SANDIA REPORT

SAND 2004-4207
Unlimited Release
Printed September 2004

Evaluation of Replacement Protocols and Modifications to TCP to Enhance ASC Wide Area Network Performance

Randy L. Romero Jr.

Prepared by
Sandia National Laboratories
Albuquerque, New Mexico 87185 and Livermore, California 94550

Sandia is a multiprogram laboratory operated by Sandia Corporation,
a Lockheed Martin Company, for the United States Department of Energy's
National Nuclear Security Administration under Contract DE-AC04-94AL85000.

Issued by Sandia National Laboratories, operated for the United States Department of Energy by Sandia Corporation.

Available to DOE and DOE contractors from
U.S. Department of Energy
Office of Scientific and Technical Information
P.O. Box 62
Oak Ridge, TN 37831

Telephone: (865)576-8401
Facsimile: (865)576-5728
E-Mail: reports@adonis.osti.gov
Online ordering: http://www.osti.gov/bridge

Available to the public from
U.S. Department of Commerce
National Technical Information Service
5285 Port Royal Rd
Springfield, VA 22161

Telephone: (800)553-6847
Facsimile: (703)605-6900
E-Mail: orders@ntis.fedworld.gov
Online order: http://www.ntis.gov/help/ordermethods.asp?loc=7-4-0#online

SAND2004-4207
Unlimited Release
Printed September 2004

Evaluation of Replacement Protocols and Modifications to TCP to Enhance ASC Wide Area Network Performance

Randy L. Romero Jr.
Advanced Networking Integration Department
Sandia National Laboratories
P.O. Box 5800
Albuquerque, New Mexico 87185-0806

Abstract

Historically, TCP/IP has been the protocol suite used to transfer data throughout the Advanced Simulation and Computing (ASC) community. However, TCP was developed many years ago for an environment very different from the ASC Wide Area Network (WAN) of today. There have been numerous publications that hint of better performance if modifications were made to the TCP algorithms or a different protocol was used to transfer data across a high bandwidth, high delay WAN. Since Sandia National Laboratories wants to maximize the ASC WAN performance to support the Thor's Hammer supercomputer, there is strong interest in evaluating modifications to the TCP protocol and in evaluating alternatives to TCP, such as SCTP, to determine if they provide improved performance. Therefore, the goal of this project is to test, evaluate, compare, and report protocol technologies that enhance the performance of the ASC WAN.

Sandia is a multiprogram laboratory operated by Sandia Corporation, a Lockheed Martin Company for the United States Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000.

This page intentionally left blank

Table of Contents

Figures

Tables

Background

The Advanced Simulation and Computing Program (ASC) is an activity of the National Nuclear Security Administration (NNSA), which relies on collaboration among Lawrence Livermore, Los Alamos, and Sandia National Laboratories. The ASC program seeks to ensure the safety and reliability of the nation's nuclear weapons stockpile. The ASC was established in 1995 to help shift from test-based confidence to simulation-based confidence. The mission of the ASC is to analyze and predict the performance, safety, and reliability of nuclear weapons and certify their functionality [1]. The three laboratories have developed long-distance computing capabilities to share large amounts of data.

Transfer Control Protocol/Internet Protocol, commonly referred to as TCP/IP, is a set of protocols developed to allow cooperating computers to communicate and share resources across a set of interconnected networks [2].

Network congestion occurs when a machine (computer, router, switch, etc) receives data faster than it can process. Network congestion such as this leads to dropped packets. In connection oriented protocols, such as TCP, the destination machine does not receive the dropped packet, so no acknowledgement (ACK) is sent to the sender. The sender then retransmits the lost packet to the receiver, therefore leading to more network congestion. In the late 80's, network congestion was a huge factor for several collapses to the Internet.

TCP was first standardized in 1980 and did not include any congestion control mechanisms [4]. TCP congestion control was created to allow TCP connections to recover from a lost packet. A congestion window was implemented in the control mechanism to limit the amount of outstanding data that a connection have during sending and receiving. These are known as unacknowledged packets. To avoid overflowing the buffer, TCP sets a Window Size field in each packet it transmits. This field contains the amount of data that may be transmitted into the buffer. If this number falls to zero, the remote TCP can send no more data. It must wait until buffer space becomes available and it receives a packet announcing a non-zero window size [5].

There are several TCP congestion control algorithms used today. The three national laboratories, Sandia, Los Alamos, and Lawrence Livermore, are currently connected using a 2.5 Gbps network that allows computing resources to share large amounts of data over long distances. These circumstances create a very challenging environment that can lead to network congestion with very poor recovery performance due to their large distances apart. With this high speed, long delay network, an alternative congestion control algorithm could provide better utilization of the channel between laboratories.

The Red Storm Challenge

Since Sandia National Laboratories wants to maximize the ASC WAN performance to support the Thor's Hammer supercomputer, there is strong interest in evaluating modifications to the TCP protocol and in evaluating alternatives to TCP to determine if they provide improved performance. Therefore, the goal is to test, evaluate, compare, and report protocol technologies that enhance the performance of the ASC WAN. The alternative TCP congestion control algorithms currently being tested are called High-Speed TCP and Scalable TCP. To provoke congestion, High Speed TCP and Scalable TCP is compared to Standard TCP in a test where a client machine sends data at 10 Gbps to a server machine, which can only receive data at 1 Gbps over a network that has a round trip time of 30 msec. An illustration of the Red Storm Challenge is shown in Figure 1.

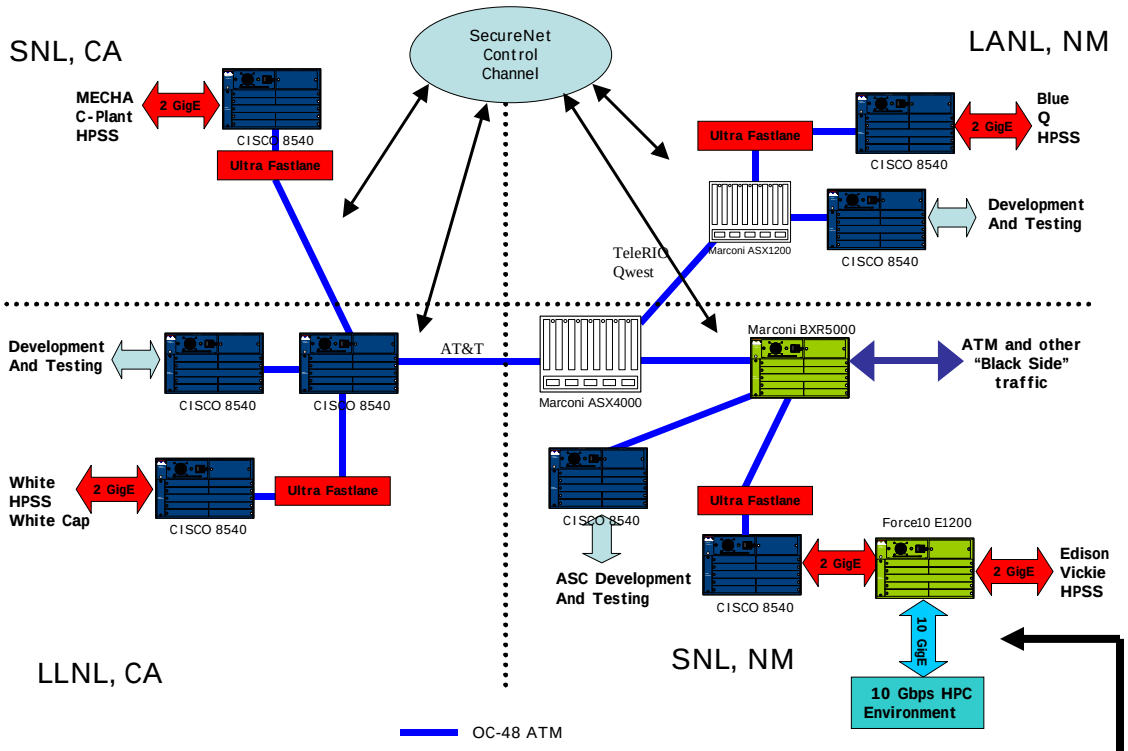

Figure 1: Red Storm Challenge

The Challenge
Interfacing the 10 Gbps HPC to the slower WAN

Introduction

Standard TCP

Standard TCP uses the congestion control algorithms described in RFC2581 [6]. For definitions, equations, and extensive research on the Standard TCP control algorithm, see the SAND2003-4404 document [3]. The SAND2003-4404 document demonstrates the evaluation of the Standard TCP algorithms used such as: slow start, congestion avoidance, fast retransmit, and fast recovery. A TCP connection is always using one of these four algorithms throughout the life of the connection.

Round-Trip Time (RTT)

The time required for a network communication to travel from the source (Client) to the destination (Server) and back (Client) is known as round-trip time (RTT). When a host (client) transmits a TCP packet to its peer (server), it must wait a period of time for an acknowledgment. If the reply does not come within the expected period, the packet is assumed to have been lost and the data is retransmitted. The time taken for a reply should be no more than a few milliseconds. On the Internet, sending and receiving data quickly is very important especially during long delay networks that are hundreds of miles apart. All TCP implementations will eventually drop packets and retransmit them no matter how good the quality of the connection is. The round trip time estimated from Sandia National Laboratory to Lawrence Livermore is 30ms. This is a large delay and can lead to large amounts of time waited for data lost.

Outstanding Data

For tests done on the Standard TCP connection the maximum outstanding data possible given by the bandwidth-delay product is approximately 3.75 MB. This is realized using a 30ms delay for the round-trip time (RTT). The bandwidth-delay product is the bandwidth multiplied by the delay; 1Gbps * 30ms = 30 Mbits or approximately 3.75 MBytes. The bandwidth-delay product equals the number of bytes it takes to fill the channel that you are using.

High-Speed TCP

High-Speed TCP was proposed by Sally Floyd as a sender-side alternative congestion control algorithm [4]. High-Speed TCP attempts to improve the performance of TCP connections with large congestion windows. A goal of High-Speed TCP is to behave similarly to Standard TCP when using small congestion windows. For definitions, equations, and extensive research on the High-Speed TCP control algorithm, see the SAND2003-4404 document [3].

Scalable TCP

Scalable TCP was proposed by Tom Kelly as another alternative sender-side congestion control algorithm [14]. The goal of Scalable TCP is to quickly recover from short congestion periods. Scalable TCP is a simple change to the traditional TCP congestion control algorithm (RFC2581), which dramatically improves TCP performance in high-speed wide area networks [13]. For definitions, equations, and extensive research on the Scalable TCP control algorithm, see the SAND2003-4404 document [3].

