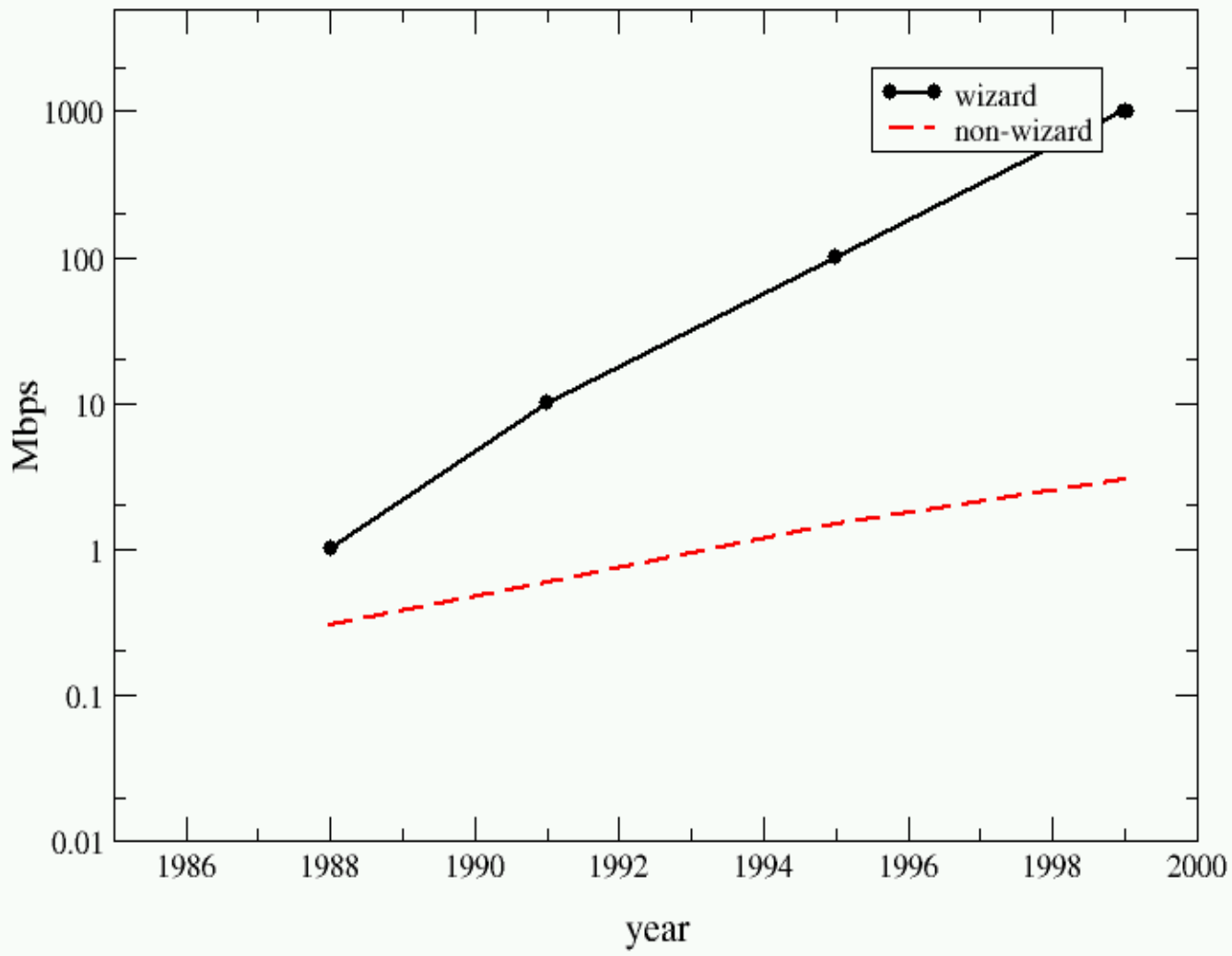
Year	Wizards	Non-wizards	Ratio
¥	1Mb/s	300kb/s	3:1
¥	10Mb/s		
1995	100Mb/s		
¥	1Gb/s	3Mb/s	300:1

Scientists/researchers not happy with this

NCAR

The Wizard Gap

(ratio has gone from 3:1 to 300:1 in last decade)



TCP tuning is painful debugging

- All problems limit performance
  - IP routing, long round trip times
  - Improper MSS negotiations or path MTU discovery
  - IP Packet reordering
  - Packet losses, congestion, lame hardware
  - TCP sender or receive buffer space
  - Inefficient applications
- Any one problem can mask all the others and confound all but the best (and few) tuning gurus
- Need for better diagnostics and visibility into problems

NCAR

Goal and Method

- Make it easy (transparent) for non-experts to achieve higher throughput performance
- Enhance TCP capabilities with better (finer grain) kernel instrumentation and automatic controls
- Real time triage capability determines sender, receiver, and/or network bottlenecks

NCAR

Why Focus on TCP

- TCP has an ideal vantage point into throughput problem space
- TCP can identify bottleneck subsystem(s)
- TCP already measures the network (some)
- TCP can measure the application
- TCP can adjust itself (auto-tuning feedback)

NCAR

Status

- Over a year of ~ 30 alpha testers from SLAC, ORNL, LBNL, and universities
  - —www.net100.org
- Modified Linux kernel supports 2.4.16
- Separation between KIS and library functions
- draft-ietf-tsvwg-tcp-mib-extension-00.txt
- draft-ietf-ipngwg-rfc2012-update-01.txt

NCAR

Looking Ahead

- New pathprobe diagnostic tool (wip, unreleased)
- Add another 10-12 instruments
- Include ideas from Wu Feng s Dynamic Right Sizing
- Review instruments and code with other wizards
- Gain vendor support for ideas and code
- Finalize IETF draft by fall meeting

NCAR

Summary

- Freely available software distribution
  - www.web100.org/download
  - hundreds of downloads
- Please be cognizant of impacts on others
- Please use, test, provide feedback, contribute code
- IETF standards process to benefit all
- Attention turning to working with OS vendors to incorporate standards enhancements into their stacks

