Experiences Using Web100 for End-To-End Network Performance Tuning

Thomas J. Hacker
Center for Parallel Computing
University of Michigan

Brian D. Athey
Cell & Developmental Biology
University of Michigan

Jason Sommerfield
Pittsburgh Supercomputing Center

Email:{hacker, bleu}@umich.edu, jasons@psc.edu

Abstract:
A major source of performance degradation in high-performance distributed applications has been attributed to poor end-to-end TCP performance. The root causes of poor TCP performance are difficult to isolate and diagnose, and the efficacy of tuning efforts are often difficult to gauge. This paper describes some sources of poor TCP performance, and describes a method to diagnose some of these problems based on a combination of existing performance tools and the Web100 tuning package. Using this methodology, the TCP performance of an application developed for the Visible Human project is shown to significantly improve.

1 Introduction

Distributed application performance problems traceable to poor TCP performance has been identified as a major source of performance degradation in high-performance applications [1]. Appropriately provisioned network infrastructure is essential for providing support for high-performance networking. However, lack of proper host tuning and unexpected levels of packet loss can adversely affect actual end-to-end network performance to such an extent that it can nullify the benefits of network infrastructure investments. Most operating systems are shipped with an overly conservative set of network tuning parameters that can severely degrade aggregate TCP performance on wide area networks. System level effects other than inadequate host tuning can also affect end-to-end performance. For example, software bugs in the implementation of TCP on host systems can contribute to poor TCP performance in very subtle ways [4, 5].

Applications can greatly benefit from application and host tuning efforts targeted at improving aggregate network performance. Measurements of the Edgewarp application [2] written for the Visible Human project at the University of Michigan [3] has shown that the effects of poor host and application tuning can seriously degrade bulk transport performance necessary for delivering images by at least a factor of four.

1.1 Actual TCP Bandwidth Delivered to the Application

If a host and application are properly tuned, effects outside the control of the host and application can adversely affect network performance. Limitations on TCP bandwidth arise from the effects of packet loss and packet round trip time on the network path between hosts. The TCP Slow Start and Congestion Control algorithms [6] probe the network path between the sender and receiver to both discover the maximum available transfer capacity of the network and at the same time minimize the effects of overloading the network and causing congestion.

Mathis [7] described the relationship between the upper bound of TCP bandwidth BW, packet round trip time RTT, and packet loss p with the equation

$$BW \leq \frac{MSS}{RTT}\frac{C}{\sqrt{p}} \qquad (1)$$

To achieve substantial network performance over a wide area network that has a relatively large RTT, the required maximum packet loss rate p must be very low. The relationship derived by Mathis for the maximum packet loss rate required to achieve a target bandwidth is defined by the relationship

$$p < \left(\frac{MSS}{BW * RTT}\right)^2 \qquad (2)$$

For example, if the minimum link bandwidth between two hosts is OC-12 (622 Mbps), and the average round trip time is 20 msec, the maximum packet loss rate necessary to achieve 66% of the link speed (411 Mbps) is approximately 0.00018%, which represents only two packets lost out of every 100000 packets. Current loss rates on the commercial Internet backbone [8] are on the order of 0.1%, which puts a hard upper limit on the potential bandwidth available to an application.

Recent work [28] has demonstrated that packet loss events occur in bursts of consecutive packet losses that may be due to the drop-tail queuing mechanism in routers [29]. As the implementation and use of the Random Early Detection (RED) queuing mechanism

becomes widely deployed across the Internet, this characteristic may change. Given the burst behavior of packet loss, obtaining significantly small packet loss rates can be very difficult. Looking at equation (1), it is apparent that increasing the MTU from the usual default value of 1500 bytes to the "jumbo frame" size of 9000 bytes can increase the upper limit on TCP bandwidth by a factor of six. The recent experience of one of the authors demonstrated that increasing the MTU by a factor of three from 1500 to 4470 bytes increased TCP throughput by a roughly equivalent factor.

1.2 Uncooperative Network Application Behaviors

Application developers have learned to overcome poor TCP performance with a toolkit of "bad" (from the network administrator's perspective) behaviors.