Lab Experiments

Test Environment

The test environment consisted of several Dell Power Edge™ 2650 servers. Each system had dual 2.8 GHz Intel® Xeon™ processors, 1GB of memory, and Intel® PRO/1000 Network Interface Cards (NICs). The systems were connected through a Foundry MG8. The operating system used in the test environment was Red Hat Linux. Separate kernels were built for each congestion control algorithm. The kernel used for Standard TCP was a standard linux-2.4.20-SingleProcessor. High-Speed TCP used a linux-2.4.19-hstcp-SingleProcessor kernel; downloaded from [10]. Scalable TCP used a linux-2.4.19-scale0.5.1-SingleProcessor kernel. The Spirent Communications Adtech AX/4000™ was used to simulate delays in the network.

The hardware used was a Spirent Communications Adtech AX/4000™. The Adtech AX/4000™ was used to simulate delays in the network. The test setup to install and evaluate replacement protocols or modifications to TCP to enhance ASC wide area network performance is provided with complete detail. There are several utility programs that need to be installed in order to create a Linux-like environment for Windows. The software programs used to test network performance, full function emulation, and plotting were:

Iperf for Linux libc 2.3

Iperf is a tool to measure maximum TCP bandwidth, allowing the tuning of various parameters and UDP characteristics. Iperf measures the maximum bandwidth by performing memory-to-memory data transfers.

Tcpdump

Tcpdump captures the headers of packets on a network interface that match a Boolean expression for later analysis.

Tera Term Web 3.1.3

TeraTerm Pro Web is the next generation tool for connecting with remote Telnet and SSH hosts.

Cygwin/X

Cygwin is a Linux-like environment emulator for Windows.

WinSCP 3.6.1
WinSCP is an open source SFTP (SSH File Transfer Protocol) and SCP (Secure CoPy) client for Windows using SSH (Secure SHell). Its main function is safe copying of files between a local and a remote computer.

WinPcap 3.0
WinPcap is an architecture for packet capture and network analysis for the Win32 platforms.

Tcptrace v6.6.0 for Windows
*Note: Please make sure to have a working installation of WinPcap before installing tcptrace. Tcptrace is a tool for analyzing TCP dump files.

Xplot
Compiled for Windows. You will need to install an XServer for Windows, like Cygwin/X. Xplot reads from a text file (for example, xplotFile.xpl) to generate its plots.

Xpl2gpl
Xpl2gpl is a utility that converters tcptrace-style xplot input to gnuplot input. This converter gives a nearly perfect gnuplot reproduction of the input xplot graph.

GnuPlot 4.0.0
Gnuplot is a portable command-line driven interactive datafile (text or binary) and function plotting utility for UNIX, IBM OS/2, MS Windows, DOS, Apple Macintosh, VMS, Atari and many other platforms.

Kernel Build Problems
There were multiple repeated errors that kept on occurring during the bootup process after building a Linux kernel to run on the Dual Processor Opteron. An attempt to boot Linux kernel versions 2.4.22, 2.4.23, 2.6.5, and 2.6.6, resulted in errors on bootup such as a getting hung up in a blank screen, couldn't find valid RAM disk image, and a VFS message. Booting the 2.4.22 kernel resulted in a kernel panic error, VFS: Unable to mount root fs. After several failed attempts it was determined that the Linux kernel versions 2.4.xx wasn't new enough to work on the Dual Processor Opteron. A build attempt for a kernel 2.6.6 was made on the Opteron since this kernel is the latest version on Linux. Booting the 2.6.6 kernel resulted in a kernel panic error, Initiating IOC0 recovery. Future work can be done on the installation of a new kernel version for the Dual Processor Opterons since it includes the features of High-Speed TCP and SCTP.

Network Setup 1
The 10 Gbps to 1 Gbps test was performed on the Foundry interconnect where a client machine sends data at 10 Gbps to a server machine, which can only

receive data at 1 Gbps over a network that has a round trip time of 30 msec. The setup used for network testing is shown in Figure 2.

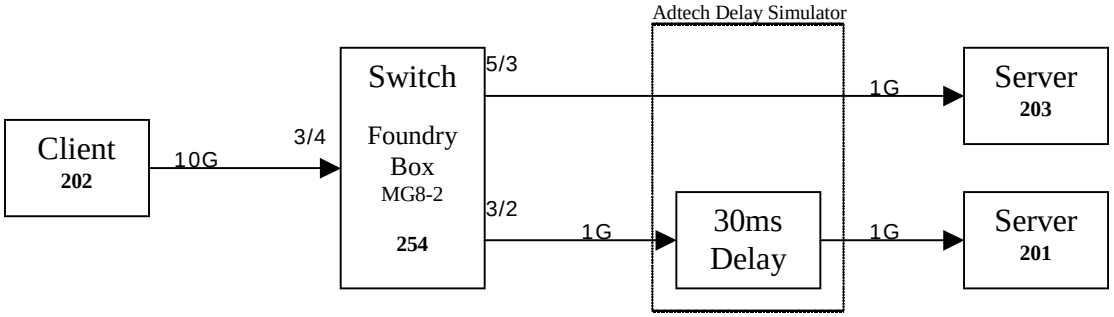

Figure 2: Network Setup Diagram 1

Tuning parameters 1

The following tuning parameters were used on both senders and receivers.

```
echo 1 >/proc/sys/net/ipv4/tcp_timestamps
echo 1 >/proc/sys/net/ipv4/tcp_window_scaling
echo 1 >/proc/sys/net/ipv4/tcp_sack
echo 8388608 > /proc/sys/net/core/wmem_max
echo 8388608 > /proc/sys/net/core/rmem_max
echo "4096 87380 4194304" >/proc/sys/net/ipv4/tcp_rmem
echo "4096 87380 4194304" >/proc/sys/net/ipv4/tcp_wmem
ifconfig eth3 mtu 9000
ifconfig eth3 txqueuelen 1000
#ifconfig eth2 down
#rmmod ixgb
#insmod ixgb
#ifup eth2
```

The window sizes were chosen based on the bandwidth-delay product of 3.75MB. The following command was also run before each test was executed.

```
sysctl -w net.ipv4.route.flush=1
```

Flushing the route resets the slow start threshold (ssthresh) for all connections. This needed to be done to allow for repeatable test results. Iperf tests were used to generate test streams to measure maximum TCP bandwidth and average throughput of the system. Iperf testing was done from the Client to Server with no delay and from the Client to Server with a 30 msec delay.

Single Data Flow with no Delay (Standard TCP)

The single data flow test with no delay was run to verify that the congestion control algorithms did not affect network performance given ideal network conditions. A network diagram is shown in Figure 3.

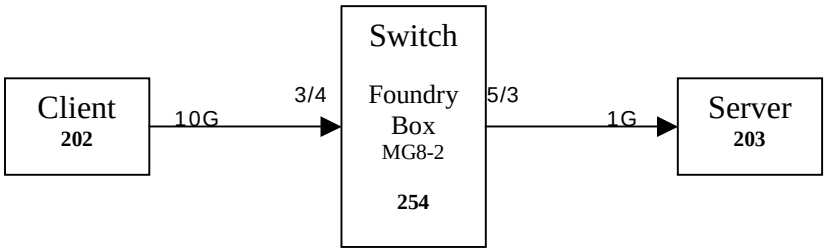

Figure 3: Network Diagram 1: No Delay

The Adtech delay simulator is set to 0 msec (no delay), which sets a 0 msec delay into the 1 Gbps link to the Dell 2650 Iperf server. The Iperf tests were run for 60 seconds using a single data stream with MTU set to 9000 bytes and a window size of 5MB.

The first test consisted of a single data flow from the 10 Gbps Iperf data client to the 1 Gbps server with no delay. The Standard TCP algorithm was used for this test to see the performance under optimal size buffer settings. A sequence of 10 identical runs was performed in order to find the average throughput of the system. The test runs are shown in Table 1.

Run	Time (sec)	Average Throughput (Gbytes)*	Bandwidth (Mbits/sec)
1	0-60	6.92	990
2	0-60	6.92	990
3	0-60	6.91	989
4	0-60	6.92	990
5	0-60	6.92	990
6	0-60	6.92	990
7	0-60	6.81	974
8	0-60	6.92	990
9	0-60	6.92	990
10	0-60	6.92	990

Table 1:Standard TCP no delay

*Average Throughput - The total amount of data, in Gigabytes, transferred during the 60 second test.

On a Standard TCP with no delay, the Average Throughput is 6.918 Gigabytes. The average bandwidth is 988.3 Megabits/sec.

Figure 4 is a sample plot of the outstanding or unacknowledged data of the Standard TCP connection and the congestion control algorithm that it uses for one test run.