NCAR

NET100

Development of network-aware operating systems

DOE Funded Project — PSC, ORNL, LBL, NCAR

www.net100.org

NCAR

Net100 project

- DOE-funded (Office of Science) project ($1M/yr, 3 yrs)
- Principal investigators
  - Wendy Huntoon and the PSC/Web100 team (Janet Brown, Matt Mathis)
  - Brian Tierney, LBNL
  - Tom Dunigan, ORNL
  - collaborators: Marla Meehl, Peter O Ne il, Bill Wing, Nageswara Rao
- Objective: develop network aware operating systems
  - optimize and understand end-to-end network and application performance
  - eliminate the w izard gap
  - Web100 to Web1000?
- Motivation
  - DOE has a large investment in high speed networks (ESnet) and distributed applications

NCAR

Net100 approach

- Deploy/enhance Web100 into DOE network applications
  - auto-tune network applications to optimize performance
  - collect performance statistics to understand/tune networks and applications
  - evaluate network applications over DOE s ESnet (OC12, OC48?)
    - bulk transfers over high bandwidth/delay network
    - distributed applications (grid)
- Develop Network Tools Analysis Framework (NTAF)
  - configure/launch network tools (*pathrate, pipechar*,)
  - aggregate and transform output from tools and Web100
- Develop Network Analysis Information Base (NAIB)
  - repository for NTAF data
  - API to collect and query

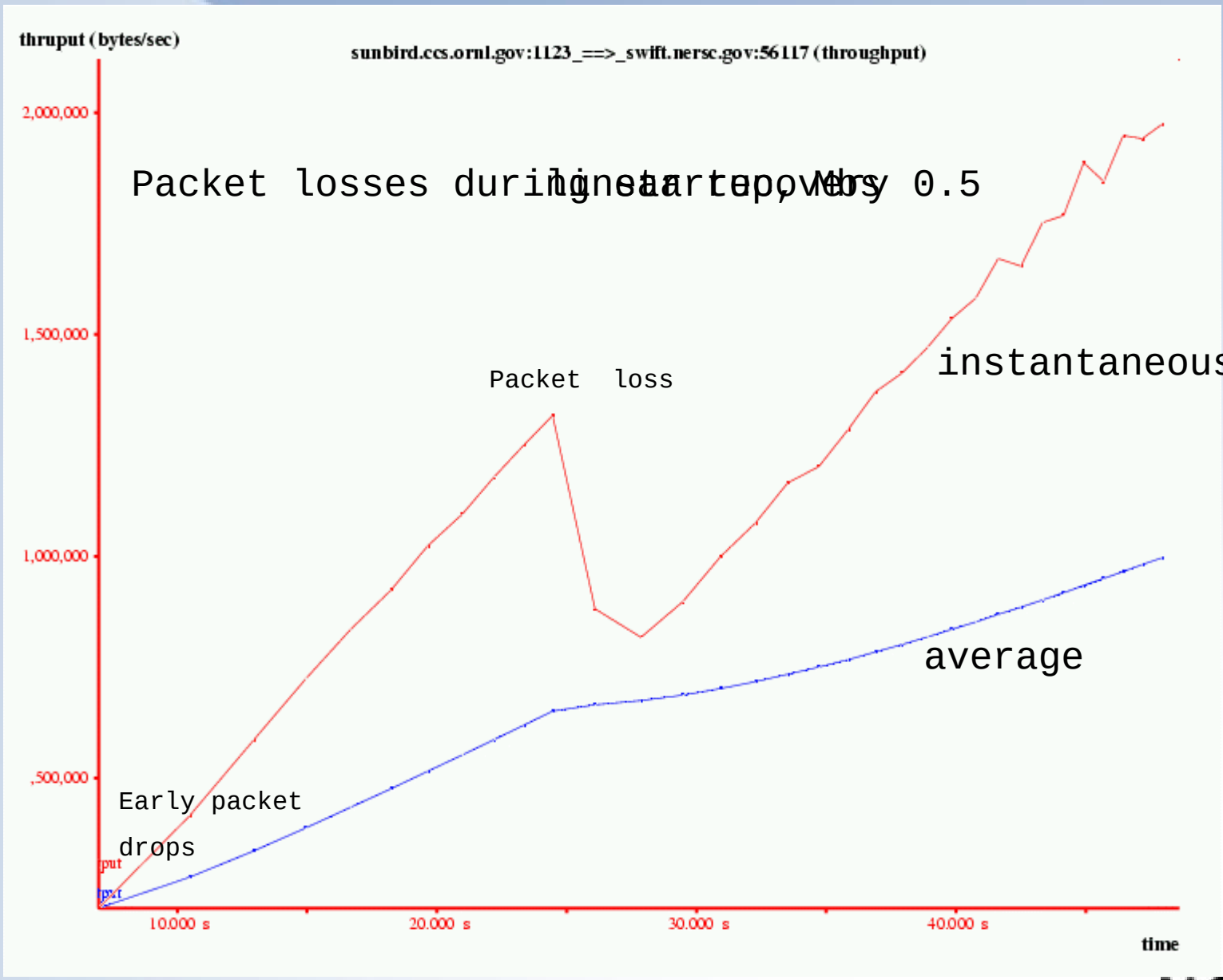
NCAR

Bulk transfers

- ORNL/NERSC Probe project
  - —wide-area distributed storage testbed (HPSS)
  - —investigate protocols, software, devices
- climate model data transfers were slow
  - — OC3 with 60 ms RTT
  - —classic TCP tuning problem
  - — also broken TCP stacks
  - — developed (almost) TCP-over-UDP test harness
    - instrumented and tunable
- Recent upgrade to OC12, 100 ms RTT

