



www.internet2.edu

Network Performance Measurement Tools: An Internet2 Cookbook

Disclosure/Disclaimer

This document is based on a presentation given by Matt Zekauskas, entitled Finding Network Problems that Influence Applications: Measurement Tools. The presentation was prepared by Matt Zekauskas, using original material inspired by Matt Mathis, NLANR DAST, and his own experience; example material created by Rich Carlson and Russ Hobby, with information from the e-VLBI project at MIT Haystack Observatory.

This document was developed to be used in conjunction with a Network Performance Workshop; for more information on these workshops (upcoming and past), see: <http://e2epi.internet2.edu/network-perf-wk/>.

Copyright © 2005, Internet2. All Rights Reserved, except that permission is expressly granted for others to use in noncommercial educational materials as long as attribution is given to Internet2 and the authors.

This cookbook will discuss:

- Problems, typical causes, diagnostic strategies;
- Examples showing how the recommended tools are used; and
- End-to-End Measurement Infrastructure.

We don't have all the answers. This book is about common problems, and tools and techniques we have found useful. Internet2's End-to-End Performance Initiative (E2Epi) would like your help. What problems are you experiencing? Have you used a good tool? Give us the benefit of your experience, especially if there was a successful problem resolution! However, if you have a common problem, or a particularly difficult problem, we'd like to hear about it. In fact, E2Epi collects "war stories" for publication on a web site.

You can learn more about specific tools described in this book – each tool has its own cookbook and is available online at: <http://e2epi.internet2.edu/library-list.html>.

What Are the Problems?

Problems come in several areas: network, [transport protocol](#) (TCP/UDP), and [applications](#). This section also covers the [usual suspects](#) for causes, [strategies](#) for problem resolution, and offers a [case-in-point](#).

Network Problems

Here's a list of network problems that we find affect performance of most applications:

- Packet loss
- Jitter
- Out-of-order packets
- Excessive latency
- Duplicated packets

Packet loss slows TCP (bulk data transfer), and causes dropouts with voice and artifacts like "jaggies" with video. Extreme packet loss will prevent connections from making progress. One reason packets could be lost is congestion within the network – some link that just has too much traffic to carry. Most links have buffers, and build up queues, but when these fill, packets are dropped. This is called *congestive* packet loss. If a path is congested, it is obvious (at least once the problem link is found) because utilization is high. However, losses can also be caused for other reasons (which we will get to in a moment) and these *non-congestive* losses are especially hard to track down (often requiring intermediate test points along a path).

Jitter, or the change in the rate that packets arrive, can also cause TCP to slow down, or at least react to problems more slowly; excessive jitter in a real-time application can cause some packets to be treated as if they were lost, causing dropouts and video problems. If you have an interactive application, say remote control of a scanning-tunneling microscope, jitter makes it hard for humans to react. We can learn how to deal with

latency, but we can't adjust for arbitrary changes in latency. An Ohio state study of H.323 codecs found that jitter caused more problems than loss (up to some point).

Out-of-order packets can be viewed as extreme jitter; in addition to the problems already listed, many applications are not well written (i.e., early MPEG-2 and HDTV codecs), and out-of-order packets can cause more problems than lost packets. Applications should be written to be tolerant of out-of-order packets (since parallelism in the network can generate them naturally, and they actually occur quite frequently), but for now, reducing the number of out-of-order packets will improve application efficiency.

Excessive latency makes it difficult for such interactive applications as video conferencing and remote instrument control, although humans can compensate to some extent. TCP's control system also has more trouble as latency increases, since it reacts more slowly. In general, it is best to engineer paths so that latency is minimized.

Duplicate packets waste bandwidth, and, in extreme cases, can cause TCP to slow down or confuse real-time applications.

How Loss Affects TCP Performance

Loss (especially coupled with excessive latency) is the major network-related problem that causes TCP to perform poorly. Let's look at how "vanilla" Reno TCP (still the most commonly deployed TCP stack) reacts to losses to see how important limiting loss along a path is.

Let's say our goal is modest (for modern workstations): send 100 megabits from coast-to-coast. With full size Ethernet packets (1500 bytes for 100Mbps interfaces), you need a probability of packet loss on the order of one in a million ($10^{-6} P_{\text{loss}}$ [Mathis]). That's one loss every 83 seconds (see <http://www.psc.edu/~mathis/papers/JTechs200105/>).

[Note: This is an equation to give you a general idea; exact semantics depend on the version of TCP that is deployed; a single loss may get covered by "fast retransmit," and the sender may never slow down.]

How about if you have gigabit Ethernet? Then, the loss probability must be less than one in ten to the negative eighth power (10^{-8}), or one loss every 497 seconds. The situation gets better if you can use so-called "jumbo-frames," or 9000 byte packets; it goes back to close to the 100 megabit case. That's one reason to try and make high-performance paths "9000-byte clean."

The situation also gets better with some of the newer TCP algorithms (high-speed TCP, BIC TCP, FAST TCP), which is why there's a lot of research into new bulk-transfer control algorithms. We would still, however, like a way to find and remove non-congestive losses as much as possible.

