NASA/TM—1999-209040

High-Speed TCP Testing

David E. Brooks
Sterling Software, Cleveland, Ohio

Holly Gassman
Case Western Reserve University, Cleveland, Ohio

Dave R. Beering
Infinite Global Infrastructures, LLC, Chicago, Illinios

Arun Welch
Anzus Consulting, Orleans, Massachusetts

Douglas J. Hoder and William D. Ivancic
Glenn Research Center, Cleveland, Ohio

National Aeronautics and
Space Administration

Glenn Research Center

May 1999

Trade names or manufacturers' names are used in this report for identification only. This usage does not constitute an official endorsement, either expressed or implied, by the National Aeronautics and Space Administration.

Available from

NASA Center for Aerospace Information
7121 Standard Drive
Hanover, MD 21076
Price Code: A03

National Technical Information Service
5285 Port Royal Road
Springfield, VA 22100
Price Code: A03

HIGH-SPEED TRANSMISSION CONTROL PROTOCOL (TCP) TESTING

David E. Brooks
Sterling Software
Cleveland, Ohio

Holly Gassman
Case Western Reserve University
Cleveland, Ohio

Dave R. Beering
Infinite Global Infrastructures, LLC
Chicago, Illinois

Arun Welch
Anzus Consulting
Orleans, Massachusetts

Douglas J. Hoder and William D. Ivancic
National Aeronautics and Space Administration
Glenn Research Center
Cleveland, Ohio

SUMMARY

Transmission Control Protocol (TCP) is the underlying protocol used within the Internet for reliable information transfer. As such, there is great interest to have all implementations of TCP efficiently interoperate. This is particularly important for links exhibiting long bandwidth-delay products. The tools exist to perform TCP analysis at low rates and low delays. However, for extremely high-rate and long-delay links such as 622 Mbps over geosynchronous satellites, new tools and testing techniques are required. This paper describes the tools and techniques used to analyze and debug various TCP implementations over high-speed, long-delay links.

BACKGROUND

The Space Program Office and the Communication Technology Division of the NASA Glenn Research Center have been working with the United States satellite communication industry over the past 15 years to develop advanced technologies and networking techniques to improve commercial satellite communications. In general, these technologies and networking techniques can be directly applied to NASA operations and NASA missions. The Advanced Communication Technology Satellite (ACTS) and associated experiments are a primary example of this technology development.

With the recent explosion of the Internet and the enormous business opportunities available to communication system providers, great interest has developed in improving the efficiency of data transfer using Transmission Control Protocol (TCP). The satellite system providers are interested in solving TCP efficiency problems associated with long delays. Similarly, the terrestrial community is interested in solving TCP problems over high-bandwidth links. The common factor is a communication link exhibiting a large bandwidth-delay product.

Even before the recent explosion of the Internet, NASA Glenn Research Center had been working with various users such as Boeing Aircraft (ref. 1), Ohio Super Computer (ref. 2), and the Aries Project (ref. 3) to distribute large data sets over satellites. During this time, NASA Glenn heavily researched the current state of the TCP protocol and its limitations. As a result, NASA Glenn realized that solutions had already been proposed for most of the problems associated with efficient data transfer over large bandwidth-delay links (which include satellite links). The solutions are detailed in various Internet Engineering Task Force (IETF) Request For Comments (RFC's). Unfortunately, most of these solutions had not been implemented at high speed (155+ Mbps). Therefore, the NASA ACTS Experiments

Program initiated a series of TCP experiments to demonstrate scalability of Transmission Control Protocol/Internet Protocol (TCP/IP) and determine how far the protocol can be optimized over a 622-Mbps satellite link. These experiments were known as the 118i and 118j experiments. During the 118i and 118j experiments, NASA Glenn worked closely with SUN Microsystems and FORE Systems to improve the operating system, TCP stacks, and network interface card drivers. This collaboration resulted in the ability to obtain data throughput rates of greater than 520 Mbps by using TCP over Asynchronous Transfer Mode (ATM) over a 622-Mbps Synchronous Optical Network (SONET) OC12 link. Following the success of these experiments and the successful government/industry collaboration, a new series of experiments—the 118x experiments—were developed. The objective of 118x was to demonstrate the interoperability of TCP/IP over OC12 ATM in a multivendor environment that uses ACTS. Participants included FORE, CISCO, SUN, Microsoft, Compac, Lockheed/Martin, Hughes, NASA Glenn, Sprint, and AMPEX.

During the 118x experiments, it became evident that better tools were needed to debug and analyze the TCP stack—particularly with multiple vendors involved. The remaining sections of this paper briefly describe some of the issues being addressed with TCP over a satellite link (or other large-bandwidth-delay link). A detailed description of the testing methods and the tools used in the extremely high-speed environment is provided.

CONDITIONS WHICH AFFECT TCP EFFICIENCY

Three issues needed to be addressed when considering TCP performance: congestion, the bandwidth-delay product, and bit errors.

Beginning in the fall of 1986, the Internet began showing signs of congestion collapse. Van Jacobson studied this phenomenon in 1988, and congestion control algorithms such as the slow start algorithm were proposed (ref. 4). These algorithms have been continually enhanced and provide an elegant solution to congestion control in an environment consisting of multifaceted users operating on a variety of interconnected networks, the Internet. These algorithms—slow start in particular—result in inefficient bandwidth utilization for end-to-end communications where a moderate amount of data is being transferred over a link exhibiting large bandwidth-delay characteristics.

