TCPM Working Group
Internet-Draft
Obsoletes: 2861 (if approved)
Updates: 5681 (if approved)
Intended status: Standards Track
Expires: June 19, 2014

G. Fairhurst
A. Sathiaseelan
R. Secchi
University of Aberdeen
December 16, 2013

Updating TCP to support Rate-Limited Traffic
draft-ietf-tcpm-newcwv-04

Abstract

This document proposes an update to RFC 5681 to address issues that arise when TCP is used to support traffic that exhibits periods where the sending 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 rate-limited interval. This method is expected to benefit applications that send rate-limited traffic using TCP, while also providing an appropriate response if congestion is experienced.

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

NOTE: The standards status of this WG document is under review for consideration as either Experimental (EXP) or Proposed Standard (PS). This decision will be made later as the document is finalised.

This Internet-Draft will expire on June 19, 2014.

Copyright Notice

Table of Contents

1. Introduction 4
2. Reviewing experience with TCP-CWV 5
3. Terminology 5
4. An updated TCP response to idle and application-limited periods 6
4.1. A method for preserving cwnd during the idle and application-limited periods. 7
4.2. Initialisation 8
4.3. The nonvalidated phase 8
4.4. TCP congestion control during the nonvalidated phase . . . 8
4.4.1. Response to congestion in the nonvalidated phase . . . 9
4.4.2. Sender burst control during the nonvalidated phase . . 10
4.4.3. Adjustment at the end of the nonvalidated phase . . . 11
4.4.4. Examples of Implementation 11
5. Determining a safe period to preserve cwnd 13
6. Security Considerations 14
7. IANA Considerations 14
8. Acknowledgments 14
9. Author Notes 14
9.1. Other related work 14
9.2. Revision notes 17
10. References 19
10.1. Normative References 19
10.2. Informative References 19
Authors' Addresses 20

1. Introduction

TCP is used to support a range of application behaviours. The TCP congestion window (cwnd) controls the number of unacknowledged packets/bytes that a TCP flow may have in the network at any time, a value known as the FlightSize [RFC5681]. 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 advertised window and the sender congestion window (cwnd). 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 rate-limited. The focus of this document is on the operation of TCP in such an idle or rate-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 impose additional restrictions on the growth of the cwnd when a TCP sender is rate-limited. A rate-limited sender may therefore grow a cwnd far 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 the CWV algorithm in RFC 2861 with an application are discussed in Section 2.

Section 3 defines relevant terminology.

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 sending rate. The method described applies to both a rate-limited and an idle condition. Section 5 describes the rationale for selecting the safe period to preserve the cwnd.

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 a rate-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] [Fai12]) 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 the 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 RTCP Restart Window (RW) after this period.

Experience with RFC 2861 suggests that although the CWV method benefited the network in a rate-limited scenario (reducing the probability of network congestion), the behaviour was too conservative for many common rate-limited applications. This mechanism did 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, as defined in RFC2681, was often a poor solution for many rate-limited applications. It had the correct motivation, but had 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].

The following new terminology is introduced:

cwnd-limited: A TCP flow that sends the number of segments permitted by the cwnd, where the application utilises the allowed sending rate.

pipeACK sample: A meaure of the volume of data acknowledged by the network within an RTT.

pipeACK variable: A variable that measures the available capacity using the set of pipeACK samples.

pipeACK Sampling Period: The maximum period that a measured pipeACK sample may influence the pipeACK variable.

Non-validated phase: The phase where the cwnd reflects a previous measurement of the available path capacity.

Non-validated period, NVP: The maximum period for which cwnd is preserved in the non-validated phase.

Rate-limited: A TCP flow that does not consume more than one half of cwnd, and hence operates in the non-validated phase.

Validated phase: The phase where the cwnd reflects a current estimate of the available path capacity.

