

Network Working Group  
Internet-Draft  
Intended status: Experimental  
Expires: 22 April 2026

C. Huitema  
Private Octopus Inc.  
S. Nandakumar  
C. Jennings  
Cisco  
19 October 2025

Specification of Christian's Congestion Control Code (C4)  
draft-huitema-ccwg-c4-spec-00

## Abstract

Christian's Congestion Control Code is a new congestion control algorithm designed to support Real-Time applications such as Media over QUIC. It is designed to drive towards low delays, with good support for the "application limited" behavior frequently found when using variable rate encoding, and with fast reaction to congestion to avoid the "priority inversion" happening when congestion control overestimates the available capacity. The design emphasizes simplicity and avoids making too many assumptions about the "model" of the network.

## 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 22 April 2026.

## Copyright Notice

Copyright (c) 2025 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document.

Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Revised BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Revised BSD License.

## Table of Contents

1. Introduction . . . . .	2
2. Key Words . . . . .	3
3. C4 variables . . . . .	3
3.1. Nominal rate . . . . .	3
3.2. Nominal max RTT . . . . .	4
3.3. Global variables . . . . .	5
3.4. Per era variables . . . . .	6
4. States and Transition . . . . .	6
4.1. Setting pacing rate, congestion window and quantum . . . . .	7
4.2. Initial state . . . . .	8
4.3. Recovery state . . . . .	9
4.3.1. Restarting Initial if High Jitter . . . . .	10
4.4. Cruising state {#c4-cruising } . . . . .	10
4.5. Pushing state . . . . .	10
5. Handling of congestion signals . . . . .	10
5.1. Variable Sensitivity . . . . .	11
5.2. Detecting Excessive Delays . . . . .	11
5.3. Detecting Excessive Losses . . . . .	12
5.3.1. Do not react to Probe Time Out . . . . .	12
5.4. Detecting Excessive CE Marks . . . . .	12
5.5. Rate Reduction on Congestion . . . . .	12
6. Security Considerations . . . . .	13
7. IANA Considerations . . . . .	13
8. References . . . . .	13
8.1. Normative References . . . . .	13
8.2. Informative References . . . . .	13
Acknowledgments . . . . .	14
Authors' Addresses . . . . .	14

## 1. Introduction

Christian's Congestion Control Code (C4) is a congestion control algorithm designed to support Real-Time multimedia applications, specifically multimedia applications using QUIC [RFC9000] and the Media over QUIC transport [I-D.ietf-moq-transport].

The two main variables describing the state of a flow are the "nominal rate" (see Section 3.1) and the "nominal max RTT" (see Section 3.2). C4 organizes the management of the flow through a series of states: Initial, during which the first assessment of

nominal-rate and nominal max RTT are obtained, Recovery in which a flow is stabilized after the Initial or Pushing phase, Cruising during which a flow uses the nominal rate, and Pushing during which the flow tries to discover if more resource is available -- see Section 4.

C4 divides the duration of the connection in a set of "eras", each corresponding to a packet round trip. Transitions between protocol states typically happen at the end of an era, except if the transition is forced by a congestion event.

C4 assumes that the transport stack is capable of signaling events such as acknowledgements, RTT measurements, ECN signals or the detection of packet losses. It also assumes that the congestion algorithm controls the transport stack by setting the congestion window (CWND) and the pacing rate (see Section 5).

C4 introduces the concept of "sensitivity" (see Section 5.1) to ensure that flows using a large amount of bandwidth are more "sensitive" to congestion signals than flows using fewer bandwidth, and thus that multiple flows sharing a common bottleneck are driven to share the resource evenly.

## 2. Key Words

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. C4 variables

In addition to the nominal rate and the nominal max RTT, C4 maintains a set of variables per flow (see Section 3.3) and per era (see Section 3.4).

### 3.1. Nominal rate

The nominal rate is an estimate of the bandwidth available to the flow. On initialization, the nominal rate is set to zero, and default values are used when setting the pacing rate and CWND for the flow.

C4 evaluates the nominal rate after acknowledgements are received using the number of bytes acknowledged since the packet was sent (`bytes_acknowledged`) and the time delay it took to process these packets.

