TCPM Working Group
Internet-Draft
Obsoletes: 2861 (if approved)
Updates: 5681 (if approved)
Intended status: Standards Track
Expires: February 7, 2013

G. Fairhurst
A. Sathiaseelan
University of Aberdeen

August 06, 2012

Updating TCP to support Application-Limited Traffic
draft-fairhurst-tcpm-newcwv-04

Abstract

This document addresses issues that arise when TCP is used to support traffic that exhibits periods where the transmission rate is limited by the application rather than the congestion window. It updates TCP to allow a TCP sender to restart quickly following either an idle or application-limited interval. The method is expected to benefit application-limited TCP applications, while also providing an appropriate response if congestion is experienced.

It also evaluates TCP Congestion Window Validation, CWV, an IETF experimental specification defined in RFC 2861, and concludes that CWV sought to address important issues, but failed to deliver a widely used solution. This document therefore proposes an update to the status of RFC 2861 by recommending it is moved from Experimental to Historic status, and that it is replaced by the current specification.

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). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

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."

This Internet-Draft will expire on February 7, 2013.

Copyright Notice

Table of Contents

1. Introduction

TCP is used to support a range of application behaviours. The TCP congestion window (cwnd) controls the number of packets/bytes that a TCP flow may have in the network at any time. A bulk application will always have data available to transmit. The rate at which it sends is therefore limited by the maximum permitted by the receiver and congestion windows. In contrast, a rate-limited application will experience periods when the sender is either idle or is unable to send at the maximum rate permitted by the cwnd. This latter case is called application-limited. The focus of this document is on the operation of TCP in such an idle or application-limited case.

Standard TCP [RFC5681] requires the cwnd to be reset to the restart window (RW) when an application becomes idle. [RFC2861] noted that this TCP behaviour was not always observed in current implementations. Recent experiments [Bis08] confirm this to still be the case.

Standard TCP does not control growth of the cwnd when a TCP sender is application-limited. An application-limited sender may therefore grow a cwnd beyond that corresponding to the current transmit rate, resulting in a value that does not reflect current information about the state of the network path the flow is using. Use of such an invalid cwnd may result in reduced application performance and/or could significantly contribute to network congestion.

[RFC2861] proposed a solution to these issues in an experimental method known as Congestion Window Validation (CWV). CWV was intended to help reduce cases where TCP accumulated an invalid cwnd. The use and drawbacks of using CWV with an application are discussed in Section 2.

Section 4 specifies an alternative to CWV that seeks to address the same issues, but does this in a way that is expected to mitigate the impact on an application that varies its transmission rate. The method described applies to both an application-limited and an idle condition.

2. Reviewing experience with TCP-CWV

RFC 2861 described a simple modification to the TCP congestion control algorithm that decayed the cwnd after the transition to a "sufficiently-long" idle period. This used the slow-start threshold (ssthresh) to save information about the previous value of the congestion window. The approach relaxed the standard TCP behaviour [RFC5681] for an idle session, intended to improve application

performance. CWV also modified the behaviour for an application-limited session where a sender transmitted at a rate less than allowed by cwnd.

RFC 2861 has been implemented in some mainstream operating systems as the default behaviour [Bis08]. Analysis (e.g. [Bis10]) has shown that a TCP sender using CWV is able to use available capacity on a shared path after an idle period. This can benefit some applications, especially over long delay paths, when compared to slow-start restart specified by standard TCP. However, CWV would only benefit an application if the idle period were less than several Retransmission Time Out (RTO) intervals [RFC6298], since the behaviour would otherwise be the same as for standard TCP, which resets the cwnd to the RW after this period.

Experience with CWV suggests that although CWV benefits the network in an application-limited scenario (reducing the probability of network congestion), the behaviour can be too conservative for many common rate-limited applications. This mechanism does not therefore offer the desirable increase in application performance for rate-limited applications and it is unclear whether applications actually use this mechanism in the general Internet.

It is therefore concluded that CWV is often a poor solution for many rate-limited applications. It has the correct motivation, but has the wrong approach to solving this problem.

3. 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 [RFC2119].

The document assumes familiarity with the terminology of TCP congestion control [RFC5681].