The first approach usually taken is to abandon the TCP transport service and to rely on UDP along with a transport layer written for the application. In this approach, the application simply transmits packets as fast as it can. If any packets are lost, the application either drops them (as in the case of multimedia applications), or performs packet retransmission on an application level. This approach is considered "bad" for several reasons. If an application is injecting UDP packets into the network at a high rate, the network infrastructure has no way of signaling back to the application that the flow is congesting the network and affecting other users of the network. If the other users of the network are being "good" and using TCP for their connection, the UDP stream is able to take an unfair share of the available network bandwidth [9, p. 246].

The second approach is to open parallel TCP network sockets between applications, and utilize software controlled striping of the data across the sockets, similar to disk striping [10]. This approach attempts to take more than the host's normal share of network bandwidth from other users of the network to deliver a higher aggregate network bandwidth to the end hosts. When there is a significant amount of random packet loss experienced by the end hosts that is not due to congestion, parallel TCP sockets can be a fair and effective method to improve aggregate throughput.

1.3 Tuning Methodology and Other Sources of Poor Network Performance

Even if the end hosts are properly tuned and the network packet loss rate is acceptable, other factors may come into play that can adversely affect network performance. Each of these factors should be considered in turn when diagnosing poor network performance, since a fault at a lower layer will affect performance in all of the layers above it.

In the Physical Layer, network cables that are not within specification limits can be a significant source of poor performance. A general rule of thumb is that Cat-5 cables are good for 10-BaseT, Cat-5 enhanced cables (Cat-5e) are good for 100-BaseT, and Cat-6 cables are good for gigabit Ethernet over copper. Network adapters are very good at getting around bad cables by decreasing their throughput or using data-link layer CRC correction to compensate for a cable that is operating below specification. An additional source of problems are host network adapters that are configured to operate at half-duplex mode rather than full-duplex mode. If both the network switch and the network adapters support full-duplex transfers, both sides should be set to full duplex. If excessive losses are encountered in full-duplex mode, the cabling between the host and switch should be tested or replaced.

In the Data Link Layer, there are several potential sources of problems. First, if the maximum transmission unit size (MTU) for packets is set too low, TCP connections will suffer from poor performance. On 10 and 100 Mb/sec Ethernet, all adapter cards enforce a 1500 byte MTU limit. On some Gigabit Ethernet cards, the MTU can be set to a "jumbo size" 9000 byte frame. If we look back at equation (1), it's apparent that increasing the MTU size (which is MSS + IP header) by a factor of six can increase TCP bandwidth by a factor of six! Unfortunately, most network switches and routers have a hard 1500 byte MTU limit that cannot be changed.

On the host side, another source of problems in the Data Link Layer is the number of CPU interrupts per second that are required to service the network adapter. If a transfer is occurring at gigabit Ethernet speeds, with a limited 1500 byte MTU, the network adapter and CPU must service over 83,000 packets per second. If the network adapter requires service from the CPU for a small number of packets, the CPU will be overwhelmed with servicing network adapter interrupts [11]. The device driver must be configured to permit an appropriate degree of packet coalescing to take advantage of the network adapter's packet buffer. Additionally, the size of the transmission queue in the operating system (txqueuelen in Linux) can affect the packet loss rate on the host. Finally, the PCI slot where the network adapter card is placed can have an impact on performance. Some motherboards, such as the Intel L440GX+[40], have a dual PCI bus architecture, with specialized PCI slots that are enhanced for specific functions (such as RAID adapters). If a host contains RAID adapters along with network adapters, improper adapter card placement can have an impact on the aggregate performance of the complete system.

In the Network Layer, there are several potential sources of problems. First, excessive packet loss and round trip time affects TCP bandwidth as described

above in equation (1). Second, there may be network configuration errors that forces traffic through an inadequate data link, or that adds unnecessary additional hops in the path between the hosts. This problem can be especially difficult to diagnose if IP encapsulation (such as AAL5 for IP over ATM) occurs on the network path, since IP based network tools (such as traceroute) do not have the ability to adequately penetrate an ATM cloud to diagnose ATM problems.

In the Transport Layer, mistuned host TCP options are a very common source of problems. Section 3 of this paper will describe some of these options and demonstrate how they can affect TCP performance.

Finally, the network I/O characteristics of the application can dramatically impact TCP performance. Application developers should consider multithreading their applications to decouple network I/O from computation. Chapter 6 of Steven's text [31] is a good starting point for these efforts.