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

This section proposes an update to the TCP congestion control behaviour during a rate-limited period. The new method permits a TCP sender to preserve the cwnd when an application becomes idle or when the sender is unable to send at the maximum rate permitted by the cwnd (the non-validated period, NVP, see section 5). The period where actual usage is less than allowed by cwnd, is named as the non-validated phase. This 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 the NVP, the value of cwnd is reduced, ensuring the value better reflects the capacity that was recently actually used.

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 rate-limited. This is partly a response to recognition that some applications wish to transmit at a rate less than allowed by the sender cwnd, and that it can be hard to make a distinction between rate-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 during the idle and application-limited periods.

[RFC5681] defines a variable, FlightSize, that indicates the amount of outstanding data in the network. This is assumed to be 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 a new variable "pipeACK" is introduced to measure the acknowledged size of the pipe, which is used to determine if the sender has validated the cwnd.

A sender determines a pipeACK sample by measuring the volume of data that was acknowledged by the network over the period of a measured Round Trip Time (RTT). Using the variables defined in [RFC3517], a value could be measured by caching the value of HighACK and after one RTT measuring the difference between the cached HighACK value and the current HighACK value. Other equivalent methods may be used.

A sender is not required to continuously update the pipeACK variable after each received ACK, but SHOULD perform a pipeACK sample at least once per RTT when it has sent unacknowledged segments.

The pipeACK variable MAY consider multiple pipeACK samples over the pipeACK Sampling Period. The value of the pipeACK variable MUST NOT exceed the maximum (highest value) within the sampling period. This specification defines the pipeACK Sampling Period as Max(3*RTT, 1 second). This period enables a sender to compensate for large fluctuations in the sending rate, where there may be pauses in transmission, and allows the pipeACK variable to reflect the largest recently measured pipeACK sample.

When no measurements are available, the pipeACK variable is set to the "undefined value". This value is used to inhibit entering the nonvalidated phase until the first new measurement of a pipeACK sample.

The method RECOMMENDS that the TCP SACK option [RFC3517] is enabled. This allows the sender to more accurately determine the number of missing bytes during the loss recovery phase, and using this method will result in a higher cwnd following loss.

4.2. Initialisation

A sender starts a TCP connection in the Validated phase and initialises the pipeACK variable to the "undefined" value. This value inhibits use of the value in cwv calculations.

4.3. 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:

o Validated phase: pipeACK >=(1/2)*cwnd, or pipeACK is undefined. This is the normal phase, where cwnd is expected to be an approximate indication of the capacity currently available along the network path, and the standard methods are used to increase cwnd (currently [RFC5681]). The rule for transitioning to the non-validated phase is specified in section 4.4.

o Non-validated phase: pipeACK <(1/2)*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 pipeACK Sampling Period. That is, when it is not known whether the cwnd reflects the currently available capacity along the network path. The mechanisms to be used in this phase seek to determine a safe value for cwnd and an appropriate reaction to congestion. These mechanisms are specified in section 4.4.

The value 1/2 was selected to reduce the effects of variations in the pipeACK variable, and to allow the sender some flexibility in when it sends data.

4.4. TCP congestion control during the nonvalidated phase

A TCP sender MUST enter the non-validated phase when the pipeACK is less than (1/2)*cwnd.

A TCP sender that enters the non-validated phase will preserve the cwnd (i.e., this neither grows nor reduces while the sender remains in this phase). If the sender receives an indication of congestion (loss or Explicit Congestion Notification, ECN, mark [RFC3168]) it uses the method described below. The phase is concluded after a

fixed period of time (the NVP, as explained in section 4.4.2) or when the sender transmits sufficient data so that pipeACK > (1/2)*cwnd (i.e. it is no longer rate-limited).

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

- o A cwnd-limited sender uses the standard TCP method to increase cwnd (i.e. a TCP sender that fully utilises the cwnd is permitted to increase cwnd each received ACK).

- o A sender that is not cwnd-limited MUST NOT increase the cwnd when ACK packets are received in this phase.

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

- o 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 described in section 4.3.2 is appropriate, since any accumulated path history is considered unreliable).