NCAR

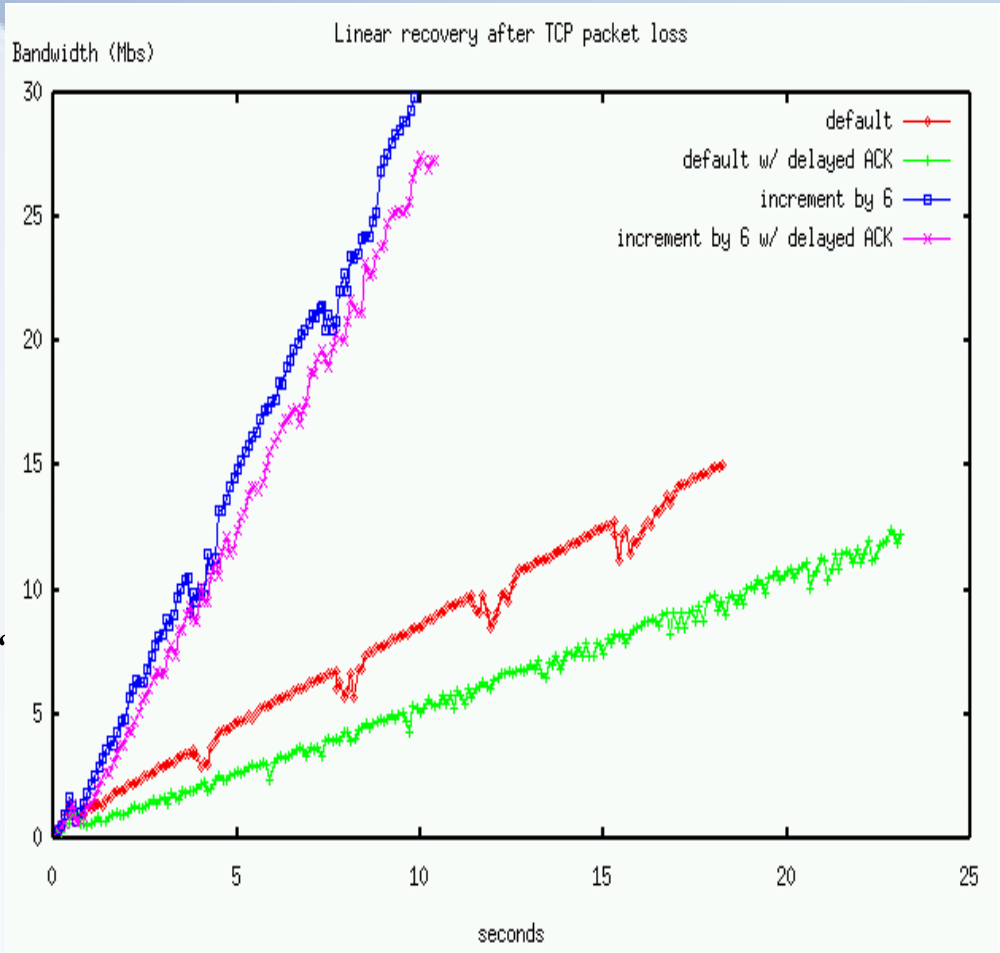
TCP losses



NCAR

Early Impact of Net100

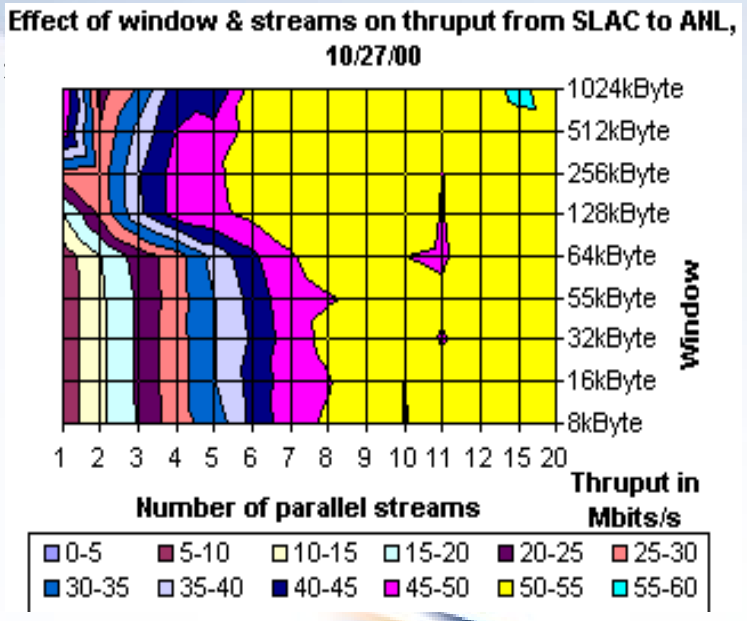
- Avoid losses
  - retain/probe for optimal buffer sizes
  - autotuning (Web100/Net100)
  - ECN capable routers/hosts
  - reduce bursts
- Faster recovery
  - shorter RTT (fix routes)
  - bigger MSS (jumbo frames)
  - speculative recovery
  - modified congestion avoidance‘
- SCTP, out-of-order delivery



NCAR

Bulk transfer speedups

- Parallel streams (*psockets*)
  - —how to choose number of streams, buffer sizes?
  - —Web100 autotune ?
- Application routing daemons
  - —indirect TCP
  - —multipath (Rao, ORNL)
- Are these fair?



