TCPM Working Group   J. Touch
Internet Draft   USC/ISI
Intended status: Standards Track   July 16, 2012
Expires: January 2013

                  Automating the Initial Window in TCP
                  draft-touch-tcpm-automatic-iw-03.txt

Status of this Memo

   This Internet-Draft is submitted in full conformance with the
   provisions of BCP 78 and BCP 79.

   Internet-Drafts are working documents of the Internet Engineering
   Task Force (IETF), its areas, and its working groups. Note that
   other groups may also distribute working documents as Internet-
   Drafts.

   Internet-Drafts are draft documents valid for a maximum of six
   months and may be updated, replaced, or obsoleted by other documents
   at any time. It is inappropriate to use Internet-Drafts as
   reference material or to cite them other than as "work in progress."

   The list of current Internet-Drafts can be accessed at
   http://www.ietf.org/ietf/1id-abstracts.txt

   The list of Internet-Draft Shadow Directories can be accessed at
   http://www.ietf.org/shadow.html

   This Internet-Draft will expire on January 16, 2011.

Copyright Notice

Abstract

   The Initial Window (IW) provides the starting point for TCP's
   feedback-based congestion control algorithm. Its value has increased
   over time to increase performance and to reflect increased
   capability of Internet devices. This document describes a mechanism
   to adjust the IW over long timescales, to make future changes more
   safely deployed and to potentially avoid reexamination of this value
   in the future.

Table of Contents

1. Introduction

   TCP's congestion control algorithm uses an initial window value
   (IW), both as a starting point for new connections and after one RTO
   or more [RFC2581][RFC2861]. This value has evolved over time,
   originally one maximum segment size (MSS), and increased to the
   lesser of four MSS or 4,380 bytes [RFC3390][RFC5681]. For typical
   Internet connections with an maximum transmission units (MTUs) of
   1500 bytes, this permits three segments of 1,460 bytes each.

   The IW value was originally implied in the original TCP congestion
   control description, and documented as a standard in 1997
   [RFC2001][Ja88]. The value was last updated in 1998 experimentally,
   and moved to the standards track in 2002 [RFC2414][RFC3390]. There
   have been recent proposals to update the IW based on further
   increases in host and router capabilities and network capacity, some

focusing on specific values (e.g., IW=10), and others prescribing a
schedule for increases over time (e.g., IW=6 for 2011, increasing by
1-2 MSS per year).

This document proposes that TCP can objectively measure when an IW
is too large, and that such feedback should be used over long
timescales to adjust the IW automatically. The result should be
safer to deploy and might avoid the need to repeatedly revisit IW
size over time.

2. Conventions used in this document

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].

In this document, these words will appear with that interpretation
only when in ALL CAPS. Lower case uses of these words are not to be
interpreted as carrying RFC-2119 significance.

In this document, the characters ">>" preceding an indented line(s)
indicates a compliance requirement statement using the key words
listed above. This convention aids reviewers in quickly identifying
or finding the explicit compliance requirements of this RFC.

3. Design Considerations

TCP's IW value has existed statically for over two decades, so any
solution to adjusting the IW dynamically should have similarly
stable, non-invasive effects on the performance and complexity of
TCP. In order to be fair, the IW should be similar for most machines
on the public Internet. Finally, a desirable goal is to develop a
self-correcting algorithm, so that IW values that cause network
problems can be avoided. To that end, we propose the following list
of design goals:

o Impart little to no impact to TCP in the absence of loss, i.e.,
   it should not increase the complexity of default packet
   processing in the normal case.

o Adapt to network feedback over long timescales, avoiding values
   that persistently cause network problems.

o Decrease the IW in the presence of sustained loss of IW segments,
   as determined over a number of different connections.

o Increase the IW in the absence of sustained loss of IW segments,
   as determined over a number of different connections.

o Operate conservatively, i.e., tend towards leaving the IW the
   same in the absence of sufficient information, and give greater
   consideration to IW segment loss than IW segment success.

We expect that, without other context, a good IW algorithm will
converge to a single value, but this is not required. An endpoint
with additional context or information, or deployed in a constrained
environment, can always use a different value. In specific,
information from previous connections, or sets of connections with a
similar path, can already be used as context for such decisions
[RFC2140].

However, if a given IW value persistently causes packet loss during
the initial burst of packets, it is clearly inappropriate and could
be inducing unnecessary loss in other competing connections. This
might happen for sites behind very slow boxes with small buffers,
which may or may not be the first hop.