4. An updated TCP response to idle and application-limited periods

This section proposes an update to the TCP congestion control behaviour during an idle or application-limited period. The new method permits a TCP sender to preserve the cwnd when an application becomes idle for a period of time (set in this specification to 5 minutes, see section 5). This period, where actual usage is less than allowed by cwnd, is named the non-validated phase. The method allows an application to resume transmission at a previous rate without incurring the delay of slow-start. However, if the TCP

sender experiences congestion using the preserved cwnd, it is required to immediately reset the cwnd to an appropriate value specified by the method. If a sender does not take advantage of the preserved cwnd within five minutes, the value of cwnd is reduced, ensuring the value then reflects the capacity that was recently actually used.

The method requires that the TCP SACK option is enabled. This allows the sender to select a cwnd following a congestion event that is based on the measured path capacity, better reflecting the fair-share. A similar approach was proposed by TCP Jump Start [Liu07], as a congestion response after more rapid opening of a TCP connection.

It is expected that this update will satisfy the requirements of many rate-limited applications and at the same time provide an appropriate method for use in the Internet. It also reduces the incentive for an application to send data simply to keep transport congestion state. (This is sometimes known as "padding").

The new method does not differentiate between times when the sender has become idle or application-limited. This is partly a response to recognition that some applications wish to transmit at a rate-limited, and that it can be hard to make a distinction between application-limited and idle behaviour. This is expected to encourage applications and TCP stacks to use standards-based congestion control methods. It may also encourage the use of long-lived connections where this offers benefit (such as persistent http).

The method is specified in following subsections.

4.1. A method for preserving cwnd in idle and application-limited periods.

The method described in this document updates [RFC5681]. Use of the method REQUIRES a TCP sender and the corresponding receiver to enable the TCP SACK option [RFC3517].

[RFC5681] defines a variable FlightSize , that indicates the amount of outstanding data in the network. This equal to the value of Pipe calculated based on the pipe algorithm [RFC3517]. In RFC5681 this value is used during loss recovery, whereas in this method it is also used during normal data transfer. A sender is not required to continuously track this value, but SHOULD measure the volume of data in the network with a sampling period of not less than one RTT period.

4.2. The nonvalidated phase

The updated method creates a new TCP sender phase that captures whether the cwnd reflects a validated or non-validated value. The phases are defined as:

- Validated phase: FlightSize >=(3/4)*cwnd. This is the normal phase, where cwnd is expected to be an approximate indication of the available capacity currently available along the network path, and the standard methods are used to increase cwnd (currently [RFC5681]).

- Non-validated phase: FlightSize <(1/4)*cwnd. This is the phase where the cwnd has a value based on a previous measurement of the available capacity, and the usage of this capacity has not been validated in the previous RTT. That is, when it is not known whether the cwnd reflects the currently available capacity available along the network path. The mechanisms to be used in this phase seek to determine whether any resumed rate remains safe for the Internet path, i.e., it quickly reduces the rate if the flow is known to induce congestion. These mechanisms are specified in section 4.3.

The values 1/4 and 3/4 were selected to reduce the effects of variations in the measured FlightSize.

4.3. TCP congestion control during the nonvalidated phase

A TCP sender that enters the non-validated phase MUST preserve the cwnd (i.e., this neither grows nor reduces while the sender remains in this phase). The phase is concluded after a fixed period of time (five minutes, as explained in section or when the sender transmits using the full cwnd (i.e. it is no longer application-limited).

The behaviour in the non-validated phase is specified as:

- If the sender consumes all the available space within the cwnd (i.e., the remaining unused cwnd in bytes is less than one Sender Maximum Segment Size, SMSS), then the sender MUST exit the non-validated phase. The threshold value of cwnd required for the sender to enter the non-validated phase is intentionally different to that required to leave the phase. This introduces hysteresis to avoid rapid oscillation between the phases. Note that a change between phases does not significantly impact an application-limited sender, but serves to determine its behaviour if it substantially increases its transmission rate.

- If the sender receives an indication of congestion while in the non-validated phase (i.e. detects loss, or an Explicit Congestion Notification, ECN, mark [RFC3168]), the sender MUST exit the non-validated phase (reducing the cwnd as defined in section 4.3.1).

- If the Retransmission Time Out (RTO) expires while in the non-validated phase, the sender MUST exit the non-validated phase. It then resumes using the Standard TCP RTO mechanism [RFC5681]. (The resulting reduction of cwnd describe din section 4.3.2 is appropriate, since any accumulated path history is considered unreliable).

4.3.1. Response to congestion in the nonvalidated phase

Reception of congestion feedback while in the non-validated phase is interpreted as an indication that it was inappropriate for the sender to use the preserved cwnd. The sender is therefore required to quickly reduce the rate to avoid further congestion. Since the cwnd does not have a validated value, a new cwnd value must be selected based on the utilised rate.

