Stream: Internet Engineering Task Force (IETF)
RFC: 9959
Category: Standards Track
Published: April 2026
ISSN: 2070-1721
Authors:
N. Kuhn
Thales Alenia Space
E. Stephan
Orange
G. Fairhurst
University of Aberdeen
R. Secchi
University of Aberdeen
C. Huitema
Private Octopus Inc.

RFC 9959
Convergence of Congestion Control from Retained State

Abstract

This document specifies a cautious method for Internet transports that enables fast startup of congestion control for a wide range of connections, known as "Careful Resume". It reuses a set of computed congestion control parameters that are based on previously observed path characteristics between the same pair of transport endpoints. These parameters are saved, allowing them to be later used to modify the congestion control behaviour of a subsequent connection.

This document describes the assumptions and defines the requirements for how a sender utilises these parameters to provide opportunities for a connection to more rapidly get up to speed and utilise available capacity. It discusses how the use of this method impacts the capacity at a shared network bottleneck and the safe response that is needed after any indication that the new rate is inappropriate.

Status of This Memo

This is an Internet Standards Track document.

This document is a product of the Internet Engineering Task Force (IETF). It represents the consensus of the IETF community. It has received public review and has been approved for publication by the Internet Engineering Steering Group (IESG). Further information on Internet Standards is available in Section 2 of RFC 7841.

Information about the current status of this document, any errata, and how to provide feedback on it may be obtained at https://www.rfc-editor.org/info/rfc9959.

Copyright Notice

Table of Contents

1. Introduction

All Internet transports are required to either use a Congestion Control (CC) algorithm or to constrain their rate of transmission [RFC2914] [RFC8085]. In 2010, a survey of alternative CC algorithms [RFC5783] noted that there are challenges when a CC algorithm operates across an Internet path with a high and/or varying Bandwidth-Delay Product (BDP). The specified method targets a solution for these challenges.

A CC algorithm typically takes time to ramp up the sending rate. This is called the "Slow-Start Phase" and is informally known as the time to "get up to speed". This defines a time during which a sender intentionally uses less capacity than might be available, with the intention to avoid or limit overshoot of the available capacity for the path. In the context of CC, a path is associated with the end-to-end communication between a pair of transport endpoints, each identified by a source IP address and a unicast or anycast destination IP address. (This document does not define support for broadcast or multicast destination addresses.) A path can also be associated with a specific Differentiated Services Code Point (DSCP). Below the transport layer, a specific path could be realised in various ways, but this is not normally evident to the transport endpoints. (When known, additional path information could potentially provide an explicit signal to the CC algorithm to allow it to detect a change in the path.)

Any overshoot of the bottleneck rate can have a detrimental effect on other flows that share a common bottleneck. A sender can also use a method that observes the rate of acknowledged data to seek to avoid an overshoot of this bottleneck capacity (e.g., Hystart++ [RFC9406]).

In the extreme case, an overshoot can result in persistent congestion with unwanted starvation of other flows that share a common capacity bottleneck (i.e., preventing other flows from successfully sharing the capacity at a common bottleneck [RFC2914]).

A separate instance of a CC algorithm typically executes over a transport path. This seeks to avoid an increase in the queuing (latency or jitter) and/or congestion packet loss for the flow. In the case of a multipath transport, there can be more than one path with a separate CC context for each path.

This document specifies Careful Resume, a method that seeks to reduce the time to complete a transfer when the sending rate is limited by the congestion controller using the congestion window (CWND). That is, when a transfer seeks to send significantly more data than allowed by the initial congestion window (IW) and where the BDP of the path is also significantly more than the product of the IW and path Round Trip Time (RTT).

Careful Resume introduces an alternative method to select initial CC parameters that seeks to more rapidly and safely grow the sending rate controlled by the CWND.

Careful Resume is based on temporal sharing (sometimes known as "caching") of a saved set of CC parameters that relate to previous observations of the same path. The parameters are saved and used to modify the CC behaviour of a subsequent connection between the same endpoints.

CC algorithms that are rate based can make similar adjustments to their target sending rate. When saving the observed capacity, some CC algorithms might save a different parameter that is equivalent to the saved_cwnd. For example, a rate-based CC algorithm such as Bottleneck Bandwidth and Round-trip propagation time (BBR) [BBR-CC] can retain the value of the bottleneck bandwidth required to reach the capacity available to the flow (e.g., BBR.max_bw).

1.1. Use of Saved CC Parameters by a Sender

CC parameters are used by Careful Resume for three functions:

1. Information to confirm whether a saved path corresponds to the current path.
2. Information about the utilised path capacity and observed RTT to set CC parameters for the current connection.
3. Information to check the CC parameters are not too old.

CC algorithms need to be cautious when using saved CC parameters on a new path (see [RFC9000] and [RFC9040]). Care is therefore needed to assure safe use and to be robust to changes in traffic patterns, network routing, and link/node conditions. There are cases where using the saved parameters of a previous connection is not appropriate (see Section 4).

1.2. Receiver Preference

Whilst the sender could take optimisation decisions without considering the receiver's preference, there are cases where a receiver could have information that is not available at the sender or might benefit from understanding that Careful Resume might be used. In these cases, a receiver could use a transport mechanism to explicitly ask to either enable or inhibit Careful Resume when an application initiates a new connection.

Examples where a receiver might request to inhibit using Careful Resume include:

1. a receiver that can predict the pattern of traffic (e.g., insight into the volume of data to be sent, the expected length of a connection, or the requested maximum transfer rate);
2. a receiver with a local indication that a path/local interface has changed since the CC parameters were saved;
3. knowledge of the current hardware limitations at a receiver;
4. a receiver that can predict additional capacity will be needed for other concurrent or later flows (i.e., it prefers to activate Careful Resume for a different connection).

1.3. Transport Protocol Interaction

The CWND is one factor that limits the sending rate of a transport protocol. Other mechanisms also constrain the maximum sending rate. These include the sender pacing rate and the receiver-advertised window [RFC9293] or flow credit [RFC9000]; see Section 4.7.

1.4. Examples of Scenarios of Interest

This section provides a set of examples where Careful Resume is expected to improve performance. Either endpoint can assume the role of a sender or a receiver. Careful Resume can also be independently used for each direction of a bidirectional connection.

For example, consider an application that uses a series of connections over a path: Without a new method, each connection would need to individually discover appropriate CC parameters, whereas Careful Resume allows the flow to use a rate based on the previously observed CC parameters.

Another example considers an application that connects after a disruption had temporarily reduced the path capacity: When this endpoint returns to use the path using Careful Resume, the sending rate can be based on the previously observed CC parameters.

There is a particular benefit for any path with an RTT that is much larger than for typical Internet paths. In a specific example, an application connected via a geo-stationary satellite access network [IJSCN] could take 9 seconds to complete a 5.3 MB transfer using standard CC, whereas a sender using Careful Resume could reduce this transfer time to 4 seconds. The time to complete a 1 MB transfer could similarly be reduced by 62 % [MAPRG111]. This benefit is also expected for other sizes of transfer and for different path characteristics when a path has a large BDP. [CR25] provides further discussion of the method defined in this document and includes analysis over various types of paths.

