Internet Engineering Task Force
INTERNET DRAFT
File: draft-blanton-dsack-use-02.txt

Ethan Blanton
Purdue University
Mark Allman
BBN/NASA GRC

October, 2002
Expires: April, 2003

Using TCP DSACKs and SCTP Duplicate TSNs to Detect Spurious Retransmissions

Status of this Memo

Abstract

TCP and SCTP provide notification of duplicate segment receipt through DSACK and Duplicate TSN notification, respectively. This document presents a conservative method of using this information to identify unnecessary retransmissions.

Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

1 Introduction

TCP [RFC793] and SCTP [RFC2960] provide notification of duplicate segment receipt through DSACK [RFC2883] and Duplicate TSN notifications, respectively. Using this information, a TCP or SCTP sender can generally determine when a retransmission was sent in error. This document presents a conservative algorithm to disambiguate unnecessary retransmissions from loss events for the

purpose of undoing unnecessary congestion control changes (although specifying methods for reversing unneeded congestion control changes is beyond the scope of this document).

While DSACKs and duplicate TSN notifications can be caused by segments being duplicated by the network, [Pax97] shows this is rare. Some network paths may exhibit this problem more than others, but we do not believe it to be a general problem. Therefore the algorithm presented in this document is disabled when duplication of segments by the network is detected.

This document is intended to outline a reasonable and safe algorithm for detecting spurious retransmissions and discuss some of the considerations involved. It is not intended to describe the only possible method for achieving the goal, although the guidelines in this document should be taken into consideration when designing alternate algorithms. Additionally, this document does not outline what a TCP or SCTP sender may do after a spurious retransmission is detected. Some possibilities are mentioned in [RFC2883], such as reverting changes made to the congestion control state. Discussion of this topic is left to a companion document that makes no assumptions about the manner in which spurious retransmissions are detected [BA01].

2 The Algorithm

The complexity of the algorithm used for detecting spurious retransmits depends on the purpose in determining this information. For instance, if a sender is only interested in keeping a count of the number of spurious retransmits the information can be derived directly from the returning DSACK or duplicate TSN notifications.

However, if the purpose of detecting spurious retransmissions is to ``undo'' unnecessary changes made to the congestion control state, as suggested in [RFC2883], the data sender needs to ensure that spurious retransmissions in a particular window of data do not mask real segment loss before reverting the congestion control state.

For example, say segments N and N+1 are retransmitted. Assume that segment N was dropped by the network and segment N+1 was needlessly retransmitted. When the sender receives the notification that segment N+1 arrived more than once it can conclude that segment N+1 was needlessly resent. However, it cannot conclude that it is appropriate to revert the congestion control state because the window of data contained at least one real congestion indication (i.e., segment N was lost).

The following algorithm ensures that all retransmissions sent in a particular window are, in fact, needless. We assume the TCP sender has a data structure to hold selective acknowledgment information (e.g., as outlined in [BA02b]). The following steps MUST be taken upon the receipt of each DSACK or duplicate TSN notification:

(A) Check the corresponding sequence range or TSN to determine whether the segment has been retransmitted.

(A.1) If the segment was retransmitted, mark it as a duplicate.

(A.2) If the segment was not retransmitted the incoming DSACK indicates that the network duplicated the segment in question. Processing of this DSACK MUST be terminated. In addition, the algorithm specified in this document MUST NOT be used for the remainder of the connection, as future DSACK reports may be indicating network duplication rather than unnecessary retransmission. Note that some techniques to further disambiguate network duplication from unnecessary retransmission (e.g., the TCP timestamp option [RFC1323]) may be used to refine the algorithm in this document further. Using such a technique in conjunction with an algorithm similar to the one presented herein may allow for the continued use of the algorithm in the face of duplicated segments. We do not delve into such an algorithm in this document due the current rarity of network duplication. However, future work should include tackling this problem.

(B) Check all retransmitted segments in the previous window of data.

(B.1) If all segments or chunks marked as retransmitted have also been marked as duplicate, we conclude that all retransmissions in the previous window of data were spurious and no loss occurred.

(B.2) If any segment or chunk is still marked as retransmitted but not marked as duplicate, there are outstanding retransmissions that could indicate loss within this window of data. We can make no conclusions based on this particular DSACK/duplicate TSN notification.

In addition to keeping the state mentioned in [BA02b] (for TCP) and [RFC2960] (for SCTP), an implementation of this algorithm must track all sequence numbers or TSNs that have been acknowledged as duplicates.

3 Related Work

In addition to the mechanism for detecting spurious retransmits outlined in this document, several other proposals for finding needless retransmits have been developed.

[BA02a] uses the algorithm outlined in this document as the basis for investigating several methods to make TCP more robust to reordered packets.

The Eifel detection algorithm [LM02] uses the TCP timestamp option [RFC1323] to determine whether the ACK for a given retransmit is for the original transmission or a retransmission. More generally,

[LK00] outlines the benefits of detecting spurious retransmits and reverting from needless congestion control changes using the timestamp-based scheme or a mechanism that uses a "retransmit bit" to flag retransmits (and ACKs of retransmits). The Eifel detection

algorithm can detect spurious retransmits more rapidly than a DSACK-based scheme. However, the tradeoff is that the overhead of the 12-byte timestamp option must be incurred in every packet transmitted.