Networks with bandwidth-delay products greater than 65,535 bytes are referred to as long fat networks (LFN's). The 16-bit Window field in standard TCP results in this 65,535-byte Window limitation. Also, there is a possibility that packet sequence numbers could be used more than once in an LFN. Adding extensions to TCP for scaled windows and time stamps solves these problems. The specification that defines these extensions is found in RFC 1323, TCP Extensions for High Performance (ref. 5).

Currently, any loss of TCP data is considered to be caused by congestion. As such, congestion control algorithms may be triggered for congestion or when data experiences corruption. The TCP fast retransmit and fast recovery[1] and the selective acknowledgment options (ref. 6) improve TCP performance in many situations where congestion and/or corruption may occur (ref. 7).

SOFTWARE TESTING TOOLS

To capture and analyze TCP performance data, three basic operations must be performed. The TCP transmission information must be captured. The captured data must be formatted into usable information, and the usable information must be displayed in a way that enables quick accurate analysis.

To capture data, network interface card (NIC) monitoring programs are used. The interfaces being monitored are ATM interfaces from FORE and SUN. The programs used prior to September 1998 were *snoop* and *atmsnoop*. *Snoop* was run on the FORE interface while *atmsnoop* was run on the SUN interface. These programs detect and show the status of incoming data streams and identify all packets traversing the interface. In addition, the programs are capable of identifying packet types. From September 1998 on, a modified version of *tcpdump* which performs the same general functions as *snoop* and *atmsnoop* was used.

Tcpdump, tcptrace (ref. 8) with slight modifications, and XPLOT were used to analyze the data. *Tcpdump* outputs the raw data in text format. This is most useful for performing a detailed analysis of specific areas of interest that are identified by using other analysis tools such as XPLOT (ref. 9). *Tcptrace* is a *tcpdump* file analysis tool program. *Tcptrace* reads output dump files in the formats of several popular packet-capturing programs such as

[1]Van Jacobson's Note to the IEFT-end2end Internet Working Group, April 1990.

tcpdump and *snoop*. *Tcptrace* analyzes numerous parameters. A small sample of the commonly referenced parameters include transmission time, round-trip time, window size, segment size, total number of packets sent, elapsed time, bytes/segments sent and received, rétransmissions, window advertisements, throughput, and numerous other parameters. *Tcptrace* can also produce output files for three different types of graphs: throughput, round-trip time, and time sequence. These graphs can be displayed by using XPLOT. The throughput graph shows the instantaneous throughput of the connection as a function of time (averaged over a specified number of segments). The round-trip time graph shows the round-trip times for the acknowledgments (ACK's) as a function of time. The time-sequence graph shows segments sent and ACK's returned as a function of time. This is the most widely used method to analyze TCP operation and implementations. Figure 1 shows an example of an XPLOT output. The abscissa is the TCP packet sequence number and the ordinate is time. The bottom line represents the acknowledgments, and the top line represents the congestion window. The marks between these two lines are the transmitted packets. With XPLOT one can easily visualize the acknowledgments, retransmissions, selective retransmission, congestion window, individual packets, and retransmitted packets and identify unusual situations occurring in the protocol. Thus, this is a very useful tool for troubleshooting TCP implementation or interoperability problems.

The program used to test TCP interoperability and performance is *ttcp*,[2] the "Test TCP" program. *Ttcp* is useful for network performance testing with both TCP and User Datagram Protocol (UDP). *Ttcp* sends normal Internet Protocol (IP) datagrams, which are handled just like any other user data within the network. *Ttcp* uses memory-to-memory transfers rather than disk-to-disk transfers thereby allowing the traffic transmitter and receiver to operate at