NCAR

Network Tool Analysis Framework (NTAF)

- Configure and launch network tools
  - measure bandwidth/latency (*iperf, pchar, pipechar*)
  - collect passive data (SNMP from routers, OS counters)
  - forecast bandwidth/latency for grid resource scheduling
  - augment tools to report Web100 data
- Collect and transform tool results into a common format
- Save results for short-term auto-tuning and archive (NAIB) for later analysis
  - compare predicted to actual performance
  - measure effectiveness of tools and auto-tuning
- Use NetLogger to format and send data to NAIB

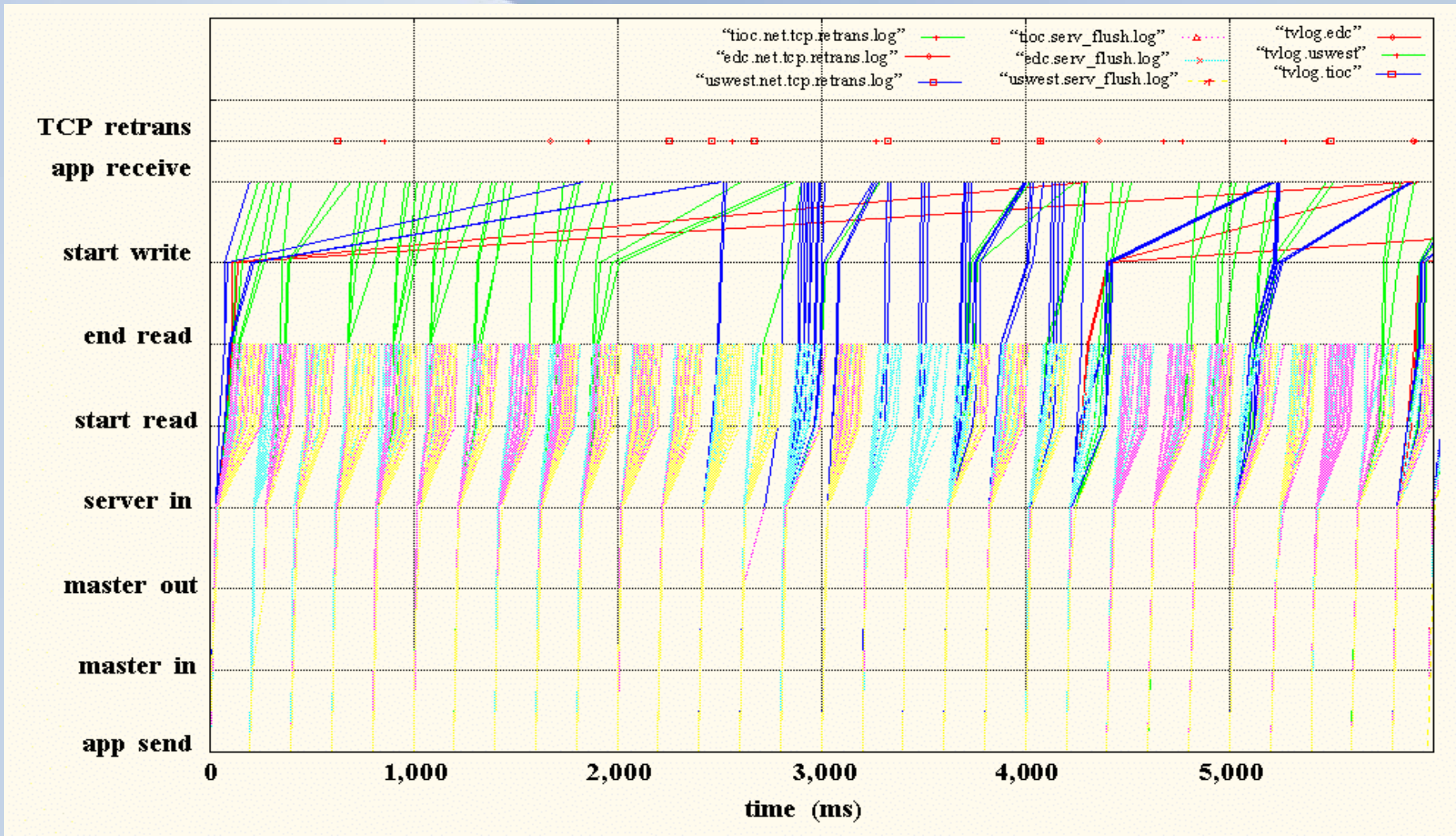
NCAR

NetLogger

- End-to-end performance monitoring tool
- Modify application to log interesting events
- Support for distributed applications (NTP timestamps)
- Identify application/network bottlenecks
- Components
  - —IETF draft standard message format
  - —API for event logging
  - —tools for collecting/sorting log files
  - —visualization tool for monitoring/playback