The process of examining the network from the Physical Layer up to the Application Layer represents an orderly methodology that should be followed when attempting to diagnose and correct network performance problems. It is important to note that if a problem exists at a lower layer in the network, such as the physical layer, efforts directed at tuning components at a higher layer to improve performance may not deliver the expected results. For example, if a physical link is improperly configured to operate at half duplex, attempts to increase performance by optimizing the end-to-end network path may yield little if any results. Thus, when diagnosing application network performance problems, it is important to make sure that tuning opportunities at each layer are explored.

1.4 Web100

The remainder of this paper will discuss experiences using Web100 for host and application TCP tuning. Web100 [12] has been used with great success for identifying and diagnosing the symptoms and causes of network performance problems and for immediately measuring the effects of performance tuning. It is hoped that the work described in this paper will be useful to application developers and system administrators for tuning their host systems and improving network and application performance

2 Related Work

The suite of tools that are currently available for measurement and diagnosis focus on specific characteristics of network performance. The tools most frequently used include ping, traceroute, tcpdump, pchar, and Iperf. Tools designed for network specialists include Treno, and TCP testrig. This section will briefly describe each of these tools and how they are currently used for diagnosing network performance problems.

2.1 General Tools

The network measurement tools available to application developers and system administrators are used to measure physical data-link bandwidth, round trip time, loss rate, router buffer sizes at each hop in the network, and measure end-to-end network bandwidth.

The UNIX ping utility is used to transmit and receive ICMP Echo packets to a destination host to determine if the host is reachable, to measure round trip time (RTT), and to measure packet loss on the network path to the host. The RTT measurements made by ping can be used to estimate the "pipe" capacity (capacity = BW * RTT) of the network between two hosts. Since the test load put on the network by ping consists of small periodic ICMP packets, the packet loss rate measured by ping is not very useful for estimating available TCP bandwidth using equation (1). The RTT measurement, however, is useful for deriving the maximum packet loss rate necessary to support a desired TCP bandwidth in equation (2).

Traceroute [16] is used to discover the IP network route between two hosts and the RTT to each hop in the network route. Traceroute is used to diagnose routing problems between hosts.

Tcpdump [17] is a packet capturing and display utility that displays all packets on a network segment connected to a network adapter that is configured in "promiscuous mode". Tcpdump is used to debug network protocols and to passively monitor the network traffic on a LAN segment.

Pchar[13] is a tool that measures bandwidth, round trip time, and router buffer space on every data link on the network path between two hosts. Pchar is used to diagnose and identify data link bottlenecks in the network path between two hosts. Pipechar [14] is another tool that can be used in conjunction with Pchar to further examine network bottleneck characteristics.

Iperf [15] is a tool that measures TCP and UDP transfer rates between host pairs. Iperf is used to estimate the maximum network bandwidth available to an application and to investigate the relationship between UDP packet injection rate and packet loss on a network between two hosts.

There are many projects working on designing and developing user-level network bandwidth prediction and management tools. These projects include Network Weather Service [18], NetLogger [19], and Gloperf [20]. A complete list of network measurement tools can be found at the NLANR website (http://ncne.nlanr.net/software/tools/).

2.2 Network Specialist's Tools

A small set of end-to-end performance measurement tools, such as Treno [21] and TCP Testrig [22] are available, but the use of these tools requires an extensive knowledge of networking and the characteristics of TCP along with privileged access to network devices in the host operating system.

Treno is a tool that performs a single stream transfer over a simulated TCP connection to diagnose TCP performance problems. TCP Testrig is a TCP test harness that is used in combination with tcptrace, xplot, tcpdump, and a TCP debugging flowchart [23] to aid specialists in characterizing and diagnosing TCP tuning problems.

Both Treno and TCP Testrig require users to have an in-depth knowledge of the TCP protocol and network characteristics to realize maximum results.

3 Web100

An application developer or systems administrator can make use of a combination of these tools to diagnose and correct host and application network problems, but there are inherent problems with the measurement methodologies within each tool that must be taken into account.

First, to make a fair estimate of the characteristics of the system under measurement, many measurements and data points must be collected, and systematic sources of error (such as time of day) need to be taken into account to eliminate artificial effects. Second, some of the tools (pchar, for example) require such a long time to run that the results of the measurement may not accurately reflect the current state of the system under measurement. Third, some components of the network path (such as switched ATM clouds) are resistant to IP based measurement techniques. Finally, a high degree of expertise in networking and operating systems is required to realize the full benefits from the use of these tools.

