INTERNATIONAL COMPUTER SCIENCE INSTITUTE

1947 Center St. • Suite 600 • Berkeley, California 94704-1198 • (510) 666-2900 • FAX (510) 666-2956

Using Spurious Retransmissions to Adapt the Retransmission Timeout

Josh Blanton[*] **Ethan Blanton**[†]

Mark Allman[‡]

TR-08-005

August 2008

Abstract

This report describes a method for using spurious retransmission timeouts to determine when the retransmission timeout is not accurately capturing the delay variance in the network. We account for this by adapting the way TCP's retransmission timeout is computed in an effort to avoid subsequent unnecessary retransmissions.

[*]Ohio University
[†]Purdue University
[‡]International Computer Science Institute

Background

This technical report contains material that was issued in Internet-Draft form but never adopted by the IETF. This report is published in the hopes of both (*i*) capturing the ideas for anyone who is interested in exploring them in the future and (*ii*) as potentially helpful to the standards process at some point in the future.

1 Introduction

Various studies have shown that TCP's retransmission timeout (RTO) estimator specified in [PA00] can trigger spurious retransmissions. [AP99] shows that such unnecessary retransmissions are generally fairly rare. However, [LK00] shows that in some networks (e.g., wireless networks) spurious retransmissions are more frequent due to occasional delay spikes that are not well predicted by TCP's RTO estimator. In this report we outline one possible approach to mitigate the impact of pre-mature RTO firings by altering the RTO estimator specified in [PA00].

Several methods for detecting spurious timeouts have been developed [LM03, BA04, SK05]. Additionally, [LG05] outlines one possible response to detecting spurious timeouts. This report outlines an alternative to [LG05]. In general terms, [LG05] specifies two actions upon the detection of an unnecessary RTO-based retransmission. First, the sending rate prior to the spurious retransmission is restored. Furthermore, the RTO is adapted by re-initializing the RTO estimator with the long round-trip time (RTT) measurement that caused the spurious RTO. The approach given in [LG05] is reasonable if the underlying cause of the problem is a shift in the path RTT. For instance, if the route a TCP connection is traversing changes and the new path's RTT is significantly longer than the previous path's RTT then simply re-initializing the RTO is a reasonable action.

As specified in the next section, we take a different approach than [LG05]. Generally, we use the failure of the RTO to wait long enough before triggering a retransmit as an indication that the RTO estimator itself is not properly capturing the variance present in the RTTs experienced by the TCP connection. Therefore, this report calls for an additive contribution to the variance component in the RTO estimator upon the detection of retransmission timeouts in an effort to cope. This change represents a preference to try to avoid future spurious timeouts rather than simply reacting to each spurious retransmission.

We note that TCP implementations using the RTTM mechanism [JBB92] to assess the RTT multiple times per RTT with the standard exponentially-weighted moving average (EWMA) gains from [PA00] retain less RTT history than when taking one RTT measurement per RTT. [AP99] shows that such "fast" EWMAs yield more spurious retransmissions than when using the standard gains with one RTT sample per RTT. Therefore, an orthogonal change to TCP implementations that use RTTM and may prevent spurious RTOs is to set the EWMA gains based on the number of RTT samples taken per RTT such that the amount of history kept, in terms of time, is the same regardless of the number RTT samples taken [Flo98, LS00].

This report was originally written as a standards contribution and therefore uses standards language. The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this report are to be interpreted as described in [Bra97].

The reader is expected to be familiar with the algorithm and terminology from [PA00].

2 Algorithmic Changes

As the basis for the changes proposed below, a TCP MUST support a spurious timeout detection method. Several such methods exist, including [LK00], [LM03], [BA04] and [SK05].

We also note that [PA00] explicitly allows for an RTO estimator that is more conservative than that given in [PA00] (which this report specifies). Therefore, we believe that any TCP implementation can use the changes specified in this report without being non-compliant with regards to the current specification.