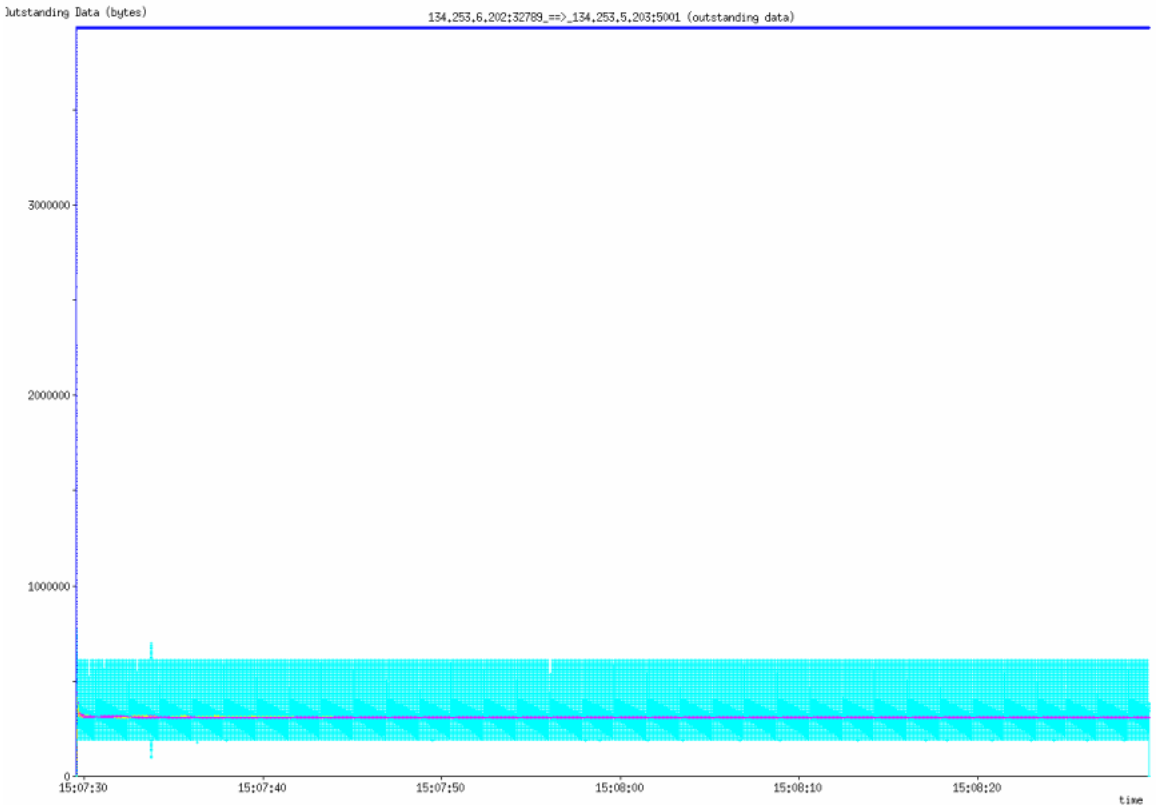

Figure 4: Outstanding Data Plot no delay for Standard TCP

The outstanding data plot clearly shows that with no delay for a Standard TCP connection, the buffer is hardly used as the outstanding data is approximately 300 Kbytes. The buffer used remains constant and does not reach anywhere near the bandwidth-delay product of 3.75Mbytes. The data transfer rate is constant since there were no dropped packets.

Single Data Flow with 30 msec Delay (Standard TCP)

The single data flow test with a 30 msec delay was added to the network test setup. The Adtech Delay Simulator was used to simulate this delay. A 30 msec delay was chosen to simulate the RTT from Sandia to Lawrence Livermore. A network diagram is shown in Figure 5.

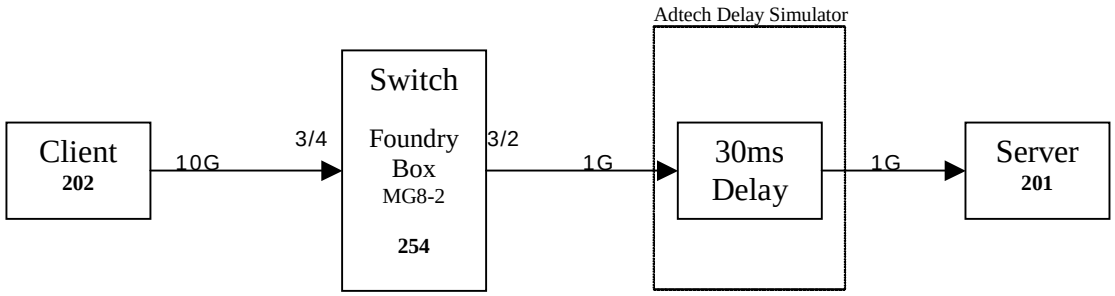

Figure 5 : Network Diagram 1: With 30 msec Delay

The Adtech delay simulator is set to 30 msec, which sets a 30 msec delay into the 1 Gbps to the Dell 2650 Iperf server. The Iperf tests were run for 60 seconds using a single data stream with MTU set to 9000 bytes and a window size of 5 MB.

The first test consisted of a single data flow from the 10 Gbps Iperf data client to the 1 Gbps server with a 30 msec delay. The Standard TCP algorithm was used for this test to see the performance under optimal size buffer settings. A sequence of 10 identical runs was performed in order to find the average throughput of the system. The test runs are shown in Table 2.

Run	Time (sec)	Average Throughput (Gbytes)*	Bandwidth (Mbits/sec)
1	0-60	4.64	664
2	0-60	4.56	652
3	0-60	5.21	746
4	0-60	3.99	570
5	0-60	3.60	515
6	0-60	4.15	593
7	0-60	4.27	611
8	0-60	3.81	546
9	0-60	4.72	674
10	0-60	4.35	622

Table 2: Standard TCP 30 msec delay

*Average Throughput - The total amount of data, in Gigabytes, transferred during the 60 second test.

Using the standard TCP algorithms, with a 30 msec delay, the average throughput is 4.33 Gigabytes. The average bandwidth is 619.3 Megabits/sec. Figure 6 plots the outstanding data bytes or unacknowledged data of a Standard TCP connection and the congestion control algorithm that it uses.

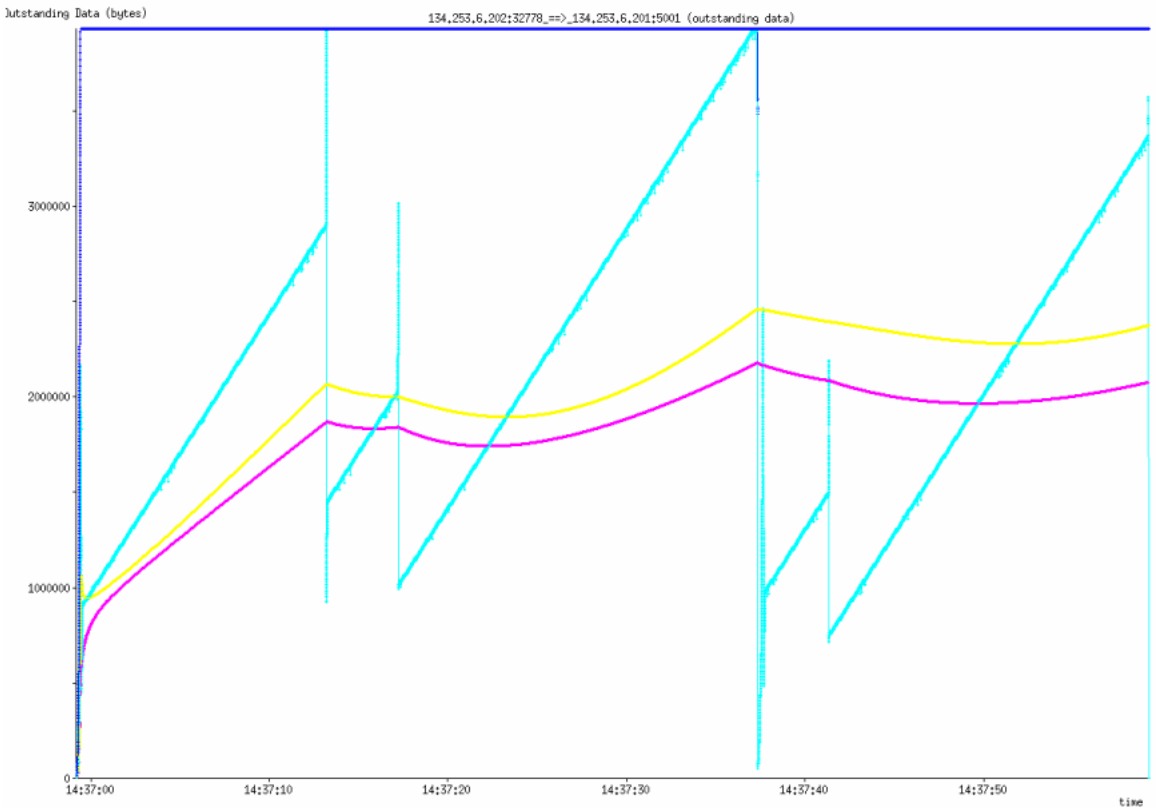

Figure 6: Outstanding Data Plot 30 ms delay for Standard TCP

The outstanding data plot clearly shows that when a delay of 30 msec is added for a Standard TCP connection, the buffer is used a lot where there are peaks up to the bandwidth-delay product of 3.75MB. The outstanding data plot for Standard TCP shows the slow start algorithm as it ramps up until a dropped packet occurs. There are indications of packets being dropped through many instances of the transfer of data. A sequence of five dropped packets occurs during the allotted time frame of 60 seconds.

The Y-axis represents the Outstanding Data in bytes and the X-axis represents time. These graphs are for estimating the congestion window at the sender. Since this cannot be determined accurately, outstanding unacknowledged data is used as an estimate. Light blue line represents instantaneous outstanding data samples at various points in the lifetime of the connection. Yellow line tracks the average outstanding data up to that point. Pink line tracks the weighted average of outstanding data up to that point. Dark blue line tracks the window advertised by the opposite end-point, the receiving window.

Single Data Flow with no Delay (High-Speed TCP)

The same tests were done for High-Speed TCP as Standard TCP. The 10 Gbps to 1 Gbps was performed on the Foundry interconnect with a 2.4.19 kernel with High-Speed TCP patch sending 10 Gbps Iperf data to the Dell 1 Gbps server.

The Iperf tests were run for 60 seconds using a single data stream with MTU set to 9000 bytes and a window size of 5MB. A sequence of 10 identical runs was performed in order to find the average throughput of the system. The test runs are shown in Table 3.

Run	Time (sec)	Average Throughput (Gbytes)*	Bandwidth (Mbits/sec)
1	0-60	6.92	990
2	0-60	6.92	990
3	0-60	6.92	990
4	0-60	6.92	990
5	0-60	6.91	988
6	0-60	6.92	990
7	0-60	6.81	974
8	0-60	6.92	990
9	0-60	6.92	990
10	0-60	6.92	990

Table 3: High-Speed TCP no delay

*Average Throughput - The total amount of data, in Gigabytes, transferred during the 60 second test.

For High-Speed TCP with no delay, the Average Throughput is 6.908 Gigabytes. The average bandwidth is 988.2 Megabits/sec.

Figure 7 plots the outstanding data bytes or unacknowledged data of a High-Speed TCP connection and the congestion control algorithm that it uses for one test run.

