SANDIA REPORT

SAND2007-4526
Unlimited Release
Printed July 2007

Optimizing the ASC WAN: Evaluating Network Performance Tools for Comparing Transport Protocols

Christopher L. Lydick

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-AC04-94-AL85000.

Sandia National Laboratories

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 ordering: http://www.ntis.gov/help/ordermethods.asp?loc=7-4-0#online

SAND2007-4526
Unlimited Release
Printed July 2007

Optimizing the ASC WAN: Evaluating Network Performance Tools for Comparing Transport Protocols

Christopher L. Lydick
Graduate Student Intern
Advanced Networking Integration (9336)
Sandia National Laboratories
P.O. Box 5800
Albuquerque, NM 87185-0806

Abstract

The Advanced Simulation & Computing Wide Area Network (ASC WAN), which is a high delay-bandwidth network connection between US Department of Energy National Laboratories, is constantly being examined and evaluated for efficiency. One of the current transport-layer protocols which is used, TCP, was developed for traffic demands which are different from that on the ASC WAN. The Stream Control Transport Protocol (SCTP), on the other hand, has shown characteristics which make it more appealing to networks such as these. Most important, before considering a replacement for TCP on any network, a testing tool that performs well against certain criteria needs to be found. In order to try to find such a tool, two popular networking tools (Netperf v.2.4.3 & v.2.4.6 (OpenSS7 STREAMS), and Iperf v.2.0.6) were tested. These tools implement both TCP and SCTP and were evaluated using four metrics: (1) How effectively can the tool reach a throughput near the bandwidth? (2) How much of the CPU does the tool utilize during operation? (3) Is the tool freely and widely available? And, (4) Is the tool actively developed? Following the analysis of those tools, this paper goes further into explaining some recommendations and ideas for future work.

Acknowledgment

Special thanks to Tan Hu, Larry Tolendino, and Jason Wertz for their help and guidance.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In an effort to find an alternative protocol(s) for use on a network, it is necessary to evaluate various performance tools that could be used to compare the available data communication protocols. This report compares two performance testing tools (Netperf v.2.4.3 & v.2.4.6 (OpenSS7 STREAMS), and Iperf v.2.0.6) against TCP (Transmission Control Protocol) and SCTP (Stream Control Transmission Protocol) by using four metrics:

- How effectively can the tool reach a throughput near the bandwidth?
- How much of the CPU does the tool utilize during operation?
- Is the tool freely and widely available?
- Is the tool actively developed? To what extent?

The process by which the network performance tools were selected for evaluation was done based on pre-existing implementations and knowledge within the Advanced Networking Integration Department at Sandia National Laboratories. Netperf and Iperf were already widely used within the department. This report will expand on that knowledge through testing within a laboratory setting.

Background

The ASC WAN

The Advanced Simulation Computing Wide Area Network (ASC WAN, formerly known as the ASCI WAN) was developed to connect supercomputers within the US Department of Energy National Laboratories, allowing the various labs to share resources [1]. The locations which utilize this network are Sandia National Laboratories, Albuquerque, NM (SNL); Sandia National Laboratories, Livermore, CA (SNL-CA); Lawrence Livermore National Laboratories, Livermore, CA (LLNL); and Los Alamos National Laboratories, Los Alamos, NM (LANL). When the ASC WAN is being used,

there are large flows of data which are transferred between the labs [4]. This requires extremely large network throughput. When coupling that characteristic with the unavoidable physical distances between the laboratories, these network connections have a high delay-bandwidth product, leading to issues within the traditional TCP implementation. In order to efficiently utilize the ASC WAN network with TCP, it is necessary to use multiple simultaneous TCP connections [4]. This has led the Advanced Networking Integration Department to research other alternatives, the most promising being Stream Control Transport Protocol (SCTP).

TCP vs. SCTP

TCP has been around for quite some time. It was developed long ago when the Internet consisted of hundreds or thousands of hosts, not in today's world with over a billion hosts. The newest alternative, SCTP was submitted to the RFC database in mid 2000 (RFC 2690, 3286), and has since been adopted by the IETF as a recognized Internet transport layer standard.