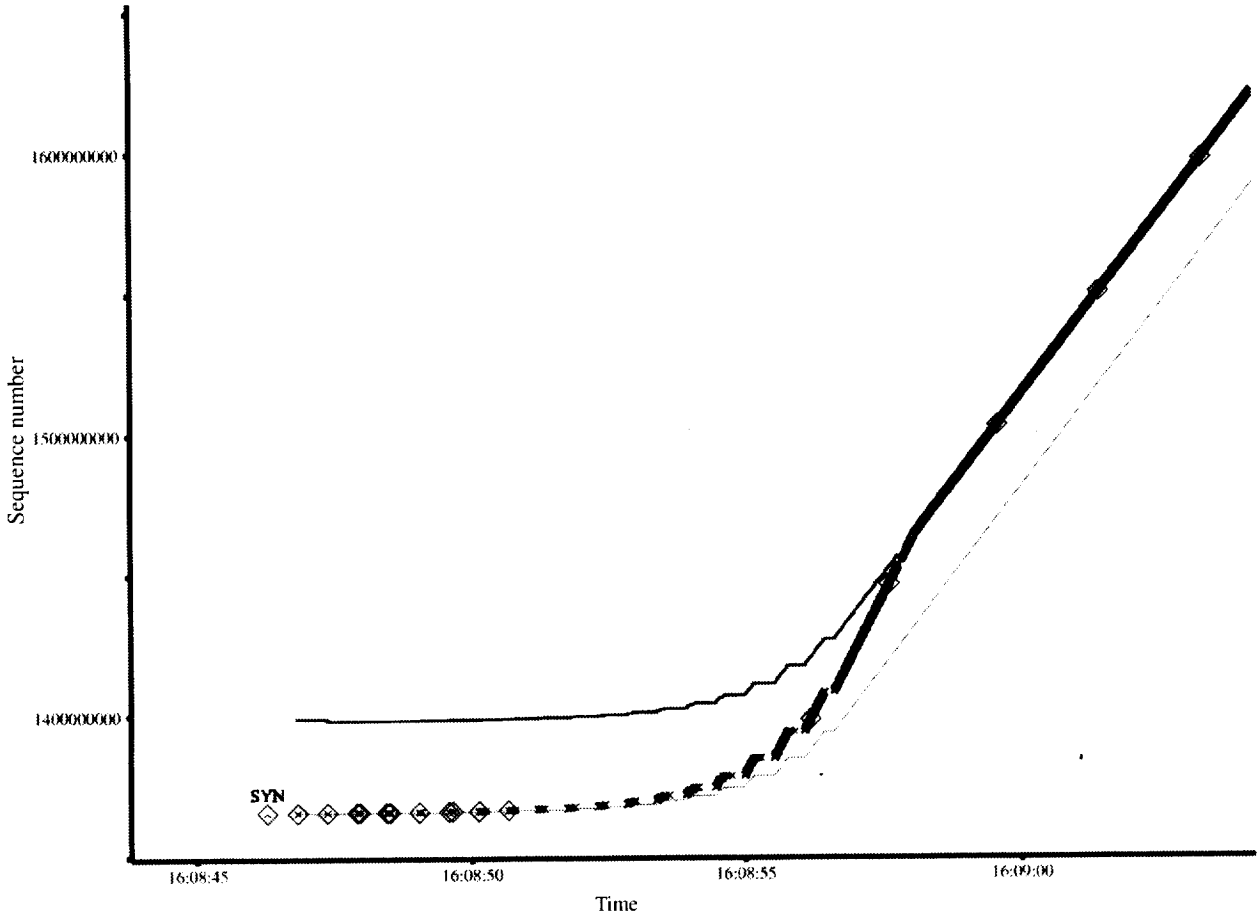


Figure 1.—Example of XPLOT output.

[2]The source for TTCP is available from a variety of sources on the Internet. A good explanation of the use and operation of *ttcp* is available at http://www.ccci.com/product/network_mon/tnm31/ttcp.htm.

TABLE I.—TCP PERFORMANCE

TTCP PERFORMANCE	
Mode	TCP - Receive
Packet size	257000 bytes
Number of packets	3334
Socket buffer size	14283500 bytes
REAL buffer size	14294448 bytes
Connection from	10.0.0.1 (tardis-oc12)
Throughput	161.84 (Mbits/sec)
Elapsed time: 41.36 sec	
CPU usage: 37%	
Received: 42488 pkts	
856838000 bytes in 15.43 CPU sec = 54229.15 Kb/CPU sec	
42489 I/O calls, msec/call = 1.00, calls/sec = 1027.26	
0.2user 15.1sys 0:41real 37% 0i+0d 0maxrss 0+0pf 17364+678csw	
buffer address 0x28000	

true network speeds. This makes *ttcp* extremely useful for the evaluation of high performance networks. The output of a typical *ttcp* run is shown in table I. Note that care should be taken when reading the throughput numbers because often the number is calculated using 1024 bytes as a kilobyte rather than 1000 bytes resulting in a 2.4 percent mismatch in actual and displayed throughput results. In order to determine whether this has occurred, the *ttcp* source code has to be analyzed (ref. 10).

For this testing, some modifications were made to *tcpdump* and *tcptrace*. *Tcpdump* was modified to handle two interfaces on one machine (See the section Experiment Configuration). Since the modified *tcpdump* saved information in a slightly different format than public domain *tcpdump, tcptrace* was modified to handle this format. A wrapper was written around *ttcp* to include scripts for recording additional information related to the experiment configuration. Such information included the ground station statistics, software versions, workstation statistics, SONET ATM layer statistics from the switches at both the ACTS High Data Rate Terminal's ATM port and the workstation's ATM port, TCP/IP statistics, TCP/IP settings, workstation driver information, type of ATM interface, special workstation settings, and other statistics.

EXPERIMENT CONFIGURATION

Figure 2 shows the general configuration of the experiment. The bulk data flow is from server to client. The server and client workstations may be SUN UltraSparcs, Intel Xeons, Compac Alphas, SGI Onyx2s, or any other computer. The network interface card in these machines are ATM based and use SONET OC3 or OC12. The ATM switches are FORE ASX1000s. The FORE ATM switches allow for multicast PVC's—instead of the ATM switches, optical splitters could have been used to effectively multicast the signals to the monitor. The monitor is a SUN 2xUltraSpac2 with dual 200-MHz processors, and the operating system is SUN's Solaris 2.7. The monitor has 512 Mbytes of system memory and can write to the disk drive at a peak rate of 70 Mbytes/sec. The server's data packets are forwarded to the client and to the OC12 port (SUNATM 622 SBUS) of the monitor. The client's acknowledgment packets are forwarded to the server and to the OC3 port (FORE SBA 200E) of the monitor.

To capture TCP data under normal low-speed conditions, *tcpdump* can operate on the same machines as *ttcp* or some other application that uses TCP. However, at the operation speeds of interest, running *tcpdump* on the same machine as the application severely affects the throughput. Therefore a machine other than the server and client is needed to capture packets. This machine is called the monitor. Two separate machines can be used to monitor the links: one for transmitted packets and one for acknowledgment. However, using one machine allows for more accurate time stamps and does not require synchronization between machines. Even when only one machine is used for monitoring, there still is some minor inaccuracy in the time stamp between interface cards, but not enough to affect the overall measurements of interest.