A sender that detects a packet-drop or receives an ECN marked packet MUST calculate a safe cwnd, by setting it to the value specified in Section 3.2 of [RFC5681].

At the end of the recovery phase, the TCP sender MUST reset the cwnd using the method below:

cwnd = ((FlightSize - R)/2).

Where, R is the volume of data that was reported as unacknowledged by the SACK information. This follows the method proposed for Jump Start [[Liu07].

The inclusion of the term R makes this adjustment is more conservative than standard TCP. This is required, since the sender may have sent more segments than Standard TCP would have done.

If the sender implements a method that allows it to identify the number of ECN-marked segments within a windowthat were observed by the receiver, the sender SHOULD use the method above, further reducing R by the number of marked segments.

4.3.2. Adjustment at the end of the nonvalidated phase

During the non-validated phase, the sender may produce bursts of data of up to the cwnd in size. While this is no different to standard TCP, it is desirable to control the maximum burst size, e.g. by setting a burst size limit, using a pacing algorithm, or some other

method [Hug01].

An application that remains in the non-validated phase for a period greater than five minutes is required to adjust its congestion control state. At the end of the non-validated phase, the sender MUST update the ssthresh:

sthresh = max(ssthresh, 3*cwnd/4).

(This adjustment of ssthresh ensures that the sender records that it has safely sustained the present rate. The change is beneficial to application-limited flows that encounter occasional congestion, and could otherwise suffer an unwanted additional delay in recovering the transmission rate.)

The sender MUST then update cwnd:

cwnd = max(FlightSize*2, IW).

Where IW is the TCP inital window [RFC5681].

(This allows an application to continue to send at the currently utilised rate, and not incur delay should it increase to twice the utilised rate.)

After completing this adjustment, the sender MAY re-enter the non-validated phase, if required (see section 4.2).

5. Determining a safe period to preserve cwnd

This section documents the rationale for selecting the maximum period that cwnd may be preserved.

Preserving cwnd avoids undesirable side effects that would result if the cwnd were to be preserved for an arbitrary long period, which was a part of the problem that CWV originally attempted to address. The period a sender may safely preserve the cwnd, is a function of the period that a network path is expected to sustain the capacity reflected by cwnd. There is no ideal choice for this time.

The period of five minutes was chosen as a compromise that was larger than the idle intervals of common applications, but not sufficiently larger than the period for which the capacity of an Internet path may commonly be regarded as stable. The capacity of wired networks is usually relatively stable for periods of several minutes and that load stability increases with the capacity. This suggests that cwnd may be preserved for at least a few minutes.

There are cases where the TCP throughput exhibits significant variability over a time less than five minutes. Examples could include wireless topologies, where TCP rate variations may fluctuate on the order of a few seconds as a consequence of medium access protocol instabilities. Mobility changes may also impact TCP performance over short time scales. Senders that observe such rapid changes in the path characteristic may also experience increased congestion with the new method, however such variation would likely also impact TCP's behaviour when supporting interactive and bulk applications.

Routing algorithms may modify the network path, disrupting the RTT measurement and changing the capacity available to a TCP connection, however such changes do not often occur within a time frame of a few minutes.

The value of five minutes is therefore expected to be sufficient for most current applications. Simulation studies also suggest that for many practical applications, the performance using this value will not be significantly different to that observed using a non-standard method that does not reset the cwnd after idle.

Finally, other TCP sender mechanisms have used a 5 minute timer, and there could be simplifications in some implementations by reusing the same interval. TCP defines a default user timeout of 5 minutes [RFC0793] i.e. how long transmitted data may remain unacknowledged before a connection is forcefully closed.

6. Security Considerations

General security considerations concerning TCP congestion control are discussed in [RFC5681]. This document describes an algorithm that updates one aspect of the congestion control procedures, and so the considerations described in RFC 5681 also apply to this algorithm.

7. IANA Considerations

There are no IANA considerations.

8. Acknowledgments

The authors acknowledge the contributions of Dr I Biswas and Dr R Secchi in supporting the evaluation of CWV and for their help in developing the mechanisms proposed in this draft. We also acknowledge comments received from the Internet Congestion Control

Research Group, in particular Yuchung Cheng, Mirja Kuehlewind, and Joe Touch.