NCAR

NetLogger



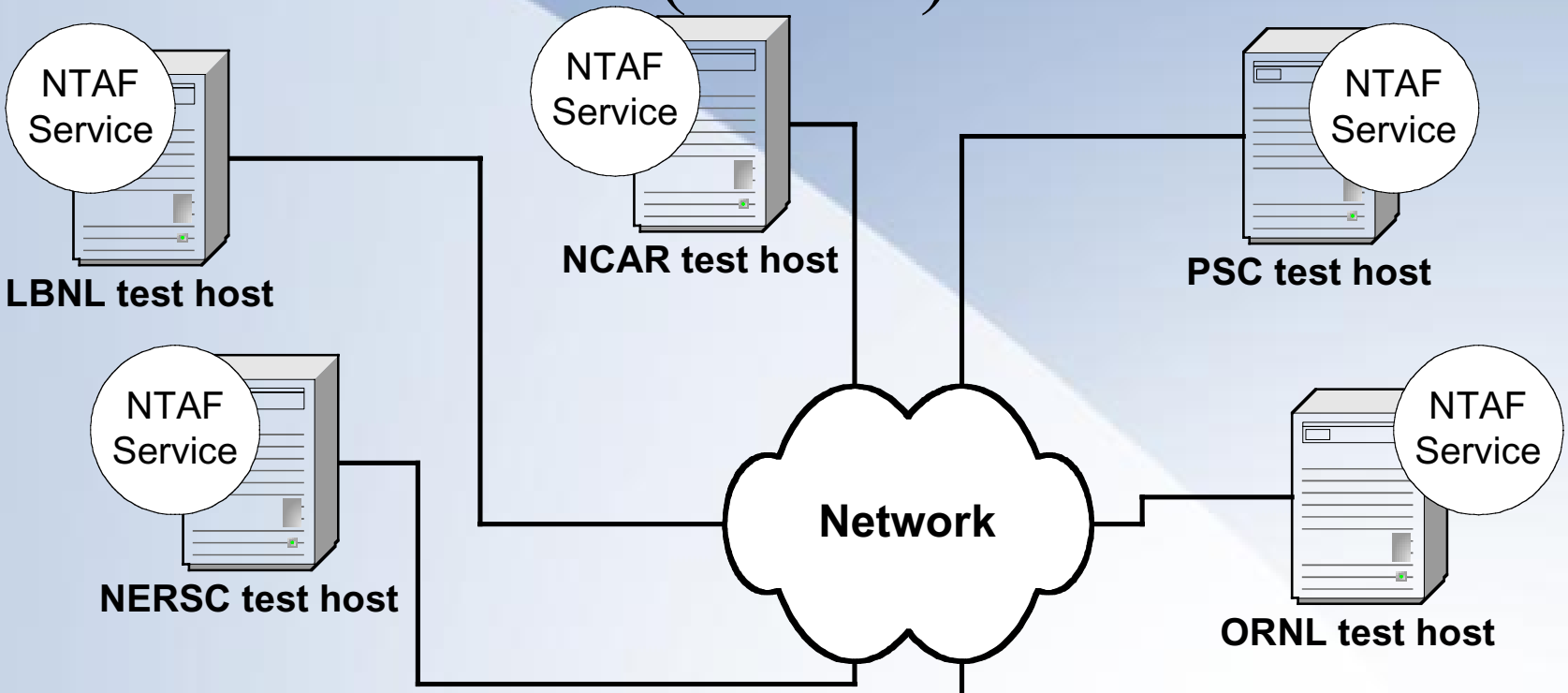
NCAR

Network Analysis Information Base (NAIB)

- Extensible infrastructure for performance data
- Collect data from active and passive Net100 probes via NetLogger
- Gather and serve data via programmatic and graphical interfaces

NCAR

Network Tool Analysis Framework (NTAF)



Configured to perform tests from each host to all other hosts

ping, traceroute, iperf, pipechar, etc.

can query any NTAF service for recent results FROM that server

all results sent to archive

NCAR

NTAF Use Case

- The NTAF is configured to run the following network tests every few hours over a period of several days:
  - ping -- measure network delay
  - pipechar -- actively measure speed of the bottleneck link
  - iperf -- actively measure TCP throughput. Multiple *iperf* tests could be run with different parameters for the number of parallel streams {e.g.: 1,2,4} and the method of tuning the TCP buffers {Web100 auto-tuned, hand-tuned}
- All tools will use the Web100 TCP-KIS interface to collect TCP information from the Web100 kernel, and then use NetLogger to format and send this data to the database.

NCAR

Use Case (cont.)

- Analysis based on this test configuration includes:
  - The ability to compare Web100 tuned throughput to hand-tuned throughput.
  - The ability to compare predicted bandwidth with application and *iperf* bandwidth.
  - The ability to determine the advantage, if any, of parallel data streams, using both hand-tuned and autotuned (Web100-tuned) TCP.
  - The ability to see the variability of the results over time.
  - The ability to compare *pipechar* and *pathrate* to see which is most accurate.
  - The ability to measure the impact of tuned TCP streams on non-tuned streams.

NCAR

GridFTP

- Working with the Globus project to add NetLogger instrumentation to GlobusIO and GridFTP,
  - provides detailed analysis of how modifying TCP parameters effects GridFTP application performance.
  - Will begin testing the use of the WAD to improve GridFTP performance.

¥ GridFTP (a real application)

—Deployment

—Full install or NTAF install?

