

**Effective Diagnostic Strategies for Wide
Area Networks
aka
Network Path and Application Diagnosis
(NPAD)**

Peter O'Neil

CENIC: Pathways to Discovery

March 8, 2005

poneil@ucar.edu

NSF STI Award

- ◆ 3 year PSC & NCAR project
 - Matt Mathis (PSC) & Peter O’Neil (NCAR) Co-PI’s
 - Senior Personnel
 - ◆ John Heffner & Raghu Reddy from PSC
 - ◆ Pete Siemsen & David Mitchell from NCAR
- ◆ Builds on work from Web100 and Net100 projects
- ◆ Symptoms of most application and network defects scale with increasing path delay
 - Uncertainty and lack of proper tools leads to reasonable but flawed assumption that problem lies with wide-area path
 - Audiences include: end users, application developers, network admins, & wizards
- ◆ NPAD continues our joint focus on e2e performance issues
 - where previous efforts dealt with end systems, we now turn our attention to application flows across the path
 - “my application takes a long time to run from here to there”
 - what’s a network engineer to do to understand and improve flow problems?

Problem Statement

- ◆ Our work on Net100 and Web100 helped us recognize why e2e diagnosis is such a difficult problem: the symptoms of all flaws scale with path delay, and those symptoms that scale with path delay cause classical diagnostic strategies to yield misleading results
 - Uncertainty and lack of proper tools leads to reasonable but flawed assumption that problem lies with wide-area path

NPAD Summary Rationale

- ◆ Single point failures are (relatively) easy to find and fix
- ◆ Remaining failures are interactive & complex involving:
 - RTT, packet loss, and MTU size
 - RTT and application design
 - Packet rate limit and MTU
 - Queue size buffers in routers and burstyness due to
 - ◆ ACK compression and cross traffic or
 - ◆ Application design
- ◆ Web100 tcp mib extensions draft scales to find multiple events on long paths without having to serially test each path segment

Observed Behavior



- Symptoms appear to scale with increasing RTT path delay
 - Servers and Local Clients see throughput as good, but for remote client throughput performance experienced as poor
 - Reflective of many (most) types of flaws
 - Impacts are multiplicative since the delay of the backbone path length magnifies symptoms of an existing flaw

Example Impacts

- ◆ Chat application (e.g., 50ms RTT per user request)
 - On 1ms LAN, this adds 50ms to user response time
 - On 100ms WAN, the same 50 transactions add 5s to user response time
- ◆ Fixed TCP socket buffer space (e.g., 32kBytes)
 - Reliable delivery of data to the application requires sufficient buffer space to hold one round trip of data
 - On a 1ms LAN, about 200Mb/s throughput can be supported
 - On a 100ms WAN, only about 2Mb/s throughput actually achieved
- ◆ Packet Loss (e.g., 1% loss with 9kB MTU packets on a 100Mb/s network)
 - On a 1ms LAN, 500 Mb/s throughput calculated data rate for link
 - On a 100ms WAN, calculated rate drops to about 5Mb/s throughput