Briefly, some of the negative issues with TCP are:

- TCP is Byte-Oriented. Because of this, message boundaries are not preserved. This can be an issue if a piece of the data becomes lost or corrupt during transfer because the receiver in the TCP connection will not deliver data to the application until it receives the missing data. This causes *Head-Of-The-Line Blocking* (HoL) [5].
- TCP uses a 3-way handshake to initiate data transfers. This characteristic is prone to *Blind SYN Attacks* [5]. These attacks occur when a malicious user sends multiple requests (SYN packets) for establishing a connection to a single host. The host receiving these requests allocates resources for each request, and after some time can be depleted of those resources and rendered useless. This is also known as a Denial of Service (DoS) attack, because once the host is rendered useless, it appears to be unavailable and passively denies any further connection requests.
- In terms of the ASC WAN, multiple TCP connections need to be established to transfer data using the parallel FTP (PFTP) utility [4]. PFTP is one of the current methods used to transfer data simultaneously through multiple TCP connections over the ASC WAN.

While on the other hand:

- SCTP is Message-Oriented. This enables SCTP to preserve the message boundaries, and allows it to get around the TCP HoL blocking [5].

- SCTP uses a 4-way handshake. SCTP allows for data transfer after three messages have been exchanged (just like TCP), but also implements a cookie and cookie verification before allocating resources. This insures that the connection request is indeed valid.

- Because SCTP uses *multi-homing*, a single SCTP association can contain multiple streams or connections. This requires less overhead (than the multi-connection TCP implementation described above), and provides SCTP with redundancy [5].

Available Network Performance Tools

Iperf 2.0.6 (OpenSS7)

The Iperf network performance tool was initially developed by NLANR/DAST (National Laboratory for Applied Network Research/Distributed Applications Support Team) as an alternative to the already available and somewhat cumbersome throughput performance measuring tools. Its user interface is fairly straight-forward and is easy to use, making it fairly popular within the networking community.

The current public release of Iperf from NLANR/DAST is 2.0.3, released in May 2005. This is also the current release available through SourceForge (*http://www.sourceforge.net*). This version of Iperf does not support SCTP testing.

OpenSS7 is an organization which provides GPL (GNU Public License, *http://www.gnu.org/copyleft/gpl.html*) versions of the SS7 (Signalling System #7) stack for various Linux and UN*X systems. SS7 is a broad range of telephony signalling protocols which provides the backbone for nearly all of the switched telephony networks in the world. Along with their development of SS7, they have actively developed SCTP libraries and have modified preexisting network performance tools to implement SCTP. Among the tools, they have modified Iperf to include their own OpenSS7 SCTP libraries (for version 2.4.x Linux Kernels), as well as support for LKSCTP (Linux-Kernel SCTP libraries).

Netperf 2.4.3

The Netperf utility, currently copyrighted by Hewlett-Packard (HP) and offered freely, is primarily used for benchmarking network throughput and latency between hosts. The project is currently maintained by Rick Jones, and is provided free. Under its license, HP grants users permission to modify, copy and use the software freely. It's worth noting that contained within its license is a provision that in the future Netperf may or may not be offered as a product of Hewlett-Packard. The current

public release includes SCTP capabilities using the LKSCTP libraries. It is available for download through Netperf's site (*http://www.netperf.org*).

Netperf 2.4.6 (OpenSS7)

This version of Netperf is also copyrighted by HP, but is currently maintained by its modifier, OpenSS7. This particular release was configured for support with the OpenSS7 STREAMS. (See section *Previous Work*.)

Others

Other network performance measuring tools which include SCTP and TCP were identified but not included within this report. Some of the other tools which were considered were sctpperf v.0.1, and SCTP TestTool (stt) v.0.9.6. Future work may include integrating those tools into the work presented in this paper.

Previous Work

Along with the widely available LKSCTP (Linux Kernel SCTP libraries using Sockets), OpenSS7 has developed their own STREAMS from scratch. STREAMS uses full duplex character device drivers to interface the user processes and kernel system calls. The Sockets, however, uses a much tighter interface to implement the SCTP protocol using traditional Linux kernel sockets.

This STREAMS has been shown to outperform SCTP Sockets using multiple distributions of Linux [2]. Along with this finding, it was also shown that much of the stigma associated with using STREAMS, such as being slow and inefficient, is not accurate [2].

Other previous work includes work from [3]. This paper has shown that enabling Nagle's algorithm during SCTP stream testing can cause a sharp drop in reported throughput as the RTT (round-trip time) value is increased. Unfortunately only the Netperf version with STREAMS could disable the Nagle Algorithm. The LKSCTP libraries did not allow for this feature to be tested.

Taking this information, as well as other work that has been done comparing SCTP and TCP, this paper attempts to find the best tool for analyzing protocol performance in the ASC WAN network. It should be understood that an optimal SCTP performance testing tool which performs well on the ASC WAN network may not be optimal for different environments.