Also we note that, given that a TCP implementation is savvy enough to untangle needed and unneeded retransmission timeouts, the TCP does not need to use Karn's algorithm [KP87, PA00] and can accurately determine the RTT that causes spurious retransmissions.

The general idea behind the mechanism is to introduce an additive variance term, V, in addition to the multiplier K which is applied to RTTVAR in the RTO calculation given in step (2.3) of [PA00], to allow for additional variance in the path's RTT. The specific mechanism for TCPs using this change is:

1. A TCP using this method MUST replace the calculation of RTO in step (2.3) of [PA00] with:

$$RTO \leftarrow SRTT + max(G, K * RTTVAR) + V \quad (1)$$

   to include the additional variance term.

2. When a TCP connection is initiated, V is set to 0. I.e., until we observe otherwise we assume the standard estimator can handle the delay variance across the network path.

3. Upon the first expiration of the retransmission timer for a given sequence number, the values of $SRTT$ and $RTTVAR$ MUST be saved as $SRTT_{prev}$ and $RTTVAR_{prev}$, respectively.

4. Upon detecting that a previous RTO-based retransmission was spurious, a TCP MUST calculate a V' using the RTT sample R', which is the time between when the original transmission of the given segment was sent and when the that original transmission is acknowledged, as follows:

$$V' \leftarrow R' - SRTT_{prev} + max(G, K * SRTTVAR_{prev}) \quad (2)$$

   V' then becomes the difference between the previously calculated RTO and the RTO value which would have prevented the spurious retransmission.

   The value of V' MUST NOT be reduced for the remainder of the connection (as discussed in more detail below).

5. The values of $SRTT$ and $RTTVAR$ in use when the spurious retransmit occurred MUST replace the current values:

$$SRTT \leftarrow SRTT_{prev} \quad (3)$$

$$RTTVAR \leftarrow RTTVAR_{prev} \quad (4)$$

6. The R' RTT sample MUST be used to adjust $SRTT$ and $RTTVAR$ and therefore the RTO, per [PA00].

The actual V that is used in the RTO calculation is determined by the size of the congestion window. When a TCP has only a small number of outstanding segments, advanced loss recovery that relies on the receipt of three duplicate acknowledgments as a recovery trigger is not as effective as when the congestion window is larger. Therefore, TCP relies more heavily on the RTO in this regime. Furthermore, the impact caused by spurious timeouts in this situation—in terms of congestion window reduction and resource wastage by go-back-N transmission—is small. Hence, when the congestion window is less than or equal to 4*SMSS bytes then a V of 0 SHOULD be used when calculating the RTO. Once the congestion window size grows beyond 4*SMSS bytes, the calculated value of V' SHOULD be used in the calculation of the RTO.

This specification explicitly offers no way to reduce V' after it has been inflated. V' is never reduced because the presence of spurious timeouts which inflated V' indicates that the standard estimator is inadequate for accurately estimating the variance of the RTT across the network path and therefore reducing V' would increase the chances of further spurious retransmissions.

Finally, we note that bounding V' is not advisable. Say V' would be set to 20 via equation 2. If V' were, instead, bound to 10 then legitimate RTOs would be forced to wait longer without offering solid protection against delay spikes (given that delay spikes that a V' of 10 will not handle have been observed).

3 Advantages

The advantage of tuning the RTO calculation to be more conservative after detecting spurious RTO-based retransmissions is in preventing further spurious RTOs and their attendant performance impact. In addition, spurious RTOs can cause go-back-N behavior [LK00] which can also be avoided by adapting the RTO to be more conservative.

4 Disadvantages

The disadvantage of tuning the RTO calculation to be more conservative is that legitimate RTO firings take longer and could hurt performance. However, an important note is that the RTO should not be TCP's primary loss recovery strategy. [FHG04] and [BAFW03] provide methods for TCP to effectively repair multiple lost segments from a single window of data without falling back to using the RTO. Further, research shows that these changes are widely implemented [MAF05]. Therefore, making TCP's RTO calculation more conservative should not hinder performance under normal circumstance. Put differently, when using advanced loss recovery techniques the firing of the RTO should be an indication that the congestion situation in the network is fairly bad. In this case, it may well be that making the RTO estimator more conservative is the right general approach.