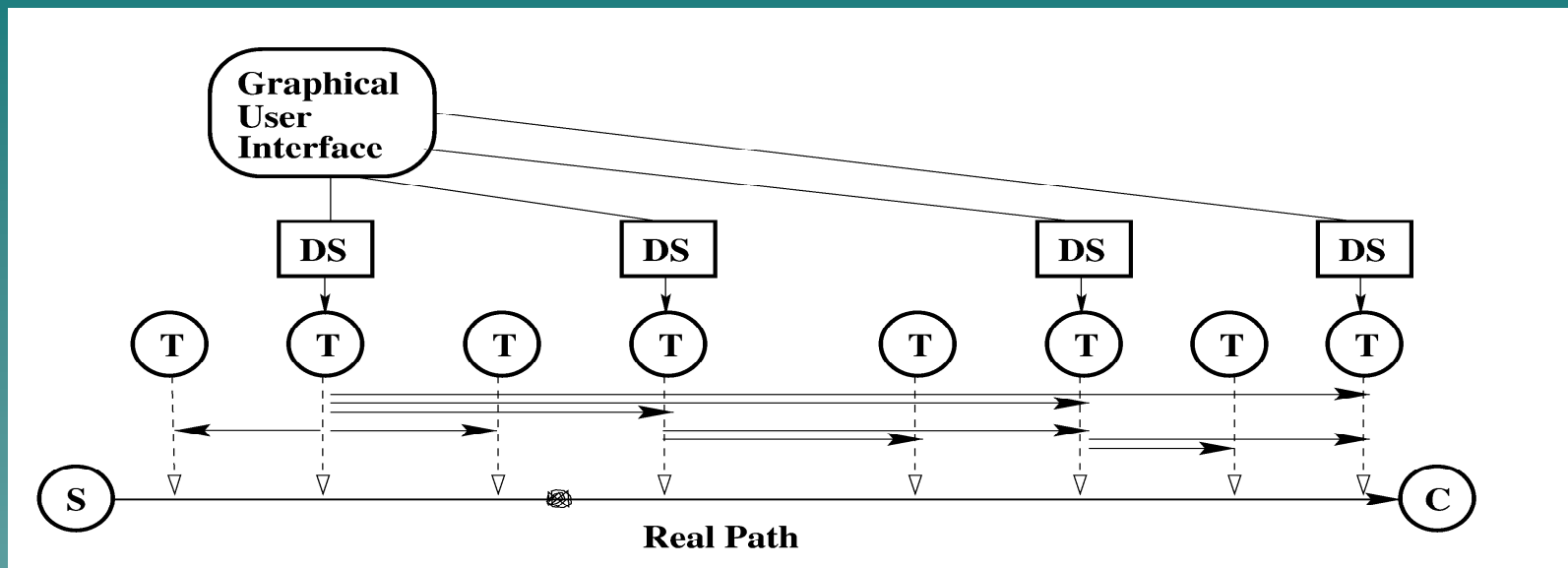
Symptoms Present Diagnostic Quandaries

- ◆ False reassurance on short path flows
 - Obscures actual local bugs/problems
 - Incorrectly points elsewhere
- ◆ Stymies e2e diagnosis and thus improvements
- ◆ Promotes diagnosis as “one off workarounds”
 - Unscalable time sync for network engineers

NPAD Approach

- ◆ Web100 based diagnostic server
 - Simple TCP test to a test target
 - Use Web100/IETF TCP MIB extensions to rescale and model results
- ◆ TCP discard server for test target
 - Believe trivial to widely deploy on Unix boxes at GigaPoPs
 - C or Java
- ◆ Provide good estimates of expected results for long paths

e2e Diagnosis



- Need 1 diagnostic server (DS) per campus/GigaPoP/backbone node
- Test targets at lots of hubs, LANs (or WS/clusters), GigaPoPs
- Allows for isolating a flaw (slowdown) across each element of a long path

End User Tool

- ◆ Web/Java client
 - With built in Test Target
 - Invokes pathdiag test on Diagnostic Server to Test Target machine and back to self


Combine with Other Approaches

- ◆ Bench test applications and end-systems (stacks)
- ◆ Use long ideal (virtual) paths for application developers to help test for delay induced impacts
 - Dummynet style of emulated delay
 - Tunnel or VPN style of “scenic” routing

SCP/SSH Protocol Illustration

- ◆ Effort of Chris Rapier, PSC and Michael Stevens, CMU
 - protocol network performance limited by statically defined internal flow control buffers, buffers act as brake on throughput of SCP especially on long paths
 - high bandwidth, high latency links now prevalent - “elephants”
 - apps that use windowing need to ensure window size at least equals BDP (bandwidth delay product) to obtain maximum link utilization
 - TCP window size can be tuned manually or use Web100 auto-tuning
 - problem arises when apps above TCP layer implement windowing, resulting in throughput becoming the lesser of either TCP or the app
 - two changes required:
 - ◆ enable buffer code to allocate larger sizes
 - ◆ get TCP window size from getsockoptopt and adjust window size to match, but only if new size is larger than old one
 - before and after tests demonstrated dramatic throughput increases - limitation no longer TCP or SSH window size, but ability of host to encrypt at fast enough rate
 - <http://www.psc.edu/networking/projects/hpn-ssh/>

Project Goals

- ◆ Design, improve, and package pathdiag for wide use
 - ◆ Analyze log files to expose (make explicit) and learn about network pathologies to suggest remedies
 - ◆ Study effects of various delay times on throughput
 - ◆ Test the diagnostic tools effectiveness with network users and operators using actual high performance applications
 - ◆ Contribute to life of a network engineer being better!
- 

Status

- ◆ Pathdiag (v0.9alpha) code runs between a diagnostic server and a test target machine
 - diagnostic server must have latest Web100 kernel, C compiler, and Python
- ◆ Test target machine runs trafficrcv code (simple C program)
- ◆ Pathdiag will create 3 html files (written to a directory with the name and time stamp of the test target host name)
- ◆ Just beginning to examine initial test logs to determine how tools can report sensible messages about problems discovered
- ◆ Pathdiag becomes more sensitive on short paths

Work Still to Do

- ◆ Packaging for autoconfig for wider use by community
- ◆ Improve pathdiag to allow results from one test run to lead to another to refine diagnosis and compare results
- ◆ Laminar mode addition to pathdiag to mimic a well behaved smooth bulk transport application such as FTP on a quiet network
 - While useful, its insufficient to detect flaws associated with inadequate queuing in the network, so will compliment burst mode which mimics TCP with a bursty application (e.g., sending large messages separated by short idle intervals), or TCP under effect of ACK compression or other cross traffic

User Interface Development

- ◆ Java User Interface to allow user control of what is happening with test(s) and be able to stop/restart
 - protocol will run between Diagnostic Server and Test Target server
 - needs to support discard server establish TCP connections to the pathdiag server so functionality will work when client and discard server are inside a security perimeter and the pathdiag server is outside
 - Java code layered on top of BEEP which will handle client/server hello/goodbye and version/capability negotiations
 - ◆ BEEP code available for Debian, Java, and Python
 - server to maintain queue of test requests with estimated wait time; when test slot available, server listens on ephemeral port and informs client of port number and IP address; when client connects to this port, server runs the test; when test(s) complete, it responds with an html report or url.
 - ◆ test types include: pathdiag and rerun with different tuning variables
 - ◆ scenic pathways via tunnels to emulate delay of long paths

Questions/Comments?

<http://www.ucar.edu/npad/>