Workgroup: TCPM Working Group
Internet-Draft:
draft-nishida-ccwg-standard-cc-analysis-01
Published: 22 July 2023
Intended Status: Informational
Expires: 23 January 2024
Authors: Y. Nishida
AWS

Analysis for the Differences Between Standard Congestion Control Schemes

Abstract

Reno-based congestion control has been referred as the standard document from IETF for long time that describes congestion control principle of the Internet. In the meantime, IETF recently has published two new congestion control standards that use slightly different schemes from the previous one. This document provides analysis for the differences between these standards in order to provide helpful information when an unified congestion control principles for the Internet is standardized in the future.

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 https://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 23 January 2024.

Table of Contents

1. Introduction
2. Conventions and Definitions
3. Analysis for differences between RFC9002 and RFC5681
  3.1. Initial Window
  3.2. Loss Window
  3.3. SlowStart Threshold After Packet loss
  3.4. Window Growth in Slow Start Period
  3.5. Loss Recovery Algorithm
  3.6. Loss Detection Schemes
  3.7. Minimal RTO Values
4. Analysis for differences between CUBIC and RFC5681
  4.1. Multiplicative Window Decrease Factor
  4.2. Reno-friendly Model
  4.3. SlowStart Threshold After Packet loss
5. Security Considerations
6. IANA Considerations
7. References
  7.1. Normative References
  7.2. Informative References
Acknowledgments
Appendix A: Deriving increase factor for CUBIC from AIMD model
Contributors
Author's Address

1. Introduction

[RFC5681] specifies Reno-based congestion control and it has been referred as the standard document from IETF that outlines congestion control principle of the Internet. On the other hand, IETF recently have published two new congestion control standards; [RFC9002] as Reno-based congestion control for QUIC [RFC9000] and [I-D.ietf-tcpm-rfc8312bis] as CUBIC congestion control for various transport protocols. We believe all transport protocols should share the same congestion control principle so that they can share network resources mostly fairly. From this point, we believe the concepts described in these standards should not conflict each other. In our study, the new standards mostly follow the principles described in [RFC5681], however, there are certain differences in their schemes or the constant values, which may create certain performance differences.

This document provides a list of such differences as a result of our study, but does not provide any evaluations nor analysis for the performance impacts by them. Hence, some differences described in the document might be proved to be negligible in further analysis. Or, others may be considered to create distinct performance differences so that they will need to be updated to avoid conflicts between the standards. However, given that the scale of the Internet, we think such evaluations will not be easy as it would require large-scale and long-term analysis.

The aim of the document is to simply describe the differences in them and discuss their potential impacts as a reference for further analysis. We hope the document will be an useful resource when an unified congestion control principles for the Internet is needed to be standardized in the future.

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. Analysis for differences between RFC9002 and RFC5681

We think there are the following differences between RFC9002 and RFC5681.

3.1. Initial Window

RFC5681 specifies Initial Window to be at most 4 segments or 4380 bytes as specified in [RFC3390] while RFC9002 allows it to be up to 10 segments or 14720 bytes. [RFC6928] allows TCP connections to use up to 10 segments or 14600 bytes for Initial Window, however RFC5681 does not adopt it as it is an experimental document. The difference in the choice of Initial Window will have certain impacts on the growth rate of congestion window.

3.2. Loss Window

RFC5681 specifies Loss Window Size to be 1 segment while RFC9002 uses 2 segments for it. As Section 4.8 in RFC9002 describes, using 2 segments for Loss Window can reduce the chance of RTO and avoid additional delays caused by delayed ack algorithm. However, this could mean when an RFC5681 connection and an RFC9002 connection get RTOs at the same time, the RFC9002 connection can recover congestion window size more than 2 times faster than the RFC5681 connection.

3.3. SlowStart Threshold After Packet loss

In RFC9002, a sender sets the slow-start threshold to the half value of the congestion window when packet loss is detected. However, in RFC5681, it uses the half value of the flight size instead of congestion window. As there can be some situations where there is a significant difference between flightsize and congestion window, the choice of them will have considerable impacts.

Even more, RFC5681 implicitly disallows to use congestion window here as it states:

> " Implementation Note: An easy mistake to make is to simply use cwnd, rather than FlightSize, which in some implementations may incidentally increase well beyond rwnd. "

We fathom this sentence stemmed from the fact that there were some implementations which increment congestion window on every acknowledgment even though receiver's window was fully subscribed in the past. However, In RFC9002, it is prohibited to increase congestion window when it is underutilized to prevent this situation (Section 7.8). RFC9002 also allows to use other mechanisms to update its congestion window during idle periods such as [RFC7661].