Transport Protocol Problems

The number one reason for TCP not running at “full-speed” is for it to be starved for buffer space. Vendors ship TCP stacks with buffers that are tuned for the commercial Internet. If the buffer is too small, TCP, which uses a “sliding window” for flow control, must wait for packets to be acknowledged to advance the window and send more data. Essentially, the sender is forced to stop and wait. You need to be able to buffer the number of bits you can send in one round-trip time at your desired speed. [Note: There are also send and receive buffers; if either is too small you can end up with this problem.]

For example, with a 70 millisecond round-trip time (more-or-less trans-continental North America), to sustain one gigabit per second you need 8.4 megabytes of buffer space ($70\text{ms} * 1\text{Gbps} = 70 * 10^6$ bits). For 100Mbps at the same distance, you need 855 kilobytes ($70\text{ms} * 100\text{Mbps} = 855\text{KB}$). Many stacks default to 64 kilobytes, which only allows 7.4 Mbps.

One word of caution – network kilobits, megabits, gigabits are powers of 10. Memory kilobytes and megabytes are in powers of two, a kilobyte being 1024 bytes (2^{10}) and a megabyte being 1,048,576 bytes (2^{20}).

TCP has two buffers – send and receive. The sliding window is always used, not only when the buffer is too small. Since TCP delivers a reliable, in-order packet delivery service, it needs to detect and recover from loss and out-of-ordered packets. The sender must retain a copy of all packet sent in the event that IP packets are lost. If this buffer fills up, the sender must stop sending until ACKs are received.

The receiver must also deal with out-of-order packets, so it maintains a reassembly buffer. This buffer also holds packets when the application is unable to process them. If the receive buffer fills up, the sender must stop sending until the application can process the data.

As you see, either buffer filling up can cause the sender to stop. Both sender and receiver must be tuned to eliminate buffer stalls. In some cases, it may not be possible for the local host or sys-admin to fix the problem (i.e., the remote host has mis-set buffers).

Application Problems The same problem carries up to the applications themselves. For video and audio (streaming media), the lack of buffer space in the application (in our world, MPEG-2-based applications are especially bad) means the application is very sensitive to packet loss or reordering. Of course, if your application is interactive, then increased buffering can lead to lag in response, which is not desirable, either.

This generalizes to bad network application behavior, so that they are not robust to network changes or anomalies. Drops will occur. Reordering will occur. Even if only very occasionally.

Even applications that would like to use TCP to do bulk transfer can do such things as not hand enough data to TCP to allow it to stream over long distances. One that was brought to light recently is scp (and therefore ssh; this problem can also occur with standard FTP); popular versions of scp do not provide large enough buffers for TCP to stream. (There is a pointer to a good version of scp off a tcp tuning page at PSC mentioned later.)

The Usual Suspects: Causes for the Above Problems

There is *always* a list of the usual suspects – the ones that keep the trouble ticket queue in business:

- Host configuration errors (TCP buffers)
- Duplex mismatch (Ethernet)
- Wiring/Fiber problem
- Bad equipment
- Bad routing
- Congestion

First on the list, and most common, is a bad host configuration. As we just mentioned, this is usually because operating systems ship tuned to the commercial Internet and we have very different paths over the Internet2 infrastructure (in particular the “bandwidth delay product” is much greater).

The second most common is duplex mismatch, usually due to auto-configuration failure, with one side believing it is full duplex (can send and receive simultaneously), and the other side believing it is half-duplex (can only send or receive one at a time). This is a legacy of how the Ethernet standard has evolved. This is the major cause of “non-congestive” packet losses.

Wiring or fiber problems also can cause “non-congestive” packet losses. Bad equipment (anything from host interfaces that cannot run full-speed, to host, switch, router, or fiber equipment failure) can cause excessive delays, jitter, or “non-congestive” packet loss.

Bad routing can cause excessive latency, or sometimes jitter due to multiple different length paths being used. Congestion – both “Real” traffic and unnecessary traffic (broadcasts, multicast, denial of service attacks) – causes varying delays and packet loss.

General Strategy

Most problems are local... Here’s a playbook for locating the problem. First and foremost, if you are planning a demo, or other event, test ahead of time. If you have a concerned application community, this may mean periodic testing among points close to key equipment. For example, all of the Radio Astronomy Very Long Baseline Interferometry (VLBI) sites may test among each other. It may also mean periodic testing within your network to points in Abilene, or other campuses you talk to frequently.

Now, say you have a problem that the periodic testing did not pick up (there are just too many paths to test them all). The first question – do you have connectivity and reasonable

latency? Ping will give you round-trip times, assuming it isn't blocked along the way. One of the recommended tools described later, One-Way Ping (OWAMP), measures one-way delay, which allows you to disambiguate problems that might occur asymmetrically – asymmetric routing, asymmetric traffic queuing, a dirty fiber can cause asymmetric problems (since each fiber transmits light in one direction). Are you seeing many losses with these low-rate tests? If so, there is probably something terribly wrong.

If the latency is not what you expect, there may be a routing problem. The best-known tool is traceroute, and you can use that to make sure the path looks reasonable. It goes through your campus, possibly through a gigapop, across Abilene and down to the other side in a reasonable fashion (not taking a scenic tour of the US, for example). Remember that you have to test in the opposite direction; the Abilene router proxy and traceroute servers can help. If you are in the US, and expecting a research path, if no node has the suffix “abilene.ucaid.edu” or the middle nodes contain recognizable commercial names (“att.net”, “sprint”, “mci”, etc.), then routing is directing you over commercial paths where performance may not be what you expect.