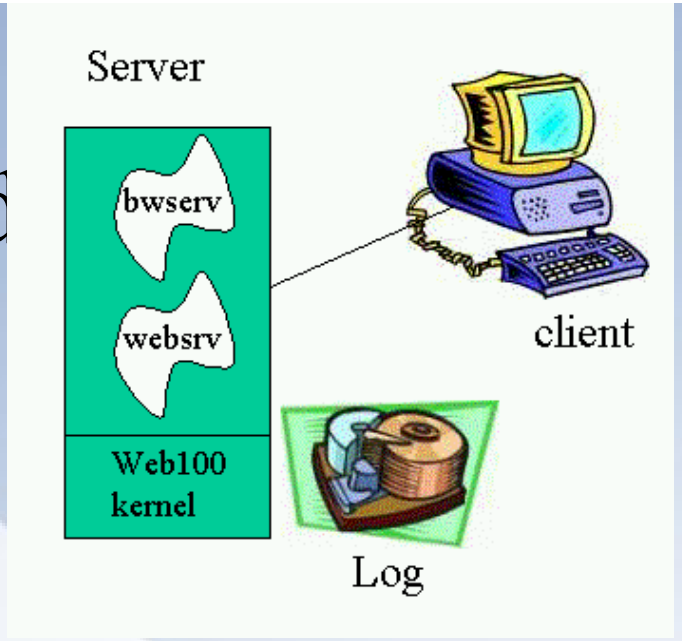
¥ will require work for sysadmins

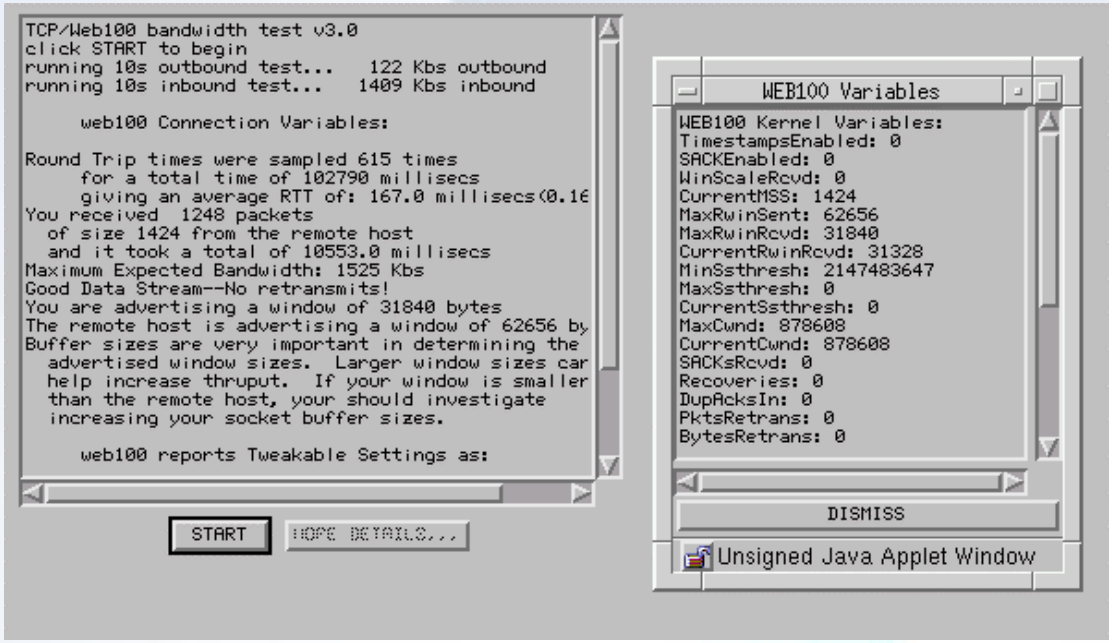
¥ PROBE?

NCAR

Net100: applied

- Web100
  - Linux 2.4 kernel mods
  - 100+ TCP variables per flow
- Net100
  - Add Web100 to iperf/ttcp
  - Monitoring/tuning daemon
- Java applet bandwidth/client tester
  - fake WWW server provides html and applet
  - applet connects to *bwserver*
    - 3 sockets (control, bwin, bwout)
    - server reports Web100 variables to applet (window sizes, losses, RTT)
  - **Try it**
    http://firebird.ccs.ornl.gov:7123

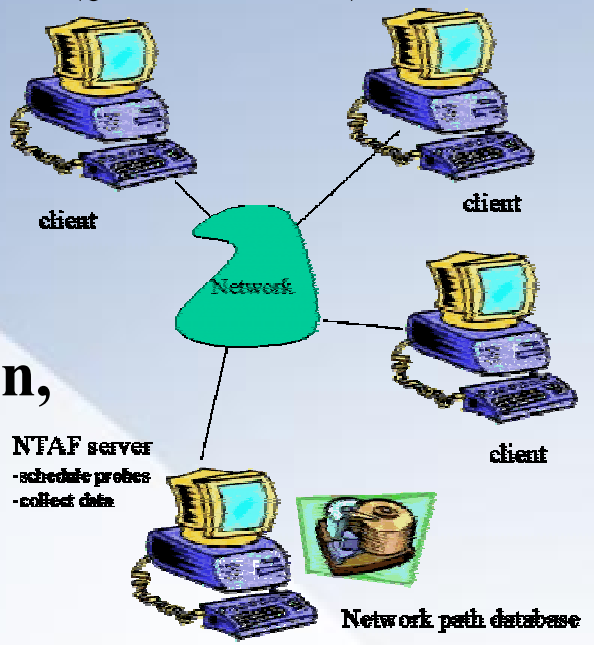


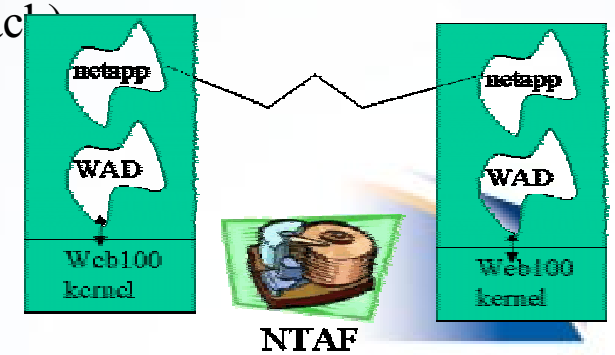


NCAR

Net100 concept (year 1)

- **Path characterization (NTAF)**
  - — both active and passive measurement
  - — data base of measurement data
- **Application tuning (tuning daemon, WAD)**
  - — Work around network problems
  - — daemon tunes application at start up
    - static tuning information
    - query data base and calculate optimum TCP parameters
  - — dynamically tune application (Web100 feedback)
    - recalculate parameters during flow
    - split optimum among parallel flows
- **Transport protocol optimizations**
  - — what to tune?
  - — is it fair? stable?