To address these problems, Web100 [12] was developed by a team at Pittsburgh Supercomputing Center to provide a window into the characteristics of a TCP connection for application developers and systems administrators, and to provide an integrated performance measurement and diagnosis tool. Web100 provides kernel level access to internal TCP protocol variables, settings, and performance characteristics for instantaneous feedback on TCP performance characteristics.

The current implementation of Web100 consists of two major components. The first component is the set of Linux kernel modifications that export TCP measurements, variables, and settings through the Linux '/proc' interface. The second major component of Web100 is the graphical user tool, Diagnostic Tool Builder (dtb), which provides an interface to the Web100 TCP instrumentation in the form of numerical displays, bar graphs, and pie charts of the data values provided by Web100.

3.1 Web100 and the Visible Human Project

The Visible Human project at the University of Michigan [3, 24] is a data and visualization intensive Grid computing project that is designed to deliver volumetric three-dimensional rendered human anatomy images along with pedagogical content to students at teaching hospitals and medical centers across the nation. The goal of the Visible Human project is to support the simultaneous access of content through the Internet by 40 teaching stations for each class session. There may be many of these training sessions occurring simultaneously throughout the nation. To deliver these services interactively, the Visible Human project will require guaranteed end-to-end network performance (Quality of Service reservation and provision) along with high performance data delivery and volume rendering systems.

The edgewarp [2, 32] application was developed in conjunction with the Visible Human project to retrieve image voxels from a server and to render the resulting anatomical images. To improve the performance of edgewarp along with other applications developed to support the Visible Human Project (VHP), Web100 was used along with a toolset consisting of ping, traceroute, pchar, and Iperf to improve the network performance of VHP data servers and applications [25]. The features of Web100 that proved to be most useful for tuning were the real-time measurements of data bytes transmitted, packet retransmission, receiver TCP window size, and the display of TCP options negotiated by the sender and receiver on connection establishment.

4 Using Web100 to Tune End-to-end Performance

To test the ability of the voxel server to deliver voxels to a client application, a test rig that connects to the Pittsburgh Supercomputing Center Visible Human voxel server was developed by the VHP development group to simulate the retrieval characteristics of the edgewarp browser. The results of the test rig were analyzed in combination with the use of Iperf, pchar and pipechar to determine if there were any network performance bottlenecks in the network path between the voxel server at Pittsburgh Supercomputing Center and the edgewarp client at the University of Michigan in Ann Arbor.

Pchar indicated that the structural bottleneck in the network path between PSC and U-M was a 100 Mb/sec connection to the edgewarp client at U-M. Traceroute,

pchar and pipechar indicated that the network path (other than the 100 Mb/sec client connection) consisted of a combination of OC-12 and Gigabit Ethernet data links, with approximately 7 network hops between the client and server. The voxel server ran on a Compaq ES-40 server with a Netgear GA620 Gigabit Ethernet network adapter. The edgewarp client was configured as an Intel 500 Mhz processor Linux host with an Elsa NVIDIA graphics card used to render edgewarp graphics on the user's desktop.

Once the structural properties of the network path were determined, the ability of the EdgeWarp server to deliver data from PSC to U-M was tested. Web100 was used on the server at PSC to monitor in real-time the characteristics of the TCP transfer.

The next section will show the transfer characteristics of a mistuned Linux host. The following section will describe the application of the network tuning methodology described earlier along with the results of the tuning efforts measured using Web100.

4.1 Characteristics of a Mistuned TCP Receiver

Figure 1 shows the initial Web100 window running on the PSC VH server. This window provides an interface to allow a user to select a TCP connection of interest, and to display and modify TCP variables of interest.

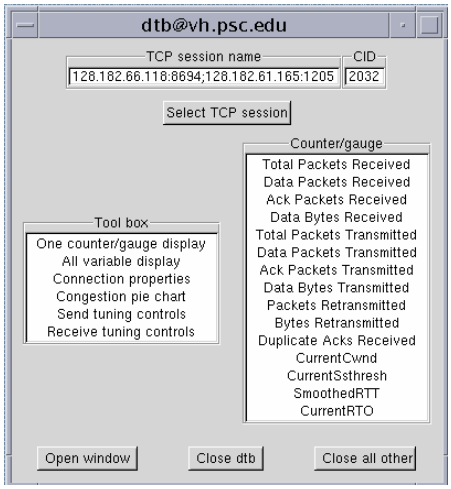


Figure 1. Web100 Main Window

The TCP session between the voxel server at PSC and the test rig client U-M was selected in the Web100 main window.