1.5. Design Principles

Resuming a connection with CC parameters that were observed during a previous connection is inherently a tradeoff between the potential performance gains for the new connection and the risks of degraded performance for other connections that share a common bottleneck. The specified method is designed to obtain good performance when resuming is appropriate, while seeking to minimise the impact on other connections when it is not appropriate.

The following precautions mitigate the risk of a sender adding excessive congestion to a path:

1. The first precaution is to recognise whether the conditions have changed so much that the saved values are no longer valid. We describe that as the "Reconnaissance Phase". During that phase, the sender will not send more data than allowed for any new connection, e.g., using the recommended maximum IW for the first RTT of transmitting data [RFC6928] [RFC9002]. The sender will only proceed with the resume process if the reconnaissance succeeds. If it fails (for example, if previous packets in a connection experience congestion or the RTT is significantly different), the sender will follow the standard process for a new connection. This provides some protection against aggravating severe congestion and to establish the minimum RTT.
2. The second precaution is to cautiously use the saved parameters when resuming. This is called the "Unvalidated Phase". For example, the jump in the size of CWND/rate is restricted to a fraction (1/2) of the saved_cwnd, to avoid starving other flows that may have started or

increased their capacity after the last capacity measurement. The same principle applies for CC algorithms that use different parameters to classic TCP CC: i.e., a sender must not send faster than allowed by a fraction of the saved CC parameters. For example, a connection using a rate-based CC algorithm (e.g., BBR) will set the pacing rate to half the remembered value of the "bottleneck bandwidth". The sender also needs to pace all unvalidated packets, to ensure the rate does not exceed the previously used rate. This is intended to avoid a sudden influx of packets that could result in building a bottleneck queue and disrupting existing flows. Successful validation can allow further increases of the CWND, after first validating that the used rate did not result in congestion.

3. The third precaution is to enter a "Safe Retreat Phase" if the validation fails, for example, if congestion is detected during validation. The risk here is that the trial use of the saved CC parameters could have disrupted existing connections. For example, consider a connection using Reno CC: When exiting "slow start" mode due to loss, Reno would normally update the CWND to a "slow start threshold" set to half the volume of data in flight. However, during this validation, the CWND is restored from the saved CC parameters. The resultant sending rate could be much larger than the value that would have been reached by a "standard" slow start process, resulting in an overload of the path that potentially could cause significant congestion to other flows. Instead of continuing with that "too large" value, the retreat process resets the congestion window (rate) to a value no greater than what a standard process would have discovered. For other CC algorithms, such as Cubic [RFC9438] or BBR, the implementation details may differ, but the principle remains: Trying and failing should not unduly disadvantage existing connections that share a common bottleneck (e.g., resulting in starving these connections).

2. Language, Notation, and Terms

This section provides a brief summary of key terms and the requirements language.

2.1. Requirements Language

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.

2.2. The Remote Endpoint

The Remote Endpoint is an implementation-dependent value that identifies the sender's view of the network path being used. This is used to match the current path with a set of CC parameters associated with a previously observed path. It includes:

- an identifier representing the sending interface (e.g., a globally assigned address/prefix or other local identifier);
- an identifier representing the destination (e.g., a unicast or anycast IP address).

The Remote Endpoint could also include information such as the DSCP. If included, such information needs to be set consistently for a resumed connection to the same endpoint. Although additional information could improve the path differentiation, it could reduce the reusability of saved parameters.

The saved CC parameters can only be used to modify the startup when the Remote Endpoint is not known to have changed (see Section 3.2).

2.3. Logging Support

This document defines triggers to support logging key events. For example, [LOG] provides definitions that enable a Careful Resume implementation to generate qlog events when using QUIC.

2.4. Notation and Terms

The document uses language drawn from a range of RFCs. The following terms are defined:

ACK: The indication at the transport layer that the Remote Endpoint has correctly received the acknowledged data. In a CC algorithm, an ACK also confirms that the acknowledged data is no longer in flight.

Beta: A scaling factor between 0.5 and 1; the default value is 0.5.

Careful Resume (CR): The method specified in this document to select initial CC parameters and to more rapidly and safely increase the initial sending rate.

Congestion Control (CC) parameters: A set of saved CC parameters from observing the capacity of an established connection (see Section 1.1).

congestion window (CWND): The congestion window or equivalent CC variable limiting the maximum sending rate (see [RFC5681]).

current Round Trip Time (RTT): A sample measurement of the current RTT measured using the most recent ACK.

flight_size: The current volume of unacknowledged data (see [RFC5681]).

jump_cwnd: The resumed CWND, used in the Unvalidated Phase.

Lifetime: The configured time after which a set of saved CC parameters can no longer be safely reused.

max_jump: The configured maximum jump_cwnd.

PipeSize: A measure of the validated available capacity based on the acknowledged data.

Remote Endpoint: The endpoint corresponding to a connection; see Section 2.2.

saved_cwnd: A CC parameter with the preserved capacity derived from observation of a previous connection (see Section 4.1).

saved_remote_endpoint: The Remote Endpoint that was associated with a set of saved CC parameters.

saved_rtt: A CC parameter with the preserved minimum RTT (see Section 4.1).

unvalidated packet: A packet sent when the CWND has been increased beyond the size normally permitted by the CC algorithm; if such a packet is acknowledged by an ACK, it contributes to the PipeSize, but if congestion is detected, it triggers entry to the Safe Retreat Phase.

3. The Phases of CC Using Careful Resume

This section defines a series of phases that the congestion controller moves through as a connection uses Careful Resume. Each rule is prefixed by the name of the relevant phase.

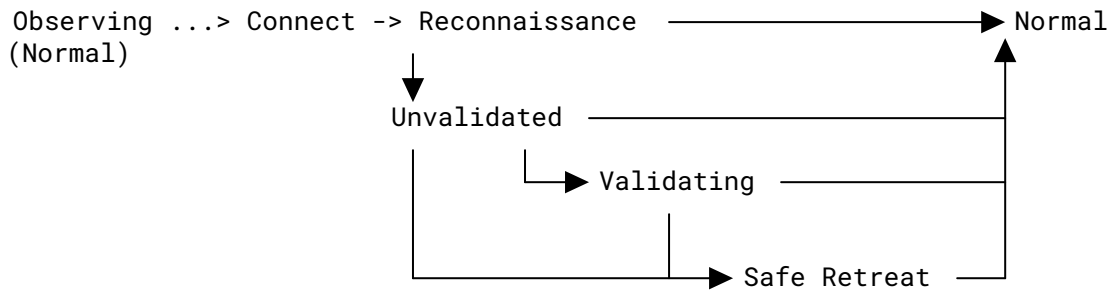

Figure 1: Key Transitions Between Phases in Careful Resume

The key phases of Careful Resume are illustrated in Figure 1. Examples of transitions between these phases are provided in Appendices A and B.

3.1. Observing

An established connection can save a set of CC parameters for the specific path to the current endpoint. A set of CC parameters includes a Lifetime (e.g., as a timestamp after which the parameters must not be used) and corresponds to one saved_remote_endpoint.