At the time of these experiments, commercial network monitoring equipment was not available to perform these tasks at 622 Mbps. It is assumed that such equipment will be available in the future. However, the tools and techniques described here should be applicable to the next increase in speed.

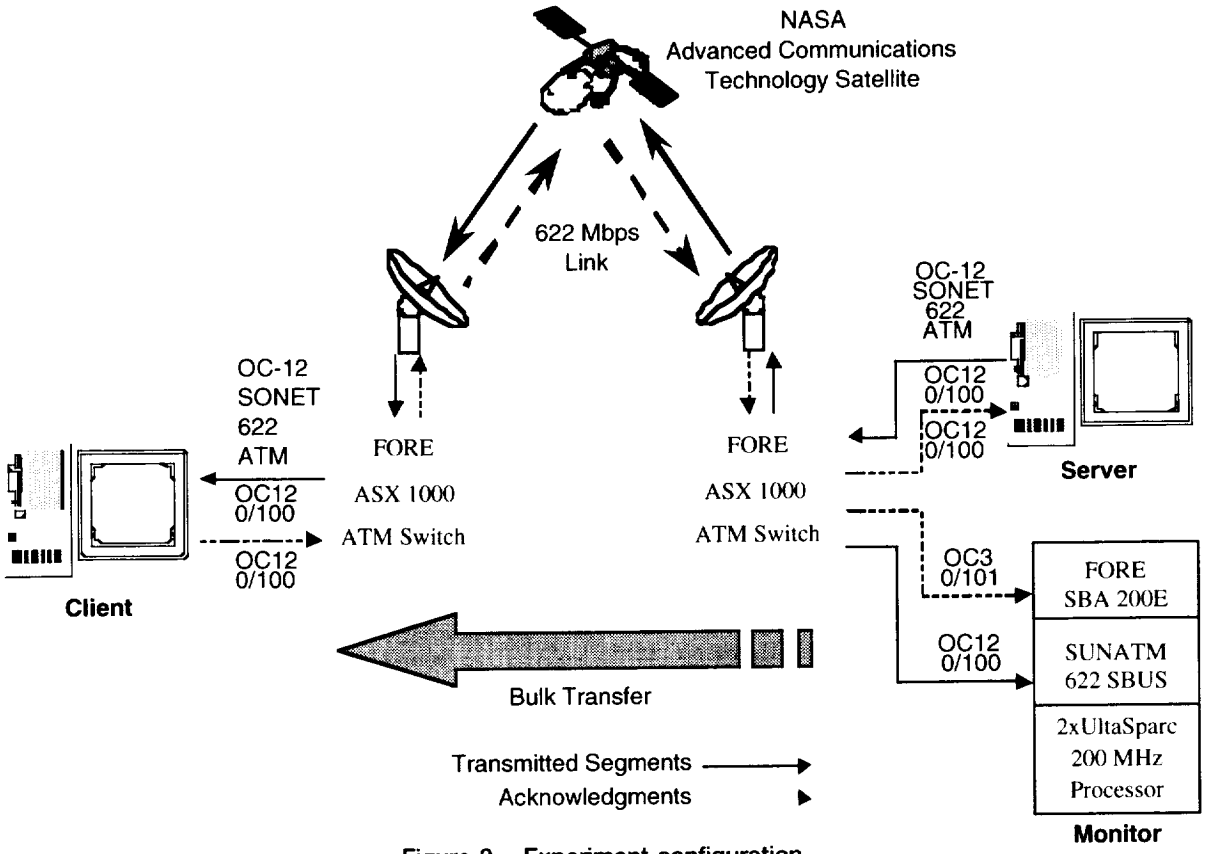


Figure 2.—Experiment configuration.

EXPERIMENT SEQUENCE

There are four basic, sequential events that must be followed for a successful TCP experiment run: configuration setup, data collection, post processing, and analysis.

The configuration setup includes ensuring that proper switch settings and network interface configurations are made. One key configuration setting that must not be overlooked is ensuring that the ARP tables are configured in the monitor. If not, the monitor will ignore data that appears to be addressed to another machine. For example, assume the server and associated port is named tardis-0c12 and that the client and associated port is named sprintsun-oc12. Then, the following entries should be in the monitor's ARP table:

```
ba0 0 100 tardis-oc12
qaa0 0 101 sprintsun-oc12
```

With these entries in the monitor, any data destined for tardis-0c12 on virtual path identifier (VPI) 0 virtual channel identifier (VCI) 100 of interface ba0 and any data destined for sprintsun-0c12 on VPI 0 VCI 101 of interface qaa0 100 will be captured. For this setup, interface qaa0 is the SUNATM 622 SBUS card, and ba0 is the FORE SBA200E card.

The data must be collected by performing the following functions:

(1) Initiate the program *tcpdump* (modified *tcpdump*) on both interfaces of the monitor machine but store only the first 124 bytes of data collected per packet. This is performed by using the following command sequence:

```
tcptrace_sunatm -s124 -i ba0 ip
tcptrace_sunatm -s124 -i qaa0 ip
```

Here, s-124 indicates save only the first 124 bytes of the packet, -i indicates which interface, and ip indicates that *tcpdump* should only capture IP packets. Classical IP is being used in these experiments. Therefore, IP packets are 9180 bytes long with 20 bytes for the IP header and up to 60 bytes for TCP header with full extensions (fig. 3). To guarantee storage of the complete TCP header data, at least 80 bytes must be captured per classical IP packet. Storing only the IP and TCP header bytes reduces the *tcpdump* output file size significantly. Furthermore, since TCP behavior is of primary interest, only the header information from each packet is needed. In addition, the hardware transfers to disk cannot keep up with the incoming data if the total 9180-byte packet is captured and stored. Trying to save all 9180 bytes will result in lost data.

