NASA Contractor Report 204151

Fixing Two BSD TCP Bugs

Mark Allman
Sterling Software, Cleveland, Ohio

October 1997

The NASA STI Program Office ... in Profile

Since its founding, NASA has been dedicated to the advancement of aeronautics and space science. The NASA Scientific and Technical Information (STI) Program Office plays a key part in helping NASA maintain this important role.

The NASA STI Program Office is operated by Langley Research Center, the lead center for NASA's scientific and technical information. The NASA STI Program Office provides access to the NASA STI Database, the largest collection of aeronautical and space science STI in the world. The Program Office is also NASA's institutional mechanism for disseminating the results of its research and development activities. These results are published by NASA in the NASA STI Report Series, which includes the following report types:

- TECHNICAL PUBLICATION. Reports of completed research or a major significant phase of research that present the results of NASA programs and include extensive data or theoretical analysis. Includes compilations of significant scientific and technical data and information deemed to be of continuing reference value. NASA counter-part of peer reviewed formal professional papers, but having less stringent limitations on manuscript length and extent of graphic presentations.

- TECHNICAL MEMORANDUM. Scientific and technical findings that are preliminary or of specialized interest, e.g., quick release reports, working papers, and bibliographies that contain minimal annotation. Does not contain extensive analysis.

- CONTRACTOR REPORT. Scientific and technical findings by NASA-sponsored contractors and grantees.

- CONFERENCE PUBLICATION. Collected papers from scientific and technical conferences, symposia, seminars, or other meetings sponsored or co-sponsored by NASA.

- SPECIAL PUBLICATION. Scientific, technical, or historical information from NASA programs, projects, and missions, often concerned with subjects having substantial public interest.

- TECHNICAL TRANSLATION. English-language translations of foreign scientific and technical material pertinent to NASA's mission.

Specialized services that help round out the STI Program Office's diverse offerings include creating custom thesauri, building customized databases, organizing and publishing research results ... even providing videos.

For more information about the NASA STI Program Office, you can:

- Access the NASA STI Program Home Page at http://www.sti.nasa.gov/STI-homepage.html

- E-mail your question via the Internet to help@sti.nasa.gov

- Fax your question to the NASA Access Help Desk at (301) 621-0134

- Phone the NASA Access Help Desk at (301) 621-0390

- Write to:
  NASA Access Help Desk
  NASA Center for AeroSpace Information
  800 Elkridge Landing Road
  Linthicum Heights, MD 21090-2934

NASA Contractor Report 204151

Fixing Two BSD TCP Bugs

Mark Allman
Sterling Software, Cleveland, Ohio

Prepared under Contract NAS3–27121

National Aeronautics and
Space Administration

Lewis Research Center

October 1997

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
800 Elkridge Landing Road
LInthicum Heights, MD 21090-2934
Price Code: A03

National Technical Information Service
5287 Port Royal Road
Springfield, VA 22100
Price Code: A03

Fixing Two BSD TCP Bugs

Mark Allman
Sterling Software
NASA Lewis Research Center
21000 Brookpark Rd. MS 54-2
Cleveland, OH 44135
mallman@lerc.nasa.gov

Abstract

This note outlines two bugs found in the BSD 4.4 Lite TCP implementation, as well as the implications of these bugs and possible ways to correct them. The first problem encountered in this particular TCP implementation is the use of a 2 segment initial congestion window, rather than the standard 1 segment initial window. The second problem is that the receiver delays ACKs in violation of the delayed ACK rules.

1 Introduction

This report discusses two bugs found in the Berkeley Software Distribution release 4.4 Lite implementation of the Transmission Control Protocol (TCP) [Pos81]. NetBSD's implementation of TCP is derived from BSD 4.4 Lite and therefore inherited these bugs[1]. In this paper, "TCP" will refer to standard TCP [Pos81] [Bra89] [Ste97], while "NTCP" will refer to the NetBSD implementation of TCP. The following two sections include an outline of the particular problem and the implications the problem. Next, each section will examine the details that cause the problem, and an outline of how to fix the bug.