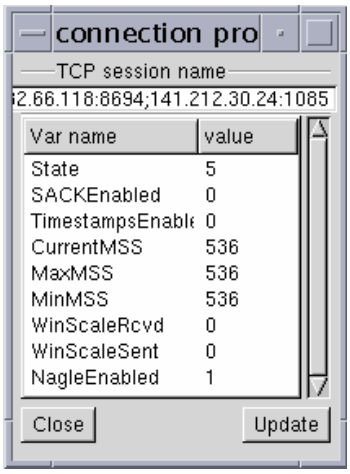


Figure 2. TCP Connection Properties for an EdgeWarp Data Transfer to a Mistuned Host

Figure 2 shows the TCP connection properties for the session. Examination of the contents of this window indicates that several TCP options are disabled. First, Selective Acknowledgement (SACK) [26], which is critical for good TCP performance in networks with packet loss, is disabled. Second, the maximum segment size (MSS) is very small. Normally over an Ethernet LAN, the maximum frame size may be up to 1500 bytes. If the TCP MSS is set smaller than the network and host can actually support, the large number of packets that must be processed (relative to the potentially smaller number if MSS is set properly) creates an additional overhead that can degrade performance. Finally, timestamps and window scaling options are not used. These options are described in RFC1323 [27] and are critical for high-speed TCP connections.

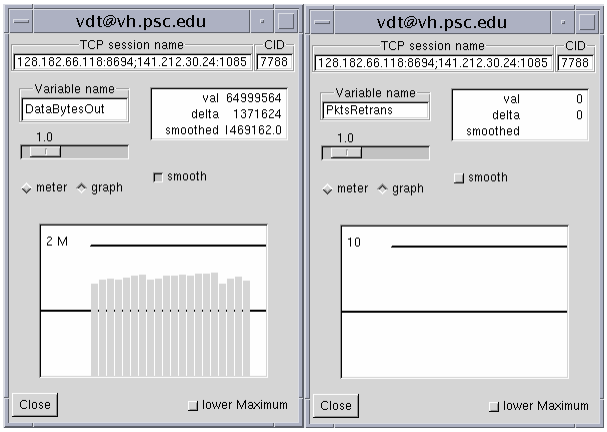


Figure 3. TCP Transfer Rate and Packet Loss Rate for EdgeWarp Data Transfer to a Mistuned Host

After examining the TCP connection properties, the "Data Bytes Transmitted" and "Packets Retransmitted" options were selected from the right panel in the Web100 window. Figure three shows the data transfer rate in units of bytes transmitted per second in the left window, and the packet loss rate in units of packets retransmitted per second in the right window.

The bar graph in the left window of figure 3 indicates that the data transfer rate is approximately 1.5 MB/sec, and the right window indicates little or no packet loss.

The "All variable display" in the left panel of the main Web100 window shown in figure 1 opens to display all of the variables maintained by Web100 in real-time for the duration of the connection. Two items of interest from this window are the CurrentCwnd and CurrentRwinRcv variables.

TCP uses the minimum of the advertised TCP receiver window size and the calculated congestion window to determine the maximum number of outstanding unacknowledged TCP segments allowed in the network. If the TCP receiver window size is larger than the pipe capacity of the network link between the sender and receiver, TCP will attempt to "probe" the maximum capacity of the network link to carry traffic by increasing the congestion window (and the corresponding number of "in flight" unacknowledged segments) until the network indicates that it is congested by dropping a packet.

In Figures 1 through 3, it is apparent that no packets are being dropped. Using Web100, the congestion window size (CurrentCwnd) was observed to be very large compared with the receiver window size. Given this observation, the number of outstanding segments is limited by the TCP receiver window size, not the maximum capacity of the network. Thus, an inappropriately sized TCP window on the receiver limits the performance of TCP at this point.

4.2 Tuning the Network and Hosts

Prior to host tuning efforts, the network connection between the client host adapter and the network switch was checked to ensure that the Data Link Layer was operating in full-duplex mode. The maximum transmission unit size (frame size) on the local network switch was then checked to ensure that it was configured to support a maximum transmission unit size (MTU) of at least 1500 bytes. Finally, the network cable was checked to verify that it was a Category-5 enhanced cable.