(2) Run *ttcp* or whatever TCP (or IP) application you wish to monitor.
(3) Stop the two *tcpdump* sessions running on the monitor.

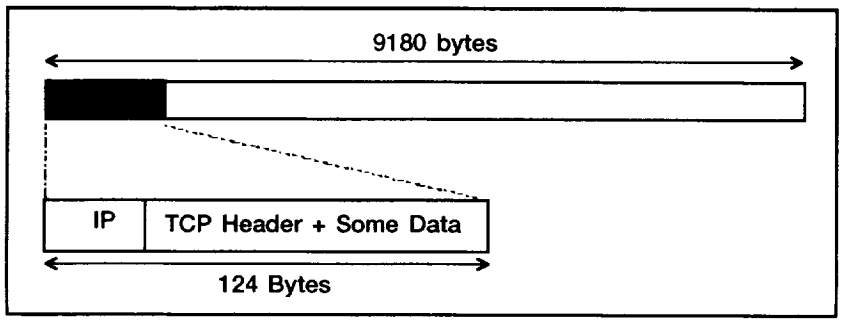


Figure 3.—Classical IP.

To perform post processing, first combine two *tcpdump* output files into a third *tcpdump* output file. This is necessary because data were captured on two unique ATM interface cards. The modified *tcpdump* program, *tcpdumpcombo*, performs this function. It assumes a SunATM-622 sbus card is being used for recording outbound traffic and a FORE SBA-200E for recording inbound traffic as two separate *tcpdump* files. Next the modified *tcptrace* program, *tcptrace_sunatm*, is performed to produce files that can be used by XPLOT. *Tcptrace_sunatm* supports output files produced from SUN's *atmsnoop* and *tcptrace_sunatm*.

The data can be analyzed using *tcpdump, tcptrace*, and XPLOT. Table II is an example of a *tcpdump* report. The time stamps, ip address, windows size, and packet sequence numbers are readily available as are the return acknowledgments. Table III is an example of a *tcptrace* long-output-format report file. Figure 4 shows sample time-sequence plots. Figure 4(a) is representative of a TCP stack that appears to be operating correctly. The exponential increase is clearly visible. Figure 4(b) shows that there are obviously problems with this TCP run—particularly since this run is over an error-free, congestion-free link. The problem may be in either or both the transmitter and receiver TCP stacks, the operating systems, the network interface card drivers, or the interaction of some or all of these entities. Figure 4(c) is a magnified portion of figure 4(b) showing the selective acknowledgment packets (noted by an "S") and retransmissions (noted by an "R"). Study of the plots in figures 4(b) and (c) indicate that some timeouts occur which may be caused by the operating systems inability to manage the large window. Note that the *tcpdump* report, *tcptrace* report, and figures 4(b) and (c) are from the same run.

TABLE II.— *tcpdump* REPORT

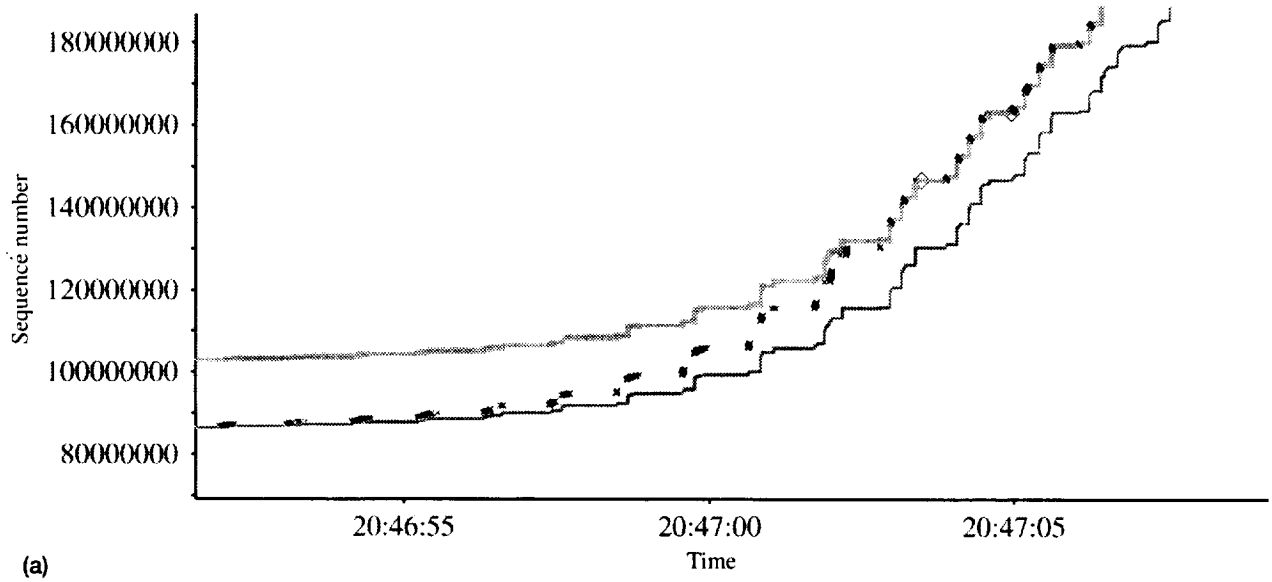
```
14:35:40.164037 10.0.0.5.5001 > 10.0.0.1.32801: . ack 2069345 win 32767 <nop,nop,timestamp 40391
1054755> (DF) [tos 0x88]
14:35:40.164093 10.0.0.1.32801 > 10.0.0.5.5001: . 2890071:2899185(9114) ack 1 win 64029
<nop,nop,timestamp 1054809 40391> (DF)
14:35:40.164114 10.0.0.5.5001 > 10.0.0.1.32801: . ack 2087573 win 32767 <nop,nop,timestamp 40391
1054755> (DF) [tos 0x88]
14:35:40.164228 10.0.0.5.5001 > 10.0.0.1.32801: . ack 2097153 win 32767 <nop,nop,timestamp 40391
1054755> (DF) [tos 0x88]
14:35:40.164230 10.0.0.1.32801 > 10.0.0.5.5001: . 2899185:2908299(9114) ack 1 win 64029
<nop,nop,timestamp 1054809 40391> (DF)
14:35:40.164381 10.0.0.1.32801 > 10.0.0.5.5001: . 2908299:2917413(9114) ack 1 win 64029
<nop,nop,timestamp 1054809 40391> (DF)
14:35:40.164493 10.0.0.5.5001 > 10.0.0.1.32801: . ack 2115381 win 32767 <nop,nop,timestamp 40391
1054755> (DF) [tos 0x88]
14:35:40.164538 10.0.0.1.32801 > 10.0.0.5.5001: . 2917413:2926527(9114) ack 1 win 64029
<nop,nop,timestamp 1054809 40391> (DF)
14:35:40.164679 10.0.0.1.32801 > 10.0.0.5.5001: . 2926527:2935641(9114) ack 1 win 64029
<nop,nop,timestamp 1054809 40391> (DF)
```