The following rules apply to observing a connection:

- Observing (saved_cwnd): The currently utilised capacity for the connection is measured as the volume of bytes sent during an RTT and is recorded in the saved_cwnd. This could be computed by measuring the volume of data acknowledged in one RTT. If the measured CWND is less than four times the IW, the sender can choose to not save the CC parameters, because the additional actions associated with performing Careful Resume for a small CWND would not justify its use.
- Observing (saved_rtt): The minimum RTT at the time of observation is saved as the saved_rrt.

- Observing (updating saved CC parameters): A sender **MUST NOT** retain more than one set of CC parameters for a Remote Endpoint, but the set of CC parameters **SHOULD** be updated (or replaced) after a later observation of a path to the same Remote Endpoint.

Implementation notes are provided in Section 4.1.

3.2. Reconnaissance Phase

During this phase, the sender attempts to retrieve CC parameters that were previously saved, then determine whether the path is consistent with a previously observed path (i.e, match the saved_remote_endpoint in a set of saved CC parameters).

The sender enters the Reconnaissance Phase after connection setup (using normal CC). In this phase, the CWND is initialised to the IW, and the sender transmits any initial data.

In the Reconnaissance Phase, the sender performs the following action:

- Reconnaissance Phase (received acknowledgment): The CWND is increased using normal CC as each ACK confirms delivery of previously unacknowledged data (i.e., the base congestion control algorithm continues to operate normally).

The sender exits the Reconnaissance Phase and stops using Careful Resume when one of the following events occurs:

- Reconnaissance Phase (detected congestion): If the sender detects congestion (e.g., packet loss or ECN-CE marking), this indicates that use of the saved CC parameters is inappropriate. The sender stops using Careful Resume and **MUST** continue using normal CC to react to the detected congestion.
- Reconnaissance Phase (using saved_cwnd): Only one connection can use a specific set of saved CC parameters. If another connection has already started to use the saved_cwnd, the sender **MUST** exit Careful Resume and return to use normal CC.
- Reconnaissance Phase (path change): If the Remote Endpoint is not the same as any saved_remote_endpoint, or the sender receives a signal from the local stack indicating that the path is now different to the observed path, the sender **MUST** stop using Careful Resume and returns to use normal CC.
- Reconnaissance Phase (lifetime of saved CC parameters): The CC parameters are temporal. If the Lifetime of the observed CC parameters is exceeded, this set of CC parameters is not used; the saved CC parameters are deleted and **MUST** stop using Careful Resume and returns to use normal CC.
- Reconnaissance Phase (minimum RTT too small): If the minimum RTT recorded in the Reconnaissance Phase is less than or equal to (saved_rtt / 2) (see Section 4.2.1), the sender **MUST** stop using Careful Resume (e.g., logged as rtt_not_validated in [LOG]) and returns to use normal CC. This is because the calculation of a sending rate from a saved_cwnd is directly impacted by the RTT; therefore, a significant change in the RTT is a strong indication that the previously observed CC parameters are not valid for the current path.

Note: When a path is not confirmed, Careful Resume does not modify the CWND before it exits to use normal CC.

The sender is permitted to enter the Unvalidated Phase as described below:

- Reconnaissance Phase (path confirmed): When the sender has received an ACK that acknowledges all the initial data (usually the IW) without reported congestion, it **MAY** then enter the Unvalidated Phase. Although a sender can immediately transition to the Unvalidated Phase, this transition **MAY** be deferred to the time at which more data is sent than would have been normally permitted by the CC algorithm.

Implementation notes are provided in Section 4.2.

3.3. Unvalidated Phase

The Unvalidated Phase is designed to enable the CWND to more rapidly get up to speed by using paced transmission of a tentatively increased CWND.

On entry to the Unvalidated Phase, the following actions are performed:

- Unvalidated Phase (initialising PipeSize): The variable PipeSize is initialised to the flight_size on entry to the Unvalidated Phase. This records the used portion of the CWND before a jump is applied.
- Unvalidated Phase (setting the jump_cwnd): To avoid starving other flows that could have either started or increased their use of capacity since observing the capacity of a path, the jump_cwnd **MUST** be no more than half of the saved_cwnd. Hence, jump_cwnd is less than or equal to Min(max_jump,(saved_cwnd/2)). CWND = jump_cwnd.

In the Unvalidated Phase, the sender performs the following actions:

- Unvalidated Phase (pacing transmission): All packets sent in the Unvalidated Phase **MUST** use pacing based on the current RTT.
- Unvalidated Phase (tracking PipeSize): The variable PipeSize is increased by the volume of data that is newly acknowledged by each received ACK. (This indicates a previously unvalidated packet has been successfully sent over the path.)

The sender exits the Unvalidated Phase and enters the Safe Retreat Phase when one of the following events occurs:

- Unvalidated Phase (confirming the path during transmission): If the sender determines that it is not valid to use the previous CC parameters due to a detected path change (e.g., a change in the RTT or an explicit signal indicating a path change), the Safe Retreat Phase **MUST** be entered (e.g., logged as path_changed in [LOG]).
- Unvalidated Phase (detected congestion): If the sender detects congestion (e.g., packet loss or ECN-CE marking), the Safe Retreat Phase **MUST** be entered (e.g., logged as either packet_loss or ECN_CE in [LOG]). (Note that insufficient time has passed in the Unvalidated Phase for a sender to receive any feedback validating the jump in the CWND. Therefore, any detected congestion must have resulted from packets sent before the Unvalidated Phase.)

The sender exits the Unvalidated Phase and evaluates whether to enter the Validating Phase when one of the following events occurs:

- Unvalidated Phase (completed sending all unvalidated packets): The sender enters the Validating Phase when the flight_size equals the CWND (e.g., logged as last_unvalidated_packet_sent in [LOG]). For an implementation that measures the CWND in bytes, this condition is also true when the remaining unused CWND is less than one maximum-sized packet.
- Unvalidated Phase (receiving acknowledgment for an unvalidated packet): The sender enters the Validating Phase when an ACK is received that acknowledges the first packet number (or higher) that was sent in the Unvalidated Phase (e.g., logged as first_unvalidated_packet_acknowledged in [LOG]).
- Unvalidated Phase (limiting time in the Unvalidated Phase): The sender enters the Validating Phase if more than one RTT has elapsed while in the Unvalidated Phase (e.g., logged as rtt_exceeded in [LOG]).

Unvalidated Phase (check flight_size): Upon any of these events (and after processing any Acknowledgments that update the PipeSize and flight_size), the sender checks if (flight_size is less than the IW) or (flight_size is less than or equal to the PipeSize) then the CWND is reset to the PipeSize (e.g., logged as rate_limited in [LOG]) and the sender stops using Careful Resume and returns to use normal CC. In the absence of detected congestion, the CWND is not reduced below the IW. (The PipeSize does not include the part of the jump_cwnd that was not utilised.) Otherwise, the CWND **MUST** be set to the flight_size and the sender progresses to the Validating Phase.

Implementation notes are provided in Section 4.3.

Notes for BBR are provided in Appendix C.1.

3.4. Validating Phase

The Validating Phase checks whether all packets sent in the Unvalidated Phase were received without inducing congestion. The CWND remains unvalidated and the sender typically remains in this phase for one RTT.