Chapter 2

Laboratory Configuration

The test bed consisted of two hosts, a client and server, connected through a Spirent Adtech AX/4000 Network Impairment Emulator. This emulator has the ability to vary the RTT value between the hosts thus imitating the performance of the ASC WAN. See Figure 2.1.

Spirent^{TM}Adtech AX/4000

This broadband test system provides many different capabilities for network testing, however only one feature, varying the RTT value, was used to benchmark the testing tools within this paper. Because of the approximate 40ms RTT between various hosts in the ASC WAN network, it's necessary to vary the RTT values during performance tests to see how well the testing tools will perform in this environment. This test system provided the ability to do that.

Ubuntu 7.04 Server

Table 2.1 shows the individual specifications for this machine. This server differs from the client only by the Ubuntu/Kubuntu Linux distributions.

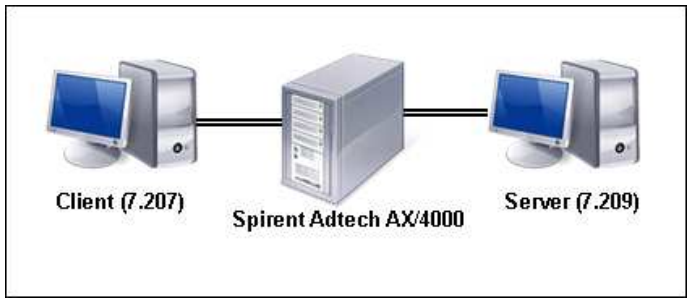

Figure 2.1. Lab Setup

Table 2.1. Client and Server Specifications

AMD©Opteron™2.0 Ghz
1 GB RAM
x86_64 2.6.20-15-generic Linux Kernel
Intel®PRO/1000 Network Card
TSO and CKO enabled
Jumbo-Frames enabled (9000 Byte MTU)
16MB [r/w]mem_max
4096 87380 16777216 tcp_rmem
4096 65536 16777216 tcp_wmem
TCP Reno
TCP SACK, Window Scaling, and Timestamps
2500 netdev_max_backlog

There are two interfaces which are used on this machine. One interface, referred to as the administrative interface, is used to connect through SSH remotely. This interface used *subnet 4*. The other interface, referred to as the test interface, used *subnet 7*. This server will be referenced from here on as the *server*.

Kubuntu 7.04 Client

For the specifications, see Table 2.1. As described above, the only difference between the client and server was the Linux distribution.

Just like the server, this client incorporates two interfaces. The administrative interface on this machine used *subnet 4*, and used the test interface on *subnet 7*. This machine will be referenced here after as the *client*.

Chapter 3

Analysis and Evaluation

The analysis and evaluation of the network test utilities (Netperf and Iperf) will be outlined as follows. Each protocol (and version) is noted, and is given a standard grade of A/B/C/D/F (A=Excellent ... F=Fail) when answering the previous listed metrics for evaluation. These metrics are:

- (1) How effectively can the tool reach a throughput near the bandwidth?
- (2) How much of the CPU does the tool utilize during operation?
- (3) Is the tool freely and widely available?
- (4) Is the tool actively developed? To what extent?

These metrics are fairly important, and collectively represent the top characteristics of a proper testing tool. The ability to reach the throughput is very critical in an effective testing tool, as is the ability to not overuse the processor during testing. Whether or not the tool is freely or widely available would probably be the least important metric listed above, while active development is very critical.

For each metric, it is important to understand what the *correct* answers to those questions are. The following are explanations of how the testing tools were rated:

(1): The theoretical bandwidth can be calculated manually by knowing a little information about the network. First, the test network uses a 1-gig Ethernet/IP network. For each data packet, the total data transferred (including headers at all layers) is 9018 bytes. That breaks down to:

	8952	data payload
+	16	SCTP chunk header
+	12	SCTP common header
+	20	IP header
+	18	Ethernet header/CRC
	9018	Total bytes transmitted per packet

Table 3.1. Grades For Given Throughput and CPU Utilizations

Grade	Throughput	CPU Utilization
A	$\geq$ 900 Mb/sec	$\leq$ 19%
B	800-899 Mb/sec	20-39%
C	700-799 Mb/sec	40-59%
D	600-699 Mb/sec	60-79%
F	$\leq$ 599 Mb/sec	$\geq$ 80%

The payload of data within that packet is 8952 bytes, giving us 99.27% of the packet for data, and 0.74% of the packet for header/footer information. Theoretically we should get throughput readings (when completely filling the pipe) at 992.7Mb/sec. The link-layer preamble was not considered in this calculation. See Table 3.1 for the grade distribution:

(2): Ideally we want to be using as little CPU resources as possible. While this is not possible, nor is it probable, Table 3.1 shows an optimistic ranking of CPU utilizations.