3.4. Window Growth in Slow Start Period

In RFC9002, when a sender is in slow start, the congestion window increases by the number of bytes acknowledged on each acknowledgment segment arrival. On the other hand, RFC5681 increases congestion window by at most 1 full segment. RFC5681 mentions RFC3465 [RFC3465] which uses similar method to RFC9002, however RFC5681 does not recommend to use it. In addition, RFC3465 defines the limited factor: L which controls the aggressiveness of the algorithm. RFC3465 recommends to use L=2. This means it can allow to increase congestion window by at most 2 full segments. This logic in RFC5681 is overridden by [RFC9406] when hystart++ is activated in the connection. When hystart++ is enabled, as long as some packet pacing mechanisms are used in the connection, congestion window can be increased by the amount of acknowledged data in an ACK packet like RFC9002. However, the increase of congestion window per ACK is limited to 8 SMSS if there is no pacing. The performance differences in this logic will be more visible in the presence of stretch ACKs.

3.5. Loss Recovery Algorithm

The gist of the loss recovery algorithm in RFC5681 is to retransmit all lost segments found in the previous round trip and once all of them have been acknowledged, it migrates to Congestion Avoidance from Recovery period. The detailed algorithms are specified in [RFC6582] and [RFC6675].

On the other hand, RFC9002 specifies the ends of Recovery period as when one of any packets sent during the Recovery period is acknowledged. This means RFC9002 can end Recovery period even not all lost segments in the previous round trip has been successfully retransmitted. Moreover, it can end Recovery period even if some segments have been lost during Recovery period as long as one or more packets have been acknowledged.

Although we think this behavior will not lead to a congestion collapse, it looks more aggressive than RFC5681. For example, when there is a congestion where some but not all segments have been lost during several round trips, RFC5681 reduces congestion window by half every round trip (as long as retransmission schemes work successfully, otherwise it will be timed out). On the other hand, RFC9002 will repeat Recovery period and Congestion Avoidance Period in turn, which reduces congestion window by half every other round trip.

Another aspect of the loss recovery in RFC9002 is persistent congestion that is equivalent to TCP's RTO. In RFC9002, data sender establishes persistent congestion only when all sent packets are lost for a long enough duration. This period is equivalent to the duration for an RTO and two TLPs[RFC8985] in TCP. This will mean RFC9002 reduces congestion window to minimal value only when there is a extreme severe congestion. On the other hand, RFC5681 has more chances for RTOs as it gets RTOs when fast retransmission/fast recovery scheme doesn't work due to insufficient number of acknowledgments.

3.6. Loss Detection Schemes

In RFC9002, in addition to acknowledgment-based loss detection scheme which is also defined in RFC5681, it specifies another loss detection scheme similar to RACK-TLP[RFC8985]. Although RACK-TCP is a standard document, RFC5681 has no description for it.

3.7. Minimal RTO Values

RFC9002 follows most of the parts of [RFC6298] which defines the standard algorithm of retransmission timer computations and managements for TCP. However, it does not follow 1 sec for minimal RTO values in RFC6298 as it does not specify minimal RTO. This might not be a major problem because it is known that various TCP implementations already adopt lower values for minimal RTO. In addition, QUIC has more explicit mechanism to identify spurious RTOs than TCP, hence we believe there is no risk for large-scale network issues in this. However, the impacts for not having minimum RTO is still an important research topic for the performance and efficiency of transport protocols.

4. Analysis for differences between CUBIC and RFC5681

We think the following points in the CUBIC specification can cause differences behavior from RFC5681.

4.1. Multiplicative Window Decrease Factor

The CUBIC specification [I-D.ietf-tcpm-rfc8312bis] uses 0.7 for Multiplicative Window Decrease factor while RFC5681 uses 0.5. We think the rationale for using 0.5 in RFC5681 is derived from the following sentences in [Jac88], hence we presume using 0.7 instead will not be too aggressive to lead to a congestion collapse. However, it can still be more aggressive than RFC5681 which may cause unfair resource sharing.

> " We usually run our nets with ρ ≤ 0.5 so it’s probable that there are now exactly two conversations sharing the bandwidth. I.e., you should reduce your window by half because the bandwidth available to you has been reduced by half. And, if there are more than two conversations sharing the bandwidth, halving your window is conservative "

In order to compensate the aggressiveness by using the aggressive decrease factor, CUBIC uses "Reno-friendly model" which employs slower window growth rate in low BDP environments. We will discuss the validity of the model in the following section, however, even if the model is valid, CUBIC can be more aggressive than RFC5681 in some situations. For example, when there are a congestion that can last several round trips, CUBIC reduces congestion window by 30% every round trip while RFC5681 reduces it by half. In this situation, the window decrease rate for CUBIC will be mostly the half of RFC5681's. In addition, during Recovery period, CUBIC transmits data with 70% of the previous congestion window size while RFC5681 uses 50% of it.