In the Validating Phase, the sender performs the following actions:

- Validating Phase (tracking PipeSize): The PipeSize is increased by the volume of acknowledged data for each received ACK that indicates a packet was successfully sent over the path.
- Validating Phase (updating the CWND): The CWND is updated using the normal rules for the current congestion controller. This typically will use "slow start", which allows the CWND to be increased for each received ACK that indicates a packet has been successfully sent across the path.

The sender exits the Validating Phase when one of the following events occurs:

- Validating Phase (detected congestion): If the sender determines that congestion was detected (e.g., packet loss or ECN-CE marking), Careful Resume enters the Safe Retreat Phase (e.g., logged as either packet_loss or ECN_CE in [LOG]).
- Validating Phase (receiving acknowledgment of the unvalidated packets): The sender stops using Careful Resume when an ACK is received that acknowledges the last packet number (or higher) that was sent in the Unvalidated Phase (e.g., logged as last_unvalidated_packet_acknowledged in [LOG]). This means that the packets sent in the Unvalidated Phase were acknowledged without congestion.

Notes for BBR are provided in Appendix C.2.

3.5. Safe Retreat Phase

This phase is entered when congestion is detected for an unvalidated packet. It drains the path of other unvalidated packets.

On entry to the Safe Retreat Phase, the following actions are performed:

- Safe Retreat Phase (removing saved information): The set of saved CC parameters for the path are deleted, to prevent these from being used again by later flows.
- Safe Retreat Phase (re-initializing the CWND): The CWND **MUST** be reduced to no more than (PipeSize/2). This avoids persistent starvation by allowing capacity for other flows to regain their share of the total capacity. (Note: The minimum CWND in QUIC is 2 packets; see [RFC9002], Section 4.8).
- Safe Retreat Phase (loss recovery): Loss recovery commences using the newly reduced CWND that was set on entry to the Safe Retreat Phase. When the CWND is reduced, a QUIC sender can immediately send a single packet prior to the reduction [RFC9002] to speed up loss recovery. A similar method for TCP is described in Section 5 of [RFC6675]. The loss recovery continues until acknowledgment of the last packet number (or a later packet) sent in the Unvalidated or Validating Phase. (Note: If the last unvalidated packet is not cumulatively acknowledged, then additional packets might also need to be retransmitted.)

In the Safe Retreat Phase, the sender performs the following actions:

- Safe Retreat Phase (tracking PipeSize): The sender continues to update the PipeSize after processing each ACK. (The PipeSize will be used to update the ssthresh when leaving this phase, but it does not affect the CWND.)
- Safe Retreat Phase (maintaining the CWND): The CWND **MUST NOT** be increased in the Safe Retreat Phase.
- Safe Retreat Phase (acknowledgment of unvalidated packets): When the last packet (or a later packet) sent during the Unvalidated Phase has been acknowledged, the sender stops using Careful Resume and returns to use normal CC.

On leaving the Safe Retreat Phase, the ssthresh **MUST** be set to no larger than the most recently measured PipeSize * Beta, where Beta is a scaling factor between 0.5 and 1. The default value is 0.5, chosen to reduce the probability of inducing a second round of congestion. Cubic defines a Beta__cubic of 0.7 [RFC9438] (e.g., logged as exit_recovery in [LOG]).

Implementation notes are provided in Section 4.5.

Notes for BBR are described in Appendix C.3.

3.6. Detecting Persistent Congestion While Using Careful Resume

A sender that experiences persistent congestion (e.g., a Retransmission Time Out (RTO) or expiry in TCP) ceases to use Careful Resume. The sender stops using Careful Resume and returns to use normal CC. If using BBR, the normal processing of packet losses will cause it to enter the Drain state while the "carefully-resuming" flag is set to True; see Appendix C.3.

As in loss recovery, data sent in the Unvalidated Phase could be later acknowledged after an RTO event.

3.7. Returning to Use Normal CC

After exiting Careful Resume, the sender returns to using the normal CC algorithm (e.g., in congestion avoidance when the CWND is more than ssthresh, or slow start when less than or equal to ssthresh).

Implementation notes are provided in Section 4.6.

4. Implementation Notes and Guidelines

This section provides guidance for implementation and use.

4.1. Observing the Path Capacity

There are various approaches to measuring the capacity used by a connection. Congestion controllers, such as such as Reno [RFC5681] or Cubic [RFC9438], could estimate the capacity based on the CWND, flight_size, acknowledged rate, etc. A different approach could estimate the same parameters for a rate-based congestion controller, such as BBR [BBR-CC], or by observing the rate at which data is acknowledged by the Remote Endpoint.

Implementations are required to calculate a saved_rtt, measuring the minimum RTT while observing the capacity. For example, this could be the minimum of a set RTT of measurements measured over the previous 5 minutes.

- There are cases where the current CWND does not reflect the path capacity. At the end of slow start, the CWND can be significantly larger than needed to fully utilise the path (i.e., a CWND overshoot). It is inappropriate to use an overshoot in the CWND as a basis for

estimating the capacity. In most cases, the CWND will converge to a stable value after several more RTTs. One mitigation when a connection is in Slow Start could be to set the saved_cwnd based on the validated pipe size (i.e., CWND / 2).

- When the sender is rate limited or in the RTT following a burst of transmission, a sender typically transmits less data than allowed by the CWND. Such observations could be discounted when estimating the saved_cwnd (e.g., when a previous observation recorded a higher value).

4.2. Confirming the Path in the Reconnaissance Phase

In the Reconnaissance Phase, the sender initiates a connection and starts sending initial data, while measuring the current RTT. The CC is not modified. A sender therefore needs to limit the initial data, sent in the first RTT of transmitted data, to no more than the IW [RFC9002]. This transmission using the IW is assumed to be a safe starting point for any path to avoid adding excessive load to a potentially congested path.

Careful Resume does not permit multiple concurrent reuse of the saved CC parameters. When multiple new concurrent connections are made to a server, each can have a valid saved_remote_endpoint, but the saved_cwnd can only be used by one connection at a time. This is to prevent the sender from performing multiple jumps in the CWND, each individually based on the same saved_cwnd, and hence creating an excessive aggregate load at the bottleneck.

The method that is used to prevent reuse of the saved CC parameters will depend upon the design of the server. For example, if a simple sender receives multiple connections from a Remote Endpoint, then the sender process could use a hash table to manage the CC parameters, whereas when using some types of load balancing, a distributed system might be needed to ensure this invariant when the load balancing hashes connections by 4-tuple and hence multiple connections from the same client device are served by different server processes; see also Section 4.2.

A sender that is rate limited [RFC7661] sends insufficient data to be able to validate transmission at a higher rate. Such a sender is allowed to remain in the Reconnaissance Phase and to not transition to the Unvalidated Phase until there is more data in the transmission buffer than would normally be permitted by the CC algorithm.

4.2.1. Confirming the Path

Path characteristics can change over time for many reasons. This can result in the previously observed CC parameters becoming irrelevant. To help confirm the path, the sender compares the saved_rrt with each current RTT sample.

If the current RTT sample is less than a half of the saved_rrt, this is regarded as too small. This is an indicator of a path change. This factor of two arises because the jump_cwnd is calculated as half the measured saved_cwnd and the sending rate ought not to exceed the observed rate when the saved_cwnd was measured.

