Internet Engineering Task Force (IETF)
Request for Comments: 6298
Obsoletes: 2988
Updates: 1122
Category: Standards Track
ISSN: 2070-1721
V. Paxson
ICSI/UC Berkeley
M. Allman
ICSI
J. Chu
Google
M. Sargent
CWRU
June 2011

Computing TCP's Retransmission Timer

Abstract

This document defines the standard algorithm that Transmission Control Protocol (TCP) senders are required to use to compute and manage their retransmission timer. It expands on the discussion in Section 4.2.3.1 of RFC 1122 and upgrades the requirement of supporting the algorithm from a SHOULD to a MUST. This document obsoletes RFC 2988.

Status of This Memo

1. Introduction

The Transmission Control Protocol (TCP) [Pos81] uses a retransmission timer to ensure data delivery in the absence of any feedback from the remote data receiver. The duration of this timer is referred to as RTO (retransmission timeout). RFC 1122 [Bra89] specifies that the RTO should be calculated as outlined in [Jac88].

This document codifies the algorithm for setting the RTO. In addition, this document expands on the discussion in Section 4.2.3.1 of RFC 1122 and upgrades the requirement of supporting the algorithm from a SHOULD to a MUST. RFC 5681 [APB09] outlines the algorithm TCP uses to begin sending after the RTO expires and a retransmission is sent. This document does not alter the behavior outlined in RFC 5681 [APB09].

In some situations, it may be beneficial for a TCP sender to be more conservative than the algorithms detailed in this document allow. However, a TCP MUST NOT be more aggressive than the following algorithms allow. This document obsoletes RFC 2988 [PA00].

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 [Bra97].

2. The Basic Algorithm

To compute the current RTO, a TCP sender maintains two state variables, SRTT (smoothed round-trip time) and RTTVAR (round-trip time variation). In addition, we assume a clock granularity of G seconds.

The rules governing the computation of SRTT, RTTVAR, and RTO are as follows:

(2.1) Until a round-trip time (RTT) measurement has been made for a segment sent between the sender and receiver, the sender SHOULD set RTO <- 1 second, though the "backing off" on repeated retransmission discussed in (5.5) still applies.

Note that the previous version of this document used an initial RTO of 3 seconds [PA00]. A TCP implementation MAY still use this value (or any other value > 1 second). This change in the lower bound on the initial RTO is discussed in further detail in Appendix A.

(2.2) When the first RTT measurement R is made, the host MUST set

```
SRTT <- R
RTTVAR <- R/2
RTO <- SRTT + max (G, K*RTTVAR)
```

where K = 4.

(2.3) When a subsequent RTT measurement R' is made, a host MUST set

```
RTTVAR <- (1 - beta) * RTTVAR + beta * |SRTT - R'|
SRTT <- (1 - alpha) * SRTT + alpha * R'
```

The value of SRTT used in the update to RTTVAR is its value before updating SRTT itself using the second assignment. That is, updating RTTVAR and SRTT MUST be computed in the above order.

The above SHOULD be computed using alpha=1/8 and beta=1/4 (as suggested in [JK88]).

After the computation, a host MUST update
RTO <- SRTT + max (G, K*RTTVAR)

(2.4) Whenever RTO is computed, if it is less than 1 second, then the RTO SHOULD be rounded up to 1 second.

Traditionally, TCP implementations use coarse grain clocks to measure the RTT and trigger the RTO, which imposes a large minimum value on the RTO. Research suggests that a large minimum RTO is needed to keep TCP conservative and avoid spurious retransmissions [AP99]. Therefore, this specification requires a large minimum RTO as a conservative approach, while at the same time acknowledging that at some future point, research may show that a smaller minimum RTO is acceptable or superior.

(2.5) A maximum value MAY be placed on RTO provided it is at least 60 seconds.

3. Taking RTT Samples

TCP MUST use Karn's algorithm [KP87] for taking RTT samples. That is, RTT samples MUST NOT be made using segments that were retransmitted (and thus for which it is ambiguous whether the reply was for the first instance of the packet or a later instance). The only case when TCP can safely take RTT samples from retransmitted segments is when the TCP timestamp option [JBB92] is employed, since the timestamp option removes the ambiguity regarding which instance of the data segment triggered the acknowledgment.