4. Proposed IW Algorithm

Below is a simple description of the proposed IW algorithm. It
relies on the following parameters:

o MinIW = 3 MSS or 4,380 bytes (as per RFC3390]

o MaxIW = 10

o MulDecr = 0.5

o AddIncr = 2 MSS

o Threshold = 0.05

We assume that the minimum IW (MinIW) should be as currently
specified [RFC3390]. The maximum IW can be set to a fixed value
[Ch10], or set based on a schedule if trusted time references are
available [Al10]; here we prefer a fixed value. We also propose to
use an AIMD algorithm, with increase and decreases as noted.

Although these parameters are somewhat arbitrary, their initial
values are not important except that the algorithm is AIMD and the
MaxIW should not exceed that recommended for other systems on the
Internet. Current proposals, including default current operation,
are degenerate cases of the algorithm below for given parameters -

notably MulDec = 1.0 and AddIncr = 0 MSS, thus disabling the
automatic part of the algorithm.

The proposed algorithm is as follows:

0. On boot:

   IW = MaxIW; # assume this is in bytes, and an even number of MSS

1. Upon starting a new connection

   CWND = IW;
   conncount++;
   IWnotchecked = 1; # true

2. During a connection's SYN-ACK processing, if SYN-ACK includes
   ECN, treat as if the IW is too large

   if (IWnotchecked && (synackecn == 1)) {
      losscount++;
      IWnotchecked = 0; # never check again
   }

3. During a connection, if retransmission occurs, check the seqno of
   the outgoing packet (in bytes) to see if the resent segment fixes
   an IW loss:

   if (Retransmitting && IWnotchecked && ((ISN - seqno) < IW))) {
      losscount++;
      IWnotchecked = 0; # never do this entire "if" again
   } else {
      IWnotchecked = 0; # you're beyond the IW so stop checking
   }

4. Once every 1000 conections, as a separate process (i.e., not as
   part of processing a given connection):

```
    if (conncount > 1000) {
        if (losscount/conncount > threshold) {
            # the number of connections with errors is too high
            IW = IW * MulDecr;
        } else {
            IW = IW + AddIncr;
        }
    }
```

We recognize that this algorithm can yield a false positive when the sequence number wraps around. This can be avoided using either PAWS [RFC1323] context or 64-bit internal sequence numbers (as in TCP-AO [RFC5925]). Alternately, false positives can be allowed since they are expected to be infrequent and thus will not affect the overall statistics of the algorithm.

The following additional constraints are imposed:

>> The automatic IW algorithm MUST initialize to MaxIW, in the absence of other context information.

If there are too few connections to make a decision, or if there is otherwise insufficient information to increase the IW, then the MaxIW defaults to the current recommended value.

>> An implementation may allow the MaxIW to grow beyond the currently recommended Internet default, but not more than 2 segments per calendar year.

If an endpoint has a persistent history of successfully transmitting IW segments without loss, then it is allowed to probe the Internet to determine if larger IW values have similar success. This probing is limited and requires a trusted time source, otherwise the MaxIW remains constant.

>> An implementation MUST adjust the IW based on loss statistics at least once every 1000 connections.

An endpoint needs to be sufficiently reactive to IW loss.

>> An implementation MUST decrease the IW by at least one MSS when indicated during an evaluation interval.

An endpoint that detects loss needs to decrease its IW by at least one MSS, otherwise it is not participating in an automatic reactive algorithm.

>> An implementation MUST increase by no more than 2 MSS per
evaluation interval.

An endpoint that does not experience IW loss needs to probe the
network incrementally.

>> An implementation SHOULD use an IW that is an integer multiple of
2 MSS.

The IW should remain a multiple of 2 MSS segments, to enable
efficient ACK compression without incurring unnecessary timeouts.

>> An implementation MUST decrease the IW if more than 95% of
connections have IW losses.

Again, this is to ensure an implementation is sufficiently reactive.

>> An implementation MAY group IW values and statistics within
subsets of connections. Such grouping MAY use any information about
connections to form groups except loss statistics.

There are some TCP connections which might not be counted at all,
such as those to/from loopback addresses, or those within the same
subnet as that of a local interface (for which congestion control is
sometimes disabled anyway). This may also include connections that
terminate before the IW is full, i.e., as a separate check at the
time of the connection closing.

The period over which the IW is updated is intended to be a long
timescale, e.g., a month or so, or 1,000 connections, whichever is
longer. An implementation might check the IW once a month, and
simply not update the IW or clear the connection counts in months
where the number of connections is too small.

5. Discussion

There are numerous parameters to the above algorithm that are
compliant with the given requirements; this is intended to allow
variation in configuration and implementation while ensuring that
all such algorithms are reactive and safe.

This algorithm continues to assume segments because that is the
basis of most TCP implementations. It might be useful to consider
revising the specifications to allow byte-based congestion given
sufficient experience.

The algorithm checks for IW losses only during the first IW after a
connection start; it does not check for IW losses elsewhere the IW
is used, e.g., during slow-start restarts.