If the current RTT is larger than the saved_rtt, this would result in a proportionally lower rate for the unvalidated packets, because the transmission is paced based on the current RTT. Hence, this rate is still safe. If the current RTT has been incorrectly measured as larger than the actual path RTT, the sender will receive an ACK for an unvalidated packet before it has completed the Unvalidated Phase. This ACK resets the CWND to reflect the flight_size, and the sender then enters the Validating Phase. The flight_size reflects the amount of outstanding data in the network rather than the maximum that is permitted by the CWND.

A current RTT that is more than ten times the saved_rrt is indicative of a path change. The value of ten accommodates both increases in latency from buffering on a path and any variation between RTT samples.

Note 1: In the Reconnaissance Phase, the sender calculates a minimum RTT over the phase and checks this on entry to the Unvalidated Phase. This avoids a need to check after each current RTT sample.

Note 2: During the Unvalidated Phase, the minimum RTT cannot increase, and hence the minimum RTT can never be larger than (saved_rtt x 10) during the Unvalidated Phase.

The sender also verifies that the initial data was acknowledged. Any loss could be indicative of persistent congestion. If a sender in the Reconnaissance Phase detects congestion, it stops using Careful Resume and returns to using normal CC. Some transport protocols implement CC mechanisms that infer potential congestion from an increase in the current RTT. Designs need to consider if such an indication is a suitable trigger to revert to stop using Careful Resume.

4.3. Safety in the Unvalidated Phase

This section considers the safety for using saved CC parameters to tentatively update the CWND. This seeks to avoid starving other flows that could have either started or increased their use of capacity since observing the capacity of a path.

To avoid inducing significant congestion to any connections that have started to use a shared bottleneck, a sender must not directly use the previous saved_cwnd to directly initialise a new flow causing it to resume sending at the same rate. The jump_cwnd is therefore limited to half the previously saved_cwnd.

4.3.1. Lifetime of CC Parameters

The long-term use of the previously observed parameters is not appropriate; a Lifetime defines the duration during which a set of saved CC parameters can be safely reused. The maximum Lifetime is a configurable parameter for a sender. An implementation also needs to provide a method to flush the set of saved CC parameters following a configuration change.

[RFC9040] provides guidance on the implementation of TCP Control Block Interdependence, but it does not specify how long a saved parameter can safely be reused. [RFC7661] specifies a method for managing an unvalidated CWND. It states:

> After a fixed period of time (the non-validated period (NVP)), the sender adjusts the CWND (Section 4.4.3). The NVP **SHOULD NOT** exceed five minutes.

Section 5 of [RFC7661] discusses the rationale for choosing that period. However, [RFC7661] targets rate-limited connections using normal CC. Careful Resume includes additional mechanisms to avoid and mitigate the effects of overshoot, and therefore a longer period can be justified when using a saved_cwnd with Careful Resume.

When the path characteristics are known to be dynamic, or the path varies, a small Lifetime is desirable (e.g., measured in minutes). For stable paths, and where the sender does not expect the path to be shared by many senders, a longer Lifetime (e.g., measured in hours) could be used. A bottleneck that is shared by a large number of senders brings greater risk that CR connections could contribute congestion that leads to prolonged overload with starvation. This can be mitigated by setting a small Lifetime.

4.3.2. Pacing in the Unvalidated Phase

A sender needs to avoid any step increase in the CWND resulting in a burst of packets that is greater than the size of the CC algorithm's IW. This is consistent with [RFC8085] and [RFC9000].

Pacing packets as a function of the current RTT, rather than the saved_rrt, provides additional safety during the Unvalidated Phase, because it avoids a smaller saved_rrt inflating the sending rate. The lower bound to the minimum acceptable current RTT avoids sending unvalidated packets at a rate that would be higher than was previously observed.

The following example provides a relevant pacing rhythm: An Inter-packet Transmission Time (ITT) is determined by using the current Maximum Packet Size (MPS), including headers, the saved_cwnd, and the current RTT. A safety margin can be configured to avoid sending more than a maximum (max_jump):

```
jump_cwnd = Min(max_jump,saved_cwnd/2)

ITT = (current RTT x MPS)/jump_cwnd
```

This follows the idea presented in [RFC4782], [INIT-SPREADING], and [CONEXT15]. Other sender mitigations have also been suggested to avoid line-rate bursts (e.g., [TCP-SSR]).

4.3.3. Exit from the Unvalidated Phase Because of Variable Network Conditions

- Careful Resume has been designed to be robust to changes in network conditions due to variations in the forwarding path (see Section 1.5), such as reconfiguration of equipment or changes in the link conditions. This is mitigated by path confirmation.
- Careful Resume has been designed to be robust to changes in network traffic, including the arrival of new flows that compete for capacity at a shared bottleneck. This is mitigated by jumping to no more than a half of the saved_cwnd and by pacing.

- Careful Resume has been designed to avoid unduly suppressing flows that have used the capacity since the capacity was observed. This is further mitigated by bounding the duration of the Unvalidated Phase and the following Validating Phase, and the conservative design of the Safe Retreat Phase.

4.4. The Validating Phase

The purpose of the Validating Phase is to trigger an entry to the Safe Retreat Phase if the capacity is not validated.

When the sender completes the Unvalidated Phase, either by sending a jump_cwnd of data or after one RTT or an acknowledgment for an unvalidated packet, it ceases to use the unvalidated CWND.

If the flight_size was less than or equal to the PipeSize, the sender resets the CWND to the PipeSize and stops using Careful Resume. Otherwise, if the CWND is larger than the flight_size, the CWND is reset to the flight_size. The sender then awaits reception of ACKs to validate the use of this capacity.

New packets are sent when previously sent data is newly acknowledged. The CWND is increased during the Validating Phase, based on received ACKs. This allows new data to be sent, but this does not have any final impact on the CWND if congestion is subsequently detected.

4.5. Safety in the Safe Retreat Phase

This section considers the safety after congestion has been detected for unvalidated packets.

The Safe Retreat Phase sets a safe CWND value to drain any unvalidated packets from the path after a packet loss has been detected or when ACKs that indicate the sent packets were marked as ECN-CE. The CC parameters that were used are invalid and are removed.

The Safe Retreat reaction differs from a traditional reaction to detected congestion, because a jump_cwnd can result in a significantly higher rate than would be allowed by Slow-Start. Such a jump could aggressively feed a congested bottleneck, resulting in overshoot where a disproportionate number of packets from existing flows are displaced from the buffer at the congested bottleneck. For this reason, a sender in the Safe Retreat Phase needs to react to detected congestion by reducing the CWND significantly below the saved_cwnd.

During loss recovery, a receiver can cumulatively acknowledge data that was previously sent in the Unvalidated Phase in addition to acknowledging the successful retransmission of data. [RFC3465] describes how to appropriately account for such ACKs. The sender tracks received ACKs that acknowledge the reception of the unvalidated packets to measure the maximum available capacity, called the "PipeSize". (The first unvalidated packet can be determined by recording the sequence number of the first packet sent in the Unvalidated Phase.) This calculated PipeSize is later used to reset the ssthresh. However, note that this is not a safe measure of the currently available share of the capacity whenever there was also a significant overshoot at the bottleneck, and it must not be used to reinitialise the CWND.

