Congestion Control Working Group
Internet-Draft
Updates: RFC5681, RFC9002, RFC9260, RFC9438 (if approved)
Intended status: Standards Track
Expires: 11 April 2026

M. Welzl
University of Oslo
T. Henderson
University of Washington
G. Fairhurst
University of Aberdeen
M. P. Tahiliani
National Institute of Technology Karnataka

8 October 2025

Increase of the Congestion Window when the Sender Is Rate-Limited
draft-ietf-ccwg-ratelimited-increase-02

Abstract

This document specifies how transport protocols increase their congestion window when the sender is rate-limited, and updates RFC 5681, RFC 9002, RFC 9260, and RFC 9438. Such a limitation can be caused by the sending application not supplying data or by receiver flow control.

About This Document

This note is to be removed before publishing as an RFC.

The latest revision of this draft can be found at https://mwelzl.github.io/draft-ccwg-ratelimited-increase/draft-ietf-ccwg-ratelimited-increase.html. Status information for this document may be found at https://datatracker.ietf.org/doc/draft-ietf-ccwg-ratelimited-increase/.

Discussion of this document takes place on the Congestion Control Working Group Working Group mailing list (mailto:ccwg@ietf.org), which is archived at https://mailarchive.ietf.org/arch/browse/ccwg/. Subscribe at https://www.ietf.org/mailman/listinfo/ccwg/.

Source for this draft and an issue tracker can be found at https://github.com/mwelzl/draft-ccwg-ratelimited-increase.

Copyright Notice

Table of Contents

1. Introduction 3
1.1. Terminology 4
2. Conventions and Definitions 4
3. Increase rules 4
3.1. Example 5
3.1.1. Unconstrained sender 5
3.1.2. Sender constrained by the increase rules 5
3.2. Discussion 6
3.2.1. Rate-based congestion control 6
3.2.2. Pacing 6
4. Security Considerations 7
5. IANA Considerations 7
6. References 7
6.1. Normative References 7
6.2. Informative References 8
Appendix A. The state of RFCs and implementations 8
A.1. TCP ("Reno" congestion control) 8
A.1.1. Specification 8
A.1.2. Implementation 8
A.1.3. Assessment 9

A.2. CUBIC 9
A.2.1. Specification 9
A.2.2. Implementation 9
A.2.3. Assessment 9
A.3. SCTP 10
A.3.1. Specification 10
A.3.2. Assessment 10
A.4. QUIC 10
A.4.1. Specification 10
A.4.2. Assessment 10
A.5. DCCP CCID2 10
A.5.1. Specification 11
A.5.2. Assessment 11
Appendix B. Change Log 11
Acknowledgments 12
Authors’ Addresses 12

1. Introduction

A sender of a congestion controlled transport protocol becomes "rate-limited" when it does not send any data even though the congestion control rules would allow it to transmit data. This could occur because the application has not provided sufficient data to fully utilise the congestion window (cwnd). It could also occur because the receiver has limited the sender using flow control (e.g., by the advertised TCP receiver window (rwnd) or by the connection or stream flow credit in QUIC). Current RFCs specifying congestion control algorithms diverge regarding the rules for increasing the cwnd when the sender is rate-limited.

Congestion Window Validation (CWV) [RFC7661] provides an experimental specification defining how to manage a cwnd that has become larger than the current flight size. In contrast, this present document concerns the increase in cwnd when a sender is rate-limited. These two topics are distinct, but are related, because both describe the management of the cwnd when the sender does not fully utilise the current cwnd.

This document specifies a uniform rule that congestion control algorithms MUST apply and provides a recommendation that congestion control implementations SHOULD follow. An appendix provides an overview of the divergence in current RFCs and some current implementations regarding cwnd increase when the sender is rate-limited.