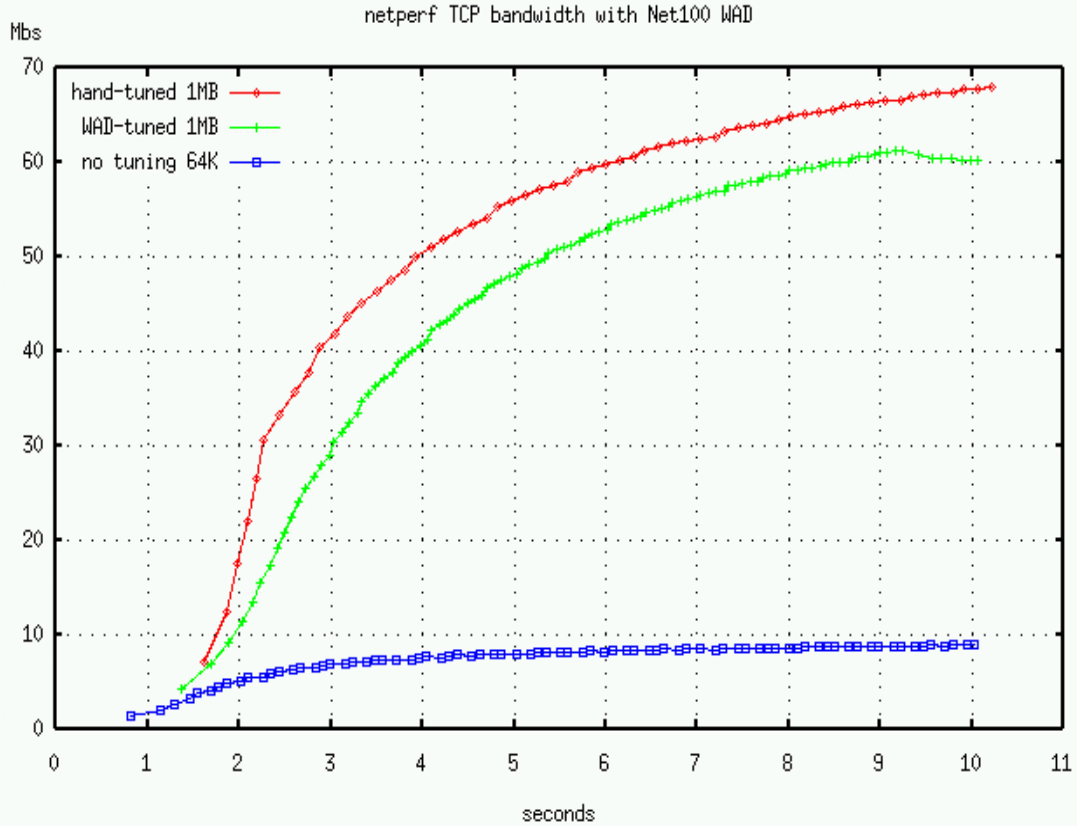




NCAR

Net100: tuning

- Work-around Daemon (WAD) Version 0
  - tune unknowing sender/receiver at startup
  - config file with static tuning data
    - {src, srcport, dst, dstport, window}
  - LBL has python version
    - expression-based tuning
- To be done
  - applying measurement info
  - tune more than window size?
  - communicating WADs
  - dynamic tuning



Example WAD Usage

- Manipulate Web100 variables based on info from other Web100 instrument variables
- Ability to generate and log derived events:

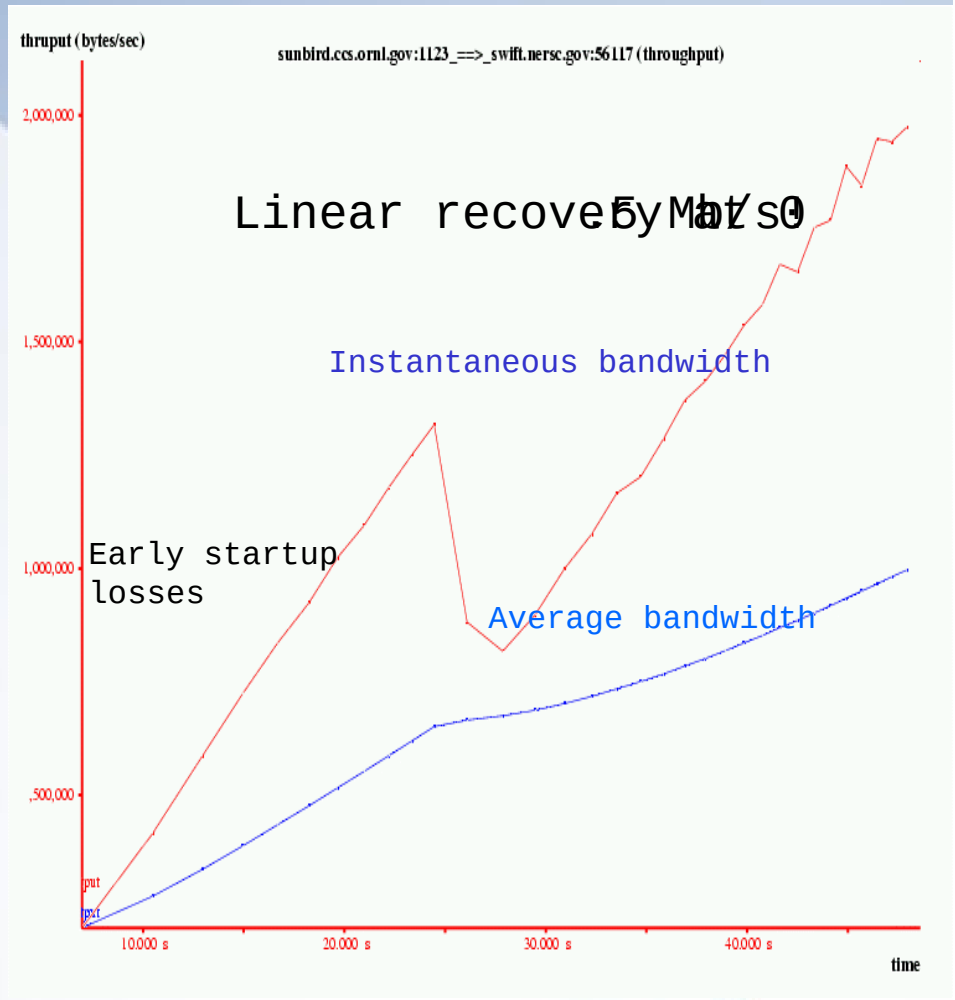
```
derived_event: BW=(DataBytesOut*8)/(SndLimTimeRwin+
               SndLimTimeCwnd+SndLimTimeSender)
```

  - uses NetLogger to send events to archive or for real-time analysis
- Ability to tune parallel streams (make them fairer?)
  - buffer size per stream = optimal buffer size for 1 stream / number of parallel streams
- WAD-to-WAD control channel
  - Receiver WAD sends tuning data to transmitter WAD

NCAR

TCP Issues

- TCP robust over 20 years
  - reliable/stable/fair
  - need window = bandwidth*delay
    - ORNL/NERSC (80 ms, OC12) need 6 MB
- **Changing**: bandwidths
  - 9.6 Kbs 1.5 Mbs ..45 10 0 1000 ? Mbs
- **Unchanging**:
  - speed of light (RTT)
  - MTU (still 1500 bytes)
  - TCP congestion avoidance
- TCP is lossy by design !
  - 2x overshoot at startup, sawtooth
  - recovery after a loss can be very slow on today s high delay/bandwidth links
  - Recovery proportional to MSS/RTT^2