(3): Whether or not a tool is freely and widely available is fairly straight forward. Considering *widely available* and *freely available* are not usually mutually exclusive, the grade will be given on how widely and freely available the tool appears to be. This will, of course, be subjective.

(4): The analysis of the level of development for any software project is also subjective. Because of this, there are descriptions for the reasoning behind the grade given for each tool in the respective sections.

Each subsection following will describe and analyze each of the testing tools using the above criteria.

Iperf 2.0.6, LKSCTP

Test Results

Figures 3.1 and 3.2 show the throughput and CPU test results, respectively. These results were obtained by adjusting the RTT value and issuing the following command on the client:

*$./iperf -c server -B client -z -w [0.5*window_size] -l 8952 -t 30*

The *window_size* is the calculated window size for that specific bandwidth-delay product. The window size specified in the command line is reported by Iperf to be doubled, which is why the value passed is half of the actual window size. At the server level,

Table 3.2. Data Collected during Iperf (LKSCTP) Test

RTT	SCTP		TCP	
	Throughput	CPU Utilized	Throughput	CPU Utilized
0 ms	989 Mb/s	100 %	994 Mb/s	63 %
5	989	100	991	52
10	999	100	987	30
15	976	100	983	28
20	927	100	979	33
25	877	95	974	32
30	842	92	969	25
35	801	82	965	23
40	740	44	959	18
45	669	32	957	34
50	620	28	951	34
55	605	26	873	25
60	590	25	800	29

the following command was issued:

$./iperf -s -B server -z -w 5M

The libraries which were used on both machines were *libsctp1 version 1.0.6.dfsg-4* and *libsctp-dev version 1.0.6.dfsg-4*. These provided the user-space access to the LKSCTP libraries.

When analyzing the data using Wireshark, it was observed that the correct slow-start for SCTP was maintained, and only single chunks with data payloads of size 8952 bytes were within the 9018 byte frames.

Throughput Grade

B

This testing tool seemed to perform fairly well in achieving the theoretical throughput at lower RTT values. The only problem shown was that after 15ms, performance on achieving the accurate throughput was declining. Between 0ms and 60ms, the average throughput achieved was 816Mb/sec.