Traditionally, TCP implementations have taken one RTT measurement at a time (typically, once per RTT). However, when using the timestamp option, each ACK can be used as an RTT sample. RFC 1323 [JBB92] suggests that TCP connections utilizing large congestion windows should take many RTT samples per window of data to avoid aliasing effects in the estimated RTT. A TCP implementation MUST take at least one RTT measurement per RTT (unless that is not possible per Karn's algorithm).

For fairly modest congestion window sizes, research suggests that timing each segment does not lead to a better RTT estimator [AP99]. Additionally, when multiple samples are taken per RTT, the alpha and beta defined in Section 2 may keep an inadequate RTT history. A method for changing these constants is currently an open research question.

4. Clock Granularity

There is no requirement for the clock granularity G used for computing RTT measurements and the different state variables. However, if the K*RTTVAR term in the RTO calculation equals zero, the variance term MUST be rounded to G seconds (i.e., use the equation given in step 2.3).

    RTO <- SRTT + max (G, K*RTTVAR)

Experience has shown that finer clock granularities (<= 100 msec) perform somewhat better than coarser granularities.

Note that [Jac88] outlines several clever tricks that can be used to obtain better precision from coarse granularity timers. These changes are widely implemented in current TCP implementations.

5. Managing the RTO Timer

An implementation MUST manage the retransmission timer(s) in such a way that a segment is never retransmitted too early, i.e., less than one RTO after the previous transmission of that segment.

The following is the RECOMMENDED algorithm for managing the retransmission timer:

(5.1) Every time a packet containing data is sent (including a retransmission), if the timer is not running, start it running so that it will expire after RTO seconds (for the current value of RTO).

(5.2) When all outstanding data has been acknowledged, turn off the retransmission timer.

(5.3) When an ACK is received that acknowledges new data, restart the retransmission timer so that it will expire after RTO seconds (for the current value of RTO).

When the retransmission timer expires, do the following:

(5.4) Retransmit the earliest segment that has not been acknowledged by the TCP receiver.

(5.5) The host MUST set RTO <- RTO * 2 ("back off the timer"). The maximum value discussed in (2.5) above may be used to provide an upper bound to this doubling operation.

(5.6) Start the retransmission timer, such that it expires after RTO seconds (for the value of RTO after the doubling operation outlined in 5.5).

(5.7) If the timer expires awaiting the ACK of a SYN segment and the TCP implementation is using an RTO less than 3 seconds, the RTO MUST be re-initialized to 3 seconds when data transmission begins (i.e., after the three-way handshake completes).

This represents a change from the previous version of this document [PA00] and is discussed in Appendix A.

Note that after retransmitting, once a new RTT measurement is obtained (which can only happen when new data has been sent and acknowledged), the computations outlined in Section 2 are performed, including the computation of RTO, which may result in "collapsing" RTO back down after it has been subject to exponential back off (rule 5.5).

Note that a TCP implementation MAY clear SRTT and RTTVAR after backing off the timer multiple times as it is likely that the current SRTT and RTTVAR are bogus in this situation. Once SRTT and RTTVAR are cleared, they should be initialized with the next RTT sample taken per (2.2) rather than using (2.3).

6. Security Considerations

This document requires a TCP to wait for a given interval before retransmitting an unacknowledged segment. An attacker could cause a TCP sender to compute a large value of RTO by adding delay to a timed packet's latency, or that of its acknowledgment. However, the ability to add delay to a packet's latency often coincides with the ability to cause the packet to be lost, so it is difficult to see what an attacker might gain from such an attack that could cause more damage than simply discarding some of the TCP connection's packets.