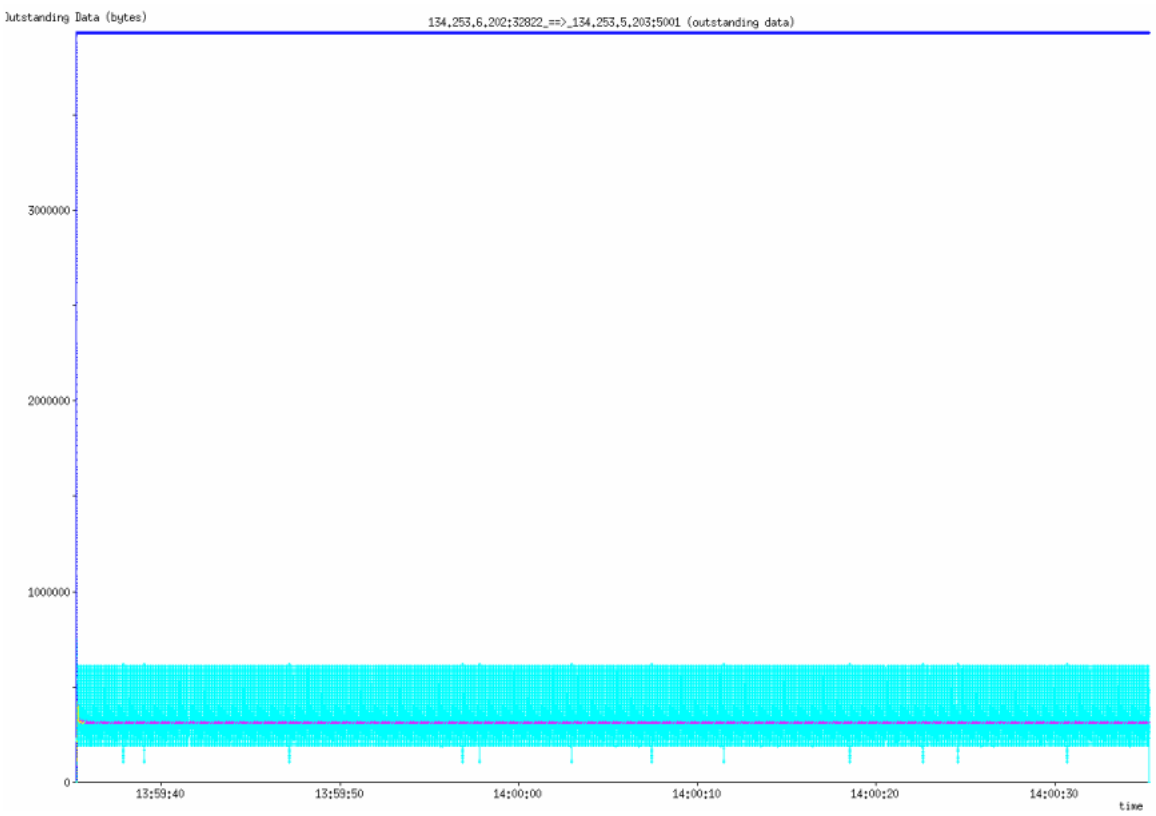

Figure 7: Outstanding Data Plot no delay for High-Speed TCP

The outstanding data plot clearly shows that with no delay for a High-Speed TCP connection, the buffer is hardly used and stays constant at approximately 300 Kbytes. The buffer used remains constant and does not reach anywhere near the bandwidth-delay product of 3.75MB.

Single Data Flow with 30 msec Delay (High-Speed TCP)

The single data flow test with a 30 msec delay was added to the network test setup. The RTT delay is set to 30 msec, which sets a 30 msec delay into the 1 Gbps to the Dell 2650 Iperf server.

The Iperf tests were run for 60 seconds using a single data stream with MTU set to 9000 bytes and a window size of 5MB. A sequence of 10 identical runs was performed in order to find the average throughput of the system. The test runs are shown in Table 4.

Run	Time (sec)	Average Throughput (Gbytes)*	Bandwidth (Mbits/sec)
1	0-60	5.31	760
2	0-60	5.48	784
3	0-60	6.24	889
4	0-60	5.37	767
5	0-60	5.29	757
6	0-60	4.81	688
7	0-60	5.31	760
8	0-60	4.41	631
9	0-60	4.93	705
10	0-60	5.98	856

Table 4: High-Speed TCP 30 msec delay

*Average Throughput - The total amount of data, in Gigabytes, transferred during the 60 second test.

On a High-Speed TCP with a 30 msec delay, the Average Throughput is 5.313 Gigabytes. The average bandwidth is 759.7 Megabits/sec.

Figure 8 plots the outstanding data bytes or unacknowledged data of a High-Speed TCP connection and the congestion control algorithm that it uses for a single test run.

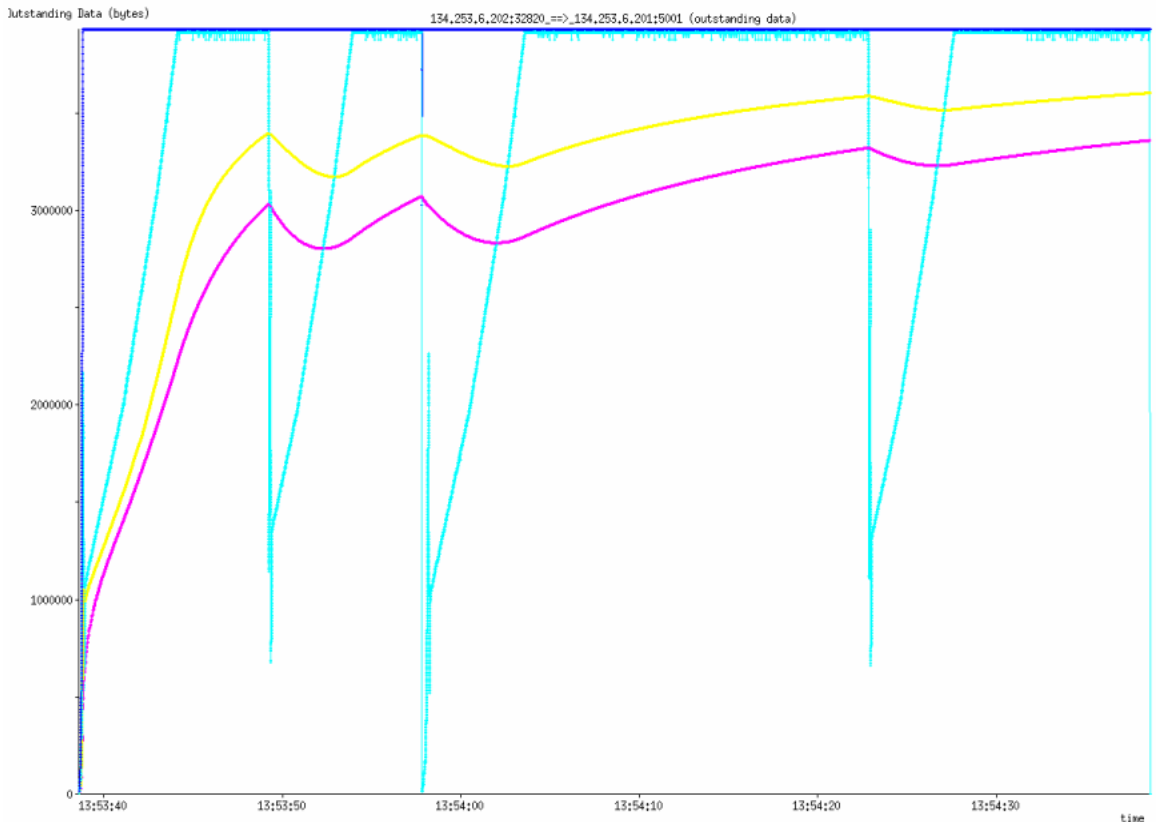

Figure 8: Outstanding Data Plot 30 ms delay for High-Speed TCP

The outstanding data plot clearly shows that when a delay of 30 msec is added for a High-Speed TCP connection, the buffer is used a lot and maxed out to the bandwidth-delay product of 3.75MB.

The example of outstanding data in Figure 7 shows that the same slow start algorithm used as Standard TCP. As the unacknowledged data ramps up to the bandwidth-delay product, it stays constant using the maximum allowable bytes to fill the channel. There are indications of packets being dropped through many instances of the transfer of data. A single drop occurs at around 11 seconds. This leads to the fast retransmit and recovery algorithms being used. A sequence of three dropped packets occurs during the allotted time frame of 60 seconds.

Single Data Flow with no Delay (Scalable TCP)

The same tests were done for Scalable TCP as Standard TCP. The 10 Gbps to 1 Gbps was performed on the Foundry interconnect with a 2.4.19 kernel with High-Speed TCP patch sending 10 Gbps Iperf data to the Dell 1 Gbps server.

The Iperf tests were run for 60 seconds using a single data stream with MTU set to 9000 bytes and a window size of 5MB. A sequence of 10 identical runs was performed in order to find the average throughput of the system. The test runs are shown in Table 5.

Run	Time (sec)	Average Throughput (Gbytes)*	Bandwidth (Mbits/sec)
1	0-60	6.91	988
2	0-60	6.92	990
3	0-60	6.91	989
4	0-60	6.92	990
5	0-60	6.76	967
6	0-60	6.90	987
7	0-60	6.84	979
8	0-60	6.92	990
9	0-60	6.90	986
10	0-60	6.90	987

Table 5: Scalable TCP no delay

*Average Throughput - The total amount of data, in Gigabytes, transferred during the 60 second test.

For Scalable TCP with no delay, the Average Throughput is 6.888 Gigabytes. The average bandwidth is 985.3 Megabits/sec.

Figure 9 plots the outstanding data bytes or unacknowledged data of a Scalable TCP connection and the congestion control algorithm that it uses for a single tet run.