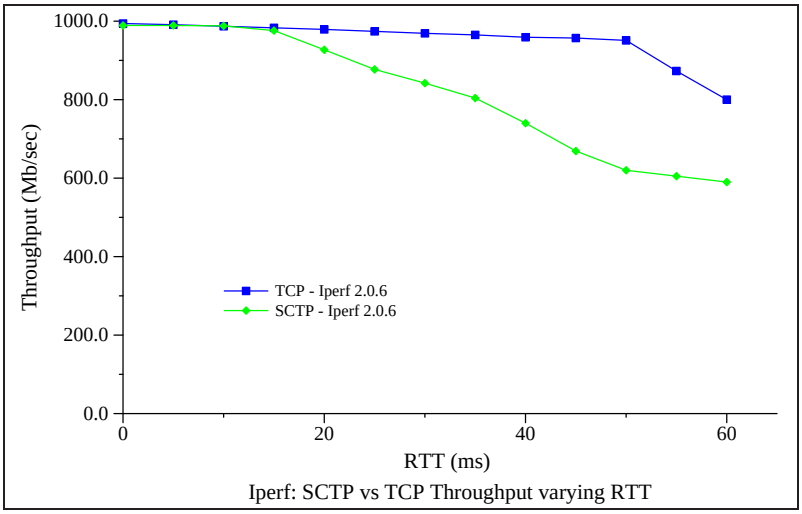

Figure 3.1. Iperf Throughput Results, SCTP vs. TCP varying RTT

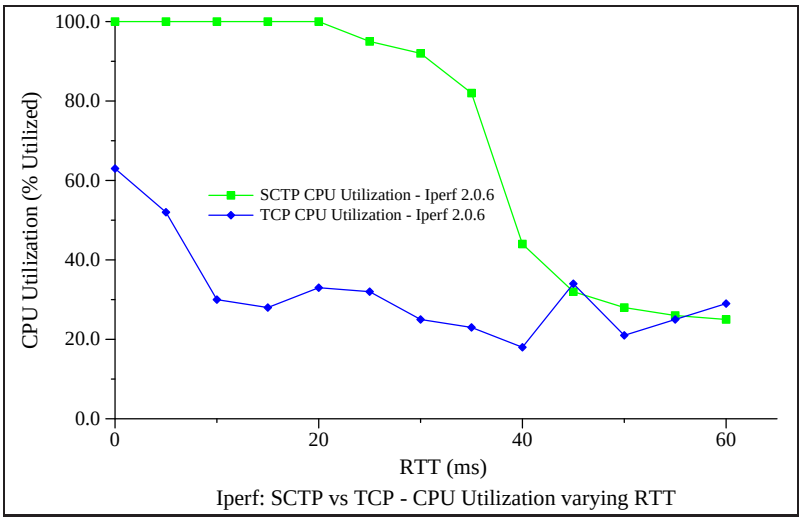

Figure 3.2. Iperf CPU Results, SCTP vs. TCP varying RTT

CPU Grade

D

The CPU utilization showed very poor performance. At low RTT values the CPU was over utilized, which is not a characteristic that is attractive in a performance testing tool. These results could be stemming from multiple areas. CKO (Checksum Offloading) and TSO (TCP Segmentation Offloading) are implemented through TCP which reduces the amount of work that the CPU must do during high levels of communication. Developers from the Netperf project described how these features are not currently implemented within SCTP. The average CPU Utilization was 71%.

Availability Grade

A

This tool is widely and freely available. Because of its easy-to-use interface, Iperf is a very popular choice for network performance testing.

Development Grade

B

It's not known whether or not NLANR/DAST is still actively developing the Iperf tool. But because their latest release was in 2005, it may be losing ground on active development. Aside from that, OpenSS7 has been creating their own modified versions of Iperf.

Overall Grade

B

Iperf has a lot of room for improvement, but has shown some characteristics of an effective network performance tool. From these tests, it probably does not fully meet the requirements for an SCTP test tool on the ASC WAN.

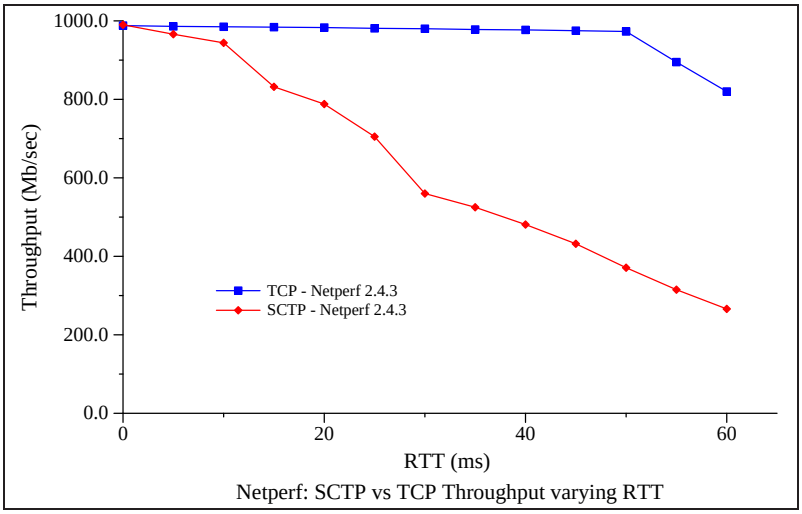

Figure 3.3. Netperf Throughput Results, SCTP vs. TCP varying RTT

Netperf 2.4.3, LKSCTP

Test Results

Figures 3.3 and 3.4 show the throughput and CPU tests, respectively. As with the Iperf tests above, these results were obtained by evaluating the CPU utilization and reported throughput as the RTT values were varied. At the server level, the following command was issued:

$./netserver

The command issued at the client level was:

$./netperf -i 9,3 -I 95,5 -t SCTP_STREAM -l 30 -H server – -m 8952 -S 16777216 -s 16777216

This provided the client with a +/- 2.5% @ 95% confidence for the results with a minimum of 3 tests and a maximum of 9 tests to validate the specific result. It also assured that the message sizes plus all header information would equal the MTU size, as well as insure that both client and server would utilize 16MB buffers. As with Iperf, the libraries which were used on both machines were *libsctp1 version 1.0.6.dfsg-4* and *libsctp-dev version 1.0.6.dfsg-4*