After the Physical and Data Link characteristics of the local network infrastructure were validated, the host tuning problems described in the previous section were addressed. The client host was tuned to support SACK, MTU discovery, Timestamps, and Window Scaling. The TCP maximum and default send and receive socket buffer, which is used by TCP to determine the receiver window size, were set to 2 MB. The server was checked to ensure that these options were enabled.

Note that setting the TCP send and receive socket buffer sizes to a large value may possibly have negative effects on the overall performance of the host in several instances. First, if the host manages a large number of TCP connections, (a webserver, for example) each TCP connection could potentially request 2 MB of socket buffer. This could easily consume the memory on a host may have a limited amount of memory. Second, it is possible to use the TCP socket buffer to control the maximum TCP bandwidth each connection can use in a local area network context the by configuring the TCP send and receive socket buffer sizes to a small maximum value. With large maximum TCP buffer sizes, each connection is allowed to fill the local network between two hosts, and other hosts on the LAN may experience congestion on the local area network. In the past, before the deployment of SACK, this could possibly lead to congestion collapse of the network, since a large percentage of the packets would be retransmitted packets. SACK, however, alleviates this problem.

Finally, note that with higher network throughput and a larger number of network packets that must be processed, the host network adapter on both the sender and receiver could demand a significantly larger percentage of the CPU to handle adapter interrupts. This could adversely affect application performance if the application is computationally intensive.

After checking the network, server host and tuning the edgewarp client host, the transfer test was retried. The output of tcpdump for the first two packets in the connection shows immediate results:

```
# /usr/sbin/tcpdump port 8694
Kernel filter, protocol ALL, datagram
packet socket tcpdump: listening on
all devices

19:07:26.172433 eth1 >
spbuild.engin.umich.edu.1088 >
vh.psc.edu.8694: S
1067517561:1067517561(0) win 32758
<mss 1460,sackOK,timestamp 29833739
0,nop,wscale 5> (DF)

19:07:26.192439 eth1 <
vh.psc.edu.8694 >
spbuild.engin.umich.edu.1088: S
1021853801:1021853801(0) ack
1067517562 win 4060 <mss
1460,sackOK,timestamp 1073113995
29833739,nop,wscale 5> (DF)
```

Both sides of the TCP connection have now agreed on the use of SACK, timestamps, an MSS value of 1460 bytes, and window scale. Figure 4 shows the effect of these changes on the TCP connection property information provided by Web100.

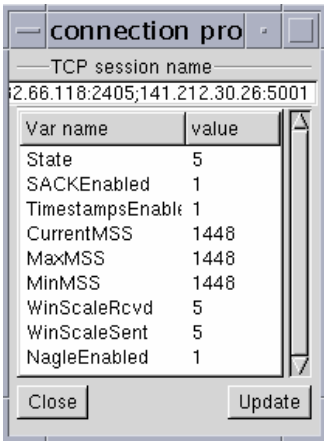


Figure 4. Effects of Host Tuning on TCP Client Settings

Figure 5 shows the effects of host tuning on the data transmission rate and on packet loss. The data transmission rate has increased from 1.5 MB/sec in the mistuned case to 5 MB/sec. Packet loss is now occurring, which indicates that the network is experiencing either congestion or random packet loss.

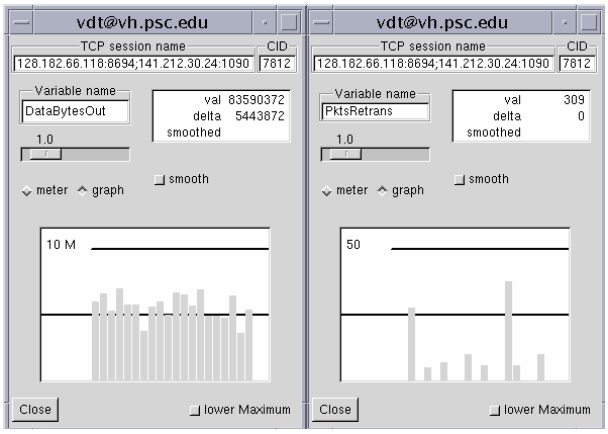
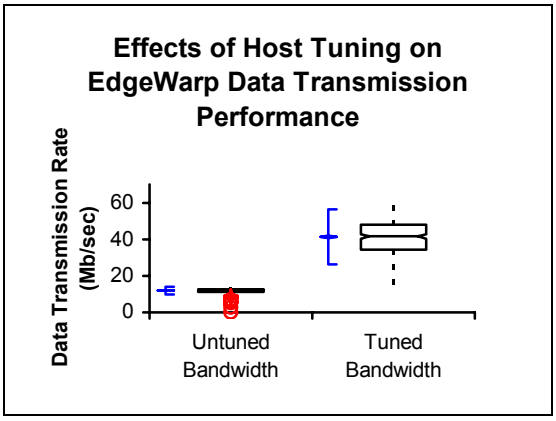


