ccwg
Internet-Draft
Intended status: Standards Track
Expires: 7 May 2026

奥 一穂 (K. Oku)
Fastly

3 November 2025

Rapid Startup of Congestion Control
draft-kazuho-ccwg-rapid-start-00

Abstract

This document defines Rapid Start, a congestion-control startup algorithm that grows the window by 3× per RTT until queue buildup is observed, so that a sender can reach the path BDP faster than with classic 2× slow start. When congestion is observed, Rapid Start immediately scales the window relative to the bytes that have passed through the bottleneck and then hands over to normal recovery and congestion avoidance.

Discussion Venues

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

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

Source for this draft and an issue tracker can be found at https://github.com/kazuho/draft-kazuho-ccwg-rapid-start.

Status of This Memo

Table of Contents

1. Introduction

New transport connections do not know the available bandwidth or the bandwidthdelay product (BDP) of the path, so TCP and QUIC start from an initial window and use an exponential startup ("slow start" ; Section 3.1 of [RFC5681], Section 7.3.1 of [RFC9002]) to probe for the bottleneck. Classic slow start doubles the congestion window once per RTT. This is safe, but on high-RTT or high-BDP paths it can still take a considerable amount of time to reach the path BDP. It is a poor fit for short-lived connections such as HTTP, where many connections complete while still in the startup phase.

Rapid Start keeps this IW-based probing model but increases the congestion window by 3× per RTT while an RTT-sized observation window shows no queueing, so that the sender reaches the path BDP in fewer RTTs than with 2× slow start. Once queue buildup is observed in that

window, Rapid Start stops using 3× growth and reverts to 2× growth. If actual congestion is signaled (for example, by packet loss or ECN), Rapid Start does not simply apply a fixed multiplicative decrease; instead it scales the window based on the amount of data that has passed the bottleneck in that round and then hands control over to normal recovery and congestion avoidance.

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. Algorithm

This section describes the algorithm used by Rapid Start.

3.1. Rapid Start Phase

When the path appears not to be building a queue, the sender uses a more aggressive startup increase than classic slow start.

Whether the path is "not building a queue" is determined by comparing the floor RTT of the most recent round trip with the connection’s minimum RTT.

Let:

* min_rtt be the minimum RTT observed for the connection so far; and

* rtt_floor be the minimum RTT sample observed during the last round trip (i.e., within the most recent interval of length min_rtt).

If rtt_floor is no greater than min(min_rtt + 4 ms, min_rtt * 1.10), the sender increases the congestion window (cwnd) by 2 bytes for every byte that is newly acknowledged, which results in a 3× growth of cwnd per round-trip time.

If rtt_floor is greater than this threshold, the sender SHOULD increase the congestion window as classic slow start does; i.e., by 1 byte for every byte that is newly acknowledged, which results in a 2× growth of cwnd per round-trip time.

The additive term (+4 ms) and the multiplicative term (× 1.10) are RECOMMENDED defaults that provide tolerance for typical jitter while keeping Rapid Start out of the range where early queueing-detection algorithms such as HyStart++ [RFC9406] are known to trigger. Therefore, HyStart++ can be used in conjunction with Rapid Start.

3.2. Pacing Requirement

Rapid Start uses a more aggressive growth factor than classic slow start. When such growth is used, sending the initial congestion window as a short burst can make the sender observe a bottleneck overflow earlier than it would under evenly paced transmission. To ensure that Rapid Start observes the path's queueing behavior rather than sender-side burstiness, the sender SHOULD pace the packets over approximately one RTT when filling the connection's congestion window for the first time.

One way to accomplish that is to use Careful Resume [CAREFUL-RESUME], which requires that all packets sent in its Unvalidated Phase be paced based on current_rtt, regardless of previous knowledge. For connections that have no prior knowledge of the path (i.e., no previously saved CC parameters applicable to the 4-tuple), the sender SHOULD limit the initial jump window (jump_cwnd) to at most 2 * IW. With this bound, the required pacing rate (pacing_rate = jump_cwnd / min_rtt) does not exceed the pacing rate that would be used by classic slow start with pacing, so Rapid Start does not create a larger burst than existing paced startup.