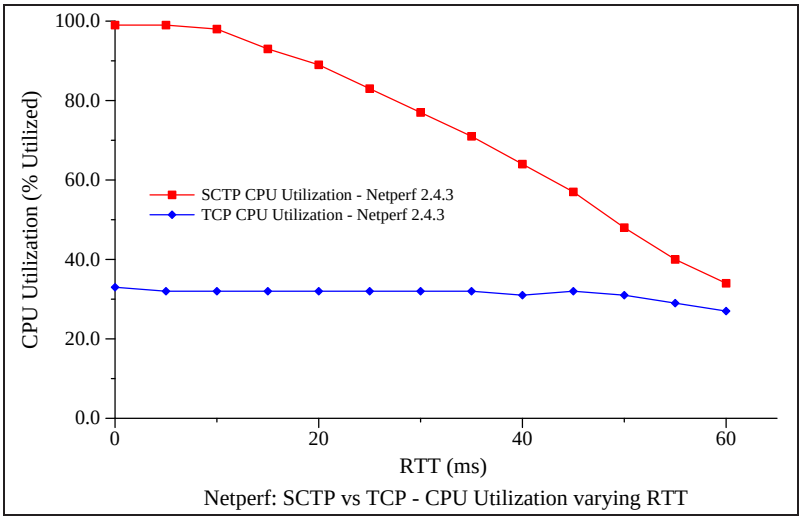

Figure 3.4. Netperf CPU Results, SCTP vs. TCP varying RTT

Throughput Grade

C

The throughput shown during this test was very similar to the Iperf test, but Netperf showed a lower average throughput than the Iperf test did. Netperf performed well up to 15 ms RTT, and then drastically dropped off from there. The average throughput between values of 0 and 60 ms RTT was 629 Mb/s. This tool would work well with very small RTT values, but is not efficient enough for use on the ASC WAN at this time.

CPU Grade

D

The average CPU Utilization from 0 to 60 ms RTT was 73%. According to Table 2, this gives this part of the tool a D grade. Much like the Iperf test, the Nagle algorithm was not allowed to be disabled, which could have improved both throughput and CPU performance.

Availability Grade

B

Netperf is widely and freely available through the Netperf website. The only reason this grade was lowered was because of the potential for HP to take full control over distribution and availability. This may cause developers to hesitate when using or helping to develop this tool, since it is not released under the standard GPL.

Development Grade

A

Development for this tool seems to be very active. The Netperf website has a very concise manual and also has active developers available through the Netperf mailing list. Along with Rick Jones, the main contact for the Netperf project, other developers are available for troubleshooting. The latest release through the Netperf site was in February of 2007, version 2.4.3.

Overall Grade

C

This version of Netperf has shown functionality for SCTP, but seems to perform worse than Iperf. It is not recommended for use on the ASC WAN because of its poor performance near the RTT values that are seen on the ASC WAN.

Netperf 2.4.6, STREAMS

Test Results

Figures 3.5 and 3.6 show the throughput and CPU utilization results from the SCTP STREAMS test using the OpenSS7 Fast-STREAMS. The TCP test results are not included because the TCP STREAMS was not working correctly. When testing the TCP STREAMS, one or both of the test machines would freeze. Regardless, that data was not critical to analyze and rank the functionality of SCTP STREAMS through Netperf 2.4.6. The command issued at the client level during the test was:

$ ./netperf -i 9,3 -I 95,5 -l 30 -t XTI_SCTP_STREAM -h server – -m 8952 -X /dev/sctp_t -D -S 16777216 -s 16777216

Table 3.3. Data Collected during Netperf (LKSCTP) Test

RTT	SCTP		TCP	
	Throughput	CPU Utilized	Throughput	CPU Utilized
0 ms	990 Mb/s	99 %	988 Mb/s	33 %
5	966	99	986	32
10	944	98	985	32
15	832	93	984	32
20	788	89	983	32
25	705	83	981	32
30	560	77	980	32
35	525	71	978	32
40	481	64	977	31
45	432	57	975	32
50	371	48	973	31
55	315	40	895	29
60	266	34	820	27

Nagle's Algorithm was disabled in this test, but did not provide any "better" results when toggling its functionality on and off. At the server level:

$ ./netserver

See Table 3.4. As is shown, there is a sharp degradation in throughput performance from 0 ms RTT to 5 ms RTT. This degradation also follows a very low CPU utilization as the RTT increases. Upon further analysis, it seemed that the STREAMS was not *opening* its congestion window correctly. By observing the data on the network being sent from client to server, no more than 2 segments of size 8952 bytes were sent at a time. At nearly zero RTT, this was acceptable, but once delay is introduced, the reported throughput was very low.

Throughput Grade

F

Because of the low throughput within this test, the throughput portion of this grade resulted in a failing grade. The average throughput reported from 0 ms to 60 ms RTT was 89.6 Mb/sec.