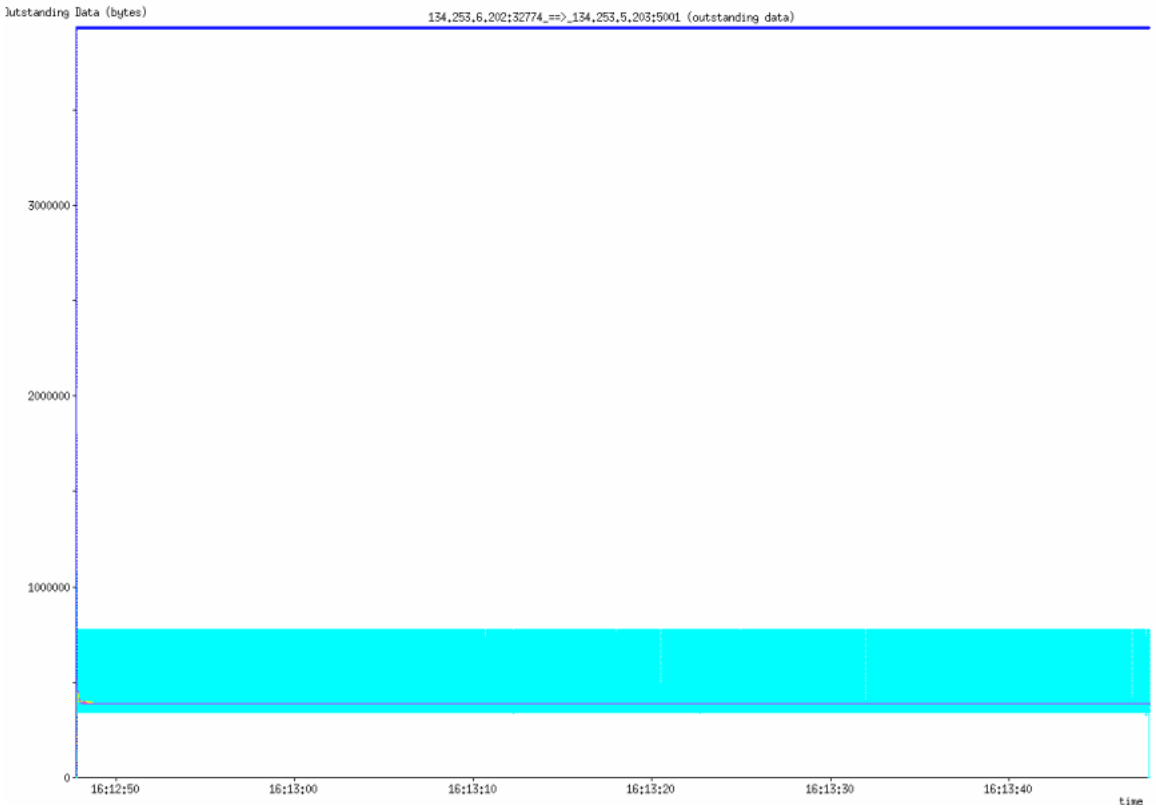

Figure 9: Outstanding Data Plot no delay for Scalable TCP

The outstanding data plot clearly shows that with no delay for a Scalable TCP connection, the buffer is hardly used. The buffer used remains constant and does not reach any where near the bandwidth-delay product of 3.75MB. The data transferred is constant which results in no significant large amount of dropped packets.

Single Data Flow with 30 msec Delay (Scalable TCP)

The single data flow test with a 30 msec delay was added to the network test setup. The RTT delay is set to 30 msec, which sets a 30 msec delay into the 1 Gbps to the Dell 2650 Iperf server.

The Iperf tests were run for 60 seconds using a single data stream with MTU set to 9000 bytes and a window size of 5MB. A sequence of 10 identical runs was performed in order to find the average throughput of the system. The test runs are shown in Table 6.

Run	Time (sec)	Average Throughput (Gbytes)*	Bandwidth (Mbits/sec)
1	0-60	6.60	945
2	0-60	5.72	818
3	0-60	6.62	947
4	0-60	6.45	922
5	0-60	6.63	949
6	0-60	6.12	876
7	0-60	6.49	929
8	0-60	6.38	913
9	0-60	6.42	918
10	0-60	6.09	869

Table 6: Scalable TCP 30 msec delay

*Average Throughput - The total amount of data, in Gigabytes, transferred during the 60 second test.

On a Scalable TCP with a 30 msec delay, the Average Throughput is 6.352 Gigabytes. The average bandwidth is 908.6 Megabits/sec.

Figure 10 plots the outstanding data bytes or unacknowledged data of a Scalable TCP connection and the congestion control algorithm that it uses for a single test run.

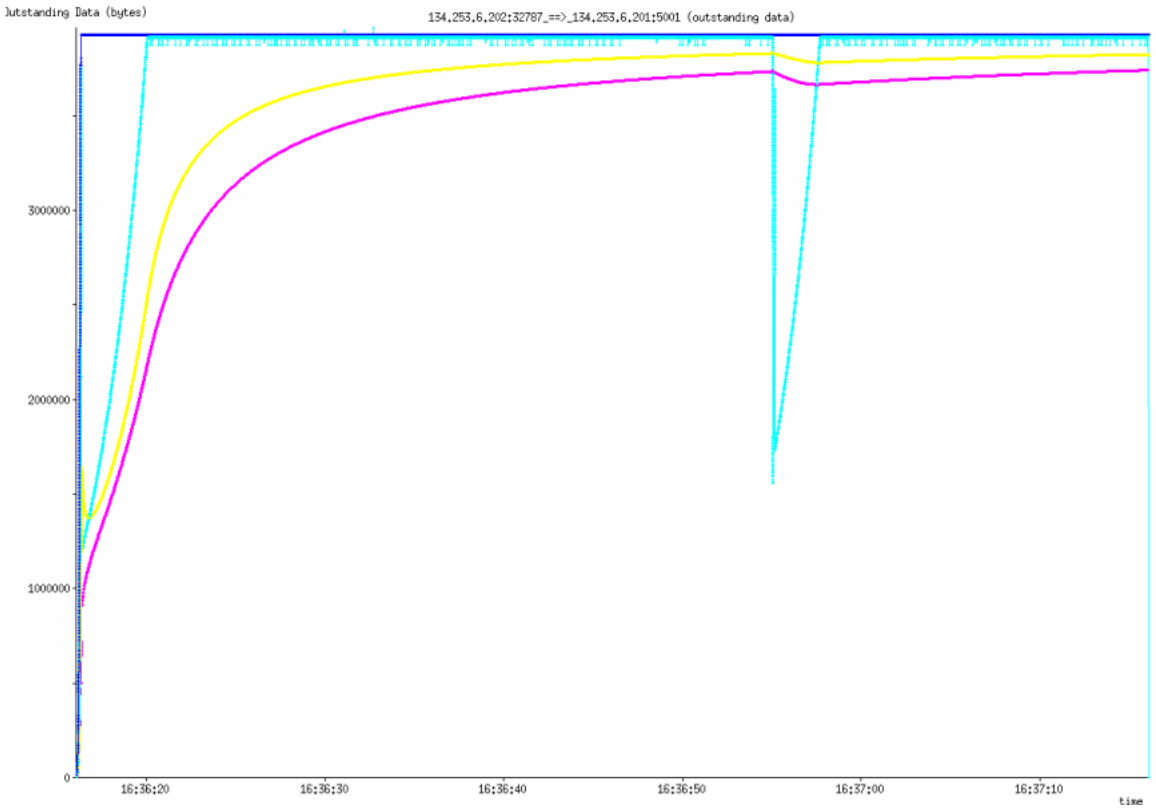

Figure 10 : Outstanding Data Plot 30 ms delay for Scalable TCP

The outstanding data plot clearly shows that when a delay of 30 msec is added for a High-Speed TCP connection, the buffer is used a lot and maxed out to the bandwidth-delay product of 3.75MB.

The example of outstanding data in Figure 9 shows that Scalable TCP quickly recovers from the short congestion periods.

As the unacknowledged data ramps up to the bandwidth-delay product, it stays constant using the maximum allowable bytes to fill the channel. There is an indication of a packet dropped through the instance of the data transfer. One dropped packet occurred during the allotted time frame of 60 seconds. A single drop occurs at around 40 seconds. The fast retransmit and recovery algorithms are clearly being used. The slope of the unacknowledged data is much steeper showing that the outstanding data recovers much faster than High-Speed and Standard TCP.

Summary of results 1

The goal was to test, evaluate, compare, and report protocol technologies that enhance the performance of the ASC WAN. The alternative TCP congestion control algorithms tested were High-Speed TCP and Scalable TCP. To provoke congestion, High Speed TCP and Scalable TCP was compared to Standard TCP in a test where a client machine sends data at 10 Gbps to a server machine, which can only receive data at 1 Gbps over a network that has a round trip time of 30 msec. Iperf tests were run for 60 seconds using a single data stream with MTU set to 9000 bytes and a window size of 5MB. A sequence of 10 identical runs each was performed in order to find the average throughput and average bandwidth of the system. The test runs are shown in Table 7.

	Average Throughput (Gbytes)*		Bandwidth (Mbits/sec)	
	0 ms	30 ms	0 ms	30 ms
Standard TCP	6.908	4.330	988.3	**619.3**
High-Speed TCP	6.908	5.313	988.2	**759.7**
Scalable TCP	6.888	6.352	985.3	**908.6**

Table 7: Summary of Results for Single Data Flows

*Average Throughput - The total amount of data, in Gigabytes, transferred during the 60 second test.

Figure 11 shows the results for the three algorithms, Standard, High-speed, and Scalable TCP for 1 Client to 1 Server tests. Their bandwidths are compared to each other for the cases when the network is set up for both non-delay and 30 msec delays.