That delay is normally set to the difference between the time at which the acknowledged packet was sent (`time_sent`), and the current time (`current_time`). However, that difference may sometimes be severely underestimated because of delay jitter and ACK compression. We also compute a "send delay" as the difference between the send time of the acknowledged packet and the send time of the oldest "delivered" packet.

```
delay_estimate = max (current_time - time_sent, send_delay)
rate_estimate = bytes_acknowledged /delay_estimate
```

If we are not in a congestion situation, we update the nominal rate:

```
if not congested and nominal_rate > rate_estimate:
    nominal_rate = rate_estimate
```

The data rate measurements can only cause increases in the nominal rate. The nominal rate is reduced following congestion events, as specified in Section 5.

The "congested" condition is defined as being in the recovery state and having either entered that state due to a congestion event, or having received a congestion event after entering recovery.

Updating the nominal rate in these conditions would cause a congestion bounce: the nominal rate is reduced because of a congestion event, C4 enters recovery, but then packets sent at the previous rate are received during recovery, generating a new estimate and resetting the nominal rate to a value close to the one that caused congestion.

### 3.2. Nominal max RTT

The nominal max RTT is an estimate of the maximum RTT that can occur on the path in the absence of queues. The RTT samples observed for the flow are the sum of four components:

- \* the latency of the path
- \* the jitter introduced by processes like link layer contention or link layer retransmission
- \* queuing delays caused by competing applications
- \* queuing delays introduced by C4 itself.

C4's goal is to obtain a estimate of the combination of path latency and maximum jitter. This is done by only taking measurements when C4 is sending data at a rate not higher than the nominal transmission rate, as happens for example in the recovery and cruising states. These measurements will happen during the following era. C4 captures them by recording the max RTT for packets sent in that era. C4 will also progressively reduce the value of the nominal max RTT over time, to account for changes in network conditions.

# on end of era

```
if alpha_previous <= 1.0:
    if era_min_rtt < running_min_rtt:
        running_min_rtt = era_min_rtt
    else:
        running_min_rtt =
            (7*running_min_rtt + era_min_rtt)/8

    if era_max_rtt > running_min_rtt + MAX_JITTER:
        # cap RTT increases to MAX_JITTER, i.e., 250ms
        era_max_rtt = running_min_rtt + MAX_JITTER
    if era_max_rtt > nominal_max_rtt:
        nominal_max_rtt = era_max_rtt
    else:
        nominal_max_rtt =
            (7*nominal_max_rtt + era_max_rtt)/8
```

The decrease over time is tuned so that jitter events will be remembered for several of the cruising-pushing-recovery cycles, which is enough time for the next jitter event to happen, at least on Wi-Fi networks.

### 3.3. Global variables

In addition to the nominal rate and nominal MAX RTT, C4 maintains a set of variables tracking the evolution of the flow:

- \* running min RTT, an approximation of the min RTT for the flow,
- \* number of eras without increase (see Section 4.2),
- \* number of successful pushes,
- \* current state of the algorithm, which can be Initial, Recovery, Cruising or Pushing.

### 3.4. Per era variables

C4 keeps variables per era:

```
era_sequence; /* sequence number of first packet sent in this era */
alpha_current; /* coefficient alpha used in the current state */
alpha_previous; /* coefficient alpha used in the previous era */
era_max_rtt; /* max RTT observed during this era */
era_min_rtt; /* min RTT observed during this era */
```