[1]We verified that these bugs exist in the BSD 4.4 Lite. NetBSD is derived from BSD 4.4 Lite and we verified that the bugs existed in NetBSD 1.1 and NetBSD 1.2.1 (the current release of NetBSD at the time this report was prepared).

2 Two Segment Initial Window

2.1 Problem and Implications

Figure 1 shows a time-sequence plot [She91] of NTCP. This figure clearly illustrates that NTCP is using an initial window of 2 segments, rather than 1 segment, as defined in the TCP standard [Bra89] [Ste97]. The size of the *congestion window* determines the amount of data the sender can transmit without receiving an acknowledgment (ACK). This bug in NTCP occurs because the congestion window is mistakenly increased during connection setup.

This bug poses only a small problem in a congested network where each connection's share of the bottleneck is greater than or equal to 1 segment but less than 2 segments. An initial window of 1 segment will experience one congestion free round-trip time (RTT), while an initial window of 2 segments will experience loss immediately. However, when using an initial window of 1 segment, the ACK of the first segment sent will trigger 2 new segments, which will cause loss. Therefore, even with an initial window of 1 segment loss is not prevented, just postponed one RTT. It is expected that extremely congested networks of this type are rare and that this bug has very little impact on the network.

A current proposal [FAP97] suggests increasing the initial window from 1 segment to 4 KBytes. Although further study is still needed, this change has been investigated and found to be safe in certain environments [AHO97] [SP97]. In particular, a 2 segment initial window did not

significantly increase loss rates in tests over dialup modem lines and tests over the Internet [AHO97].

2.2 Details

Figure 2 shows the portion of the TCP finite state machine [Pos81] [Com95] used to setup TCP connections. The following is a list of the steps involved in establishing a connection between a client (which *actively* opens the connection) and a server (which *passively* opens the connection).

1. The server moves from the **CLOSED** state to the **LISTEN** state when it is prepared to accept connections from clients.

2. When prepared to establish a connection, the client *actively* opens the TCP connection by sending a *synchronize* (SYN) packet to the server. This moves the client from the **CLOSED** state to the **SYN SENT** state.

3. When the server is in **LISTEN** state and receives a SYN packet from a client, the server sends an ACK for the incoming SYN, as well as, transmitting its own SYN (usually both the SYN and ACK are sent in the same packet). At this time, the server moves from the **LISTEN** state to the **SYN RECEIVED** state.

4. When the client is in **SYN SENT** state and receives a SYN/ACK packet from the server, the client transmits an ACK for the server's SYN and moves to the **ESTABLISHED** state.

5. When the server is in **SYN RECEIVED** state and receives an ACK for it's SYN from the client, the server moves to **ESTABLISHED** state.

6. When both the client and server are in the **ESTABLISHED** state data can be exchanged.

The bug in NTCP happens when moving from the **SYN RECEIVED** state to the **ESTABLISHED** state. After the **SYN RECEIVED** code handles the incoming ACK and changes NTCP's state to **ESTABLISHED**, the code that handles incoming ACKs when in the **ESTABLISHED** state is allow to process the ACK. So, the ACK is processed twice. An ACK received when in **ESTABLISHED** state would normally indicate that one or more packets had been successfully received. This would cause an increase in the congestion window, allowing the sender to transmit new data. However, the ACK in response to the server's SYN segment does not indicate that data has successfully arrived at the remote host and therefore should not increase the congestion window.

2.3 Fix

We fixed this bug in NTCP by setting a flag when moving from the **SYN RECEIVED** state to the **ESTABLISHED** state. When this flag is set, the congestion window is not increased. Figure 3 shows NTCP correctly beginning data transmission by sending 1 data segment.