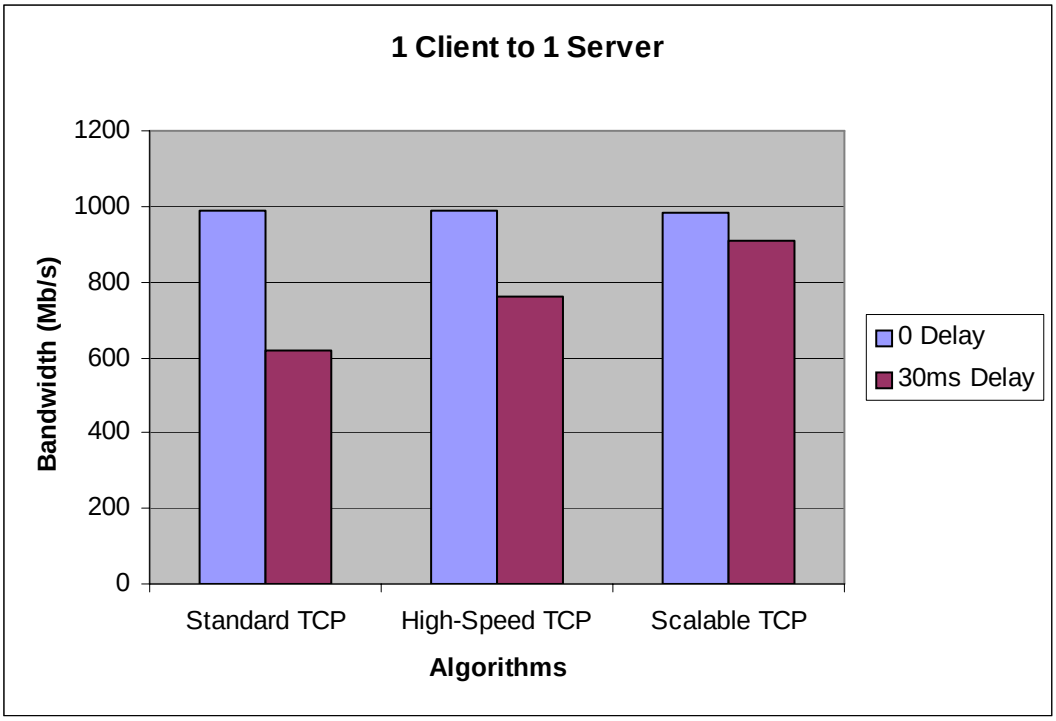

Figure 11: 1 Client to 1 Server: Overall Performance Comparison

Single Data Flow Conclusions

High-Speed and Scalable TCP implemented simple changes to the currently used Standard TCP congestion control algorithm. These changes have a positive effect on the existing network traffic. The alternative algorithm provides higher channel utilization for the high speed, long delay environment. For a 5M window, High-Speed and Scalable TCP performed better than Standard TCP. The tests with no delay on bandwidth for High-Speed, Scalable, and Standard TCP were exactly the same, which was the result expected. When a delay of 30 msec was added to the system, the bandwidth for Standard TCP was 619.3 Mbits/sec, while High-Speed TCP was 759.7 Mbits/sec, and Scalable TCP was 908.6 Mbits/sec. High-Speed TCP proved to be much more significant than Standard TCP, while Scalable TCP was the dominant protocol overall. High-Speed and Scalable TCP had a faster data recovery than Standard TCP, which can be seen by steeper sloped peaks. Scalable TCP had one dropped packet and High-Speed TCP had three dropped packets compared to Standard TCP, which had five dropped packets in the allotted time frame of 60 seconds. Testing single data flow from a client machine that sends data at 10 Gbps to a server machine, which can only receive data at 1 Gbps over a network that has a round trip time of 30 msec definitely showed improvement. The improvement was approximately a 20 percent increase in bandwidth for High-Speed TCP, resulting in much better performance over Standard TCP. Scalable TCP was approximately a 17 percent increase in bandwidth of better performance over High-Speed TCP, while Scalable TCP was approximately a 32 percent increase in bandwidth of better performance over Standard TCP.

Network Setup 2

While the initial tests performed showed a significant performance improvement attributable to the alternative congestion algorithms, the test did not examine the general case of simultaneous multiple data flows and multiple host transfers. In order to be sure that these alternative algorithms were stable in these more complicated scenarios further testing was required. Simultaneous 10 Gbps to 1 Gbps tests were performed on the Foundry interconnect where two client machines sends data at 10 Gbps to a server machine, which can only receive data at 1 Gbps over a network that has a round trip time of 30 msec. The setup used for network testing is shown in Figure 12.

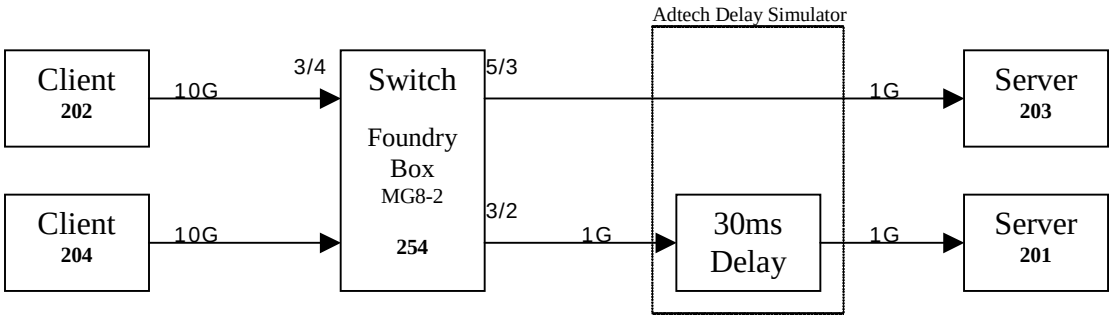

Figure 12: Network Setup Diagram 2

A series of parallel testing was performed from the two clients using Standard, High-Speed, and Scalable TCP kernels which indicates the number of connections to handle by the server before closing. The parallel stream numbers used when testing were 1, 2, 4, and 8. Likewise, the server was set up to receive these simultaneous connections of parallel streams for the cases of delay and no delay.

Tuning parameters 2

The following tuning parameters were used on both senders and receivers.

```
echo 1 >/proc/sys/net/ipv4/tcp_timestamps
   echo 1 >/proc/sys/net/ipv4/tcp_window_scaling
echo 1 >/proc/sys/net/ipv4/tcp_sack
echo 8388608 > /proc/sys/net/core/wmem_max
echo 8388608 > /proc/sys/net/core/rmem_max
echo "4096 87380 4194304" >/proc/sys/net/ipv4/tcp_rmem
echo "4096 87380 4194304" >/proc/sys/net/ipv4/tcp_wmem
ifconfig eth2 mtu 9000
ifconfig eth2 txqueuelen 1000
ifconfig eth2 down
ifconfig eth3 mtu 9000
ifconfig eth3 txqueuelen 1000
ifconfig eth3 134.253.6.204/24
ifconfig eth3 134.253.6.204/24
ifconfig eth3 broadcast 134.253.6.255
ip route add 134.253.5.0/24 via 134.253.6.254
#ifconfig eth2 down
```

```
#rmmod ixgb
#insmod ixgb
#ifup eth2
```

The following command was also run before each test was executed.

```
sysctl -w net.ipv4.route.flush=1
```

Flushing the route resets the slow start threshold (ssthresh) for all connections. This needed to be done to allow for repeatable test results. Iperf tests were used to generate test streams to measure maximum TCP bandwidth and average throughput of the system. Iperf testing was done from the Two Clients to One Server with no delay and from the Two Clients to One Server with a 30 msec delay.

Parallel Data Flow with no Delay (Standard TCP)

The parallel data flow test with no delay was run to verify that the congestion control algorithms did not affect network performance given ideal network conditions. A network diagram is shown in Figure 13.

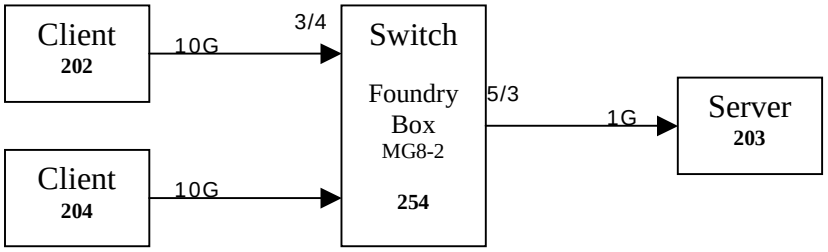

Figure 13: Network Diagram 2: No Delay

The RTT delay is set to 0 msec (no delay), which sets a 0 msec delay into the 1 Gbps link to the Dell 2650 Iperf server. The Iperf tests were run for 60 seconds using a parallel data stream with MTU set to 9000 bytes and a window size of 5MB with parallel stream numbers of 1, 2, 4, and 8.

The first test consisted of a parallel data flow from the 10 Gbps Iperf data client to the 1 Gbps server with no delay. The Standard TCP algorithm was used for this test to see the performance under optimal size buffer settings. A sequence of 5 identical runs was performed in order to find the average bandwidth and throughput of the system. The test runs for the parallel streams of 1, 2, 4, and 8 are shown in Table 8.

Parallel	Time (sec)	Average Throughput (Gbytes)*	Bandwidth (Mbits/sec)**
1	0-60	7.012	1002.2
2	0-60	7.024	1003.2
4	0-60	7.2514	1006.2
8	0-60	7.2574	999.6

Table 8: Standard TCP no delay

*Average Throughput - The total amount of data, in Gigabytes, transferred during the 60 second test.

** The total bandwidth reported can exceed the 1 Gbps line rate because of minor timing inconsistencies in the test setup.

Parallel Data Flow with 30 msec Delay (Standard TCP)

The parallel data flow test with a 30 msec delay was added to the network test setup. The Adtech Delay Simulator was used to simulate this delay. A 30 msec delay was chosen to simulate the RTT from Sandia to Lawrence Livermore. A network diagram is shown in Figure 14.

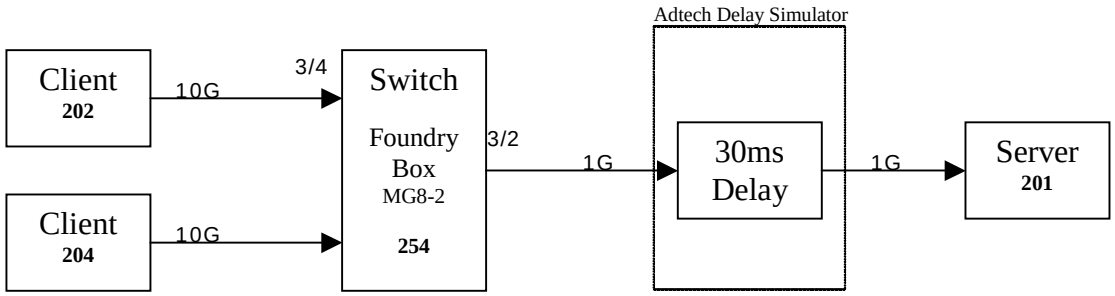

Figure 14: Network Diagram 2: With 30 msec Delay

The RTT delay is set to 30 msec, which sets a 30 msec delay into the 1 Gbps to the Dell 2650 Iperf server. The Iperf tests were run for 60 seconds using a parallel data stream with MTU set to 9000 bytes and a window size of 5 MB with parallel stream numbers of 1, 2, 4, and 8.

Testing consisted of parallel single data flows from the 10 Gbps Iperf data clients to the 1 Gbps server with a 30 msec delay. The Standard TCP algorithm was used for this test to see the performance under optimal size buffer settings. A sequence of 5 identical runs was performed in order to find the average bandwidth and throughput of the system. The test runs for the parallel streams of 1, 2, 4, and 8 are shown in Table 9.

Parallel	Time (sec)	Average Throughput (Gbytes)*	Bandwidth (Mbits/sec)
1	0-60	5.68	811.6
2	0-60	6.316	899.6
4	0-60	6.6858	927.66
8	0-60	6.9322	950.22

Table 9 : Standard TCP 30 msec delay

*Average Throughput - The total amount of data, in Gigabytes, transferred during the 60 second test.