The Internet, to a considerable degree, relies on the correct implementation of the RTO algorithm (as well as those described in RFC 5681) in order to preserve network stability and avoid congestion collapse. An attacker could cause TCP endpoints to respond more aggressively in the face of congestion by forging acknowledgments for segments before the receiver has actually received the data, thus lowering RTO to an unsafe value. But to do so requires spoofing the acknowledgments correctly, which is difficult unless the attacker can monitor traffic along the path between the sender and the receiver. In addition, even if the attacker can cause the sender's RTO to reach too small a value, it appears the attacker cannot leverage this into much of an attack (compared to the other damage they can do if they can spoof packets belonging to the connection), since the sending TCP will still back off its timer in the face of an incorrectly transmitted packet's loss due to actual congestion.

The security considerations in RFC 5681 [APB09] are also applicable to this document.

7. Changes from RFC 2988

This document reduces the initial RTO from the previous 3 seconds [PA00] to 1 second, unless the SYN or the ACK of the SYN is lost, in which case the default RTO is reverted to 3 seconds before data transmission begins.

8. Acknowledgments

The RTO algorithm described in this memo was originated by Van Jacobson in [Jac88].

Much of the data that motivated changing the initial RTO from 3 seconds to 1 second came from Robert Love, Andre Broido, and Mike Belshe.

9. References

9.1. Normative References

[APB09] Allman, M., Paxson, V., and E. Blanton, "TCP Congestion Control", RFC 5681, September 2009.

[Bra89] Braden, R., Ed., "Requirements for Internet Hosts - Communication Layers", STD 3, RFC 1122, October 1989.

[Bra97] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[JBB92] Jacobson, V., Braden, R., and D. Borman, "TCP Extensions for High Performance", RFC 1323, May 1992.

[Pos81] Postel, J., "Transmission Control Protocol", STD 7, RFC 793, September 1981.

9.2. Informative References

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

[Chu09] Chu, J., "Tuning TCP Parameters for the 21st Century", http://www.ietf.org/proceedings/75/slides/tcpm-1.pdf, July 2009.

[SLS09] Schulman, A., Levin, D., and Spring, N., "CRAWDAD data set umd/sigcomm2008 (v. 2009-03-02)", http://crawdad.cs.dartmouth.edu/umd/sigcomm2008, March, 2009.

[HKA04] Henderson, T., Kotz, D., and Abyzov, I., "CRAWDAD trace dartmouth/campus/tcpdump/fall03 (v. 2004-11-09)", http://crawdad.cs.dartmouth.edu/dartmouth/campus/tcpdump/fall03, November 2004.

[Jac88] Jacobson, V., "Congestion Avoidance and Control", Computer Communication Review, vol. 18, no. 4, pp. 314-329, Aug. 1988.

[JK88] Jacobson, V. and M. Karels, "Congestion Avoidance and Control", ftp://ftp.ee.lbl.gov/papers/congavoid.ps.Z.

[KP87] Karn, P. and C. Partridge, "Improving Round-Trip Time Estimates in Reliable Transport Protocols", SIGCOMM 87.

[PA00] Paxson, V. and M. Allman, "Computing TCP's Retransmission Timer", RFC 2988, November 2000.

Appendix A. Rationale for Lowering the Initial RTO

Choosing a reasonable initial RTO requires balancing two competing considerations:

1. The initial RTO should be sufficiently large to cover most of the end-to-end paths to avoid spurious retransmissions and their associated negative performance impact.

2. The initial RTO should be small enough to ensure a timely recovery from packet loss occurring before an RTT sample is taken.

Traditionally, TCP has used 3 seconds as the initial RTO [Bra89] [PA00]. This document calls for lowering this value to 1 second using the following rationale:

- Modern networks are simply faster than the state-of-the-art was at the time the initial RTO of 3 seconds was defined.

- Studies have found that the round-trip times of more than 97.5% of the connections observed in a large scale analysis were less than 1 second [Chu09], suggesting that 1 second meets criterion 1 above.

- In addition, the studies observed retransmission rates within the three-way handshake of roughly 2%. This shows that reducing the initial RTO has benefit to a non-negligible set of connections.

- However, roughly 2.5% of the connections studied in [Chu09] have an RTT longer than 1 second. For those connections, a 1 second initial RTO guarantees a retransmission during connection establishment (needed or not).