3.3. Congestion Handling

When Rapid Start observes the first packet loss or an explicit congestion signal (e.g., ECN-CE), the sender enters the recovery period. The purpose of this period is (1) to drain the queue and (2) after the more aggressive startup, to bring the congestion window back in line with the actual BDP of the path.

When entering the recovery period, the sender scales the current congestion window by a silence factor. This momentarily stops transmission so that the bottleneck queue can drain by a controlled amount.

```
cwnd *= silence_factor
```

During the recovery period, whenever new data is acknowledged, the sender reduces the congestion window in proportion to the amount that has been newly acknowledged:

```
cwnd -= ack_factor * bytes_newly_acked
```

Likewise, whenever packet loss is confirmed during the recovery period, the sender reduces the congestion window in proportion to the amount of data lost:

```
cwnd -= loss_factor * bytes_newly_lost
```

This approach ensures that, by the end of the recovery period, the congestion window becomes a fraction of the full BDP (the sum of the idle BDP and the bottleneck queue size), while keeping the silence period short enough that the sender is likely to resume transmission before the bottleneck is fully drained, even if the congestion window had to be reduced significantly to compensate for the aggressive ramp-up.

The sender SHOULD NOT reduce the congestion window below

```
cwnd_before_loss * (silence_factor - 1/3 * ack_factor - 2/3 * loss_factor)
```

because, if the losses are caused purely by tail drops at the bottleneck queue, the loss ratio is unlikely to exceed the reciprocal of the most aggressive growth factor.

Separately, the sender MUST NOT reduce the congestion window below the minima specified by [RFC5681] or [RFC9002].

The sender MAY stop reducing the congestion window once it reaches the initial window multiplied by the window decrease factor. Doing so preserves classic slow start's aggressiveness on connections with tiny BDPs as the sender transitions to congestion avoidance.

3.3.1. Deriving the Reduction Factors

The reduction factors are constants derived from the multiplicative window decrease factor (beta), which is used in the congestion avoidance phase. The factors are calculated as:

```
K               = 11/18
silence_factor  = beta + K * (1 - beta)
ack_factor      = K * (1 - beta)
loss_factor     = beta + K * (1 - beta)
```

Specifically, when beta is 0.5, the values are:

```
silence_factor  = 29/36
ack_factor      = 11/36
loss_factor     = 29/36
```

When beta is 0.7 (i.e., that of CUBIC [RFC9438]), the values are:

```
silence_factor  = 53/60
ack_factor      = 11/60
loss_factor     = 53/60
```

The formula guarantees the following properties:

* When the loss ratio is 2/3, the duration of the silence period is 1 - beta relative to the full BDP, the same as during the congestion avoidance phase.

* At the end of the recovery period, the congestion window becomes as large as the full BDP multiplied by beta, the same as at the end of the recovery period during the congestion avoidance phase.

4. Limitations

To estimate the BDP during the first recovery period, Rapid Start depends on the transport protocol's accurately and promptly reporting the traversal of each sent packet, even when the packet loss ratio is high. QUIC, with its explicit packet numbers and ACK frames capable of reporting many gaps, meets this criterion. However, with TCP, there can be issues producing a reliable estimate.

5. Security Considerations

TODO Security

6. IANA Considerations

This document has no IANA actions.

7. References

7.1. Normative References

[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>.

[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>.

[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>.

[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>.

7.2. Informative References

[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>.

[CAREFUL-RESUME] Kuhn, N., Stephan, E., Fairhurst, G., Secchi, R., and C. Huitema, "Convergence of Congestion Control from Retained State", Work in Progress, Internet-Draft, draft-ietf-tsvwg-careful-resume-24, 1 October 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-tsvwg-careful-resume-24>.

[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>.

Acknowledgments

"SUSS: Improving TCP Performance by Speeding Up Slow-Start" (Mahdi Arghavani, et. al.) advocates a similar approach that increases the congestion window by 4× per round-trip, using a predictive mechanism coupled with HyStart. Compared to SUSS, Rapid Start is simpler and allows reuse of existing mechanisms and specifications such as pacing, HyStart++, and Careful Resume. Rapid Start also specifies how the congestion window should be decreased upon congestion, allowing a smooth transition to congestion avoidance.

Author’s Address

Kazuho Oku
Fastly
Email: kazuhooku@gmail.com

Additional contact information:

奥 一穂
Fastly