Has the host been tuned? Is there potentially a duplex-mismatch one of the local Ethernet connections? Here, running NDT (Network Diagnostic Tool), another of the recommended tools, can point out a series of common problems. NDT itself relies on Web100, which instruments the Linux kernel. You might consider installing a Web100 machine (or using machines with Web100 code); there are additional diagnostics you can run using the Web100-provided variables, and the kernel itself is better “out of the box”: it can automatically tune buffers on some TCP connections.

If routing looks reasonable, and the host is reasonable, you may have a problem in the path. Large losses in the low rate tests also indicate path problems, assuming it isn't a duplex mismatch problem, local congestion – perhaps a denial of service attack – or even broken network hardware on the end system.

Iperf is a tool to run synthetic TCP streams (memory-to-memory) between two machines. BWCTL, another recommended tool, adds authentication and scheduling to Iperf and allows you to test to multiple points, including midpoints within Abilene.

One Technique: Problem Isolation via Divide & Conquer

For path problems, the best strategy is usually to “divide and conquer”; test to a midpoint, and see which side the problem is on, and then test to a midpoint on one side, until you've exhausted your midpoints and have localized the problem as much as you can.

Internet2 is working on tools to automate this process, but, for now, it is all manual. This picture shows testing to a bunch of points that you have access to.

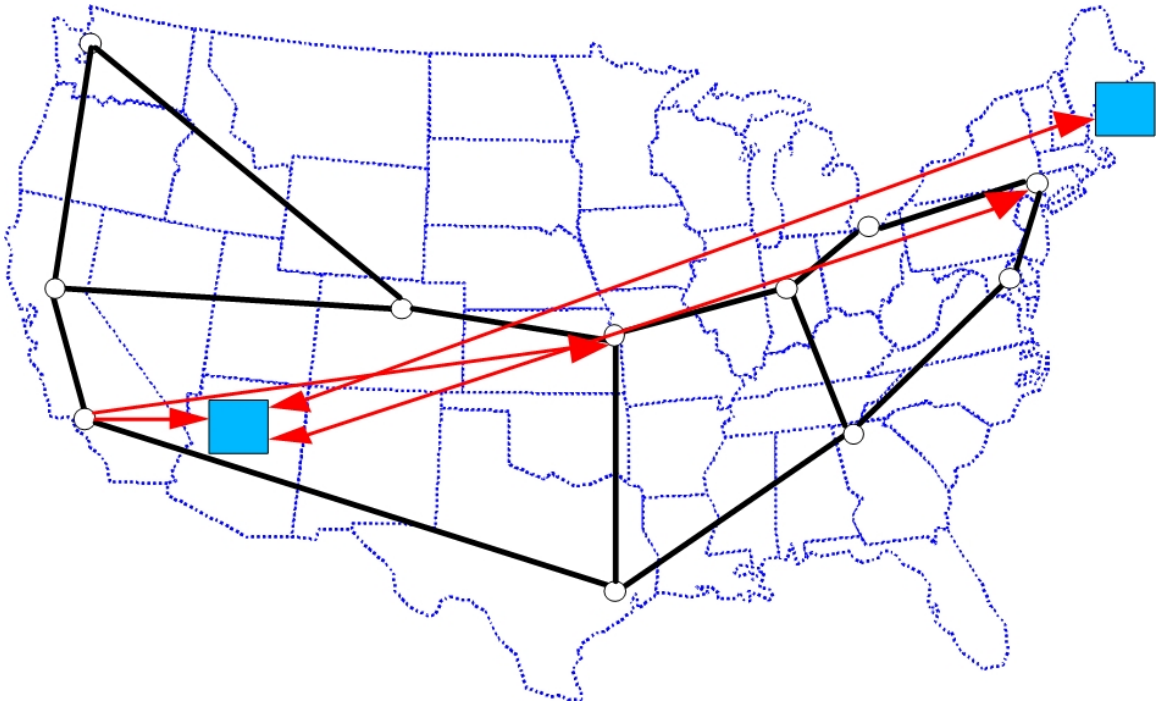


Figure 1. Divide and Conquer

Tool Examples

This section will provide examples showing usage of the tools we use and discuss, in detail, in our network performance workshops:

- When to use NDT
- NDT in action at SC'04
- When to use the Bandwidth Test Controller
- The Bandwidth Test Controller in action with e-VLBI
- When to use One-Way Ping
- One-Way Ping in action with Abilene

When to Use NDT

The NDT (Network Diagnostic Tool) is the quickest of the tools to use – the client can be used on almost any host (it is Java-based) and servers are almost always close to some point in the path along Internet2 that you are interested in. Use NDT when you or a user need:

- To know about last mile and host problems
- A quick and easy test to provide clues at possible problem cause
- To understand large segments of the path from the host view point
- To test their own host

Technique

Start by testing to the nearest NDT server from each end of the problem path. This will help you with a majority of problems. If test indicates good performance, test to a distant NDT server. If tests still indicate good performance, suspect a problem in the application, not the host or network.

SC'04 Real-Life Example

During the SC'04 (SuperComputing 2004 Conference), a booth was having trouble getting an application to run from Amsterdam to Pittsburgh. Tests between the Amsterdam SGI and Pittsburgh PC showed throughput limited to less than 20 Mbps. The immediate assumption: PC buffers too small! The question became: How do we set the WinXP send/receive buffer?