1.1. Terminology

This document uses the terms defined in Section 2 of [RFC5681] and Section 3 of [RFC7661]. Additionally, we define:

* maxFS: the largest value of FlightSize since the last time that cwnd was decreased. If cwnd has never been decreased, maxFS is the maximum value of FlightSize since the start of the data transfer.

2. Conventions and Definitions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

3. Increase rules

When FlightSize < cwnd, regardless of the current state of a congestion control algorithm, senders using a congestion controlled transport protocol:

1. MUST cap cwnd to be no larger than limit(maxFS).

2. MAY restrict maxFS as min(maxFS, pipeACK), using "pipeACK" as defined in [RFC7661].

In rule #1, the function limit() returns the maximum cwnd value the congestion control algorithm would yield by increasing from the value of the maxFS parameter within one RTT. The RTT includes the minimum path propagation delay plus any delay accumulated by queing in the stack, at the interface and in network elements along the path. For example, for Slow Start, as specified in [RFC5681], limit(maxFS)=2*maxFS, such that equation 2 in [RFC5681] becomes:

```
cwnd_new = cwnd + min (N, SMSS)
cwnd = min(cwnd_new, 2*maxFS)
```

where cwnd and SMSS follow their definitions in [RFC5681] and N is the number of previously unacknowledged bytes acknowledged in the incoming ACK.

Similarly, with rule #1 applied to Congestion Avoidance, limit(maxFS)=1+maxFS, such that equation 3 in [RFC5681] becomes:

```
cwnd_new = cwnd + SMSS*SMSS/cwnd
cwnd = min(cwnd_new, 1+maxFS)
```

where cwnd and SMSS follow their definitions in [RFC5681].

As with cwnd, without a way to reduce it when the transport sender becomes rate-limited, rule #1 allows for maxFS to stay valid for a long time, possibly not reflecting the reality of the end-to-end Internet path in use. For cwnd, this is remedied by "Congestion Window Validation" in [RFC7661], which also defines a "pipeACK" variable that measures the acknowledged size of the network pipe when the sender is rate-limited. Accordingly, to implement CWV, rule #2 can be used.

3.1. Example

We illustrate the working of the rules by showing the increase of cwnd in two scenarios: when the growth of cwnd is unconstrained, and when it is constrained by the increase rules. In both cases we assume initial cwnd (initcwnd) = 10 segments, as defined for TCP in [RFC6928], QUIC in [RFC9002], a single connection begins with Slow Start, the sender transmits a total of 14 segments but pauses after transmitting 10 segments and resumes the transmission for the remaining 4 segments afterwards, no packets are lost, and an ACK is sent for every packet.

3.1.1. Unconstrained sender

Initially, cwnd = initcwnd. Therefore, using initcwnd = 10 segments, as defined for TCP in [RFC6928], QUIC in [RFC9002], the sender transmits 10 segments and pauses. Since the sender is in the Slow Start phase, the arrival of an ACK for each of the 10 segments increases the cwnd by 1 segment, resulting in the cwnd increasing to 20 segments. Subsequently, after the pause, the sender transmits 4 segments and pauses again. As a consequence, the arrival of 4 ACKs results in cwnd further increasing to 24 segments even though the sender is rate-limited (i.e., has never sent more than 10 segments/ RTT).

3.1.2. Sender constrained by the increase rules

Initially, cwnd = initcwnd. Therefore, using initcwnd = 10 segments, as defined for TCP in [RFC6928], QUIC in [RFC9002], the sender transmits 10 segments and pauses; note that FlightSize and maxFS are 10 segments at this point. Since the sender is in the Slow Start phase, the arrival of an ACK for each of the 10 segments increases the cwnd by 1 segment, resulting in cwnd increasing to 20 segments. Subsequently, when the sender resumes and transmits 4 segments, rule

#1 constrains the growth of cwnd because FlightSize < cwnd and it caps cwnd to be no larger than limit(maxFS) = 2 X maxFS = 2 X 10 segments = 20 segments.