Another example is that congestion window size can be much larger than network capacity during slow start and CUBIC's high decrease factor may have more impacts than RFC5681. For example, let's say network capacity is 100Mbps and a TCP sender's congestion window size at a certain round trip allows to transfer data at 99Mbps, which is lower than the capacity. If this sender is in slow-start, the congestion window size may glow to transfer data at 198Mbps in the next round trip and can cause many packet losses. In case of RFC5681, congestion window will be reduced by half in the following round trip and transfer rate will be 99Mbps. However, in case of CUBIC, the transfer rate will be 138.6Mbps which still exceeds the network capacity. This would mean CUBIC can saturate network for two round trips in this example while RFC5681 does only for one round trip. However, this might be a pathological example since many

recent transport stacks support pacing mechanism and [HyStart] or [I-D.ietf-tcpm-hystartplusplus] to mitigate the overshooting during slow-start.

4.2. Reno-friendly Model

CUBIC employs Reno-friendly model which is designed to be fair to RFC5681 in low BDP environments. The model in CUBIC is derived from [FHP00] and it is based on AIMD congestion control as same as RFC5681, but adopts different increase factor α and multiplicative factor β. In RFC5681, α is 1.0 and β is 0.5. This means RFC5681 increases congestion window by 1 segment per acknowledgment when a transport protocol is in congestion avoidance and reduces congestion window by half when packet losses are detected. In the meantime, in {I-D.ietf-tcpm-rfc8312bis}}, α is around 0.529 and β is 0.7. Hence, CUBIC reduces congestion window less than RFC5681 at packet loss, but at the same time, it reduces the window growth rate so that the performance of CUBIC and RFC5681 will be mostly the same.

We explain the rationale behind the values for α and β for CUBIC in Appendix "Appendix A: Deriving increase factor for CUBIC from AIMD model", but the important point for using these values is that it is based on the following two presumptions.

- *CUBIC connection(s) and Reno connection(s) are sharing the same network resource

- *When packet losses happen, both CUBIC connection and Reno connection lost packets at the same time.

Although the first presumption can be considered as a common situation, the second presumption will not be a common one as there should be various patterns in packet losses. Moreover, in this model, CUBIC increases congestion window by 0.529 segments per acknowledgment which means CUBIC transmits a segment upon the arrival of every two acknowledgments because a transport protocol usually does not send a segment until it has at least one full segment space available in congestion window. This makes harder to establish the second presumption even more.

This might be a relatively minor point that do not have significant impacts on overall performance of the model, however, more detailed analysis with realistic packet dynamics will be desirable. A recent report on this point shows that the model in CUBIC looks mostly fair to RFC5681 in low BDP environments [AIMD-friendliness].

4.3. SlowStart Threshold After Packet loss

In CUBIC, a sender sets the slow-start threshold to the half value of the FlightSize just like specified in RFC5681. However, it is not

prohibited to use congestion widow instead. In addition, CUBIC mentions employing [RFC7661] as a more sophisticated approach.

5. Security Considerations

TODO Security

6. IANA Considerations

This document has no IANA actions.

7. References

7.1. Normative References

[I-D.ietf-tcpm-rfc8312bis] Xu, L., Ha, S., Rhee, I., Goel, V., and L. Eggert, "CUBIC for Fast and Long-Distance Networks", Work in Progress, Internet-Draft, draft-ietf-tcpm-rfc8312bis-15, 31 January 2023, <https://datatracker.ietf.org/doc/html/draft-ietf-tcpm-rfc8312bis-15>.

[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>.

[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>.

7.2. Informative References

[AIMD-friendliness] Briscoe, B. and O. Albisser, "Friendliness between AIMD Algorithms", RFC Editor, please replace this URL with the permanent arXiv one , 8 August 2022, <https://raw.githubusercontent.com/bbriscoe/cubic-reno/main/creno_tr.pdf>.

[FHP00] Floyd, S., Handley, M., and J. Padhye, "A Comparison of Equation-Based and AIMD Congestion Control", May 2000, <https://www.icir.org/tfrc/aimd.pdf>.

[HyStart] Ha, S. and I. Ree, "Hybrid Slow Start for High-Bandwidth and Long-Distance Networks", DOI 10.1145/1851182.1851192, International Workshop on Protocols for Fast Long-Distance Networks , 2008.

[I-D.ietf-tcpm-hystartplusplus] Balasubramanian, P., Huang, Y., and M. Olson, "HyStart++: Modified Slow Start for TCP", Work in Progress, Internet-Draft, draft-ietf-tcpm-hystartplusplus-14, 27 February 2023, <https://datatracker.ietf.org/doc/html/draft-ietf-tcpm-hystartplusplus-14>.

[Jac88] Jacobson, V., "Congestion Avoidance and Control", Computer Communication Review, vol. 18, no. 4, pp. 314-329 , August 1988, <ftp://ftp.ee.lbl.gov/papers/congavoid.ps.Z.>.