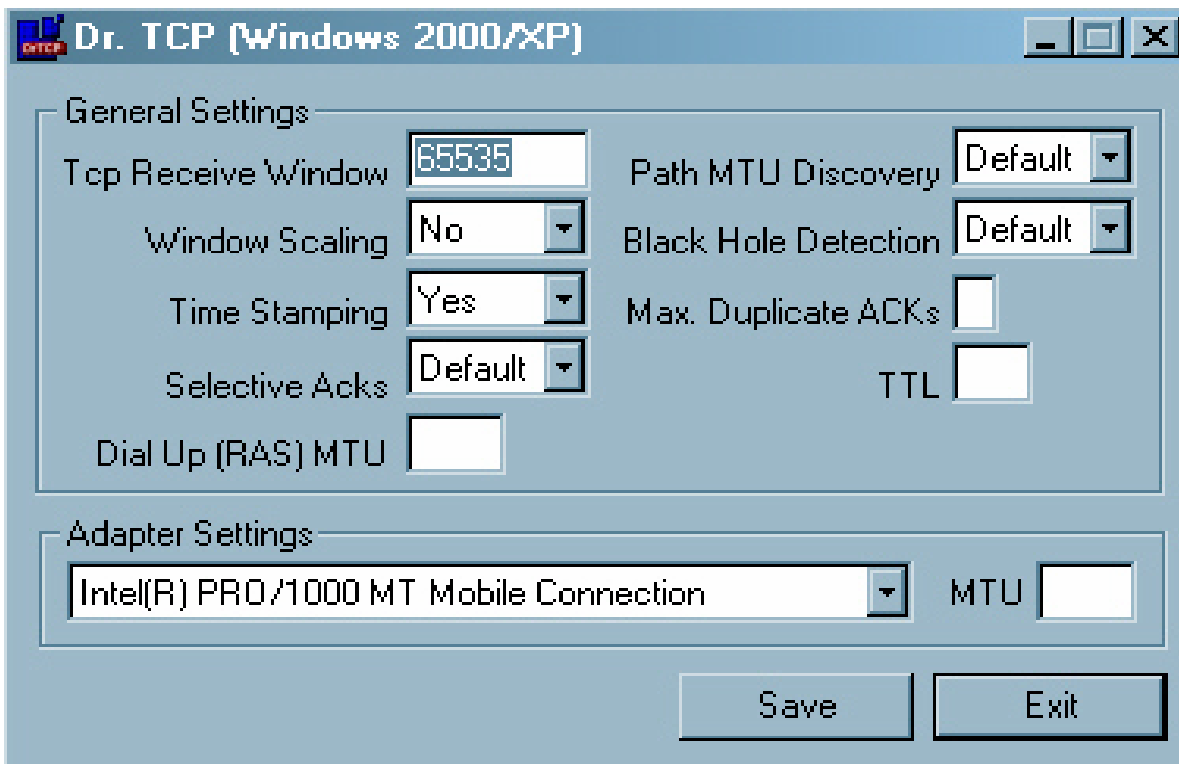


Figure 2. Determine Win XP Info

Rich Carlson, a developer of NDT, pointed the sysadmin in charge to this tool to check/set the Windows buffer size. [NOTE: In Windows, TCP transmit buffer tracks receive buffer, so setting either one sets them both. This is the output from a laptop computer.]

Next step – confirm PC settings. Results when NDT was run on sysadmin's XP system:

- Client-to-Server: 90 Mbps
- Server-to-Client: 95 Mbps
- PC Send/Recv Buffer size: 16 Mbytes (wscale 8)
- NDT Send/Recv Buffer Size: 8 Mbytes (wscale 7)

- Reported TCP average RTT: 46.2 msec – approximately 600 Kbytes of data in TCP buffer
Min buffer size / RTT: 1.3 Gbps

The sysadmin and researchers didn't believe the results because their test program kept saying 8K buffers were being used. Next, the team ran a test from the XP to the NDT server on the conference network. The report clearly showed that PC buffers were being set to 16 MB. Also, with that buffer size and the reported RTT, the link should be able to sustain 1.3 Gbps and the limit is the physical 100 Mbps Fast E interface in the PC.

DrTCP reported 16 MB buffers, but the test program was still slow; the question was: how do you confirm? The team ran a test to the SCInet NDT server (PC has Fast Ethernet Connection). This test confirmed that the local PC was configured correctly – no problem was found, it was able to run at line rate, and the test confirmed that the PC's TCP buffers were set correctly. The conclusion from the test was that the PC was operating properly.

Now they turned to the Amsterdam SGI. They ran a test from the remote SGI to the booth on the SC'04 show floor (the SGI is Gigabit Ethernet connected). They downloaded and built the command line tool on SGI IRIX:

- Client-to-Server: 17 Mbps
- Server-to-Client: 16 Mbps
- SGI Send/Recv Buffer size: 256 Kbytes (wscale 3)
- NDT Send/Recv Buffer Size: 8 Mbytes (wscale 7)
- Average RTT: 106.7 msec
- Min Buffer size / RTT: 19 Mbps

This host was located at the sysadmin's home university in Amsterdam. They needed to download and build the command line client before this test could be done. The SGI has a small (256KB) buffer and that this buffer limits the throughput to ~19 Mbps, which is in line with the measured results. Now, the group knew that the SGI needed to be tuned to run over the transatlantic path. The user was reluctant to make changes to the SGI network interface from SC'04 show floor; the NDT client tool allowed the application to change the buffer (setsockopt() function call).