TABLE III.— *tcptrace* LONG REPORT

1 args remaining, starting with 'sun2nt_sunside_combo_tcpdump.0'
Ostermann's tcptrace -- version 5.0.4 -- Thu May 14, 1998

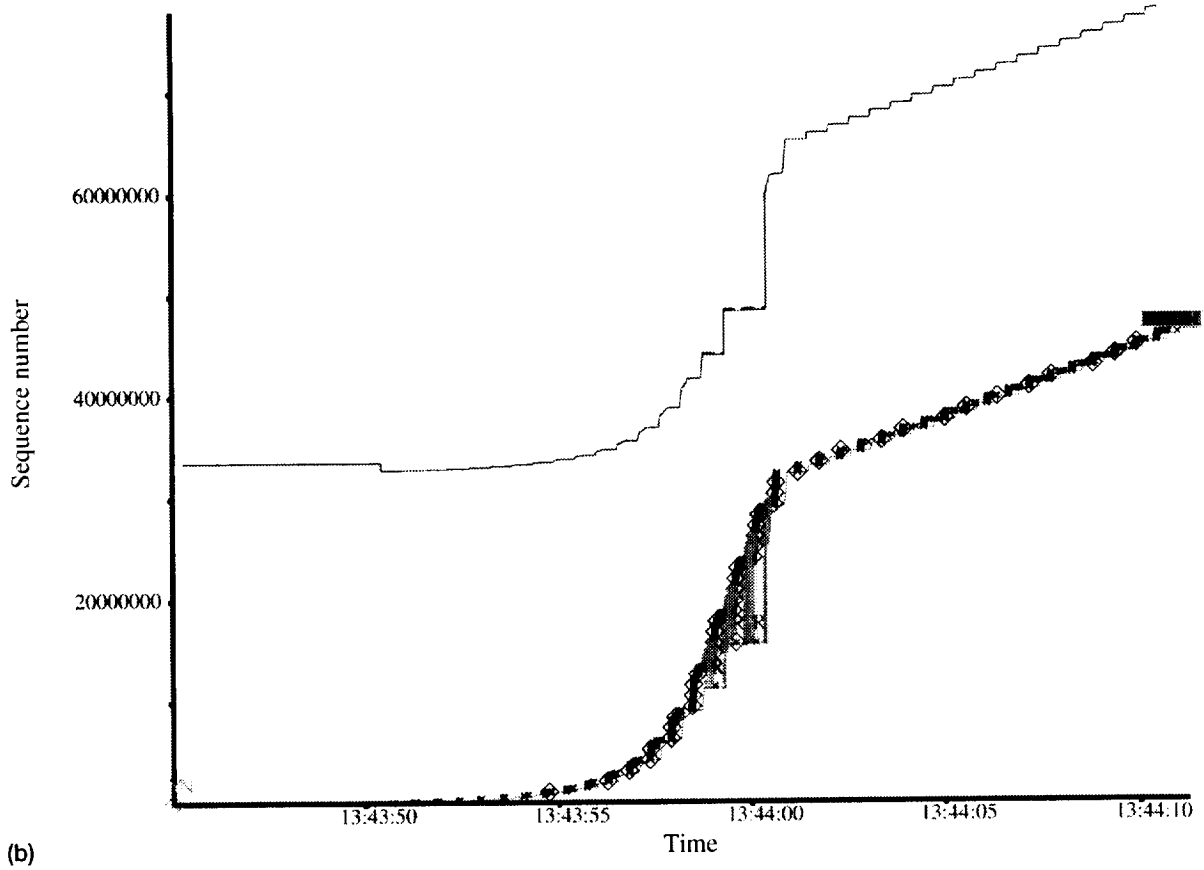
1 connection traced:
21943 packets seen, 21943 TCP packets traced
connection 1:
host a: 10.0.0.1:32801
host b: 10.0.0.5:5001
complete conn: yes
first packet: Tue Sep 22 14:35:33.951085
last packet: Tue Sep 22 14:37:05.511716
elapsed time: 0:01:31.560631
total packets: 21943
filename: sun2nt_sunside_combo_tcpdump.0