[RFC3390] Allman, M., Floyd, S., and C. Partridge, "Increasing TCP's Initial Window", RFC 3390, DOI 10.17487/RFC3390, October 2002, <https://www.rfc-editor.org/rfc/rfc3390>.

[RFC3465] Allman, M., "TCP Congestion Control with Appropriate Byte Counting (ABC)", RFC 3465, DOI 10.17487/RFC3465, February 2003, <https://www.rfc-editor.org/rfc/rfc3465>.

[RFC6298] Paxson, V., Allman, M., Chu, J., and M. Sargent, "Computing TCP's Retransmission Timer", RFC 6298, DOI 10.17487/RFC6298, June 2011, <https://www.rfc-editor.org/rfc/rfc6298>.

[RFC6582] Henderson, T., Floyd, S., Gurtov, A., and Y. Nishida, "The NewReno Modification to TCP's Fast Recovery Algorithm", RFC 6582, DOI 10.17487/RFC6582, April 2012, <https://www.rfc-editor.org/rfc/rfc6582>.

[RFC6675] Blanton, E., Allman, M., Wang, L., Jarvinen, I., Kojo, M., and Y. Nishida, "A Conservative Loss Recovery Algorithm Based on Selective Acknowledgment (SACK) for TCP", RFC 6675, DOI 10.17487/RFC6675, August 2012, <https://www.rfc-editor.org/rfc/rfc6675>.

[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>.

[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>.

[RFC8985] Cheng, Y., Cardwell, N., Dukkipati, N., and P. Jha, "The RACK-TLP Loss Detection Algorithm for TCP", RFC 8985, DOI 10.17487/RFC8985, February 2021, <https://www.rfc-editor.org/rfc/rfc8985>.

[RFC9000] Iyengar, J., Ed. and M. Thomson, Ed., "QUIC: A UDP-Based Multiplexed and Secure Transport", RFC 9000, DOI 10.17487/RFC9000, May 2021, <https://www.rfc-editor.org/rfc/rfc9000>.

[RFC9406] Balasubramanian, P., Huang, Y., and M. Olson, "HyStart++: Modified Slow Start for TCP", RFC 9406, DOI 10.17487/RFC9406, May 2023, <https://www.rfc-editor.org/rfc/rfc9406>.

Acknowledgments

add people in tcpm-wg community.

Appendix A: Deriving increase factor for CUBIC from AIMD model

This section describes how increase factor: α used in CUBIC is determined from AIMD congestion control model. We define AIMD(α, β) as AIMD congestion control that uses an increase parameter α an a decrease parameter β. Hence, AIMD(1, 0.5) represents congestion control described in RFC5681 while CUBIC can be expressed as AIMD(α, β) where β=0.7. We also define Wmax as the congestion window size that can fully utilize network capacity.

At first, it is clear that β=0.7 is more aggressive than β=0.5 when there is no other traffic. As cwnd growth is linear, if there's enough long time for the data transfer, the average congestion window for β =0.5 will be (1.0 + 0.5)/2 = 0.75 Wmax and it will be (1.0 + 0.7)/2 = 0.85 Wmax for β=0.7. Hence, it is obvious that this model does not aim for the cases where there's no other traffic.

The choice of (α, β) for CUBIC is designed to be fair only when it competes with AIMD(1.0, 0.5). Here, we define W1 as the max window size for AIMD(1.0, 0.5) and W2 as the max window size for CUBIC's AIMD(α=X, β =0.7) when they compete each other. In this situation, AIMD(1.0, 0.5) model oscillates between 0.5 W1 and 1.0 W1 in a congestion epoch. CUBIC's AIMD (α=X, β =0.7) model oscillates between 0.7 W2 and 1.0 W2 in the same congestion epoch. When these two models have the same loss ratio, it should satisfy the following equation (1).

$$(1.0 + 0.5)\ W1 = (1.0 + 0.7)\ W2 \quad (1)$$

Also, in one congestion epoch, (α=1.0, β =0.5) increases congestion window by 0.5 W1 while (α=X, β =0.7) increases 0.3 W2. The length of congestion epoch for AIMD(1.0, 0.5) can be expressed as 0.5 W1/1.0 and it will be expressed as 0.3 W2 /X for AIMD(α=X, β =0.7). Because AIMD(1.0), AIMD(α=X, β =0.7) should have the same congestion epoch when they compete equaly, it should satisfy the following equation (2).

0.5 W1 / 1.0 = 0.3 W2 / X (2)

From equations (1) and (2), we get X=0.529.

Contributors

The contents in this documents are the individual contributions from the authors and do not relate to the authors' positions at their affiliations.

Author's Address

Yoshifumi Nishida
Amazon Web Services
440 Terry Ave N
Seattle, WA 98109
United States of America

Email: nsd.ietf@gmail.com