3.2. Discussion

If the sending rate is less than permitted by cwnd for multiple RTTs, limited either by the sending application or by the receiver-advertised window, continuously increasing the cwnd would cause a mismatch between the cwnd and the capacity that the path supports (i.e., over-estimating the capacity). Such unlimited growth in the cwnd is therefore disallowed.

However, in most common congestion control algorithms, in the absence of an indication of congestion, a cwnd that has been fully utilized during an RTT is permitted to be increased during the immediately following RTT. Thus, such an increase is allowed by the first rule.

3.2.1. Rate-based congestion control

The present document updates congestion control specifications that use a congestion window (cwnd) to limit the number of unacknowledged packets a sender is allowed to emit. Use of a congestion window variable to control sending rate is not the only mechanism available and used in practice.

Congestion control algorithms can also constrain data transmission by explicitly calculating the sending rate over some time interval, by "pacing" packets (injecting pauses in between their transmission) or via combinations of the above (e.g., BBR combines these three methods [I-D.cardwell-iccrg-bbr-congestion-control]). The guiding principle behind the rules in Section 3 applies to all congestion control algorithms: in the absence of a congestion indication, a sender should be allowed to increase its rate from the amount of data that it has transmitted during the previous RTT. This holds irrespective of whether the sender is rate-limited or not.

3.2.2. Pacing

Pacing mechanisms seek to avoid the negative impacts associated with "bursts" (flights of packets transmitted back-to-back). The present specification introduces a limitation using "maxFS", which is measured over an RTT; thus, as long as the number of packets per RTT is unaffected by pacing, the rules in Section 3 also do not constrain the use of pacing mechanisms.

4. Security Considerations

While congestion control designs could result in unwanted competing traffic, they do not directly result in new security considerations.

Transport protocols that provide authentication (including those using encryption), or are carried over protocols that provide authentication, can protect their congestion control algorithm from network attack. This is orthogonal to the congestion control rules.

5. IANA Considerations

This document requests no IANA action.

6. References

6.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/rfc/rfc2119>.

[RFC5681] Allman, M., Paxson, V., and E. Blanton, "TCP Congestion Control", RFC 5681, DOI 10.17487/RFC5681, September 2009, <https://www.rfc-editor.org/rfc/rfc5681>.

[RFC7661] Fairhurst, G., Sathiaseelan, A., and R. Secchi, "Updating TCP to Support Rate-Limited Traffic", RFC 7661, DOI 10.17487/RFC7661, October 2015, <https://www.rfc-editor.org/rfc/rfc7661>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/rfc/rfc8174>.

[RFC9002] Iyengar, J., Ed. and I. Swett, Ed., "QUIC Loss Detection and Congestion Control", RFC 9002, DOI 10.17487/RFC9002, May 2021, <https://www.rfc-editor.org/rfc/rfc9002>.

[RFC9260] Stewart, R., T端xen, M., and K. Nielsen, "Stream Control Transmission Protocol", RFC 9260, DOI 10.17487/RFC9260, June 2022, <https://www.rfc-editor.org/rfc/rfc9260>.

[RFC9438] Xu, L., Ha, S., Rhee, I., Goel, V., and L. Eggert, Ed., "CUBIC for Fast and Long-Distance Networks", RFC 9438, DOI 10.17487/RFC9438, August 2023, <https://www.rfc-editor.org/rfc/rfc9438>.

6.2. Informative References

[I-D.cardwell-iccrg-bbr-congestion-control]
Cardwell, N., Cheng, Y., Yeganeh, S. H., Swett, I., and V. Jacobson, "BBR Congestion Control", Work in Progress, Internet-Draft, draft-cardwell-iccrg-bbr-congestion-control-02, 7 March 2022, <https://datatracker.ietf.org/doc/html/draft-cardwell-iccrg-bbr-congestion-control-02>.