3 Delayed ACK Violations

3.1 Problem and Implications

In addition to using a large initial window, figure 1 also shows that NTCP is violating standard ACKing behavior [Bra89]. RFC 1122 outlines a *delayed ACK* mechanism that allows a TCP receiver to refrain from sending an ACK for every incoming segment, as long as the ACK is not excessively delayed. Specifically, an ACK must be sent for every second full-sized packet. Furthermore, if a second full-sized packet does not arrive within a given timeout, an ACK must be sent (this timeout must be ≤ 0.5 seconds). As figure 1 shows, NetBSD is in violation of these rules by ACKing every third full-sized packet.

Each ACK triggers the transmission of new data (assuming we are not recovering from loss). Therefore, ACKing more data with a single ACK, allows the sender to transmit more data in response to an ACK. As this burst of traffic grows, the likelihood of overwhelming intermediate gateways and causing packet loss increases.

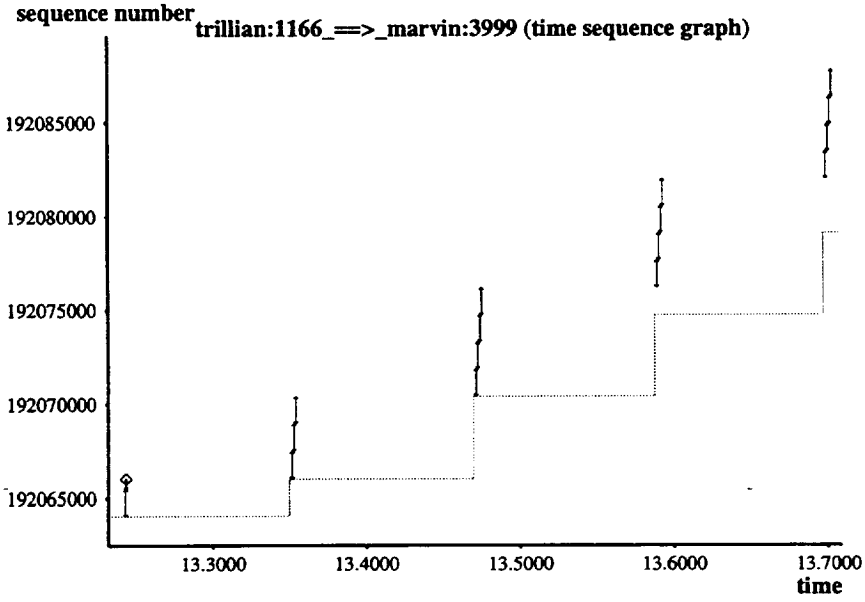

Figure 1: Standard NTCP
This figure shows the behavior of unmodified NTCP. The figure illustrates that NTCP uses an initial window of 2 segments and ACKs every third full-sized packet, rather than every second.

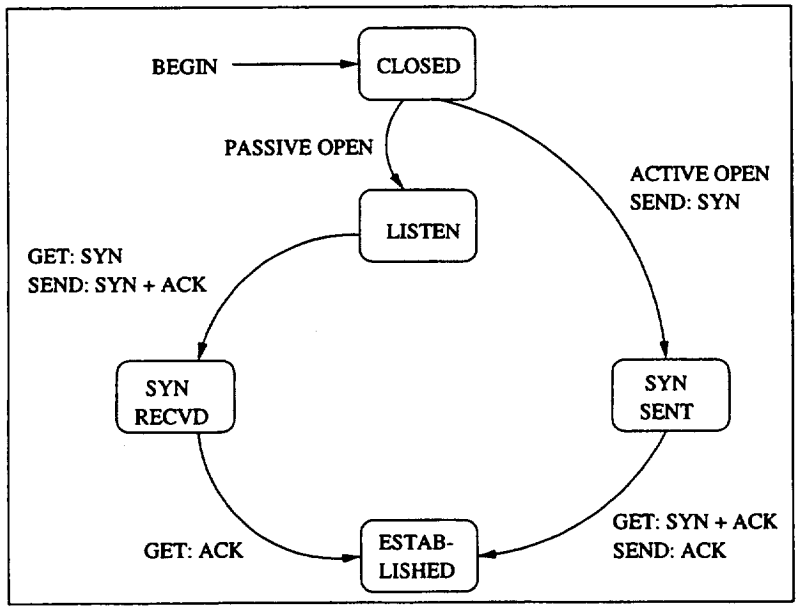

Figure 2: Partial TCP Finite State Machine
This figure shows the connection setup portion of the TCP finite state machine.