The common exception to the above argument is when the congestion window is small, such that these advanced loss recovery algorithms do not work effectively. The mechanism in this report explicitly takes this case into account by not using the more conservative RTO estimate when the congestion window is small.

5 Summary

This report specifies a small change that makes the RTO calculation given in [PA00] more conservative upon the detection of spurious RTO-based retransmissions. The root cause of spurious retransmits is an inaccurate assessment of the network conditions (in this case, of the RTT). Therefore, we tackle this by making the RTO calculation take into account an additional variance term. While this does lengthen the time required for legitimate retransmissions to fire, the RTO should not be TCP's primary means for retransmitting data and therefore this lengthened interval should only minimally impact overall performance and should only come into play when conditions along the network path have deteriorated significantly. Finally, we note that this report makes the estimator given in [PA00] strictly more conservative and is therefore allowed via [PA00].

Acknowledgments

This report has benefited from discussions with Ted Faber, Aaron Falk, Joseph Ishac, Janardhan Iyengar, Sally Floyd, Vern Paxson and Joe Touch. This work was in part funded by grant ITR/ANI-0205519 from the National Science Foundation.

References

[AP99] Mark Allman and Vern Paxson. On Estimating End-to-End Network Path Properties. In *ACM SIGCOMM*, September 1999.

[BA04] Ethan Blanton and Mark Allman. Using TCP Duplicate Selective Acknowledgement (DSACKs) and Stream Control Transmission Protocol (SCTP) Duplicate Transmission Sequence Numbers (TSNs) to Detect Spurious Retransmissions, February 2004. RFC 3708.

[BAFW03] Ethan Blanton, Mark Allman, Kevin Fall, and Lili Wang. A Conservative Selective Acknowledgment (SACK)-based Loss Recovery Algorithm for TCP, April 2003. RFC 3517.

[Bra97] Scott Bradner. Key Words for Use in RFCs to Indicate Requirement Levels, March 1997. RFC 2119.

[FHG04] Sally Floyd, Tom Henderson, and Andrei Gurtov. The NewReno Modification to TCP's Fast Recovery Algorithm, April 2004. RFC 3782.

[Flo98] Sally Floyd. Comments on RFC1323.bis, TCP-LW mailing list,, May 1998.

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

[KP87] Phil Karn and Craig Partridge. Improving Round-Trip Time Estimates in Reliable Transport Protocols. In *ACM SIGCOMM*, pages 2–7, August 1987.

[LG05] Reiner Ludwig and Andre Gurtov. The Eifel Response Algorithm for TCP, February 2005. RFC 4015.

[LK00] Reiner Ludwig and Randy Katz. The Eifel Algorithm: Making TCP Robust Against Spurious Retransmissions. *Computer Communication Review*, 30(1), January 2000.

[LM03] Reiner Ludwig and Michael Meyer. The Eifel Detection Algorithm for TCP, April 2003. RFC 3522.

[LS00] Reiner Ludwig and Keith Sklower. The Eifel Retransmission Timer. *ACM Computer Communication Review*, 30(3), July 2000.

[MAF05] Alberto Medina, Mark Allman, and Sally Floyd. Measuring the Evolution of Transport Protocols in the Internet. *ACM Computer Communication Review*, 35(2), April 2005.

[PA00] Vern Paxson and Mark Allman. Computing TCP's Retransmission Timer, November 2000. RFC 2988.

[SK05] Pasi Sarolahti and Markuu Kojo. Forward RTO-Recovery (F-RTO): An Algorithm for Detecting Spurious Retransmission Timeouts with TCP and the Stream Control Transmission Protocol (SCTP), August 2005. RFC 4138.