[RFC2861] Handley, M., Padhye, J., and S. Floyd, "TCP Congestion Window Validation", RFC 2861, DOI 10.17487/RFC2861, June 2000, <https://www.rfc-editor.org/rfc/rfc2861>.

[RFC4341] Floyd, S. and E. Kohler, "Profile for Datagram Congestion Control Protocol (DCCP) Congestion Control ID 2: TCP-like Congestion Control", RFC 4341, DOI 10.17487/RFC4341, March 2006, <https://www.rfc-editor.org/rfc/rfc4341>.

[RFC6928] Chu, J., Dukkipati, N., Cheng, Y., and M. Mathis, "Increasing TCP’s Initial Window", RFC 6928, DOI 10.17487/RFC6928, April 2013, <https://www.rfc-editor.org/rfc/rfc6928>.

Appendix A. The state of RFCs and implementations

This section is provided as input for IETF discussion, and should be removed before publication.

A.1. TCP ("Reno" congestion control)

A.1.1. Specification

[RFC7661] suggests there is no increase limitation in the standard TCP behavior (which [RFC7661] changes), on page 4:

> Standard TCP does not impose additional restrictions on the growth of the congestion window when a TCP sender is unable to send at the maximum rate allowed by the cwnd. In this case, the rate-limited sender may 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.

A.1.2. Implementation

* ns-2 allows cwnd to grow when it is rate-limited by rwnd. (Rate-limited by the sending application: not tested.)

* Until release 3.42, ns-3 allowed cwnd to grow when rate-limited, either due to an application or rwnd limit. Since release 3.42, ns-3 TCP models conform to rule #1 in Section 3, following the current Linux TCP approach in this regard (see next bullet).

* In Congestion Avoidance, Linux only allows the cwnd to grow when the sender is unconstrained. Before kernel version 3.16, this also applied to Slow Start. The check for "unconstrained" is perfomed by checking if FlightSize is greater or equal to cwnd. Since kernel version 3.16, which was published in August 2014, in Slow Start, the increase implements rule #1 in Section 3 in the tcp_is_cwnd_limited function in tcp.h.

A.1.3. Assessment

Linux implements a limit to cwnd growth in accordance with rule #1 in Section 3; in Slow Start, this limit follows the rule’s upper limit, while in Congestion Avoidance, it is more conservative than rule #1. The specification and the ns-2 and (older) ns-3 implementations are in conflict with rule #1 in Section 3.

A.2. CUBIC

A.2.1. Specification

Section 5.8 of [RFC9438] says:

> Cubic doesn’t increase cwnd when it’s limited by the sending application or rwnd.

A.2.2. Implementation

The description of Linux described in Appendix A.1.2 also applies to Cubic.

A.2.3. Assessment

Both the specification and the Linux implementation limit the cwnd growth in accordance with rule #1 in Section 3; in Congestion Avoidance, this limit is more conservative than rule #1 in Section 3, and in Slow Start, it implements the upper limit of rule #1 in Section 3.

A.3. SCTP

A.3.1. Specification

Section 7.2.1 of [RFC9260] says:

> When cwnd is less than or equal to ssthresh, an SCTP endpoint MUST use the slow-start algorithm to increase cwnd only if the current congestion window is being fully utilized and the data sender is not in Fast Recovery. Only when these two conditions are met can the cwnd be increased; otherwise, the cwnd MUST NOT be increased.

A.3.2. Assessment

The quoted statement from [RFC9260] prescribes the same cwnd growth limitation that is also specified for Cubic and implemented for both Reno and Cubic in Linux. It is in accordance with rule #1 in Section 3, and more conservative.

Section 7.2.1 of [RFC9260] is specifically limited to Slow Start. Congestion Avoidance is discussed in Section 7.2.2 of [RFC9260] However, this section neither contains a similar rule, nor does it refer back to the rule that limits the growth of cwnd in Section 7.2.1. It is thus implicitly clear that the quoted rule only applies to Slow Start, whereas the rules in Section 3 apply to both Slow Start and Congestion Avoidance.