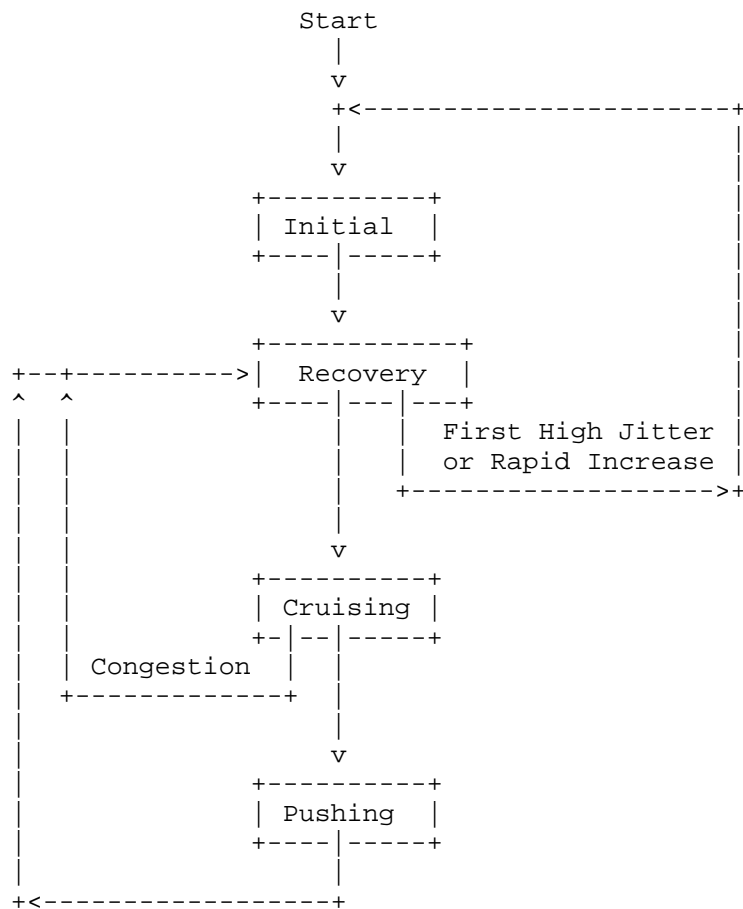
These variables are initialized at the beginning of the era.

### 4. States and Transition

The state machine for C4 has the following states:

- \* "Initial": the initial state, during which the CWND is set to twice the "nominal\_CWND". The connection exits startup if the "nominal\_cwnd" does not increase for 3 consecutive round trips. When the connection exits startup, it enters "recovery".
- \* "Recovery": the connection enters that state after "Initial", "pushing", or a congestion detection in a "cruising" state. It remains in that state for at least one roundtrip, until the first packet sent in "recovery" is acknowledged. Once that happens, the connection goes back to "startup" if the last 3 pushing attempts have resulted in increases of "nominal rate", or if it detects high jitter and the previous initial was not run in these conditions (see ). It enters "cruising" otherwise.
- \* "Cruising": the connection is sending using the "nominal\_rate" and "nominal\_max\_rtt" value. If congestion is detected, the connection exits cruising and enters "recovery" after lowering the value of "nominal\_cwnd". Otherwise, the connection will remain in "cruising" state until at least 4 RTT and the connection is not "app limited". At that point, it enters "pushing".
- \* "Pushing": the connection is using a rate and CWND 25% larger than "nominal\_rate" and "nominal\_CWND". It remains in that state for one round trip, i.e., until the first packet send while "pushing" is acknowledged. At that point, it enters the "recovery" state.

These transitions are summarized in the following state diagram.



#### 4.1. Setting pacing rate, congestion window and quantum

If the nominal rate or the nominal max RTT are not yet assessed, C4 sets pacing rate, congestion window and pacing quantum to initial values:

- \* pacing rate: set to the data rate of the outgoing interface,
- \* congestion window: set to the equivalent of 10 packets,
- \* congestion quantum: set to zero.

If the nominal rate or the nominal max RTT are both assessed, C4 sets pacing rate, and congestion window to values that depends on these variables and on a coefficient `alpha_current`:

```

pacing_rate = alpha_current_ * nominal_rate
cwnd = max (pacing_rate * nominal_max_rtt, 2*MTU)
quantum = max ( min (cwnd / 4, 64KB), 2*MTU)

```

The coefficient alpha for the different states is:

state	alpha	comments
Initial	2	
Recovery	15/16	
Cruising	1	
Pushing	5/4 or 17/16	see Section 4.5 for rules on choosing 5/4 or 17/16

Table 1

#### 4.2. Initial state

When the flow is initialized, it enters the Initial state, during which it does a first assessment of the "nominal rate" and "nominal max RTT". The coefficient `alpha_current` is set to 2. The "nominal rate" and "nominal max RTT" are initialized to zero, which will cause pacing rate and CWND to be set default initial values. The nominal max RTT will be set to the first assessed RTT value, but is not otherwise changed during the initial phase. The nominal rate is updated after receiving acknowledgements, see {#nominal-rate}.

C4 will exit the Initial state and enter Recovery if the nominal rate does not increase for 3 consecutive eras, omitting the eras for which the transmission was "application limited".