9. Other related work - Author Notes

There are several issues to be discussed more widely:

- Should the method explicitly state a procedure for limiting burstiness or pacing?

  This is often regarded as good practice, but isn't a formal part of TCP. draft-hughes-restart-00.txt provides some discussion of this topic.

- There are potential interaction with the proposal to raise the TCP initial Window to ten segments, do these cases need to be elaborated?

  This relates to draft-ietf-tcpm-initcwnd.

  The two methods have different functions and different response to loss/congestion.

  IW=10 proposes an experimental update to TCP that would allow faster opening of the cwnd, and also a large (same size) restart window. This approach is based on the assumption that many forward paths can sustain bursts of up to ten segments without (appreciable) loss. Such a significant increase in cwnd must be matched with an equally large reduction of cwnd if loss/congestion is detected, and such a congestion indication is likely to require future use of IW=10 to be disabled for this path for some time. This guards against the unwanted behaviour of a series of short flows continuously flooding a network path without network congestion feedback.

  In contrast, new-CWV proposes a standards-track update with a rationale that relies on recent previous path history to select an appropriate cwnd after restart.

  The behaviour differs in three ways:

  1) For applications that send little initially, new-cwv may constrain more than IW=10, but would not require the connection

to reset any path information when a restart incurred loss. In contrast, new-cwv would allow the TCP connection to preserve the cached cwnd, any loss, would impact cwnd, but not impact other flows.

2) For applications that utilise more capacity than provided by a cwnd=10, this method would permit a larger restart window compared to a restart using IW=10. This is justified by the recent path history.

3) new-CWV is attended to also be used for application-limited use, where the application sends, but does not seek to fully utilise the cwnd. In this case, new-cwv constrains the cwnd to that justified by the recent path history. The performance trade-offs are hence different, and it would be possible to enable new-cwv when also using IW=10, and yield the benefits of this.

o There is potential overlap with the Laminar proposal (draft-mathis-tcpm-tcp-laminar)

The current draft was intended as a standards-track update to TCP, rather than a new transport variant. At least, it would be good to understand how the two interact and whether there is a possibility of a single method.

10. References

10.1. Normative References

[RFC0793] Postel, J., "Transmission Control Protocol", STD 7, RFC 793, September 1981.

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

[RFC2861] Handley, M., Padhye, J., and S. Floyd, "TCP Congestion Window Validation", RFC 2861, June 2000.

[RFC3168] Ramakrishnan, K., Floyd, S., and D. Black, "The Addition of Explicit Congestion Notification (ECN) to IP", RFC 3168, September 2001.

[RFC3517] Blanton, E., Allman, M., Fall, K., and L. Wang, "A Conservative Selective Acknowledgment (SACK)-based Loss

Recovery Algorithm for TCP", RFC 3517, April 2003.

[RFC5681] Allman, M., Paxson, V., and E. Blanton, "TCP Congestion Control", RFC 5681, September 2009.

[RFC6298] Paxson, V., Allman, M., Chu, J., and M. Sargent, "Computing TCP's Retransmission Timer", RFC 6298, June 2011.

10.2. Informative References

[Bis08] Biswas and Fairhurst, "A Practical Evaluation of Congestion Window Validation Behaviour, 9th Annual Postgraduate Symposium in the Convergence of Telecommunications, Networking and Broadcasting (PGNet), Liverpool, UK", June 2008.

[Bis10] Biswas, Sathiaseelan, Secchi, and Fairhurst, "Analysing TCP for Bursty Traffic, Int'l J. of Communications, Network and System Sciences, 7(3)", June 2010.

[Hug01] Hughes, Touch, and Heidemann, "Issues in TCP Slow-Start Restart After Idle (Work-in-Progress)", December 2001.

[Liu07] Liu, Allman, Jiny, and Wang, "Congestion Control without a Startup Phase, 5th International Workshop on Protocols for Fast Long-Distance Networks (PFLDnet), Los Angeles, California, USA", February 2007.

Authors' Addresses

Godred Fairhurst
University of Aberdeen
School of Engineering
Fraser Noble Building
Aberdeen, Scotland AB24 3UE
UK

Email: gorry@erg.abdn.ac.uk
URI: http://www.erg.abdn.ac.uk

Arjuna Sathiaseelan
University of Aberdeen
School of Engineering
Fraser Noble Building
Aberdeen, Scotland AB24 3UE
UK

Email: arjuna@erg.abdn.ac.uk
URI: http://www.erg.abdn.ac.uk