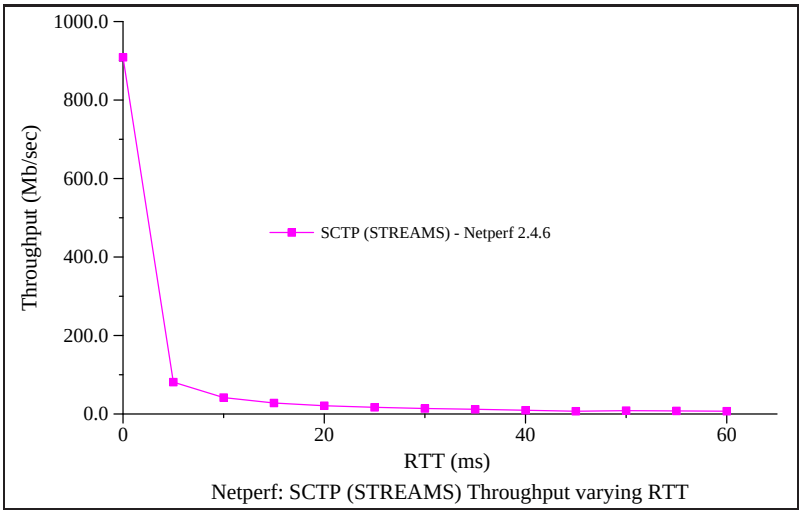

Figure 3.5. Netperf Throughput Results, SCTP (STREAMS) varying RTT

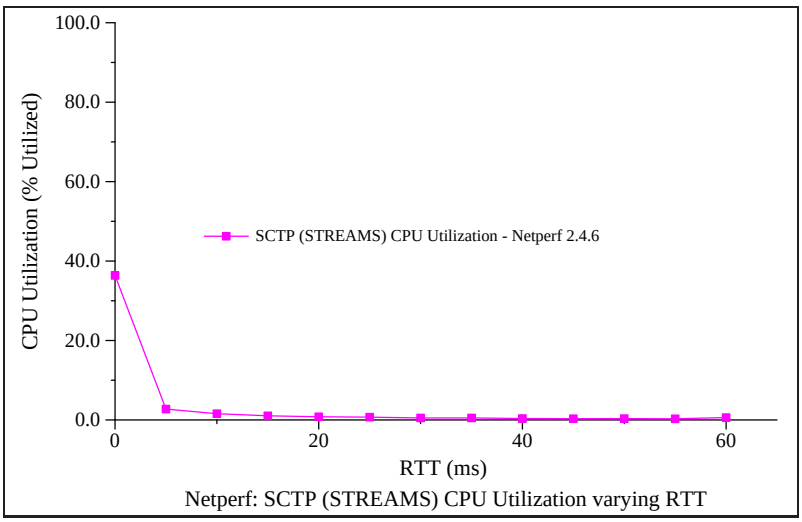

Figure 3.6. Netperf CPU Results, SCTP (STREAMS) varying RTT

Table 3.4. Data Collected during Netperf STREAMS Test

RTT	SCTP	
	Throughput	CPU Utilized
0 ms	909 Mb/s	36 %
5	81	3
10	42	2
15	28	1
20	22	1
25	17	1
30	14	1
35	12	1
40	10	0
45	7	0
50	9	0
55	8	0
60	7	1

CPU Grade

undef.

Because of the issues of low throughput, there was only one data point which could be considered for the CPU utilization. The first data point had a utilization of 36%, which may have resulted in a B grade, but because of the lack of other data points, this metric wasn't able to be graded.

Availability Grade

B

The OpenSS7 STREAMS is available freely through the OpenSS7 website under the GNU public license. The Netperf utility (as stated before) is not available through the GNU public license, but is currently available free through HP.

Development Grade

A

Frequent updates have been released through the OpenSS7 website for some time. The newest STREAMS, version 0.9.2.3, was released in June 2007.

Table 3.5. Grade Card

	Iperf 2.0.6	Netperf 2.4.3	Netperf 2.4.6
Throughput Grade	B	C	F
CPU Grade	D	D	*undef.*
Availability Grade	A	B	B
Development Grade	B	A	A
Overall Grade	B	C	F

Overall Grade

F

In its current implementation, Netperf 2.4.6 with SCTP STREAMS is unable to meet the needs of the ASC WAN as a functional network SCTP testing tool. It showed very poor performance with throughput, and left the CPU test up in the air.

Chapter 4

Future Work

Based on [3], the Nagle Algorithm should improve the performance for SCTP at higher RTT values. Future work should definitely include reproducing those results, and seeing if they raised the grades of either of these tools.

Other work could include an analysis of the performance through the LKSCTP and STREAMS implementation. This analysis should delve into whether or not TSO (Transmission Sequence Offloading) or CKO (Checksum Offloading) are viable options for SCTP as they are for TCP. If so, they may lower the CPU utilizations at the lower RTT values by routing some redundant computation to the NIC cards.