Once the Amsterdam SGI was tuned, the team re-ran the test from the remote SGI to the SC'04 show floor, using the `-b #` option:

- Client-to-Server: 107 Mbps
- Server-to-Client: 109 Mbps
- SGI Send/Recv Buffer size: 2 Mbytes (wscale 5)
- NDT Send/Recv Buffer Size: 8 Mbytes (wscale 7)
- Reported average RTT: 104 msec
- Min Buffer size / RTT: 153.8 Mbps

Experiments showed that 2 MB was the maximum value the SGI would allow. It would require a system configuration change to go above this value. Notice that the tests get 100 Mbps, which is sufficient because the client was only connected via a Fast Ethernet link.

The sysadmin now knew that the network path could support full line rate between the convention floor and the remote site in Amsterdam.

SC'04 Debugging Results – the sysadmin spent over one hour looking at Win XP configuration, trying to verify buffer size, and determine why the XP host wasn't operating properly. The two tools used gave different results; the data provided either was not correct or was not believed because test results did not match expected results. A single NDT test verified this in under 30 second.

It took about 10 minutes to download and build the NDT command line client on the SGI. This was done remotely from the SC'04 convention floor via an ssh terminal session. After running the test, it took another 15 minutes to evaluate the results, identify the options, and specify how to measure improvement if the default buffer sizes were increased. It took eight minutes to find the SGI limits and determine maximum allowable buffer setting (2 MB). Total time – 34 minutes to verify problem was with remote servers' TCP send/receive buffer size.

When to Use the Bandwidth Test Controller (BWCTL)

It is a good idea to use the Bandwidth Test Controller if you need to:

- Understand segments of the path
- Know if each segment can handle flows of a specific size
- Know parameters such as bandwidth and packet loss
- Help design or tune an application based on available performance

Another reason to use the Bandwidth Test Controller is that you do not have access to the end hosts; using this tool, you can still get partial information. Remember that the Bandwidth Test Controller only allows testing only between known entities (other allowed Bandwidth Test Controller servers), it ensures that only one test at a time is done between servers, and these servers will have been tuned to insure that the results reflect the network performance. [Note that to use this tool, you will have to obtain access to intermediate servers. Participants in our workshops will likely obtain access to Abilene servers as part of the workshop; they or others can use the web site [\[http://e2epi.internet2.edu/pipes/ami/bwctl/\]](http://e2epi.internet2.edu/pipes/ami/bwctl/). Another web site acts as an interim directory of servers; if your path crosses one of the listed networks, instructions are given as to how to obtain access. The directory web site is [\[http://e2epi.internet2.edu/pipes/pmp/pmp-dir.html\]](http://e2epi.internet2.edu/pipes/pmp/pmp-dir.html).]

Technique

The general technique to use for the Bandwidth Test Controller is divide-and-conquer. If you have a performance problem, and NDT says that the “last mile” is performing correctly, and the host has no tuning issues, then Bandwidth Test Controller is the tool to use to diagnose which intermediate path segment contains the problem. In general, TCP testing is done among intermediate points. In certain cases, moderate-rate UDP testing can be used to deduce particular loss patterns.

Case A: A performance problem between two nodes, but NDT test at each node pass. Using traceroute, deduce the path; using the directory web site above, see if any test points are along your path. Pick an intermediate node along the path. Try from one edge to the intermediate node. If that has the problem, recursively pick another intermediate node between those two points until there is no intermediate node to pick. (If that half-segment didn't have a problem, try from the intermediate node to the other end, it likely has the problem, and recurse again.) Once you have minimized the path distance there should be only one or few possible troublesome networks, and your network operations center should be able to contact them with the test results you have, and resolve or get an explanation for the problem (the Abilene NOC can also be helpful, as can Internet2 engineers).

Case B: Low-rate tests show loss. Earlier, we mentioned that if low-rate latency tests show loss (say with One-Way Ping), then there is often a serious problem with the path. In Abilene, One-Way Ping and Bandwidth Test Controller nodes are co-located. If One-Way Ping tests show loss, you can use the Bandwidth Test Controller in UDP mode to make these explicit. If you only have TCP access (because UDP streams can cause congestion, and the stream itself won't react), you can still find which segments are having problems because TCP performance will be unnaturally low. Say you have access to the two nodes where One-Way Ping reports loss (One-Way Ping is, by nature, one-way, so you know which direction is having problems). Try a 10 second TCP Bandwidth Test Controller test in the same direction. Performance should be much less than what the end-hosts are capable of (say 1Mbps instead of 100Mbps). Pick an intermediate node, and begin divide and conquer with the TCP tests as in Case A above.

Case Study: e-VLBI

The e-VLBI (electronic Very Long Baseline Interferometry) project needed to move massive amounts of data between several radio telescope sites around the world. They found that performance from some sites was only in the 1 Mbps range, which would not permit them to transfer data from the telescopes to the correlators in real time.