C4 exit the Initial if receiving a congestion signal and the following conditions are true:

- 1- If the signal is due to "delay", C4 will only exit the initial state if the `nominal_rate` did not increase in the last 2 eras.
- 2- If the signal is due to "loss", C4 will only exit the initial state if more than 20 packets have been received.



The restriction on delay signals is meant to prevent spurious exit due to delay jitter. The restriction on loss signals is meant to ensure that enough packets have been received to properly assess the loss rate.

#### 4.3. Recovery state

The recovery state is entered from the Initial or Pushing state, or from the Cruising state in case of congestion. The coefficient `alpha_current` is set to 15/16. Because the multiplier is lower than 1, the new value of CWND may well be lower than the current number of bytes in transit. C4 will wait until acknowledgements are received and the number of bytes in transit is lower than CWND to send new packets.

The Recovery ends when the first packet sent during that state is acknowledged. That means that acknowledgement and congestion signals received during recovery are the consequence of packets sent before. C4 assumes that whatever corrective action is required by these events will be taken prior to entering recovery, and that events arriving during recovery are duplicate of the prior events and can be ignored.

Rate increases are detected when acknowledgements received during recovery reflect a successful "push" during the Pushing phase. The prior "Pushing" is considered successful if it did not trigger any congestion event, and if the data rate increases sufficiently between the end of previous Recovery and the end of this one, with sufficiently being defined as:

- \* Any increase if the prior pushing rate (`alpha_prior`) was 17/16 or less,
- \* An increase of at least 1/4th of the expected increase otherwise, for example an increase of 1/16th if `alpha_previous` was 5/4.

C4 re-enters "Initial" at the end of the recovery period if the evaluation shows 3 successive rate increases without congestion, or if high jitter requires restarting the Initial phase (see Section 4.3.1. Otherwise, C4 enters cruising.

Reception of a congestion signal during the Initial phase does not cause a change in the `nominal_rate` or `nominal_max_RTT`.

#### 4.3.1. Restarting Initial if High Jitter

The "nominal max RTT" is not updated during the Initial phase, because doing so would prevent exiting Initial on high delay detection. This can lead to underestimation of the "nominal rate" if the flow is operating on a path with high jitter.

C4 will reenter the "initial" phase on the first time high jitter is detected for the flow. The high jitter is detected after updating the "nominal max RTT" at the end of the recovery era, if:

$$\text{running\_min\_rtt} < \text{nominal\_max\_rtt} * 2/5$$

This will be done at most once per flow.

#### 4.4. Cruising state {#c4-cruising }

The Cruising state is entered from the Recovery state. The coefficient `alpha_current` is set to 1.

C4 will normally transition from Cruising state to Pushing state after 4 eras. It will transition to Recovery before that if a congestion signal is received.

#### 4.5. Pushing state

The Pushing state is entered from the Cruising state. The coefficient `alpha_current` is set to 5/4 if the previous pushing attempt was successful (see Section 4.3), or 17/16 if it was not.

C4 exits the pushing state after one era, or if a congestion signal is received before that. In an exception to standard congestion processing, the reduction in `nominal_rate` and `nominal_max_RTT` are not applied if the congestion signal is tied to a packet sent during the Pushing state.

#### 5. Handling of congestion signals

C4 responds to congestion events by reducing the nominal rate, and in some condition also reducing the nominal max RTT. C4 monitors 3 types of congestion events:

1. Excessive increase of measured RTT,
2. Excessive rate of packet losses (but not mere Probe Time Out, see Section 5.3.1),
3. Excessive rate of ECN/CE marks

C4 monitors successive RTT measurements and compare them to a reference value, defined as the sum of the "nominal max rtt" and a "delay threshold". C4 monitors the arrival of packet losses computes a "smoothed error rate", and compares it to a "loss threshold". When the path supports ECN, C4 monitors the arrival of ECN marks and computes a "smoothed CE rate", and compares it to a "CE threshold". These coefficients depend on the sensitivity coefficient defined in Section 5.1.

### 5.1. Variable Sensitivity

The three congestion detection thresholds are function of the "sensitivity" coefficient, which increases with the nominal rate of the flow. Flows operating at a low data rate have a low sensitivity coefficient and reacts slower to congestion signals than flows operating at a higher rate. If multiple flows share the same bottleneck, the flows with higher data rate will detect congestion signals and back off faster than flow operating at lower rate. This will drive these flows towards sharing the available resource evenly.