It's also curious whether or not STREAMS has correctly implemented the slow-start congestion control algorithm that the RFC specifies. Analysis of why the throughput and CPU utilization show sharp degradations after small delays would provide progress for that SCTP implementation.

Finally, it would be interesting seeing results of how the other tools such as STT and sctpperf were to perform with the LKSCTP and STREAMS implementations of SCTP.

Chapter 5

Conclusion

The goal of this research was to find an appropriate performance testing tool for comparing SCTP and TCP traffic accurately. At low RTT values, Netperf 2.4.3 (LKSCTP) and Iperf 2.0.6 (LKSCTP) worked very well in effectively comparing SCTP and TCP, but as the RTT values increased, especially near the RTT values of the ASC WAN, neither was sufficient.

Netperf 2.4.6 (STREAMS), on the other hand, was unable to provide any comparative results for SCTP and TCP traffic. This version of Netperf seemed to work well at 0 ms RTT for SCTP, but did not work as well at increased RTT values. It should be noted that this version of Netperf (2.4.6) is not as popular as the LKSCTP version (2.4.3), purely because the future development of STREAMS is unknown and not as widely used to date.

In terms of ranking the three test tools which were analyzed from best to worst, they are: Iperf 2.0.6 (LKSCTP), Netperf 2.4.3 (LKSCTP), and finally Netperf 2.4.6 (STREAMS). See the table below for a review of the grades given to each test tool.

Grade Card

	Iperf 2.0.6	Netperf 2.4.3	Netperf 2.4.6
Throughput Grade	B	C	F
CPU Grade	D	D	*undef.*
Availability Grade	A	B	B
Development Grade	B	A	A
Overall Grade	B	C	F

References

[1] Judy I. Beiriger, Hugh P. Bivens, Steven L. Humphreys, Wilbur R. Johnson, and Ronald E. Rhea. Constructing the asci computational grid. In *IEEE International Symposium on High Performance Distributed Computing, Proceedings.*, pages 193–199. Sandia National Laboratories, 2000.

[2] Brian F. G. Bidulock. Streams vs. sockets performance comparison for sctp. OpenSS7 Corporation, June 2007.

[3] Flavius Copaciu, Virgil Dobrota, Tudor Blaga, and Bagdan Moraru. Performance analysis of stream transmission control protocol. In *Symposium of Electronics and Telecommunications (ETC2004)*. Technical University of Cluj-Napoca, 2004.

[4] Tan C. Hu, Lawrence F. Tolendino, and Jason S. Wertz. Upgrading the snl/nm asc wan router: Connecting the snl/nm hpc to the tri-lab community. Sandia National Laboratories, 2007.

[5] Randall R. Steward and Qiaobing Xie. *Stream Control Transmission Protocol (SCTP) A Reference Guide*. Addison Wesley, 2002.

DISTRIBUTION:

1	MS 0630	A.L. Hale, 9600
1	MS 0662	T. Klitsner, 9330
1	MS 0788	P.A. Manke, 9338
1	MS 0788	V.K. Williams, 9334
1	MS 0795	P.C. Jones, 9317
1	MS 0801	R.W. Leland, 9300
1	MS 0801	D.S. Rarick, 9310
1	MS 0801	D.R. White, 9340
4	MS 0806	Len Stans, 9336
1	MS 0806	J.L. Akins, 9336
1	MS 0806	J.P. Brenkosh, 9336
1	MS 0806	J.M. Eldridge, 9336
1	MS 0806	A. Ganti, 9336
1	MS 0806	T.C. Hu, 9338
1	MS 0806	S.A. Gossage, 9336
1	MS 0806	B.R. Kellogg, 9336
6	MS 0806	C.L. Lydick, 9336
1	MS 0806	J.H. Maestas, 9336
1	MS 0806	J.H. Naegle, 9336
1	MS 0806	T.J. Pratt, 9338
1	MS 0806	L.F. Tolendino, 9334
1	MS 0806	J.S. Wertz, 9336
1	MS 0813	G.K. Rogers, 9329
1	MS 0813	R.M. Cahoon, 9311

1	MS 0823	J.D. Zepper, 9320
1	MS 9012	C.T. Deccio, 8949
1	MS 9012	R.D. Gay, 8949
1	MS 9151	C.T. Oien, 8940
1	MS 9158	H.Y. Chen, 8961
2	MS 9018	Central Technical Files, 8944
2	MS 0899	Technical Library, 4536