- o A sender with a pipeACK variable greater than (1/2)*cwnd SHOULD enter the validated phase. (A rate-limited sender will not normally be impacted by whether it is in a validated or non-validated phase, since it will normally not consume the entire cwnd. However a change to the validated phase will release the sender from constraints on the growth of cwnd, and restore the use of the standard congestion response.)

4.4.1. 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 indication of an ECN marked packet, MUST record the current FlightSize in the variable LossFlightSize and MUST calculate a safe cwnd for loss recovery using the method below:

    cwnd = (Max(pipeACK,LossFlightSize))/2.

If there is a valid pipeACK value, the new cwnd is adjusted to reflect that a nonvalidated cwnd may be larger than the actual FlightSize, or recently used FlightSize (recorded in pipeACK). The updated cwnd therefore prevents overshoot by a sender significantly increasing its transmission rate during the recovery period.

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

cwnd = (Max(pipeACK,LossFlightSize) - R)/2.

Where, R is the volume of data that was retransmitted during the recovery phase. This follows the method proposed for Jump Start [Liu07]. The inclusion of the term R makes an adjustment more conservative than standard TCP. (This is required, since the sender may have sent more segments than a Standard TCP sender would have done. The additional reduction is beneficial when the LossFlightSize significantly overshoots the available path capacity incurring significant loss, for instance an intense traffic burst following a non-validated period.)

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

The sender MUST also re-initialise the pipeACK variable to the "undefined" value. This ensures that standard TCP methods are used immediately after completing loss recovery until a new pipeACK value can be determined.

ssthresh is adjusted using the standard TCP method.

4.4.2. Sender burst control during the nonvalidated phase

TCP congestion control allows a sender to accumulate a cwnd that would allow it to send a bursts of segments with a total size up to the difference between the FlightsSize and cwnd. Such bursts can impact other flows that share a network bottleneck and/or may induce congestion when buffering is limited.

Various methods have been proposed to control the sender bustiness [Hug01], [All05]. For example, TCP can limit the number of new segments it sends per received ACK . This is effective when a flow of ACKs is received, but can not be used to control a sender that has not send appreciable data in the previous RTT [All05].

This document recommends using a method to avoid line-rate bursts after an idle or rate-limited period when there is less reliable

information about the capacity of the network path: A TCP sender in the non-validated phase SHOULD control the maximum burst size, e.g. using a rate-based pacing algorithm in which a sender paces out the cwnd over its estimate of the RTT, or some other method, to prevent many segments being transmitted contiguously at line-rate. The most appropriate method(s) to implement pacing depend on the design of the TCP/IP stack, speed of interface and whether hardware support (such as TCP Segment Offload, TSO) is used. The present document does not recommend any specific method.

4.4.3. Adjustment at the end of the nonvalidated phase

An application that remains in the non-validated phase for a period greater than the NVP is required to adjust its congestion control state. If the sender exits the non-validated phase after this period, it MUST update 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 rate-limited flows that encounter occasional congestion, and could otherwise suffer an unwanted additional delay in recovering the sending rate.)

The sender MUST then update cwnd to be not greater than:

    cwnd = max(1/2*cwnd, IW).

Where IW is the appropriate TCP initial window, used by the TCP sender (e.g. [RFC5681]).

(This adjustment ensures that sender responds conservatively at the end of the non-validated phase by reducing the cwnd to better reflect the current rate of the sender. The cwnd update does not take into account FlightSize or pipeACK value because these values only reflect historical data and do not reflect the current sending rate.)

4.4.4. Examples of Implementation

This section is intended to provide informative examples of implementation methods. Implementations may choose to use other methods that comply with the normative requirements.

A pipeACK sample may be measured once each RTT. This reduces the sender processing burden for calculating after each acknowledgement and also reduces storage requirements at the sender.