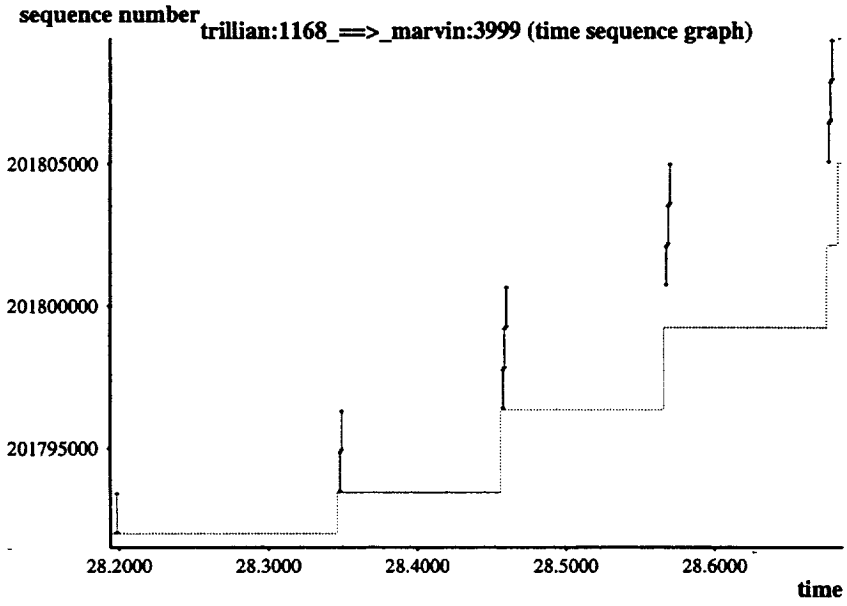

Figure 3: Fixed NTCP
This figure shows the behavior of fixed NTCP. The figure illustrates NTCP using an initial window of 1 segment and correctly ACKing every second full-sized segment.

The impact of these "stretch ACKs" is discussed in [Pax97].

In addition, increasing the ACK interval negatively impacts the slow start phase of a TCP transfer [Pax97]. During slow start [JK88], the congestion window is incremented by 1 segment for each ACK received, providing exponential increase in the size of the congestion window. Therefore, by decreasing the number of ACKs being transmitted, the receiver is slowing the rate at which the sender can increase the transmission rate. This can have a particularly large impact on long-delay connections, such as those over satellite channels.

3.2 Details

The bug in NTCP is caused by the use of TCP options (e.g., window scaling [JBB92], selective acknowledgments [MMFR96], T/TCP [Bra94]). The NTCP connection shown in figure 1 uses RFC 1323 TCP extensions [JBB92] and utilizes a maximum segment size (MSS) of 1460 bytes. NTCP sends an ACK after receiving data greater than or equal to twice the MSS. The length of the TCP options is not considered when determining whether to send an ACK. In the example given in figure 1, the options take 12 bytes of each packet which would otherwise hold data. Therefore, each packet contains 1448 bytes of data. So, two full-sized packets contain 2896 bytes of data, which is less than the 2920 bytes ($2 \times$ MSS) needed to trigger an ACK. Therefore, the receiver waits for a third segment to arrive before sending an ACK.

3.3 Fix

We fixed this bug by adding an entry to the TCP control block that keeps track of the length of the options contained on incoming TCP data segments. The length of the options received is added to the length of the data when determining whether two full-sized packets have been received. Figure 3 shows that this fix yields correct ACKing behavior.