The sensitivity coefficient varies from 0 to 1, according to a simple curve:

- \* set sensitivity to 0 if data rate is lower than 50000B/s
- \* linear interpolation between 0 and 0.92 for values between 50,000 and 1,000,000B/s.
- \* linear interpolation between 0.92 and 1 for values between 1,000,000 and 10,000,000B/s.
- \* set sensitivity to 1 if data rate is higher than 10,000,000B/s

The sensitivity index is then used to set the value of delay and loss and CE thresholds.

### 5.2. Detecting Excessive Delays

The delay threshold is function of the nominal max RTT and the sensitivity coefficient:

```
delay_fraction = 1/16 + (1 - sensitivity)*3/16
delay_threshold = min(25ms, delay_fraction*nominal_max_rtt)
```

A delay congestion signal is detected if:

```
rtt_sample > nominal_max_rtt + delay_threshold
```

### 5.3. Detecting Excessive Losses

C4 maintains an average loss rate, updated for every packet as:

```
if packet_is_lost:
    loss = 1
else:
    loss = 0
smoothed_loss_rate = (loss + 15*smoothed_loss_rate)/16
```

The loss threshold is computed as:

```
loss_threshold = 0.02 + 0.50 * (1-sensitivity);
```

A loss is detected if the smoothed loss rate is larger than the threshold. In that case, the coefficient beta is set to 1/4.

#### 5.3.1. Do not react to Probe Time Out

QUIC normally detect losses by observing gaps in the sequences of acknowledged packet. That's a robust signal. QUIC will also inject "Probe time out" packets if the PTO timeout elapses before the last sent packet has not been acknowledged. This is not a robust congestion signal, because delay jitter may also cause PTO timeouts. When testing in "high jitter" conditions, we realized that we should not change the state of C4 for losses detected solely based on timer, and only react to those losses that are detected by gaps in acknowledgements.

### 5.4. Detecting Excessive CE Marks

TBD. The plan is to mimic the L4S specification.

### 5.5. Rate Reduction on Congestion

On entering recovery, C4 reduces the nominal\_rate by the factor "beta" corresponding to the congestion signal:

```
nominal_rate = (1-beta)*nominal_rate
```

The coefficient beta differs depending on the nature of the congestion signal. For packet losses, it is set to 1/4, similar to the value used in Cubic.

For delay based losses, it is proportional to the difference between the measured RTT and the target RTT divided by the acceptable margin, capped to 1/4:

```
beta = min(1/4,  
           (rtt_sample - (nominal_max_rtt + delay_threshold)/  
            delay_threshod))
```

If the signal is an ECN/CE rate, this is still TBD. We could use a proportional reduction coefficient in line with [RFC9331], but we should use the sensitivity coefficient to modulate that signal.

## 6. Security Considerations

We do not believe that C4 introduce new security issues. Or maybe there are, such as what happen if applications can be fooled in going to fast and overwhelming the network, or going too slow and underwhelming the application. Discuss!

## 7. IANA Considerations

This document has no IANA actions.

## 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>>.
- [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

- [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>>.
- [I-D.ietf-moq-transport] Nandakumar, S., Vasiliev, V., Swett, I., and A. Frindell, "Media over QUIC Transport", Work in Progress, Internet-Draft, draft-ietf-moq-transport-14, 2 September 2025, <<https://datatracker.ietf.org/doc/html/draft-ietf-moq-transport-14>>.

[RFC9331] De Schepper, K. and B. Briscoe, Ed., "The Explicit Congestion Notification (ECN) Protocol for Low Latency, Low Loss, and Scalable Throughput (L4S)", RFC 9331, DOI 10.17487/RFC9331, January 2023, <<https://www.rfc-editor.org/info/rfc9331>>.

#### Acknowledgments

TODO acknowledge.

#### Authors' Addresses

Christian Huitema  
Private Octopus Inc.  
Email: [huitema@huitema.net](mailto:huitema@huitema.net)

Suhas Nandakumar  
Cisco  
Email: [snandaku@cisco.com](mailto:snandaku@cisco.com)

Cullen Jennings  
Cisco  
Email: [fluffy@iii.ca](mailto:fluffy@iii.ca)