In the past, the telescopes saved the data on magnetic tapes, shipped those tapes to the correlator, and, several weeks later, received results. There were several problems with this approach, not least of which is that the telescopes could not react to changing conditions or new discoveries in a timely manner. However, the real impetus for changing to real time data downloads was that the manufacturer of the magnetic tapes was no longer producing new ones and the supply was quickly running out.

Test Infrastructure

David Lapsley, one of the MIT Haystack research engineers, established Bandwidth Test Controller servers at the sites of the project:

- Japan: Kashima Observatory
- Sweden: Onsala Observatory
- US: Haystack (BOS)

He performed a full mesh of tests between all of the servers.

Test Results

They used Abilene nodes to divide the problem path; David found that there was considerable packet loss in the area of Haystack Observatory and, working with network folk from the area, the problem was quickly isolated and resolved in time for the real-time data transfer experiment.

Abilene nodes were in the middle – so they could act to see which side of the path they needed to focus on. What we have discovered is that problems are often easy (at least conceptually) to fix, once you can focus on the problem segments and identify the problem. Sometimes problem resolution requires the application of funds (upgrade equipment or links) but not always.

For one site that was using a commodity Internet, only 1 Mbps was regularly seen. The application was changed to locate caching to reduce dependence on that site.

Regular Testing

They found the testing to be very useful in understanding the network status so they established a regular testing schedule. They established a web site for reporting the results and all researchers can check the network status at any time:

<http://web.haystack.mit.edu/staff/dlapsley/tsev7.html>.

When to Use One-Way Ping (OWAMP)

The One-Way Ping tool should be used when you want baseline “heartbeat” information. Asymmetric routes can make problem location more difficult; the One-Way Ping tool can provide detailed performance on one direction in the path. This is also useful when you want to know precise latency information; this is particularly useful for real-time applications.

Why You Would Use One-Way Ping

It is very sensitive to minor network changes, including route changes and packet queuing. Also, it tells you about one-direction of the path, allowing you to spot difficulties in one direction of your round-trip loop. Because this is most useful when data is collected on a regular basis, you also have a long-term baseline, so you can note trends and changes.

Case Study: Queuing on Abilene

A brief background on what we *think* was going on: it was Tuesday, 2004-08-17, from 16:05-16:20 UTC (that’s 11:05 to 11:20 EDT). A path running from Caltech to CERN was performing a 10GE throughput experiment:

- Single adapter to date, PCI-X
- Theoretical limit of ~8.5 Gbps
- Practical limit closer to 7.5 Gbps
- Exactly what was tested at that time is unknown

What we saw: the “Worst 10” delay list had some larger than normal variances. The minimum never varies – typical of a link, unless it is fully congested. The 95th percentile

is rising, and occasionally the 50th percentile is rising too – so a “typical” packet will see some queuing. For research links, we ideally want to keep the 95th percentile equal to the minimum, unless very large tests are being run.

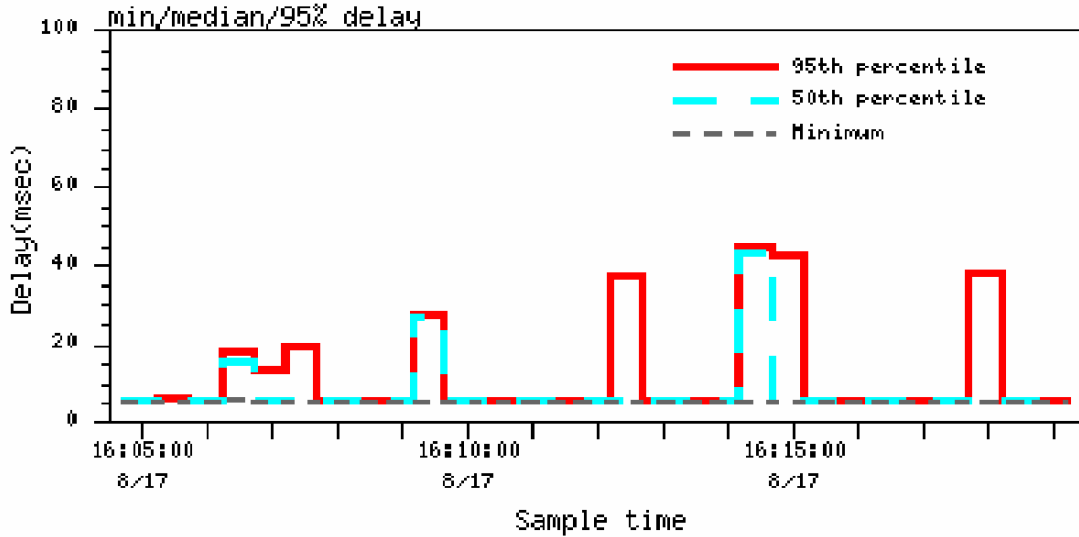


Figure 3. Denver to Kansas City link

What does this show? Only paths that traverse the DNVR>KSCY link showed additional delay during this period; some were delayed by ~ an extra 35msec. Probable cause: a router started queuing packets, which created a small delay. This tells you that there is congestion on the link.