A.4. QUIC

A.4.1. Specification

Section 7.8 of [RFC9002] states:

> When bytes in flight is smaller than the congestion window and sending is not pacing limited, the congestion window is underutilized. This can happen due to insufficient application data or flow control limits. When this occurs, the congestion window SHOULD NOT be increased in either slow start or congestion avoidance.

A.4.2. Assessment

With the exception of pacing, this specification conservatively limits the growth in cwnd, similar to Cubic and SCTP. It is in accordance with rule #1 in Section 3, and more conservative.

A.5. DCCP CCID2

A.5.1. Specification

Section 5.1 of [RFC4341] states: >There are currently no standards governing TCP's use of the congestion window during an application-limited period. In particular, it is possible for TCP's congestion window to grow quite large during a long uncongested period when the sender is application limited, sending at a low rate. [RFC2861] essentially suggests that TCP's congestion window not be increased during application-limited periods when the congestion window is not being fully utilized.

A.5.2. Assessment

A DCCP Congestion Control ID (CCID) specifing TCP-like behaviour ought to follow the method specified in this document. The current guidance relates only to [RFC2861]. The text in Section 5.1 of [RFC4341] is updated by this document to specify the management of the cwnd during an application-limited period.

Appendix B. Change Log

* -00 was the first individual submission for feedback by CCWG.

* -01 includes editorial improvements

  - Removes application interaction with QUIC pacing, since pacing might be within the QUIC stack.

  - Adds explicit mention of DCCP/CCID2.

  - Adds this change log.

* -02 addresses comments from IETF-119

  - Discusses rate-based controls and pacing.

  - Trims the list of possible RFCs to update.

  - Some editorial fixes: "congestion control algorithm" instead of "mechanism" for consistency with RFC5033.bis; earlier definition of maxFS; explicit mention of RFCs to update in abstract.

* -03 addresses comments from IETF-120

  - Introduces a third rule, with MAY, that avoids having an unvalidated long-lived maxFS (using pipeACK from RFC 7661).

- Changes "inc" to "limit" and adapts the wording of rule 2 to make it clearer (thanks to Neal Cardwell).
- Appendix: updates ns-3 in line with the recent implementation.
- Appendix: makes the RFC 9002 text clearer and shorter.

* draft-ietf-ccwg-ratelimited-increase-00
  - adds Mohit Tahiliani as a co-author
  - refines the "rule" text (shorter, clearer)
  - adds an example
* draft-ietf-ccwg-ratelimited-increase-01
  - Clarified what we mean with an RTT
  - rephrased example regarding initcwnd, citing RFCs 6928 and 9002
  - removed the too vague rule 1 and made rule 2 (now rule 1) a MUST
* draft-ietf-ccwg-ratelimited-increase-02
  - Improved the last sentence of section 3.1.2.
  - Removed a confusing and unnecessary sentence about pacing (as suggested at IETF-123).

Acknowledgments

The authors would like to thank Neal Cardwell and Martin Duke for suggesting improvements to this document.

Authors’ Addresses

Michael Welzl
University of Oslo
PO Box 1080 Blindern
0316 Oslo
Norway
Email: michawe@ifi.uio.no
URI: http://welzl.at/

Tom Henderson
University of Washington
185 Stevens Way
Seattle, WA 98195,
United States
Email: tomh@tomh.org
URI: https://www.tomh.org/

Godred Fairhurst
University of Aberdeen
Fraser Noble Building
Aberdeen, AB24 3UE
United Kingdom
Email: gorry@erg.abdn.ac.uk
URI: https://www.erg.abdn.ac.uk/

Mohit P. Tahiliani
National Institute of Technology Karnataka
P. O. Srinivasnagar, Surathkal
Mangalore, Karnataka - 575025
India
Email: tahiliani@nitk.edu.in
URI: https://tahiliani.in/