Figure 5.
TCP Transfer Rate and Packet Loss Rate on Tuned Host

Recall that the structural bottleneck in the network path between the server and client is a 100 Mb/sec link. The reasonable maximum bandwidth one could hope to achieve on this link would be approximately 80 Mb/sec.

Figure 6 shows a series of 600 measurements of the edgewarp test rig prior to and after host tuning. The results indicate that by simply changing a few TCP tuning parameters on the host, TCP performance was increased by a factor of four.



	n	SD	95% CI of Mean	Median
Mistuned Bandwidth	601	1.3121	11.728 to 11.938	12.395
Tuned Bandwidth	601	9.0751	40.613 to 42.067	41.578

Figure 6. Effects of Host Tuning on EdgeWarp Testrig Performance

5.0 Conclusions and Future Work

This paper demonstrated that Web100 can be effectively used in combination with network tuning and the suite of network performance tools currently available to identify structural and host tuning problems that can adversely effect end-to-end TCP performance. Web100 can be thought of in some respects as a TCP "oscilloscope" that provides a real time window into the characteristics of the TCP protocol that directly affect TCP performance.

To improve the performance of the application beyond the results presented, several approaches can be taken. First, a thorough examination of the characteristics of the application should be performed to discover any tuning opportunities. Second, attempts should be made in concert with Network Administrators to determine if the MTU of the network path between the server and client can be increased. Finally, an investigation of the sources of packet loss for reasons other than congestion will be undertaken. Potential sources of packet loss include operating system implementation errors, improperly configured network equipment, and all of the other sources mentioned in section 1.3. If the network bottleneck proves to be

uncongested the use of parallel TCP connections to improve throughput should be investigated.

The use of Web100 along with the other tools mentioned will be critical in assessing the impacts of these tuning efforts.

Acknowledgements

The work described in this paper would not have been possible without the help of many individuals. Anjana Kar at PSC and the Web100 development team provided help in the building and installation of the Web100 software. Bill Green and Art Wetzel of PSC developed the edgewarp client, voxel server and test rig. Finally, Matt Mathis at PSC provided patient tutoring on TCP network tuning and using Web100.

REFERENCES

[1] R. Hobby, Internet2 End-to-End Performance Initiative. http://www.internet2.edu/e2eperf/papers/End-to-End-Perf-Design-Paper.pdf

[2] F. L. Bookstein, W. D. K. Green, "Edgewarp 3D: A Preliminary Manual", Posted to the Internet as ftp://brainmap.med.umich.edu/pub/edgewarp3.1/manual.html, 1998.

[3] University of Michigan Visible Human Project. http://vhp.med.umich.edu.

[4] V. Paxson, M. Allman, S. Dawson, W. Fenner, J. Griner, I. Heavens, K. Lahey, J. Semke, B. Volz, Known TCP Implementation Problems, RFC 2525, Informational, March 1999. URL *ftp://ftp.isi.edu/in-notes/rfc2525.txt*.

[5] B. Tierney, "Information on critical Linux TCP bug for high-speed WAN applications". http://www-didc.lbl.gov/Linux-tcp-bug.html. December 2000.

[6] V. Jacobson, "Congestion Avoidance and Control." Proceedings of ACM SIGCOMM '88. ftp://ftp.ee.lbl.gov/papers/congavoid.ps.Z

[7] M. Mathis, J. Semke, J. Mahdavi, T. Ott, "The Macroscopic Behavior of the TCP Congestion Avoidance Algorithm." Computer Communication Review, volume 27, number3, July 1997.

[8] Marix.net Internet Ratings. http://ratings.miq.net/

[9] J.F. Kurose, K.W. Ross: Computer Networking: A Top-Down Approach Featuring the Internet, Addison-Wesley, 2001.

[10] H. Sivakumar, S. Bailey, R. L. Grossman, "PSockets: The Case for Application-level Network Striping for Data Intensive Applications using High Speed Wide Area Networks", Proceedings of Supercomputing 2000, IEEE.