The F-RTO scheme [SK02] slightly alters TCP's sending pattern immediately following a retransmission timeout and then observes the pattern of the returning ACKs. This pattern can indicate whether the retransmitted segment was needed. The advantage of F-RTO is that the algorithm only needs to be implemented on the sender side of the TCP connection and that nothing extra needs to cross the network (e.g., DSACKs, timestamps, special flags, etc.). The downside is that the algorithm is a heuristic that can be confused by network pathologies (e.g., duplication or reordering of key packets).

Finally, [AP99] briefly investigates using the time between retransmitting a segment via the retransmission timeout and the arrival of the next ACK as an indicator of whether the retransmit was needed. The scheme compares this time delta with a fraction (f) of the minimum RTT observed thus far on the connection. If the time delta if less than f*minRTT then the retransmit is labeled spurious. When f=1/2 the algorithm identifies roughly 59% of the needless retransmission timeouts and identifies needed retransmits only 2.5% of the time.

4 Security Considerations

It is possible for the receiver to falsely indicate spurious retransmissions in the case of actual loss, potentially causing a TCP or SCTP sender to inaccurately conclude that no loss took place (and cause inappropriate changes to the senders congestion control state). Consider the following scenario:

A receiver watches every segment or chunk that arrives and acknowledges any segment that arrives out of order by more than some threshold amount as a duplicate, assuming that it is a retransmission. A sender using the above algorithm will assume that the retransmission was spurious.

As a more trivial example, a receiver could simply acknowledge every segment or chunk received as a duplicate as they arrive. This approach is more easily defeated by heuristics, but would nonetheless cause the algorithm in this document to come to an incorrect conclusion.

The ECN nonce sum proposal [WES01] would help mitigate the ability of the receiver to hide real losses from the sender.

References

[AP99] Allman, M. and V. Paxson, "On Estimating End-to-End Network Path Properties", SIGCOMM 99.

[BA01] E. Blanton, M. Allman. Adjusting the Duplicate ACK Threshold to Avoid Spurious Retransmits. Internet-Draft draft-blanton-tcp-dupack-thresh-adjust-00.txt, July 2001. Work in progress.

[BA02a] E. Blanton, M. Allman. On Making TCP More Robust to Packet Reordering. ACM Computer Communication Review, 32(1), January 2002.

[BA02b] E. Blanton, M. Allman. A Conservative SACK-based Loss Recovery Algorithm for TCP. Internet-Draft draft-allman-tcp-sack-06.txt, July 2001. Work in progress.

[FF99] S. Floyd, K. Fall. Promoting the Use of End-to-End Congestion Control on the Internet. IEEE/ACM Transactions on Networking, August 1999.

[LK00] R. Ludwig, R. H. Katz. The Eifel Algorithm: Making TCP Robust Against Spurious Retransmissions. ACM Computer Communication Review, 30(1), January 2000.

[LM02] R. Ludwig, M. Meyer. The Eifel Detection Algorithm for TCP. Internet-Draft draft-ietf-tsvwg-tcp-eifel-alg-05.txt, October 2002. Work in progress.

[Pax97] V. Paxson. End-to-End Internet Packet Dynamics. In ACM SIGCOMM, September 1997.

[RFC793] Jon Postel. Transmission Control Protocol. Std 7, RFC 793. September 1981.

[RFC1323] Van Jacobson, Robert Braden, David Borman. TCP Extensions for High Performance. RFC 1323. May 1992.

[RFC2883] S. Floyd, J. Mahdavi, M. Mathis, M. Podolsky. An Extension to the Selective Acknowledgement (SACK) Option for TCP. RFC 2883, July 2000.

[RFC2960] R. Stewart, Q. Xie, K. Morneault, C. Sharp, H. Schwarzbauer, T. Taylor, I. Rytina, M. Kalla, L. Zhang, V. Paxson. Stream Control Transmission Protocol. October 2000.

[SK02] P. Sarolahti, M. Kojo. F-RTO: A TCP RTO Recovery Algorithm for Avoiding Unnecessary Retransmissions. Internet-Draft draft-sarolahti-tsvwg-tcp-frto-01.txt, October 2002. Work in progress.

[WES00] D. Wetherall, D. Ely, N. Spring. Robust ECN Signaling with Nonces. Internet-Draft draft-ietf-tsvwg-tcp-nonce-00.txt, January 2001. Work in progress.

Authors' Addresses:

Ethan Blanton

Purdue University Computer Sciences
1398 Computer Science Building
West Lafayette, IN 47907
eblanton@cs.purdue.edu

Mark Allman
BBN Technologies/NASA Glenn Research Center
Lewis Field
21000 Brookpark Rd. MS 54-5
Cleveland, OH 44135
Phone: 216-433-6586
Fax: 216-433-8705
mallman@bbn.com
http://roland.grc.nasa.gov/~mallman