a->b:		b->a	
total packets:	13194	total packets	8749
ack pkts sent:	13193	ack pkts sent	8749
unique bytes sent:	114796054	unique bytes sent:	0
actual data pkts:	13191	actual data pkts:	0
actual data bytes:	119306086	actual data bytes:	0
rexmt data pkts:	495	rexmt data pkts:	0
rexmt data bytes:	4510032	rexmt data bytes:	0
outoforder pkts:	454	outoforder pkts:	0
pushed data pkts:	1157	pushed data pkts:	0
SYN/FIN pkts sent:	1/1	SYN/FIN pkts sent:	1/1
req 1323 ws/ts:	Y/Y	req 1323 ws/ts:	Y/Y
adv wind scale:	9	adv wind scale:	10
mss requested:	9140 bytes	mss requested:	9126 bytes
max segm size:	9114 bytes	max segm size:	0 bytes
min segm size	466 bytes	min segm size	0 bytes
avg segm size:	9044 bytes	avg segm size:	0 bytes
max win adv:	32782848 bytes	max win adv:	67107840 bytes
min win adv:	32782848 bytes	min win adv:	33545216 bytes
zero win adv:	0 times	zero win adv:	0 times
avg win adv:	227953 bytes	avg win adv:	175329 bytes
initial window:	18228 bytes	initial window:	0 bytes
initial window:	2 pkts	initial window:	0 pkts
ttl stream length:	119218208 bytes	ttl stream length:	0 bytes
missed data:	4422154 bytes	missed data:	0 bytes
truncated data:	118343143 bytes	truncated data:	0 bytes
truncated packets	13191 pkts	truncated packets	0 pkts
data xmit time:	90.408 secs	data xmit time:	0.000 secs
throughput:	1253771 Bps	throughput:	0 Bps

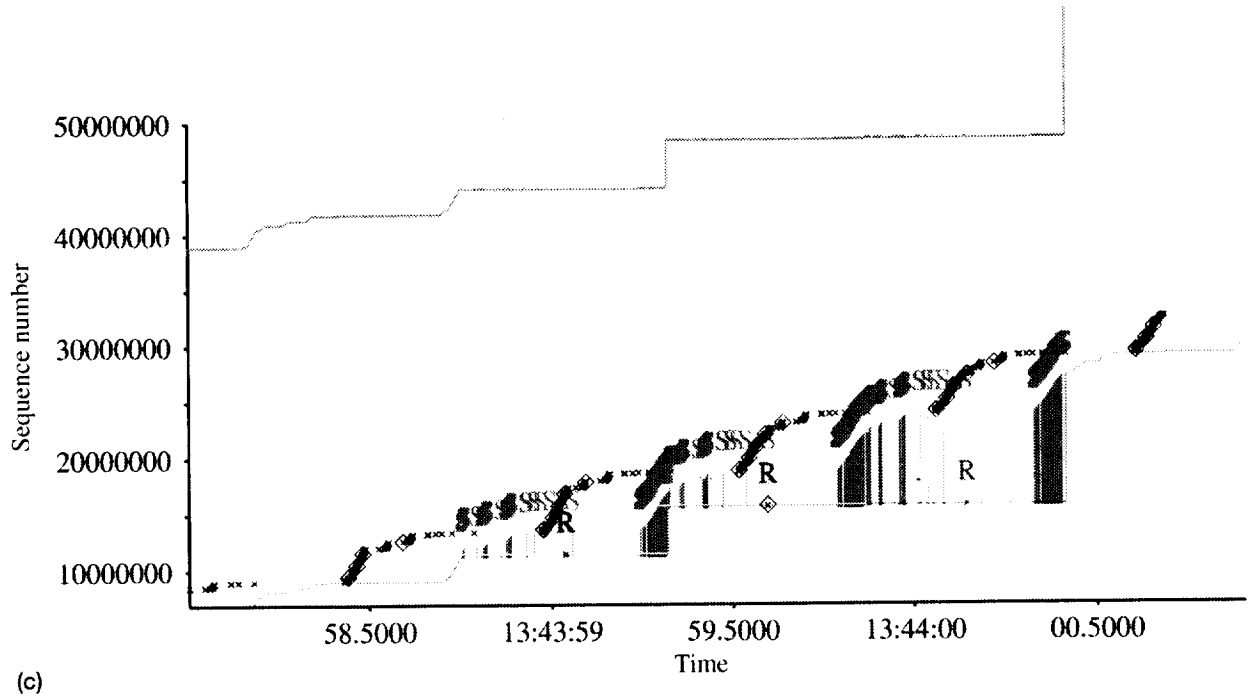


(a)

Figure 4.—Time-sequence plots. (a) TCP Stack operating correctly. (b) Problems in a TCP run. (c) Magnified portion of figure 4(b).