Parallel Data Flow with no Delay (High-Speed TCP)

Same tests were done for High-Speed TCP as Standard TCP. The 10 Gbps to 1 Gbps was performed on the Foundry interconnect sending 10 Gbps Iperf data to the Dell 1 Gbps server.

The Iperf tests were run for 60 seconds using a parallel data stream with MTU set to 9000 bytes and a window size of 5MB. A sequence of 5 identical runs was performed in order to find the average bandwidth and throughput of the system. The test runs for the parallel streams of 1, 2, 4, and 8 are shown in Table 10.

Parallel	Time (sec)	Average Throughput (Gbytes)*	Bandwidth (Mbits/sec)**
1	0-60	6.988	999
2	0-60	7.018	1000.2
4	0-60	7.2604	1007.4
8	0-60	7.3122	1007.54

Table 10: High-Speed TCP no delay

*Average Throughput - The total amount of data, in Gigabytes, transferred during the 60 second test.

** The total bandwidth reported can exceed the 1 Gbps line rate because of minor timing inconsistencies in the test setup.

Parallel Data Flow with 30 msec Delay (High-Speed TCP)

The parallel data flow test with a 30 msec delay was added to the network test setup. The RTT delay is set to 30 msec, which sets a 30 msec delay into the 1 Gbps to the Dell 2650 Iperf server.

The Iperf tests were run for 60 seconds using a single data stream with MTU set to 9000 bytes and a window size of 5MB. A sequence of 5 identical runs was performed in order to find the average bandwidth and throughput of the system. The test runs for the parallel streams of 1, 2, 4, and 8 are shown in Table 11.

Parallel	Time (sec)	Average Throughput (Gbytes)*	Bandwidth (Mbits/sec)
1	0-60	5.87	838.2
2	0-60	6.102	869.6
4	0-60	6.4598	894.9
8	0-60	7.069	959.38

Table 11: High-Speed TCP 30 msec delay

*Average Throughput - The total amount of data, in Gigabytes, transferred during the 60 second test.

Parallel Data Flow with no Delay (Scalable TCP)

Same tests were done for Scalable TCP as High-Speed and Standard TCP. The 10 Gbps to 1 Gbps was performed on the Foundry interconnect sending 10 Gbps Iperf data to the Dell 1 Gbps server.

The Iperf tests were run for 60 seconds using a parallel data stream with MTU set to 9000 bytes and a window size of 5MB. A sequence of 5 identical runs was performed in order to find the average bandwidth and throughput of the system. The test runs for the parallel streams of 1, 2, 4, and 8 are shown in Table 12.

Parallel	Time (sec)	Average Throughput (Gbytes)*	Bandwidth (Mbits/sec)**
1	0-60	7.054	1007.6
2	0-60	7.066	1008
4	0-60	7.2336	1004.6
8	0-60	7.3076	1005.9

Table 12: Scalable TCP no delay

*Average Throughput - The total amount of data, in Gigabytes, transferred during the 60 second test.

** The total bandwidth reported can exceed the 1 Gbps line rate because of minor timing inconsistencies in the test setup.

Parallel Data Flow with 30 msec Delay (Scalable TCP)

The parallel data flow test with a 30 msec delay was added to the network test setup. The RTT delay is set to 30 msec, which sets a 30 msec delay into the 1 Gbps to the Dell 2650 Iperf server.

The Iperf tests were run for 60 seconds using a single data stream with MTU set to 9000 bytes and a window size of 5MB. A sequence of 5 identical runs was performed in order to find the average bandwidth and throughput of the system. The test runs for the parallel streams of 1, 2, 4, and 8 are shown in Table 13.

Parallel	Time (sec)	Average Throughput (Gbytes)*	Bandwidth (Mbits/sec)
1	0-60	6.414	917.4
2	0-60	6.914	985.6
4	0-60	7.0916	991.36
8	0-60	7.2556	996.6

Table 13: Scalable TCP 30 msec delay

*Average Throughput - The total amount of data, in Gigabytes, transferred during the 60 second test.

Summary of results 2

The goal was to test, evaluate, compare, and report protocol technologies that enhance the performance of the ASC WAN. The alternative TCP congestion control algorithms tested were High-Speed TCP and Scalable TCP. To provoke congestion, High Speed TCP and Scalable TCP was compared to Standard TCP in a test where a two client machines sends parallel data at 10 Gbps to a server machine, which can only receive data at 1 Gbps over a network that has a round trip time of 30 msec. Iperf tests were run for 60 seconds using a single data stream with MTU set to 9000 bytes and a window size of 5MB. A sequence of 5 identical runs each was performed in order to find the average throughput and average bandwidth of the system. The test runs are shown in Table 14.

	Average Throughput (Gbytes)*		Bandwidth (Mbits/sec)	
	0 ms	30 ms	0 ms	30 ms
Parallel 1				
Standard TCP	7.012	5.68	1002.2	**811.6**
High-Speed TCP	6.988	5.87	999	**838.2**
Scalable TCP	7.054	6.414	1007.4	**917.4**
Parallel 2				
Standard TCP	7.024	6.316	1003.2	**899.6**
High-Speed TCP	7.018	6.102	1000.2	**869.6**
Scalable TCP	7.066	6.914	1008	**985.6**
Parallel 4				
Standard TCP	7.2514	6.6858	1006.2	**927.66**
High-Speed TCP	7.2604	6.4598	1007.4	**894.9**
Scalable TCP	7.2336	7.0916	1004.6	**991.36**
Parallel 8				
Standard TCP	7.2574	6.9322	999.6	**950.22**
High-Speed TCP	7.3122	7.069	1007.54	**959.38**
Scalable TCP	7.3076	7.2556	1005.9	**996.6**

Table 14: Summary of Results 2, Multiple Simultaneous Data Flows

*Average Throughput - The total amount of data, in Gigabytes, transferred during the 60 second test.

A graphical analysis of the overall performance for the algorithms tested is shown in Figures 14, 15, and 16. The results show how Standard, High-Speed, and Scalable performed under the conditions when two clients send parallel data streams to a single server.

Figure 15 shows the results for the parallel streams of 1, 2, 4, and 8. In this case, the three algorithms, Standard, High-speed, and Scalable TCP bandwidths are compared when the network is set up with no delay.

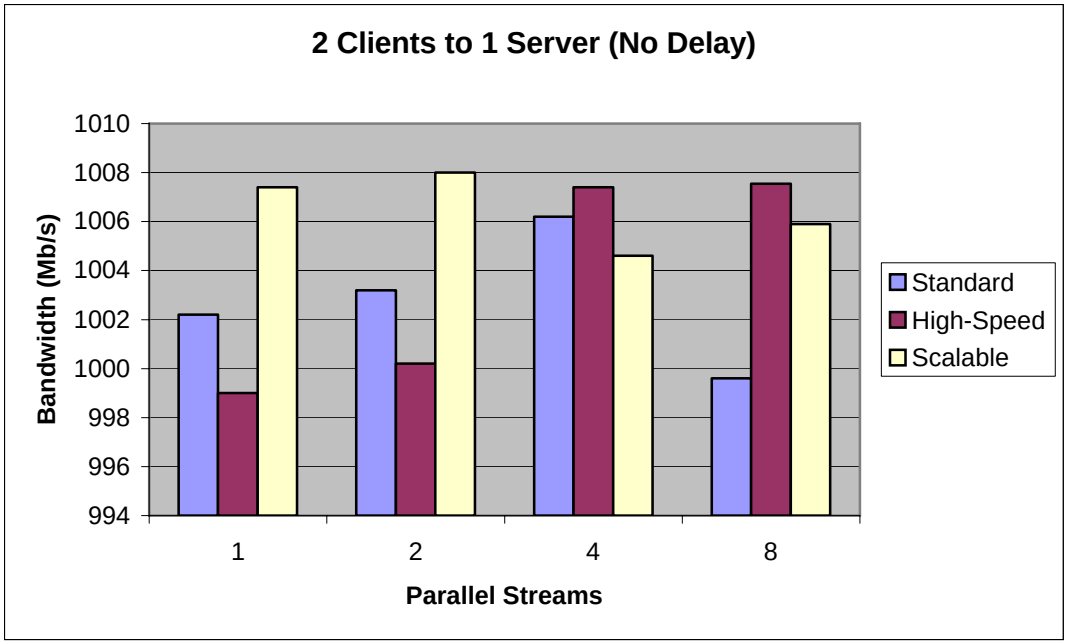

Figure 15: 2 Clients to 1 Server: No Delay

The tests with no delay on bandwidth for High-Speed, Scalable, and Standard TCP varied when applying different numbers of parallel streams. Standard TCP proved to be much more significant than High-Speed TCP, while Scalable TCP was the dominant protocol overall for the cases of 1 and 2 parallel streams of data. High-Speed was the dominant protocol over Standard and Scalable for the cases of 4 and 8 parallel streams of data.

Figure 16 shows the results for the parallel streams of 1, 2, 4, and 8. In this case, the three algorithms, Standard, High-speed, and Scalable TCP bandwidths are compared when the network is set up with a 30 msec delay.

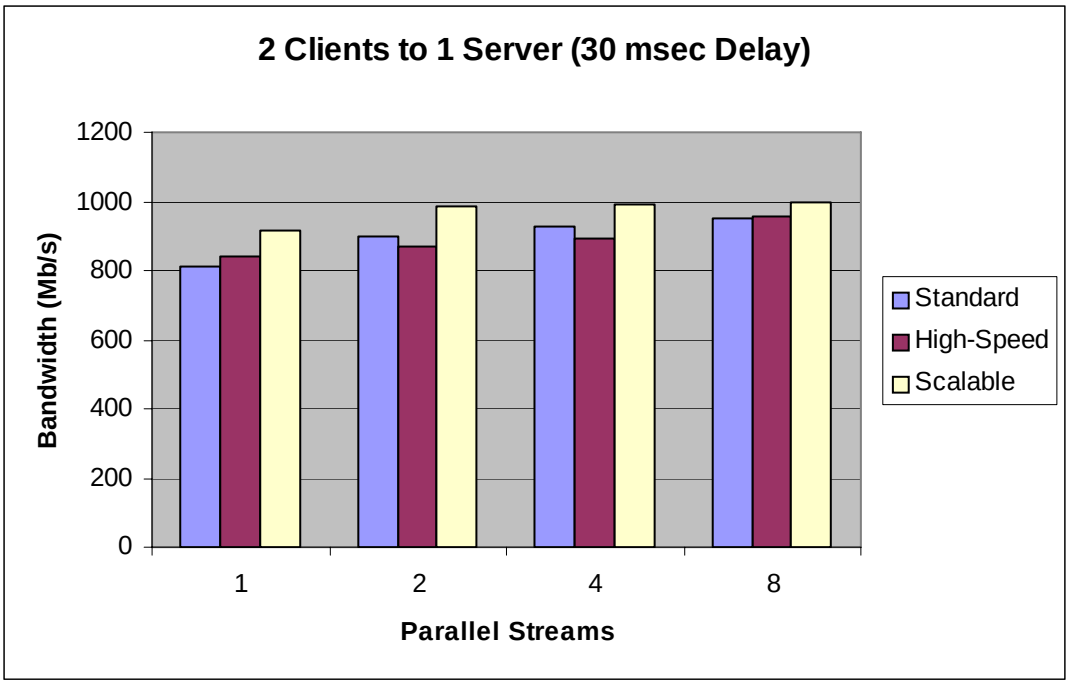

Figure 16: 2 Clients to 1 Server: With 30 msec Delay

The tests with a 30 msec delay on bandwidth for High-Speed, Scalable, and Standard TCP varied very similar when applying different numbers of parallel streams. Scalable TCP was the dominant protocol overall for all cases of parallel streams of data tested. The increase in performance is observed as the number of parallel streams is increased for each of the congestion control algorithms.