  When this happens, this document calls for reverting to an initial RTO of 3 seconds for the data transmission phase. Therefore, the implications of the spurious retransmission are modest: (1) an extra SYN is transmitted into the network, and (2) according to RFC 5681 [APB09] the initial congestion window will be limited to 1 segment. While (2) clearly puts such connections at a disadvantage, this document at least resets the RTO such that the connection will not continually run into problems with a short timeout. (Of course, if the RTT is more than 3 seconds, the connection will still encounter difficulties. But that is not a new issue for TCP.)

  In addition, we note that when using timestamps, TCP will be able to take an RTT sample even in the presence of a spurious retransmission, facilitating convergence to a correct RTT estimate when the RTT exceeds 1 second.

As an additional check on the results presented in [Chu09], we analyzed packet traces of client behavior collected at four different vantage points at different times, as follows:

Name	Dates	Pkts.	Cnns.	Clnts.	Servs.
LBL-1	Oct/05--Mar/06	292M	242K	228	74K
LBL-2	Nov/09--Feb/10	1.1B	1.2M	1047	38K
ICSI-1	Sep/11--18/07	137M	2.1M	193	486K
ICSI-2	Sep/11--18/08	163M	1.9M	177	277K
ICSI-3	Sep/14--21/09	334M	3.1M	170	253K
ICSI-4	Sep/11--18/10	298M	5M	183	189K
Dartmouth	Jan/4--21/04	1B	4M	3782	132K
SIGCOMM	Aug/17--21/08	11.6M	133K	152	29K

The "LBL" data was taken at the Lawrence Berkeley National Laboratory, the "ICSI" data from the International Computer Science Institute, the "SIGCOMM" data from the wireless network that served the attendees of SIGCOMM 2008, and the "Dartmouth" data was collected from Dartmouth College's wireless network. The latter two datasets are available from the CRAWDAD data repository [HKA04] [SLS09]. The table lists the dates of the data collections, the number of packets collected, the number of TCP connections observed, the number of local clients monitored, and the number of remote servers contacted. We consider only connections initiated near the tracing vantage point.

Analysis of these datasets finds the prevalence of retransmitted SYNs to be between 0.03% (ICSI-4) to roughly 2% (LBL-1 and Dartmouth).

We then analyzed the data to determine the number of additional and spurious retransmissions that would have been incurred if the initial RTO was assumed to be 1 second. In most of the datasets, the proportion of connections with spurious retransmits was less than 0.1%. However, in the Dartmouth dataset, approximately 1.1% of the connections would have sent a spurious retransmit with a lower initial RTO. We attribute this to the fact that the monitored network is wireless and therefore susceptible to additional delays from RF effects.

Finally, there are obviously performance benefits from retransmitting lost SYNs with a reduced initial RTO. Across our datasets, the percentage of connections that retransmitted a SYN and would realize at least a 10% performance improvement by using the smaller initial RTO specified in this document ranges from 43% (LBL-1) to 87% (ICSI-4). The percentage of connections that would realize at least a 50% performance improvement ranges from 17% (ICSI-1 and SIGCOMM) to 73% (ICSI-4).

From the data to which we have access, we conclude that the lower initial RTO is likely to be beneficial to many connections, and harmful to relatively few.

Authors' Addresses

Vern Paxson
ICSI/UC Berkeley
1947 Center Street
Suite 600
Berkeley, CA 94704-1198

Phone: 510-666-2882
EMail: vern@icir.org
http://www.icir.org/vern/

Mark Allman
ICSI
1947 Center Street
Suite 600
Berkeley, CA 94704-1198

Phone: 440-235-1792
EMail: mallman@icir.org
http://www.icir.org/mallman/

H.K. Jerry Chu
Google, Inc.
1600 Amphitheatre Parkway
Mountain View, CA 94043

Phone: 650-253-3010
EMail: hkchu@google.com

Matt Sargent
Case Western Reserve University
Olin Building
10900 Euclid Avenue
Room 505
Cleveland, OH 44106

Phone: 440-223-5932
EMail: mts71@case.edu