Since application behaviour can be bursty using CWV, it may be desirable to implement a maximum filter to accumulate the measured values so that the pipeACK variable records the largest pipeACK sample within the pipeACK Sampling Period. One simple way to implement this is to divide the pipeACK Sampling Period into several (e.g. 5) equal length measurement periods. The sender then records the start time for each measurement period and the highest measured pipeACK sample. At the end of the measurement period, any measurement(s) that are older than the pipeACK Sampling Period are discarded. The pipeACK variable is then assigned the largest of the set of the highest measured values.

```
+----------+----------+           +----------+---......
| Sample A | Sample B | No        | Sample C | Sample D
|          |          | Sample    |          |
| |\ 5     |          |           |          |
| | |      |          |           |  /\ 4    |
| | |      |  |\ 3    |           |  | \     |
| | \      | |  \---  |           |  /  \    |   /| 2
|/   \------|       - |           | /    \------/ \...
+----------+---------\+----/ /----+/---------+-------------> Time

<------------------------------------------------|
                    Sampling Period          Current Time
```

Figure 1: Example of measuring pipeACK samples

Figure 1 shows an example of how measurement samples may be collected. At the time represented by the figure new samples are being accumulated into sample D. Three previous samples also fall within the pipeACK Sampling Period: A, B, and C. There was also a period of inactivity between samples B and C during which no measurements were taken. The current value of the pipeACK variable will be 5, the maximum across all samples.

After one further measurement period, Sample A will be discarded, since it then is older than the pipeACK Sampling Period and the pipeACK variable will be recalculated, Its value will be the larger of Sample C or the final value accumulated in Sample D.

Note that the NVP period does not necessarily require a new timer to be implemented. An alternative is to record a timestamp when the sender enters the NVP. Each time a sender transmits a new segment, this timestamp may be used to determine if the NVP period has expired. If the period expires, the sender may take into account how many units of the NVP period have passed and make one reduction (as defined in section 4.3.2) for each NVP period.

A method is required to detect the cwnd-limited condition. In simple terms this method is true only when the TCP sender's FlightSize is equal to or larger than the cwnd. However, an implementation must consider other constraints on the way in which cwnd variable is used, for instance the need to support methods such as the Nagle Algorithm and TCP Segment Offload (TSO). This can result in a sender becoming cwnd-limited when the cwnd is nearly, rather than completely, equal to the FlightSize.

5. Determining a safe period to preserve cwnd

This section documents the rationale for selecting the maximum period that cwnd may be preserved, known as the non-validated period, NVP.

Limiting the period that cwnd may be preserved avoids undesirable side effects that would result if the cwnd were to be kept unnecessarily high 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.

A period of five minutes was chosen for this NVP. This is 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 (e.g. [Bis11]) 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, Mr Ziaul Hossain 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. This work was part-funded by the European Community under its Seventh Framework Programme through the Reducing Internet Transport Latency (RITE) project (ICT-317700).

9. Author Notes

RFC-Editor note: please remove this section prior to publication.

9.1. Other related work

RFC-Editor note: please remove this section prior to publication.

There are several issues to be discussed more widely:

o There are potential interactions with the Experimental update in [RFC6928] that raises the TCP initial Window to ten segments, do these cases need to be elaborated?

This relates to the Experimental specification for increasing the TCP IW defined in RFC 6928.

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

RFC 6928 proposes an experimental update to TCP that would increase the IW to ten segments. This 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, this document proposes an 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 RFC 6928, 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 of 10 segments, this method would permit a larger restart window compared to a restart using the method in RFC 6928. This is justified by the recent path history.

3) new-CWV is attended to also be used for rate-limited applications, 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 the method in RFC 6928, and yield benefits.

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.

o There is potential performance loss in loss of a short burst (off list with M Allman)

A sender can transmit several segments then become idle. If the first segments are all ACK'ed the ssthresh collapses to a small value (no new data is sent by the idle sender). Loss of the later data results in congestion (e.g. maybe a RED drop or some other cause, rather than the maximum rate of this flow). When the sender performs loss recovery it may have an appreciable pipeACK and cwnd, but a very low FlightSize - the Standard algorithm results in an unusually low cwnd (1/2 FlightSize).

A constant rate flow would have maintained a FlightSize appropriate to pipeACK (cwnd if it is a bulk flow).