4 Conclusions

This report has outlined two bugs in the BSD 4.4 Lite implementation of TCP, the implications of these bugs and fixes for them. Since a number of

TCP implementations are derived from BSD 4.4 Lite, it is expected that a number of may have similar problems.

Source Code

Our changes to the NetBSD TCP code are available at `http://gigahertz.lerc.nasa.gov/~mallman`.

References

[AHO97] Mark Allman, Chris Hayes, and Shawn Ostermann. An Evaluation of TCP Slow Start Modifications, 1997. Submitted to Computer Communications Review.

[Bra89] Robert Braden. Requirements for Internet Hosts – Communication Layers, October 1989. RFC 1122.

[Bra94] Robert Braden. T/TCP – TCP Extensions for Transactions: Functional Specification, July 1994. RFC 1644.

[Com95] Douglas E. Comer. *Internetworking with TCP/IP, Volume I, Principles, Protocols, and Architecture.* Prentice Hall, 3rd edition, 1995.

[FAP97] Sally Floyd, Mark Allman, and Craig Partridge. Increasing TCP's Initial Window, July 1997. Internet-Draft draft-floyd-incr-init-win-00.txt.

[JBB92] Van Jacobson, Robert Braden, and David Borman. TCP Extensions for High Performance, May 1992. RFC 1323.

[JK88] Van Jacobson and Michael J. Karels. Congestion Avoidance and Control. In *ACM SIGCOMM*, 1988.

[MMFR96] Matt Mathis, Jamshid Mahdavi, Sally Floyd, and Allyn Romanow. TCP Selective Acknowledgement Options, October 1996. RFC 2018.

[Pax97] Vern Paxson. Automated Packet Trace Analysis of TCP Implementations. In *ACM SIGCOMM*, September 1997. To Appear.

[Pos81] Jon Postel. Transmission Control Protocol, September 1981. RFC 793.

[She91] Tim Shepard. TCP Packet Trace Analysis. Technical Report TR-494, Massachusetts Institute of Technology, February 1991.

[SP97] Tim Shepard and Craig Partridge. When TCP Starts Up With Four Packets Into Only Three Buffers, August 1997. Internet-Draft draft-shepard-tcp-4-packets-3-buff-00.txt.

[Ste97] W. Richard Stevens. TCP Slow Start, Congestion Avoidance, Fast Retransmit, and Fast Recovery Algorithms, January 1997. RFC 2001.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (*Leave blank*)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	October 1997	Final Contractor Report

4. TITLE AND SUBTITLE

Fixing Two BSD TCP Bugs

5. FUNDING NUMBERS

WU-632-50-5A-00
C-NAS3-27121

6. AUTHOR(S)

Mark Allman

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

Sterling Software
21000 Brookpark Rd.
Mail Stop 54-2
Cleveland, Ohio 44135

8. PERFORMING ORGANIZATION REPORT NUMBER

E-10928

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

National Aeronautics and Space Administration
Lewis Research Center
Cleveland, Ohio 44135-3191

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

NASA CR-204151

11. SUPPLEMENTARY NOTES

Project Manager, Kul Bhasin, Communications Technology Division, NASA Lewis Research Center, organization code 5610, (216) 433-3676.

12a. DISTRIBUTION/AVAILABILITY STATEMENT

Unclassified - Unlimited
Subject Categories: 17, 61, and 62 Distribution: Nonstandard

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

12b. DISTRIBUTION CODE

13. ABSTRACT (*Maximum 200 words*)

This note outlines two bugs found in the BSD 4.4 Lite TCP implementation, as well as the implications of these bugs and possible ways to correct them. The first problem encountered in this particular TCP implementation is the use of a 2 segment initial congestion window, rather than the standard 1 segment initial window. The second problem is that the receiver delays ACKs in violation of the delayed ACK rules.

14. SUBJECT TERMS

Communication networks; Computer networks

15. NUMBER OF PAGES

11

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	