[11] J. Sorensen, "Alteon AceNIC / 3Com 3C985 / NetGear GA620 Gigabit Ethernet Adapter ", http://jes.home.cern.ch/jes/gige/acenic.html

[12] Web100 Project. http://www.web100.org.

[13] B. Mah, "pchar: A tool for measuring internet path characteristics," http://www.employees.org/bmah/Software/pchar/.

[14] J. Goujun. "Methods for Network Analysis and Troubleshooting" http://www-didc.lbl.gov/~jin/network/net-tools.html

[15] M. Gates, A. Warshavsky, Iperf version 1.1.1, Bandwidth Testing Tool, NLANR Applications, February 2000.

[16] V. Jacobson, "Traceroute: A tool for printing the route packets take to a network host", available from ftp.ee.lbl.gov/nrg.html.

[17] V. Jacobson, C. Leres, S. McCanne, tcpdump, available at ftp://ftp.ee.lbl.gov/tcpdump.tar.Z.

[18] M. Faerman, A. Su, R. Wolski, and F. Berman. "Adaptive Performance Prediction for Distributed Data-Intensive Applications." In Proceedings of Supercomputing 1999. IEEE Computer Society Press, 1999. Available at http://www.cs.ucsd.edu/groups/hpcl/apples/hetpubs.Html

[19] B. Tierney, W. Johnston, B. Crowley, G. Hoo, C. Brooks, D. Gunter, "The NetLogger Methodology for High Performance Distributed Systems Performance Analysis", Proceeding of IEEE High Performance Distributed Computing conference, July 1998, LBNL-42611. http://www-didc.lbl.gov/NetLogger/

[20] C. Lee, J. Stepanek, R. Wolski, C. Kesselman, I. Foster, "A Network Performance Tool for Grid Environments", in Proceedings of 7th IEEE International Symposium on High Performance Distributed Computing, pp. 260--267, 1998

[21] Pittsburgh Supercomputer Center. "About the PSC Treno Server." Available at http://www.psc.edu/pscnoc/treno info.html., November 1995.

[22] NLANR Engineering Services. "A Preconfigured TCP test rig" http://www.ncne.nlanr.net/research/tcp/testrig/

[23] NLANR Engineering Services. "TCP Trace Based Performance Diagnosis Flowchart" http://www.ncne.nlanr.net/research/tcp/debugging/

[24] M. J. Ackerman, "The Visible Human Project," J. Biocomm., vol. 18, p 14, 1991.

[25] PSC News Release. "Web100 Takes First Step Towards Improving Network Performance" March, 2001. http://www.psc.edu/publicinfo/news/2001/web100-03-19-01.html

[26] M. Mathis, J. Mahdavi, S. Floyd, A. Romanow, "TCP Selective Acknowledgement Options. RFC 2018, Proposed Standard, April 1996." URL *ftp://ftp.isi.edu/in-notes/rfc2018.txt*

[27] V. Jacobson, R. Braden, D. Borman, "RFC1323: TCP Extensions for High Performance", May 1992

[28] M. S. Borella, D. Swider, S. Uludag, G. Brewster, "Internet Packet Loss: Measurement and Implications

for End-to-End QoS," Proceedings, International Conference on Parallel Processing, Aug. 1998.
[29] Braden, B., Clark, D., Crowcroft, J., Davie, B., Deering, S., Estrin, D., Floyd, S., Jacobson, V., Minshall, G., Partridge, C., Peterson, L., Ramakrishnan, K., Shenker, S., Wroclawski, J., and L. Zhang, "Recommendations on Queue Management and Congestion Avoidance in the Internet.", RFC 2309, April 1998.
[30] Intel Corporation. Intel L440GX+ Server Board Product Guide. http://support.intel.com/support/motherboards/server/l440gx/pg.htm
[31] W.R. Stevens. Unix Network Programming Volume 1: Networking APIs: Sockets and XTI, 2nd Edition. Englewood Cliffs, New Jersey. Prentice-Hall 1997.
[32] B. D. Athey, A. W. Wetzel, and W. D. K. Green. "Navigating solid medical images by pencils of sectioning planes", Pp. 63--76 in Mathematical Modeling, Estimation, and Imaging, eds. D. Wilson, H. Tagare, F. Bookstein, F. Preteaux, and E. Dougherty, Proc. SPIE, vol. 4121, 2000.