>> An implementation MAY detect IW losses during slow-start restarts
in addition to losses during the first IW of a connection. In this
case, the implementation MUST count each restart as a "connection"
for the purposes of connection counts and periodic rechecking of the
IW value.

False positives can occur during some kinds of segment reordering,
e.g., that might trigger spurious retransmissions even without a
true segment loss. These are not expected to be sufficiently common
to dominate the algorithm and its conclusions.

This mechanism does require additional per-connection state which is
currently common in some implementations, and is useful for other
reasons (e.g., the ISN is used in TCP-AO [RFC5925]). The mechanism
also benefits from persistent state kept across reboots, as would be
other state sharing mechanisms (e.g., TCP Control Block Sharing
[RFC2140]). The mechanism is inspired by RFC 2140's use of
information across connections.

The receive window (RWIN) is not involved in this calculation. The
size of RWIN is determined by receiver resources, and provides space
to accommodate segment reordering. It is not involved with
congestion control, which is the focus of this document and its
management of the IW.

6. Observations

The IW may not converge to a single, global value. It also may not
converge at all, but rather may oscillate by a few MSS as it
repeatedly probes the Internet for larger IWs and fails. Both
properties are consistent with TCP behavior during each individual
connection.

This mechanism assumes that losses during the IW are due to IW size.
Persistent errors that drop packets for other reasons - e.g., OS
bugs, can cause false positives. Again, this is consistent with
TCP's basic assumption that loss is caused by congestion and
requires backoff. This algorithm treats the IW of new connections as
a long-timescale backoff system.

7. Security Considerations

   This algorithm presents an opportunity for an intelligent attack to
   reduce the IW of a given system, by repeatedly dropping packets
   during the IW only. An intermediate that can drop packets in a
   controlled manner can already impact the performance of a
   connection, and can reduce the congestion window of an ongoing
   connection in ways that impact performance more than just dropping
   during the IW.

8. IANA Considerations

   This document has no IANA considerations. This section should be
   removed prior to publication.

9. Conclusions

   <Add any conclusions>

10. References

10.1. Normative References

   [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate
             Requirement Levels", BCP 14, RFC 2119, March 1997.

   [RFC3390] Allman, M., Floyd, S., Partridge, C., "Increasing TCP's
             Initial Window", RFC 3390 (Standards Track), Oct. 2002.

   [RFC5681] Allman, M., Paxson, V., Blanton, E., "TCP Congestion
             Control," RFC 5681 (Standards Track), Sep. 2009.

10.2. Informative References

   [Al10]   Allman, M., "Initial Congestion Window Specification",
             (work in progress), draft-allman-tcpm-bump-initcwnd-00,
             Nov. 2010.

   [Ch10]   Chu, J., Dukkipati, N., Cheng, Y., Mathis, M., "Increasing
             TCP's Initial Window," (work in progress), draft-ietf-
             tcpm-initcwnd-04, Jun. 2012.

   [Ja88]   Jacobson, V., M. Karels, "Congestion Avoidance and
             Control", Proc. Sigcomm 1988.

   [RFC1323] Jacobson, V., Braden, R., Borman, D., "TCP Extensions for
             High Performance", RFC 1323, May 1992.

   [RFC2001] Stevens, W., "TCP Slow Start, Congestion Avoidance, Fast
             Retransmit, and Fast Recovery Algorithms", RFC2001
             (Standards Track), Jan. 1997.

   [RFC2140] Touch, J., "TCP Control Block Interdependence", RFC 2140 /
             STD 7(Informational), Apr. 1997.

   [RFC2414] Allman, M., Floyd, S., Partridge, C., "Increasing TCP's
             Initial Window", RFC 2414 (Experimental), Sept. 1998.

   [RFC2581] Allman, M., Paxson, V., Stevens, W., "TCP Congestion
             Control," RFC2581 (Standards Track), Apr. 1999.

   [RFC2861] Handley, M., Padhye, J., Floyd, S., "TCP Congestion Window
             Validation", RFC2861 (Experimental), June 2000.

   [RFC5925] Touch, J., A. Mankin, R. Bonica, "The TCP Authentication
             Option", RFC 5925 (Standards Track), June 2010.

11. Acknowledgments

   Mark Allman and Aki Nyrjinen contributed to the development of this
   algorithm. Members of the TCPM mailing list also participated in
   providing useful feedback.

   This document was prepared using 2-Word-v2.0.template.dot.

Authors' Addresses

   Joe Touch
   USC/ISI
   4676 Admiralty Way
   Marina del Rey, CA 90292-6695 U.S.A.

   Phone: +1 (310) 448-9151
   Email: touch@isi.edu