Proportional Rate Reduction (PRR) [RFC9937] assumes that it is safe to reduce the rate gradually when in congestion avoidance. PRR is therefore not appropriate when there might be significant overshoot in the use of the capacity, which can be the case when the Safe Retreat Phase is entered.

The recovery from loss depends on the design of a transport protocol. A TCP or SCTP sender is required to retransmit all lost data [RFC5681]. For some transports (e.g., QUIC and DCCP), the need for loss recovery depends on the sender policy for retransmission. On entry to the Safe Retreat Phase, the CWND can be significantly reduced. When there were multiple losses, a sender recovering all lost data could then take multiple RTTs to complete.

4.6. Returning to Normal Congestion Control

After using Careful Resume, the CC controller returns to using normal CC.

The CWND at entry to the phase will have been increased when a sender has passed through the Unvalidated Phase, unless the sender was rate limited, which causes the CWND to be reset based on the used capacity. The CWND is not reduced below the IW, unless congestion was detected. However, note that in some cases the value of the CWND could be significantly lower than the jump_cwnd (e.g., when a sender did not utilise the entire CWND in the Unvalidated Phase). The implementation details for different CC algorithms depend on the design of the algorithm.

Once a sender is no longer using Careful Resume, the sender is permitted to start observing the capacity of the path.

4.7. Limitations from Transport Protocols

The CWND is one factor that limits the sending rate of the sender. Other mechanisms can also constrain the maximum sending rate of a transport protocol. A transport protocol might need to update these mechanisms to fully utilise the CWND made available by Careful Resume:

- A TCP sender is limited by the receiver window (rwnd). Unless configured at a receiver, the rwnd constrains the rate of increase for a connection and reduces the benefit of Careful Resume.
- QUIC includes flow control mechanisms and mechanisms to prevent amplification attacks. In particular, a QUIC receiver might need to issue proactive MAX_DATA frames to increase the flow control limits of a connection that is started when using Careful Resume to gain the expected benefit.

5. Operational Considerations

This section provides some operational considerations for network providers. As noted above, using CC parameters that were observed during a previous connection is inherently a tradeoff between the potential performance gains for the new connection and the risks of degraded performance for other connections that share a common bottleneck. A transport endpoint often has no visibility of changes in the level of network traffic, nor the forwarding path over which the transport path is supported. Careful Resume is therefore a sender-side transport change that

has been designed so that any potential "harm" to other flows is constrained. It seeks to detect whether the transport path has changed since the observation of that capacity. Importantly, whenever a sender detects that assumptions about the capacity are not valid, the sender safely responds to reduce the impact on other flows (see Section 1.5).

There are three ways that the use of Careful Resume can be constrained:

- The maximum configured jump window (max_jump) (see Section 3.3),
- The Remote Endpoint identifying the client and the server that are permitted to use a specific set of saved CC parameters (see Section 2.2),
- The configured Lifetime for a set of saved CC parameters (see Section 4.3.1).

Network methods such as Equal Cost Multipath Routing, Anycast Routing, and Network Address Translation can result in changes to the forwarding path. The impact of these methods on Careful Resume can be minimised when the network is configured so that the alternative paths are provisioned to support equivalent capacity (i.e., a change to the forwarding path does not introduce a significant reduction in the capacity of the smallest bottleneck on the end-to-end path).

For many network paths, the smallest bottleneck is located in the access part of the end-to-end path. As an example, consider a typical client on an access network could connect to a remote server with a capacity bottleneck located in the access part of this path. When the client connects to a server using an anycast destination address, the anycast routing would be configured to distribute connections to a corresponding server. A client would then be unaware of whether different instances of the client's connections (with the same address pair) would terminate at the same or different servers, or at servers located at different "server farms". Hence, if a server is configured to send using Careful Resume, there is an onus to appropriately manage the use of saved CC parameters (see Section 4.2).

The way in which this is realised will depend upon the design choices in configuring the network and the servers. On the one hand, if all the servers responding to a given IP address share the same location (e.g., are in the same data center), then a method could be provided to coordinate their sharing of the CC parameters that are used to send data using Careful Resume. On the other hand, if the service configuration is such that subsquent use of the IP anycast address might result in a very different path to a server (e.g., at a different location where the path would be unable to support the same capacity), a sender should not use Careful Resume based on saved CC parameters.

6. IANA Considerations

This document has no IANA actions.

7. Security Considerations

The security considerations are the same as for other sender-based congestion control methods. Such methods rely on the receiver appropriately acknowledging receipt of data. The ability of an on-path or off-path attacker to influence congestion control depends upon the security properties of the transport protocol being used.

8. References

8.1. 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/info/rfc2119>.

[RFC2914] Floyd, S., "Congestion Control Principles", BCP 41, RFC 2914, DOI 10.17487/RFC2914, September 2000, <https://www.rfc-editor.org/info/rfc2914>.

[RFC8085] Eggert, L., Fairhurst, G., and G. Shepherd, "UDP Usage Guidelines", BCP 145, RFC 8085, DOI 10.17487/RFC8085, March 2017, <https://www.rfc-editor.org/info/rfc8085>.

[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/info/rfc8174>.

8.2. Informative References

[BBR-CC] Cardwell, N., Ed., Swett, I., Ed., and J. Beshay, Ed., "BBR Congestion Control", Work in Progress, Internet-Draft, draft-ietf-ccwg-bbr-04, 20 October 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-ccwg-bbr-04>.

[CONEXT15] Li, Q., Dong, M., and P. B. Godfrey, "Halfback: Running Short Flows Quickly and Safely", CoNEXT '15: Proceedings of the 11th ACM Conference on Emerging Networking Experiments and Technologies, DOI 10.1145/2716281.2836107, 2015, <https://doi.org/10.1145/2716281.2836107>.

[CR25] Yanev, M., Custura, A., Secchi, R., and G. Fairhurst, "Analysis of Careful Resumption of Internet Congestion Control from Retained Path State", Computer Networks, vol. 276, DOI 10.1016/j.comnet.2025.111950, 2026, <https://www.sciencedirect.com/science/article/abs/pii/S1389128625009156>.

[IJSCN] Thomas, L., Dubois, E., Kuhn, N., and E. Lochin, "Google QUIC performance over a public SATCOM access", International Journal of Satellite Communications and Networking, vol. 37, no. 6, pp. 601-611, DOI 10.1002/sat.1301, 2019, <https://doi.org/10.1002/sat.1301>.

[INIT-SPREADING] Sallantin, R., Baudoin, C., Arnal, F., Dubois, E., Chaput, E., and A. Beylot, "Safe increase of the TCP's Initial Window Using Initial Spreading", Work in Progress, Internet-Draft, draft-irtf-iccrg-sallantin-initial-spreading-00, 15 January 2014, <https://datatracker.ietf.org/doc/html/draft-irtf-iccrg-sallantin-initial-spreading-00>.

