TCPM Working Group
Internet-Draft
Intended status: Standards Track
Expires: June 26, 2012

G. Fairhurst
I. Biswas
University of Aberdeen

December 24, 2011

Updating TCP to support Variable-Rate Traffic
draft-fairhurst-tcpm-newcwv-02

Abstract

This document addresses issues that arise when TCP is used to support variable-rate traffic that exhibts 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 applications-limited interval. The method is expected to benefit variable-rate TCP applications, while also providing an appropriate response if congestion is experienced.

The document 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 recommends that the IETF should consider moving RFC 2861 from Experimental to Historic status, and that this is replaced by the current specification.

Table of Contents

1. Introduction

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 it sends is therefore limited by the maximum permitted by the receiver and congestion windows. In contrast, a variable-rate application may experience periods when the sender is either idle or is unable to send at the maximum permitted rate. This latter case is called application-limited. The focus of this document is on the operation of TCP with 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. RFC 2861 noted that this behaviour was not always observed in current implementations. Recent experiments [Bis08] confirm this to still be the case. However, standard TCP does not control growth of the cwnd when the 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. Use of such an invalid cwnd may result in reduced application performance and/or could significantly contribute to network congestion.

These issues were noted in [RFC2861], which proposed a solution, known as Congestion Window Validation (CWV). CWV was intended to help reduce the cases where TCP accumulated an invalid cwnd. The use and drawbacks of CWV 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 from a "sufficiently-long" idle period. It used the slow-start threshold (ssthresh) to save information about the previous value of the congestion window. This 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 transmits 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 have benefit, especially over long delay paths, when compared to slow-start restart specified by standard TCP. CWV offers a benefit compared to standard TCP for an application that has periods of idleness. However, CWV would only benefit the application if the idle period were less than several RTOs, 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 variable-rate applications. This mechanism does not therefore offer the desirable increase in application performance for variable rate 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 variable rate applications. In summary, CWV 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 6 minutes). 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 6 minutes, the value of cwnd is reduced, ensuring the value

then reflects the capacity that was recently actually used.

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 variable-rate, and that it can be hard to make a distinction between application-limited and idle behaviour.

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 path, 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 the update will satisfy the requirements of many variable-rate applications and at the same time provide an appropriate method for use in the Internet. The method reduces the incentive for an application to send data simply to keep transport congestion state. (This is sometimes known as padding). It 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 variable-rate, and that it can be hard to make a distinction between application-limited and idle behaviour. This update 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).

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

The method described in this document updates RFC 5681. Use of the method REQUIRES a TCP sender and the corresponding receiver to enable the TCP SACK option [RFC3517].

RFC 5681 defines a variable FlightSize, that indicates the amount of outstanding data in the network. In RFC 5681 this 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 phase that captures whether the cwnd reflects a validated or non-validated value. The phases are defined as:

- Validated phase: FlightSize >=(2/3)*cwnd. This is the normal phase, where cwnd is an approximate indication of available capacity currently available along the network path, and standard mechanisms are used [RFC5861].

- Non-validated phase: FlightSize <(2/3)*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, the transmission rate was not being constrained by the cwnd. The methods 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 induces congestion. The mechanisms are specified in the following sections.

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 (6 minutes, as explained in section 4.4) 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 SMSS), then the sender MUST exit the non-validated phase.

- 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), the sender MUST exit the non-validated phase (reducing the cwnd).

- 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 [RFC 5861]. (The resulting reduction of cwnd is appropriate, since any accumulated path history is considered unreliable).

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 the change between phases does not significantly impact an application-limited sender.

4.3.1. Adjustment at the end of the nonvalidated phase

During the non-validated phase, an application may produce bursts of data of up to the cwnd in size. This is no different to normal TCP, however, as for 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.

An application that remains in the non-validated phase for a period greater than six minutes is required to adjust its congestion control state.

At the end of the non-validated phase, the sender MUST update cwnd:

cwnd = max(FlightSize*2, IW).

Where IW is the TCP initial window [RFC5681].

(The value for cwnd was chosen to allow an application to continue to send at the currently utilised rate, and not incur delay should it increase to twice the utilised rate.)

The sender also MUST reset the ssthresh:

ssthresh = 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 MAY re-enter the non-validated phase, if required (see section 4.2).

4.3.2. Response to congestion in the nonvalidated phase

Reception of congestion feedback while in the non-validated phase, i.e., a sender that detects a packet-drop or receives an Explicit Congestion Notification (ECN), indicating 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.

When congestion is detected, the sender MUST therefore calculate a safe cwnd, based on the volume of acknowledged data:

cwnd = FlightSize - R.

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].

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

cwnd = (FlightSize/2).

4.4. Determining a safe period to preserve cwnd

Setting a limit to the period that cwnd is preserved 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 perfect choice for this time. The period of six 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 six 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 six 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.

5. Security Considerations

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

6. IANA Considerations

None.

7. Acknowledgments

The authors acknowledge the contributions of Dr A Sathiaseelan and Dr R Secchi in supporting the evaluation of CWV and for their help in developing the mechanisms proposed in this draft.

Israfil Biswas was partially supported by the School of Engineering, University of Aberdeen, Scotland, UK.

8. References

8.1. Normative References

[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.

[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.

8.2. 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, Jun. 2008.".

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

[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, Feb. 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

Israfil Biswas
University of Aberdeen
School of Engineering
Fraser Noble Building
Aberdeen, Scotland AB24 3UE
UK

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