(b)



(c)

Figure 4.— Concluded. (b) Problems in a TCP run. (c) Magnified portion of figure 4(b).

CONCLUSIONS

Performing TCP interoperability tests for high-speed, long-delay networks requires proper tools to help identify problem areas. Since beta hardware and software from various vendors is being used, it is necessary to quickly identify potential problem areas and clearly convey this information to the vendors so that all can work together to solve the interoperability problem. *Tcpdump, tcptrace, ttcp,* and XPLOT are all useful tools to perform this function. Detailed information on these four programs is available from the Web. Information on modifications that were made to *tcpdump* and *tcpdumpcombo* programs together with the source code and compile programs is available from NASA Glenn Research Center at http://mrpink.grc.nasa.gov/118x/support.html.[3]

REFERENCES

1. Carlin, C.M.; Lopez, I.; Hedges, L.S.: Remote Propulsion Simulation. Paper presented at the Advanced Communication Technology Satellite Results Conference (Cleveland, Ohio), Sept. 1995.
2. Carlson, W.E.; et al.: Visualization of Results From a Distributed, Coupled, Supercomputer-Based Mesoscale Atmospheric and Lake Models Using the NASA ACTS. Paper presented at the Advanced Communication Technology Satellite Results Conference (Cleveland, Ohio), Sept. 1995.
3. Beering, D.R.: Satellite/Terrestrial Networks for Oil Exploration. Paper presented at the Advanced Communication Technology Satellite Results Conference (Cleveland, Ohio), Sept. 1995.
4. Van Jacobson, M.K.: Congestion Avoidance and Control. Proceedings of the Symposium on Communication Architectures and Protocols. Aug. 1988. (Available online: ftp://ftp.ee.lbl.gov/papers/congavoid.ps.Z)
5. Jacobson, V.; Braden, R.; Borman, D.: TCP Extensions for High Performance. RFC 1323, May 1992. (Obsolete RFC 1072, RFC 1185).
6. Mathis, M.; et al.: TCP Selective Acknowledgment Options. RFC 2018, Oct.1996.
7. Mathis, M.; et al.: Macroscopic Behavior of the TCP Congestion Avoidance Algorithm. Comput. Communi. Rev., vol. 27, no. 3, July 1997, pp. 67–82. (Available online: http://www.psc.edu/networking/papers/papers.html)
8. Available online: http://jarok.cs.ohiou.edu/software/tcptrace/tcptrace.html.
9. Shepard, T.: TCP Packet Trace Analysis. NASA CR–188617 (MIT-LCS-TR-494), Feb. 1991. (Program available online: ftp://mercury.lcs.mit.edu/pub/shep. Full color writeup available online: http://jarok.cs.ohiou.edu/software/tcptrace/tsg.html)
10. Frantz, B.D.; Brooks, D.E.; Ivancic, W.D.: TTCP Recommendations for Large-Delay, Large-Bandwidth Networks. May 1998. (Available online: http://sulu.lerc.nasa.gov/5610/inetprotocols.html)

[3]This site is currently password protected (5-1-99); the password is expected to be removed in the future. To obtain the password, contact the ACTS Experiments Office at NASA Glenn Research Center.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (*Leave blank*)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	May 1999	Technical Memorandum

4. TITLE AND SUBTITLE

High-Speed TCP Testing

5. FUNDING NUMBERS

WU–632–50–5A–00

6. AUTHOR(S)

David E. Brooks, Holly Gassman, Dave R. Beering, Arun Welch, Douglas J. Hoder and William D. Ivancic

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

National Aeronautics and Space Administration
John H. Glenn Research Center at Lewis Field
Cleveland, Ohio 44135–3191

8. PERFORMING ORGANIZATION REPORT NUMBER

E–11565

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

National Aeronautics and Space Administration
Washington, DC 20546–0001

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

NASA TM—1999-209040

11. SUPPLEMENTARY NOTES

David E. Brooks, Sterling Software, Cleveland, Ohio; Holly Gassman, Case Western Reserve University, Cleveland, Ohio 44106; Dave R. Beering, Infinite Global Infrastructures, LLC, Chicago, Illinios; Arun Welch, Anzus Consulting, Orleans, Massachusetts; Douglas J. Hoder and William D. Ivancic, Glenn Research Center, Cleveland, Ohio. Responsible person, William D. Ivancic, organization code 5610, (216) 433–3494.

12a. DISTRIBUTION/AVAILABILITY STATEMENT

Unclassified - Unlimited
Subject Category: 17 Distribution: Nonstandard

This publication is available from the NASA Center for AeroSpace Information, (301) 621–0390.

12b. DISTRIBUTION CODE

13. ABSTRACT ***(Maximum 200 words)***

Transmission Control Protocol (TCP) is the underlying protocol used within the Internet for reliable information transfer. As such, there is great interest to have all implementations of TCP efficiently interoperate. This is particularly important for links exhibiting long bandwidth-delay products. The tools exist to perform TCP analysis at low rates and low delays. However, for extremely high-rate and long-delay links such as 622 Mbps over geosynchronous satellites, new tools and testing techniques are required. This paper describes the tools and techniques used to analyze and debug various TCP implementations over high-speed, long-delay links.

14. SUBJECT TERMS

TCP; Satellite; Modem

15. NUMBER OF PAGES

16

16. PRICE CODE

A03

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	