[LOG] Custura, A. and G. Fairhurst, "Quic Logging for Convergence of Congestion Control from Retained State", Work in Progress, Internet-Draft, draft-ietf-tsvwg-careful-resume-qlog-02, 2 February 2026, <https://datatracker.ietf.org/doc/html/draft-ietf-tsvwg-careful-resume-qlog-02>.

[MAPRG111] Kuhn, N., Stephan, E., Fairhurst, G., Jones, T., and C. Huitema, "Feedback from using QUIC's 0-RTT-BDP extension over SATCOM public access", IETF 111 Proceedings, July 2021, <https://www.ietf.org/proceedings/111/slides/slides-111-maprg-feedback-from-using-quics-0-rtt-bdp-extension-over-satcom-public-access-00.pdf>.

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

[RFC4782] Floyd, S., Allman, M., Jain, A., and P. Sarolahti, "Quick-Start for TCP and IP", RFC 4782, DOI 10.17487/RFC4782, January 2007, <https://www.rfc-editor.org/info/rfc4782>.

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

[RFC5783] Welzl, M. and W. Eddy, "Congestion Control in the RFC Series", RFC 5783, DOI 10.17487/RFC5783, February 2010, <https://www.rfc-editor.org/info/rfc5783>.

[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/info/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/info/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/info/rfc7661>.

[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/info/rfc9000>.

[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/info/rfc9002>.

[RFC9040] Touch, J., Welzl, M., and S. Islam, "TCP Control Block Interdependence", RFC 9040, DOI 10.17487/RFC9040, July 2021, <https://www.rfc-editor.org/info/rfc9040>.

[RFC9293] Eddy, W., Ed., "Transmission Control Protocol (TCP)", STD 7, RFC 9293, DOI 10.17487/RFC9293, August 2022, <https://www.rfc-editor.org/info/rfc9293>.

[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/info/rfc9406>.

[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/info/rfc9438>.

[RFC9937] Mathis, M., Cardwell, N., Cheng, Y., and N. Dukkipati, "Proportional Rate Reduction (PRR)", RFC 9937, DOI 10.17487/RFC9937, December 2025, <https://www.rfc-editor.org/info/rfc9937>.

[TCP-SSR] Hughes, A., Touch, J., and J. Heidemann, "Issues in TCP Slow-Start Restart After Idle", Work in Progress, Internet-Draft, draft-hughes-restart-00, December 2001, <https://datatracker.ietf.org/doc/html/draft-hughes-restart-00>.

Appendix A. Notes on the Careful Resume Phases

The table below is provided to illustrate the operation of Careful Resume. This table is informative; please refer to the body of the document for the normative specification. The description is based on a normal CC that uses Reno. The PipeSize tracks the validated CWND.

The table uses the following abbreviations:

SS = Slow-Start
FS = flight_size
PS = PipeSize
ACK = highest acknowledged packet

Phase	Normal	Recon.	Unvalidated	Validating	Safe Ret.
	Observing CC params	Confirm path	Send faster using saved_cwnd	Validate new CWND; Update PS	Drain path; Update PS

Phase	Normal	Recon.	Unvalidated	Validating	Safe Ret.
On entry:	-	CWND=IW	PS=FS jump_cwnd =saved_cwnd /2; CWND =jump_cwnd	If (FS>PS) {CWND=FS} else {CWND=PS; use Normal CC}	CWND=(PS/2)
CWND:	When in observe, measure saved _cwnd	CWND increases using SS	CWND is not increased	CWND can increase using normal CC	CWND is not increased
PS:	-	-	PS+=ACked		
RTT:	Measure saved_rtt	Measure current RTT	-	-	-
If loss or ECNCE:	Normal CC	Normal CC; CR is not allowed	Enter Safe Retreat		-
Next Phase:	Observing (as needed)	If (CWND, Lifetime, and RTT confirmed), enter Unvalidated; else CWND=Max (PS,IW); and use Normal CC	If (FS>CWND or >1 RTT has passed or ACK >= first unvalidated packet), If ((FS>CWND) or (FS<=PS)) cwnd=PS; and use Normal CC else cwnd=FS; enter Validating	If (ACK >= last unvalidated packet), use Normal CC	If (ACK >= last unvalidated packet), ssthresh = PS x Beta; and use Normal CC

Table 1: Illustration of the Operation of Careful Resume

Appendix B. Examples of the Careful Resume Phases

The following examples consider an implementation that keeps track of transmitted data in terms of packets and provide informative examples of use.

In the Unvalidated Phase, the first unvalidated packet corresponds to the highest sent packet recorded on entry to this phase. In the Validating Phase and Safe Retreat Phase, the sender tracks the last unvalidated packet (this is also the highest sent packet number recorded on entry to this

phase). The PipeSize (PS) tracks the validated portion of the CWND. The PS is set to the CWND on entry to the Unvalidated Phase. It is updated after receiving an ACK for each additional packet. The default value of Beta is 0.5.

Note: To simplify the description, these examples are described using packet numbers (whereas QLOG variables are expressed in bytes).

B.1. Example with No Loss

In the first example of using Careful Resume, the sender starts by sending IW packets, assumed to be 10 packets, in the Reconnaissance Phase, and then continues in a subsequent RTT to send more packets until the sender becomes CWND limited (i.e., flight_size = CWND).

The sender in the Reconnaissance Phase then confirms the RTT and other conditions for using Careful Resume. In this example, this is confirmed when the sender has 29 packets in flight.

The sender then enters the Unvalidated Phase. (This path confirmation could have happened earlier if data had been available to send.) The sender initialises the PipeSize to the flight_size (in this case, 29 packets) and then sets the CWND to 150 packets (based upon half of the previously observed saved_cwnd of 300 packets).

The sender now sends 121 unvalidated packets (the unused portion of the current CWND). Each time a packet is sent, the sender checks whether 1 RTT has passed since entering the Unvalidated Phase (otherwise, the Validating Phase is entered). This check triggers only for cases where the sender is rate limited, as shown in the following example: The PipeSize increases after each ACK is received.

When the first unvalidated packet is acknowledged (packet number 30), the sender enters the Validating Phase. (This transition would also occur if the flight_size increased to equal the CWND.) During this phase, the CWND can be increased for each ACK that acknowledges an unvalidated packet, because this indicates that the packet was validated.

When an ACK is received that acknowledges the last packet that was sent in the Unvalidated Phase, the sender stops using Careful Resume. For example, if the CWND is less than ssthresh, a Reno or Cubic sender using normal CC is permitted to use Slow-Start to grow the CWND towards the ssthresh and will then enter congestion avoidance.

B.2. Example with No Loss, Rate Limited

A rate-limited sender will not fully utilise the available CWND when using Careful Resume, and the CWND is therefore reset on entry to the Validating Phase, as described below.

The sender starts by sending up to IW packets (10) in the Reconnaissance Phase. It commences as described in the first example, transitioning to the Unvalidated Phase, where the CWND is set to 150 packets, and the PipeSize is set to the flight_size (i.e., 29 packets).

The sender then becomes rate limited, because the example only sends 50 unvalidated packets.