NCAR

Net100

¥Avoid losses

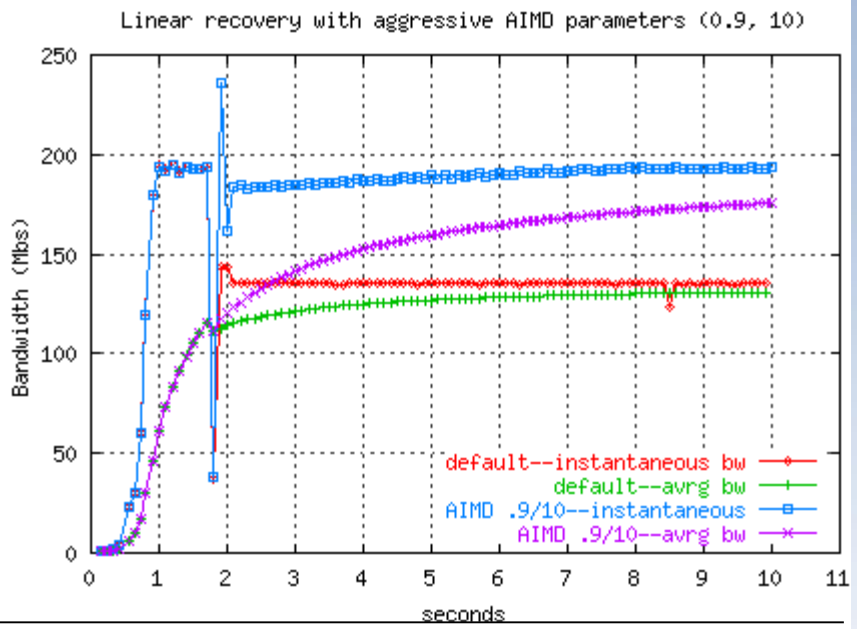
- —use optimal buffer sizes determined from network measurements
- —ECN capable routers/hosts
- —TCP Vegas
- —reduce bursts

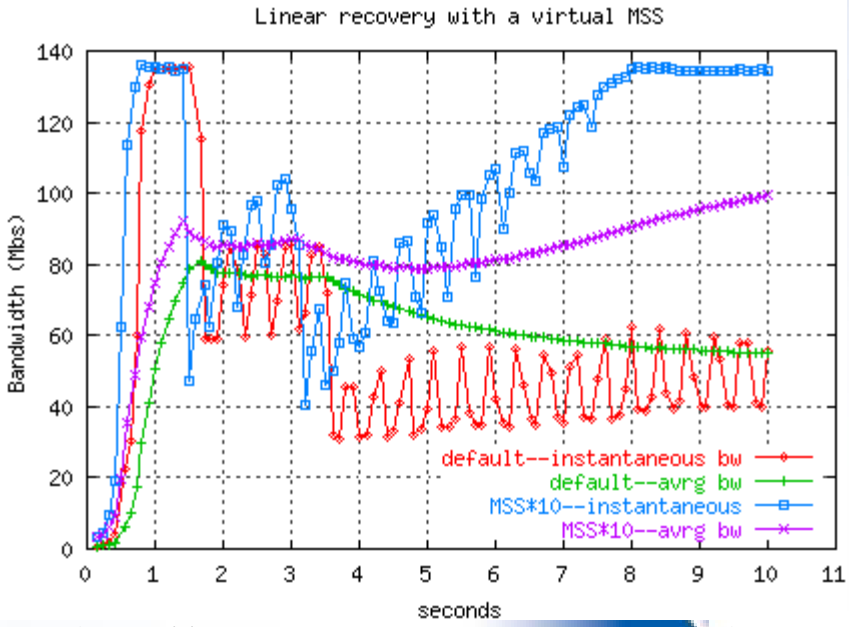
¥Faster recovery

- —bigger MSS (jumbo frames)
- —speculative recovery (D-SACK)
- —modified congestion avoidance (AIMD)
- —TCP Westwood (sender-side mods of the window congestion control scheme)

¥Autotune (WAD variables)

- —Buffer sizes
- —Dupthresh (reordering resilience)
- —Del ACK, Nagle
- —aggressive AIMD
- —Virtual MSS
- —initial window, ssthresh
- —apply only to designated flows/paths





(tests with TCP-over-UDP, *atou,* NERSC to ORNL)

NCAR

Net100 status

- Completed
  - network probes at ORNL, PSC, NCAR, LBL, NERSC
  - preliminary schema for network data
  - initial Web100 sensor daemon and tuning daemons
  - integration of DRS and Web100 (proof of principle)
- In progress
  - TCP tuning extensions to Linux/Web100 kernel
  - analysis of TCP tuning options
  - deriving tuning info from network measurements
  - tuning parallel flows and *gridFTP*
- Future
  - interactions with other network measurement sources
  - multipath/parallel path selection/tuning

NCAR

Net100 and ESnet

- GigE jumboframe experiments
- ECN experiments
  - —Supported by Linux
  - —Instrumented by Web100
- drop-tail vs RED experiments
- SNMP path data
  - —where are losses occurring?
  - —what kind of losses?
  - —SNMP mirrors (MRTG)

NCAR

Web100/Net100 outreach

- Web pages describing current results
- Downloadable Web100/Net100 software
- NAIB data available
- Tutorials, talks, and papers
  - —SC2002 paper prep
- Interact with DOE grid projects and Data Grid projects

NCAR

For more Information

- http://web100.org/
- http://www.net100.org/
- http://www-didc.lbl.gov/net100/
- http://www.csm.ornl.gov/~dunigan/net100
- http://www.csm.ornl.gov/~dunigan/net100/netlinks.html

NCAR