This could be fixed by adding a new state variable? It could also be argued this is a corner case (e.g. loss of only the last segments would have resulted in RTO), the impact could be significant.

o There is potential interaction with TCP Control Block Sharing(M Welzl)

An application that is non-validated can accumulate a cwnd that is larger than the actual capacity. Is this a fair value to use in TCB sharing?

We propose that TCB sharing should use the pipeACK in place of cwnd when a TCP sender is in the Nonvalidated phase. This value better reflects the capacity that the flow has utilised

in the network path.

9.2. Revision notes

RFC-Editor note: please remove this section prior to publication.

Draft 03 was submitted to ICCRG to receive comments and feedback.

Draft 04 contained the first set of clarifications after feedback:

- o Changed name to application limited and used the term rate-limited in all places.
- o Added justification and many minor changes suggested on the list.
- o Added text to tie-in with more accurate ECN marking.
- o Added ref to Hug01

Draft 05 contained various updates:

- o New text to redefine how to measure the acknowledged pipe, differentiating this from the FlightSize, and hence avoiding previous issues with infrequent large bursts of data not being validated. A key point new feature is that pipeACK only triggers leaving the NVP after the size of the pipe has been acknowledged. This removed the need for hysteresis.
- o Reduction values were changed to 1/2, following analysis of suggestions from ICCRG. This also sets the "target" cwnd as twice the used rate for non-validated case.
- o Introduced a symbolic name (NVP) to denote the 5 minute period.

Draft 06 contained various updates:

- o Required reset of pipeACK after congestion.
- o Added comment on the effect of congestion after a short burst (M. Allman).
- o Correction of minor Typos.

WG draft 00 contained various updates:

- o Updated initialisation of pipeACK to maximum value.

o Added note on intended status still to be determined.

WG draft 01 contained:

o Added corrections from Richard Scheffenegger.

o Raffaello Secchi added to the mechanism, based on implementation experience.

o Removed that the requirement for the method to use TCP SACK option [RFC3517] to be enabled - Although it may be desirable to use SACK, this is not essential to the algorithm.

o Added the notion of the sampling period to accommodate large rate variations and ensure that the method is stable. This algorithm to be validated through implementation.

WG draft 02 contained:

o Clarified language around pipeACK variable and pipeACK sample - Feedback from Aris Angelogiannopoulos.

WG draft 03 contained:

o Editorial corrections - Feedback from Anna Brunstrom.

o An adjustment to the procedure at the start and end of loss recovery to align the two equations.

o Further clarification of the "undefined" value of the pipeACK variable.

WG draft 04 contained:

o Editorial corrections.

o Introduced the "cwnd-limited" term.

o An adjustment to the procedure at the start of a cwnd-limited phase - the new text is intended to ensure that new-cwv is not unnecessarily more conservative than standard TCP when the flow is cwnd-limited. This resolves two issues: first it prevents pathologies in which pipeACK increases slowly and eraticaly. It also ensures that performance of bulk applications is not significantly impacted when using the method.

o Clearly identifies that pacing (or equivalent) is requiring during the NVP to control bustiness. New section added.

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.

[RFC6928] Chu, J., Dukkipati, N., Cheng, Y., and M. Mathis, "Increasing TCP's Initial Window", RFC 6928, April 2013.

10.2. Informative References

[All05] "Notes on burst mitigation for transport protocols", March 2005.

[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.

[Bis11] Biswas, "PhD Thesis, Internet congestion control for variable rate TCP traffic, School of Engineering,

University of Aberdeen", June 2011.

[Fai12] Sathiaseelan, Secchi, Fairhurst, and Biswas, "Enhancing TCP Performance to support Variable-Rate Traffic, 2nd Capacity Sharing Workshop, ACM CoNEXT, Nice, France, 10th December 2012.", June 2008.

[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

Raffaello Secchi
University of Aberdeen
School of Engineering
Fraser Noble Building
Aberdeen, Scotland AB24 3UE
UK

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