After about one RTT (e.g., by comparing the current time with local timestamps for each sent packet or by receiving an ACK for the first unvalidated packet), the sender will still not have fully used the CWND. It then enters the Validating Phase and resets the CWND to the current flight_size (i.e., 50 packets). During this phase, the CWND can be increased for each received ACK that validates reception of an unvalidated packet. The PipeSize also increases with each ACK received, to reflect the discovered capacity.

The sender completes using Careful Resume when a received ACK acknowledges the last packet that was sent in the Unvalidated Phase. It then stops using Careful Resume, as in the example with no loss.

B.3. Example with Loss Detected in the Reconnaissance Phase

When a sender detects that a packet was lost in the Reconnaissance Phase, it will stop using Careful Resume and recover the loss using the normal loss recovery algorithm and normal CC. It is considered that the sender may have discovered a capacity limit and it is not allowed to continue to use Careful Resume. In this case, there is no change to the CC algorithm and the CWND is the same as if Careful Resume had not been attempted.

B.4. Example with Loss Detected in the Validating Phase

As in the first example, the sender enters the Unvalidated Phase with a CWND of 150 packets and with the PipeSize initialised to the flight_size (i.e., 29 packets).

The sender now sends 121 unvalidated packets (consuming the remaining unused CWND). This example considers the case when one of the unvalidated packets is lost. We assume in the example that the lost packet is 64 (the 35th packet sent in the Unvalidated Phase).

The received ACKs acknowledge the reception of the first 34 unvalidated packets. The PipeSize at this time is equal to 63 (29 + 34) packets.

A loss is then detected (by a timer or by receiving three ACKs that do not acknowledge packet number 35). The sender then enters the Safe Retreat Phase because the CWND was not validated. Assuming that the IW was 10 packets, the CWND is reset to Max(10,PS/2) = Max(10,63/2) = 31 packets. This CWND is used during the Safe Retreat Phase, because congestion was detected and the sender still does not yet know if the remaining unvalidated packets will be successfully acknowledged. This conservative CWND calculation ensures the sender drains the path after this potentially severe congestion event. There is no further increase in the CWND in this phase.

The sender continues to receive ACKs that acknowledge the remaining 86 (121-35) unvalidated packets. Recall that the 35th unvalidated packet was lost and had packet number 64 (29+35). The PipeSize tracks the capacity discovered by ACKs that acknowledge the unvalidated packets (i.e., the PipeSize is increased for each received ACK that acknowledges new data). Although this PipeSize cannot be used to safety initialise the CWND (because it was measured when the sender had aggressively created overload), the estimated PipeSize (which, in this case, is 121-1 = 120 packets) can be used to set the ssthresh on exit from Safe Retreat, since it does indicate a measured upper limit to the current capacity.

At the point where all the unvalidated packets that were sent in the Unvalidated Phase have been either acknowledged or have been declared lost, the sender updates the ssthresh to be no larger than the recently measured PipeSize multiplied by Beta (the final action of the Safe Retreat Phase), and the sender stops using Careful Resume. Because the CWND will now be less than ssthresh, a sender using normal CC is permitted to use Slow-Start to grow the CWND towards the ssthresh, after which it will enter congestion avoidance.

Appendix C. Implementation Notes for Using BBR

Bottleneck Bandwidth and Round-trip propagation time (BBR) uses recent measurements of a transport connection's delivery rate, Round Trip Time (RTT), and packet loss rate to build an explicit model of the network path. BBR then uses this model to control both how fast it sends data and the maximum volume of data it allows in flight in the network at any time [BBR-CC].

When the flow is controlled using BBR Appendix C, Careful Resume is implemented by setting the pacing rate from the saved CC parameters, with the following precautions:

- The flag "carefully-resuming" is added to the BBR state to indicate that the sender is allowed to send unvalidated packets. This is initialised to False when a BBR flow starts.
- Prerequisites for using Careful Resume are described in Section 3.2.

C.1. Sending Unvalidated Packets Using BBR

Careful Resume is allowed to transmit unvalidated packets only when the BBR flow is in the Startup state.

The probing rate is configured to 1/2 of the bottleneck bandwidth, derived from the CWND calculation specified in the saved CC parameters according to the requirements in Section 3.3.

The sender starts the Unvalidated Phase at the beginning of a BBR round, and sets the "carefully-resuming" flags to True. When this "carefully-resuming" flag is set, the BBR congestion controller sets the BBR pacing rate to the larger of the nominal pacing rate (BBR.bw multiplied bytes BBRStartupPacingGain) or the calculated probing rate. Then, the CWND is set to the larger of BBR.bw and the probing rate, multiplied by BBR.rtt_min times BBRStartupCwndGain.

The "carefully-resuming" flag is reset to False two rounds after it is set (i.e., after all the packets sent in the first round of "carefully resuming" have been received and acknowledged by the peer). At that stage (after the capacity has been validated), the measured delivery rate is expected to reflect the probing rate.

If congestion is detected while the "carefully-resuming" flag is True, BBR exits the Startup state and enters the Drain state (implementing the Safe Retreat Phase).

C.2. Validation for BBR

When using BBR, the Validation Phase is realised using the BBR rules for exiting Startup. Upon exiting Startup, the connection estimates that the measured delivery rate will reflect the flow's share of the actual bottleneck bandwidth. If congestion is detected while using Careful Resume (i.e, the "carefully-resuming" flag is True), BBR then exits the Startup state and enters the Drain state.

C.3. Safe Retreat for BBR

When using BBR, the Safe Retreat Phase is entered if the Drain state is entered while the "carefully-resuming" flag (see Appendix C) is still True (i.e., if less than 2 full rounds have elapsed after the sender entered the Unvalidated Phase). The delivery rates measured in these conditions are tainted, because packets sent during the attempt are still queued at the bottleneck buffer and could have "pushed out" packets belonging to any competing flows. Therefore, any delivery rates measured in the Drain state **MUST** be discarded if the "carefully-resuming" flag is set to True. This flag is cleared upon exiting the Drain state.

Acknowledgments

The authors would like to thank John Border, Gabriel Montenegro, Patrick McManus, Ian Swett, Igor Lubashev, Robin Marx, Roland Bless, Franklin Simo, Kazuho Oku, Tong, Ana Custura, Neal Cardwell, Marten Seemann, Matthias Hofstaetter, Nicolai Fischer, Yi Huang, Mihail Yanev, and Joerg Deutschmann for their fruitful comments in developing this specification. They also thank Mike Bishop for his careful suggestions on the structure to describe the phases. Thanks also to Mohamed Boucadair and to Dan Harkins for his secdir review.

The authors would like to thank Tom Jones for co-authoring previous draft versions of this document.

Authors' Addresses

Nicolas Kuhn
Thales Alenia Space
Email: nicolas.kuhn.ietf@gmail.com

Emile Stephan
Orange
Email: emile.stephan@orange.com

Godred Fairhurst
University of Aberdeen
School of Engineering
Fraser Noble Building
Aberdeen
AB24 3UE
United Kingdom
Email: gorry@erg.abdn.ac.uk

Raffaello Secchi
University of Aberdeen
School of Engineering
Fraser Noble Building
Aberdeen
AB24 3UE
United Kingdom
Email: r.secchi@abdn.ac.uk

Christian Huitema
Private Octopus Inc.
Email: huitema@huitema.net