Detailed Tools Discussion

In this section, we will focus on tools specific to four areas:

- First mile, host issues
- Path issues
- Others to be aware of
- Within Abilene

First Mile/Host Issues

There are five specific non-commercial tools that Internet2 recommends for dealing with first mile and host issues:

- Internet2 Detective
- NLANR Performance Advisor
- NDT
- Web100 (host/OS tuning)
- Various Reference Servers (H.323 & Multicast Beacons)

Internet2 Detective:

This is a simple “is there any hope” tool. It is a Windows “tray” application that indicates, by red or green lights, whether or not the user is linked to Internet2. It is multicast and IPv6 available and has links (in the newest version) to NDT and the piPEs project. For more information, see: <http://detective.internet2.edu/>.

NLANR Performance Advisor:

This is designed for the naive user; it is run at both ends, and users can see if a standard problem is detected. This tool can also work with intermediate servers; for more information, see: <http://dast.nlanr.net/Projects/Advisor>.

NDT

The NDT (Network Debugging Tool) is a Java applet that connects to a server in the middle, runs tests, and evaluates heuristics looking for host and first mile problems. It has detailed output that is useful for an end-user to provide to sysadmins as verification of problems. There is an NDT Cookbook available at: <http://e2epi.internet2.edu/network-perf-wk/ndt-cookbook.pdf>.

Web100: Host/OS Tuning

The goal of the Web100 Project was to tune the TCP stack to remove bottlenecks. There is a large measurement component to this work. Is TCP performance not what you expect? You can ask TCP why!

- Receiver bottleneck (out of receiver window)
- Sender bottleneck (no data to send)
- Path bottleneck (out of congestion window)
- Path anomalies (duplicate, out of order, loss)

Web100 is a kernel modification, currently to Linux 2.6 series kernels. There is a TCP MIB draft in the IETF to try and standardize the export-TCP-state part of Web100, and we expect Microsoft and others to pick that up. (Microsoft already has some of the elements in recent Windows server versions). For more information, see: <http://www.web100.org>.

Reference Servers: H.323 Beacons

For H.323 videoconferencing, the goal is to have portable machines that tell you if system likely to work (and if not, why?). Examples of such tools include:

- H.323 Beacon (<http://www.osc.edu/oarnet/itecoho.net/beacon/>)
- ViDeNet Scout (<http://scout.video.unc.edu/>)

Rather than simply using the generic NDT, there are also specific tools for videoconferencing. Moderate-rate UDP is a substitute, but the H.323 Beacon from Ohio State (free) and ViDeNet Scout (uses licensed software) actually run the protocol and capture behavior.

Path issues

The two tools that are discussed in separate cookbooks should be described, briefly, here are One-Way Ping and Bandwidth Test Controller.

One-Way Ping: Latency/Loss

The One-Way Ping tool developed by Internet2 is based on the One-Way Active Measurement Protocol developed by the Internet Engineering Task Force (IETF) as a standard for determining one-way latency and loss. This tool requires a minimum of four NTP-Synchronized clocks, which makes it more useable and useful to NOCs for regularly scheduled tests than to end-users.

The tool looks for one-way latency and loss; it is designed to allow the users to set authentication and scheduling policies to meet their own needs. For more information, see the cookbook, which is available at: <http://e2epi.internet2.edu/network-perf-wk/owamp-cookbook.pdf>.

Bandwidth Test Controller: Throughput

The Bandwidth Test Controller is a tool for throughput testing that includes scheduling and authentication. (It currently uses Iperf for the actual tests.) It can assign users (or IP addresses) to classes, and then give the classes different throughput limits or time limits. This allows the user to allow periodic and on-demand testing without having to provide either user accounts to those wishing to test to the server *or* starting the test in sync with the other end of the test run.

For more information, see the Bandwidth Test Controller Cookbook at: <http://e2epi.internet2.edu/network-perf-wk/bwctl-cookbook.pdf>.

Other options

Other options include both commercial and non-commercial products.

Some Commercial Tools

As a caveat: this is only a partial list, Internet2 is always looking for more!

- Spirent (nee Netcom/Adtech) -- testers that can rigorously evaluate routers (and paths), and work at line rate.
- SmartBits -- test at low & high rates, QoS; test components or end-to-end path
- NetIQ: Chariot/Pegasus -- little drones that you can run with a command and control console. It can simulate some application behavior, and also has a capture then replay ability.
- Agilent (like SmartBits, and FireHunter) -- makes testers like those of Spirent, and also has a product called FireHunter that is used by ISPs. It does things like pings, and FTP fetches, and Web fetches, and can issue alerts when things go out of spec.
- Ixia (like SmartBits/Spirent) -- makes boxes like Agilent and Spirent.
- Brix Networks (like AMP/One-Way Ping, for 'QoS')
- Apparent Networks -- path debugger

Additionally, a commercial tool that tests for TCP buffer problems:
<http://www.dslreports.com/tweaks/>.

Some Noncommercial Tools:

This is also just a partial list: Internet2 is interested in hearing about more!

- Iperf (<http://dast.nlanr.net/Projects/iperf>) – this is what the Bandwidth Test Controller uses to run the tests. See also <http://www-itg.lbl.gov/nettest/> and <http://www-didc.lbl.gov/NCS/>.
- Flowscan – this is a tool to process netflow output and create pretty aggregate graphs. See also <http://www.caida.org/tools/utilities/flowscan/> and <http://net.doit.wisc.edu/~plonka/FlowScan/>.
- There is another set of tools, called “flow-tools” from the Ohio State University (OSU), that Abilene uses. (Note: OSU is an Internet2 technology evaluation center.) See also <http://www.splintered.net/sw/flow-tools/> .
- SLAC’s traceroute perl script; this can be used with a web server to provide traceroutes. See also <http://www.slac.stanford.edu/comp/net/wan-mon/traceroute-srv.html>.
- SLAC compiled a very large list of tools (<http://www.slac.stanford.edu/xorg/nmtf/nmtf-tools.html>).