Figure 17 shows the results for the parallel streams of 1, 2, 4, and 8 In this case, the three algorithms, Standard, High-speed, and Scalable TCP bandwidths are compared when the network is set up for both non-delay and 30 msec delays.

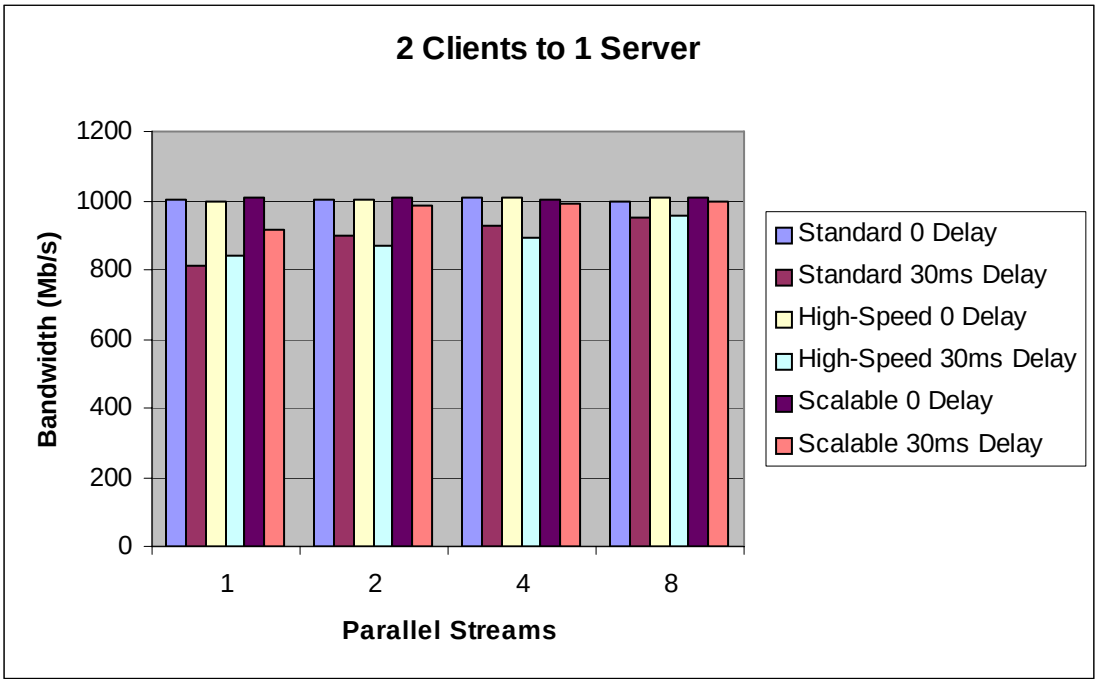

Figure 17: 2 Clients to 1 Server: Overall Performance for Multiple Data Flows

The tests with a 30 msec delay on bandwidth for High-Speed, Scalable, and Standard TCP varied very similarly when applying different numbers of parallel streams. Scalable TCP was the dominant protocol overall for all cases of parallel streams of data tested. The increase in performance is observed as the number of parallel streams is increased for each of the congestion control algorithms. Overall, the case where two clients sending 8 parallel streams of data each to a single server performed optimally compared to the rest of the parallel streams of data.

Due to the extensive amount of data gathered for the Single Data Flow and Parallel Data Flow testing, the testing results for alternative TCP flow control algorithms are reported in separate sections of this paper.

Parallel Data Flow Conclusions

High-Speed and Scalable TCP implemented simple changes to the currently used Standard TCP congestion control algorithm while performing simultaneous connections of parallel streams for the cases of delay and no delay. These changes have a positive effect on the existing network traffic. The alternative algorithms provide higher channel utilization for the high speed, long delay environment. The tests with no delay on bandwidth for High-Speed, Scalable, and Standard TCP varied when applying different numbers of parallel streams. Testing parallel data flows from two client machines that send data at 10 Gbps to a server machine, which can only receive data at 1 Gbps over a network that has a round trip time of 30 msec, definitely showed improvement.

In the case of 1 parallel stream from both clients, High-Speed TCP proved to be much more significant than Standard TCP, while Scalable TCP was the dominant protocol overall. The improvement was approximately a 3 percent increase in bandwidth for High-Speed TCP, resulting in a higher performance over Standard TCP. Scalable TCP was approximately a 9 percent increase in bandwidth of better performance over High-Speed TCP, while Scalable TCP was approximately a 12 percent increase in bandwidth of better performance over Standard TCP.

In the case of 2 parallel streams from both clients, Standard TCP proved to be much more significant than High-Speed TCP, while Scalable TCP was the dominant protocol overall. The improvement was approximately a 3 percent increase in bandwidth for Standard TCP, resulting in a higher performance over High-Speed TCP. Scalable TCP was approximately a 12 percent increase in bandwidth of better performance over High-Speed TCP, while Scalable TCP was approximately a 9 percent increase in bandwidth of better performance over Standard TCP.

In the case of 4 parallel streams from both clients, Standard TCP proved to be much more significant than High-Speed TCP, while Scalable TCP was the dominant protocol overall. The improvement was approximately a 4 percent increase in bandwidth for Standard TCP, resulting in a higher performance over High-Speed TCP. Scalable TCP was approximately a 10 percent increase in bandwidth of better performance over High-Speed TCP, while Scalable TCP was approximately a 7 percent increase in bandwidth of better performance over Standard TCP.

1

In the case of 8 parallel streams from both clients, High-Speed TCP proved to be much more significant than Standard TCP, while Scalable TCP was the dominant protocol overall. The improvement was approximately a 1 percent increase in bandwidth for High-Speed TCP, resulting in a higher performance over Standard TCP. Scalable TCP was approximately a 4 percent increase in bandwidth of better performance over High-Speed TCP, while Scalable TCP was approximately a 5 percent increase in bandwidth of better performance over Standard TCP.

References

[1] Introduction to TCP/IP, URL http://www.yale.edu/pclt/COMM/TCPIP.HTM

[2] D. E. Comer. Internetworking with TCP/IP Principles, Protocols, and Architectures, 4th edition, New Jersey, Prentice-Hall 1995.

[3] Robert M. Long, "Evaluation of TCP Congestion Control Algorithms", Sandia National Laboratories, Albuquerque, New Mexico, SAND2003-4404, December 2003.

[4] W. Noureddine, F. Tobagi. The Transmission Control Protocol, URL http://citeseer.nj.nec.com/noureddine02transmission.html, 2002

[5] V. Jacobsen, R. Braden. TCP Extensions for Long Delay Paths, URL http://www.freesoft.org/CIE/RFC/1072/index.htmb *RFC 1072*, October 1988.

[6] Yi-Ting Li. High-Speed TCP, URL http://www.hep.ucl.ac.uk/~ytl/tcpip/highspeedtcp/hstcp/index.html

[7] M. Allman, V. Paxson, and W. Stevens. TCP Congestion Control, URL http://www.faqs.org/rfcs/rfc2581.html *RFC 2581*, April 1999.

[8] S. Floyd. High-Speed TCP for Large Congestion Windows, URL http://www.icir.org/floyd/papers/draft-floyd-tcp-highspeed-00c.txt

[9] Advanced Simulation and Computing, URL http://www.lanl.gov/asci/

[10] High-Speed TCP, S. Floyd, URL http://www.hep.man.ac.uk/u/garethf/hstcp

[11] High-Speed TCP, SuSE Linux, URL http://www.suse.com/us/private/download/updates/91_x86_64.html

[12] High-Speed TCP patches, Web 100, URL http://tiki.is.os-omicron.org/en/tiki.cgi?c=v&p=Web100

[13] Scalable TCP, URL http://www-lce.eng.cam.ac.uk/~ctk21/scalable/

[14] M. Allman, V. Paxson, and W. Stevens. TCP Congestion Control, *Internet RFC 2581*, April 1999.

Distribution

1	MS 0805	W.D. Swartz, 9329
1	MS 0806	Len Stans, 9336
1	MS 0806	J.P. Brenkosh, 9336
1	MS 0806	J.M. Eldridge, 9336
1	MS0806	A. Ganti, 9336
1	MS 0806	S.A. Gossage, 9336
1	MS 0806	T.C. Hu, 9338
1	MS 0806	B.R. Kellogg, 9336
1	MS 0806	R. Romero, 9336
1	MS 0806	L.G. Martinez, 9334
1	MS 0806	M.M. Miller, 9336
1	MS 0806	J.H. Naegle, 9336
1	MS 0806	R.R. Olsberg, 9336
1	MS 0806	L.G. Pierson, 9336
1	MS 0806	T.J. Pratt, 9338
1	MS 0806	J.A. Schutt, 9336
1	MS 0806	J.D. Tang, 9336
1	MS 0806	T.D. Tarman, 9336
6	MS 0806	L.F. Tolendino, 9334
1	MS 0806	D.J. Wiener, 9336
1	MS 0806	E.L. Witzke, 9336
1	MS 0812	M.J. Benson, 9334
1	MS 9018	Central Technical Files, 8945-1
2	MS 0899	Technical Library, 9616 (2)