Tools within Abilene: Measurements from the Center

Abilene measurements include both active and passive efforts. The active tests (latency and throughput) include both measurements within Abilene and measurements to the edge. The passive measurements include:

- SNMP stats (esp. core Abilene links)
- Variables via router proxy
- Router configuration
- Route state
- Characterization of traffic (using Netflow; OCxMON)

Of particular interest is the router proxy. You can give mediated commands to the router to query state. This can be very useful. You can also issue traceroutes and pings from the Abilene routers.

Goal

Why does Abilene take all these measurements, and publish the results? Abilene’s goal is to be an exemplar.

- Measurements are open (nothing up my sleeve!)
- Tests are possible to router nodes
- Throughput tests routinely run through the backbone
- ...as well as existing utilization, etc.
- The “Abilene Observatory” (<http://abilene.internet2.edu/observatory>)

Abilene: Machines

We currently have four machines at each router node. Here are their roles:

- GigE connected high-performance tester for the Bandwidth Test Controller – ‘nms1’ – has a 9000 byte MTU
- Latency tester for One-Way Ping – ‘nms4’ – has 100bT
- Stats collection for SNMP and flow-stats – ‘nms3’ – has 100bT
- Ad-hoc tests, such as are run by the NDT server – ‘nms2’ – are also GigE with a 1500 byte MTU

Throughput

Abilene uses the Bandwidth Test Controller for throughput. We take one test per hour, each for 20 seconds:

- IPv4 TCP
- IPv6 TCP (no discernable difference)
- IPv4 UDP (on our platforms flakey at 1G)
- IPv6 UDP (ditto)

Others test to our nodes and amongst themselves; the net result: 25% of traffic (NOT capacity) is measurement.

Latency

Abilene uses One-Way Ping for latency. CDMA is used to synchronize NTP (<http://www.endruntechnologies.com>). Regularly-scheduled tests for both IPv4 and IPv6 between all router node pairs are 10 per second; these are minimally sized packets on a Poisson schedule.

Passive – Utilization

The Abilene NOC takes passive measurements for:

- Packets in,out
- Bytes in,out
- Drops/Errors
- ..for all interfaces, publishes internal links & peering points (at 5 min intervals)
- ..via SNMP polling – every 60 sec

See also: <http://loadrunner.uits.iu.edu/weathermaps/abilene/abilene.html>.

Abilene Pointers

There are lots of tools at the Abilene NOC page (<http://www.abilene.iu.edu/>). Netflow data is currently at OSU (<http://www.itec.oar.net/abilene-netflow>). We make weekly summaries to try and understand what traffic is passing over the network, and watch for trends (<http://netflow.internet2.edu/weekly/>).

End-to-End Performance Initiative (E2Epi)

One of the major projects of the E2Epi is the piPEs Project. The focus of this effort is to develop an end-to-end measurement infrastructure capable of finding network problems. The tools used by this project include the Bandwidth Test Controller (latency), One-Way Ping (throughput), and NDT (last mile issues). Each of these tools has a cookbook similar to this one. They can all be accessed through <http://e2epi.internet2.edu/library-list.html>.

E2Epi Vision

The E2Epi vision includes ongoing monitoring to test major elements and end-to-end paths. Of particular importance are the set of end-to-end paths that a particular application community cares about (such as nuclear physics site, medical research sites, and radio astronomy sites). There are always many more end-to-end paths than can be monitored so the group has focused on diagnostic tools that are available for on-demand (with proper authorization) testing.

- Elements: gigaPoP links, peering, ...
- Utilization
- Delay
- Loss
- Occasional throughput
- Multicast connectivity

Tools should:

- Show routes
- Perform flow tests (perhaps app tests)
- Parse/debug flows (a-la tcpdump or OCXmon with heuristic tools)

This is not limited to just the United States; Abilene users work with folks in Europe, Asia, and elsewhere. So there must be a way for measurements infrastructures to interoperate. That is one of the areas Internet2 is working on in 2005.

What Campuses Can Do

So, what can you do? A few simple things. Make utilization data available, at least at the edge of your campus. Monitor not only utilization, but things that can cause losses... packet drop and error counters.

Place measurement points at the edge of your campus; this will allow you to test ad-hoc from within your campus to the edge, and allow you to constantly monitor any campus connectivity you think is important. (One possible use for this is just making sure your university-to-university traffic goes over the high-performance network).

Strategy References

See also:

- <http://e2epi.internet2.edu/> (stories, documents, tools)

- <http://e2epi.internet2.edu/ndt/> (pointer to the tool, and using it for debugging the last mile)
- <http://www.psc.edu/networking/projects/tcptune/> (how to tweak OS parameters and an scp pointer)
- <http://www.ncne.org/research/tcp/> (TCP debugging the detailed way)
- <http://dast.nlanr.net/Guides/WritingApps/> (tips for application developers)
- <http://dast.nlanr.net/Guides/GettingStarted> (and some checking to do by hand & debugging)