Internet Engineering Task Force
INTERNET-DRAFT
draft-ietf-dccp-ccid3-09.ps
Expires: 20 June 2005

Sally Floyd
ICIR
Eddie Kohler
UCLA
Jitendra Padhye
Microsoft Research
20 December 2004

Profile for DCCP Congestion Control ID 3: TFRC Congestion Control

Status of this Memo

Copyright Notice

Abstract

This document contains the profile for Congestion Control Identifier 3, TCP-Friendly Rate Control (TFRC), in the Datagram Congestion Control Protocol (DCCP). CCID 3 should be used by senders that want a TCP-friendly sending rate, possibly with Explicit Congestion Notification (ECN), while minimizing abrupt rate changes.

TO BE DELETED BY THE RFC EDITOR UPON PUBLICATION:

Changes from draft-ietf-dccp-ccid3-08.txt:

* Add description of data and sequence loss interval lengths.

* Change Loss Intervals option to include loss interval data lengths.

* Some rephrasing, as a result of working group feedback.

* Added section numbers to many references.

* Referred to RFC 3448 for the definition of the first loss interval, and for the definition of the beginning and end of a loss interval.

* Clarified that X_inrecv is in bytes per second, and changed "X_inrecv - 3*s" to "X_inrecv - 3*s/RTT", to keep all of the units straight.

Changes from draft-ietf-dccp-ccid3-07.txt:

* Loss Intervals is mandatory.

* Elapsed Time is mandatory, even if there's a Timestamp Echo.

* Send Loss Event Rate defaults to zero.

* Rewrite Section 5.

* IANA Considerations.

* Wording nits.

Changes from draft-ietf-dccp-ccid3-06.txt:

* Moved the sections on Possible Changes to the Initial Window and Other Possible Changes to TFRC to be the section on Possible Future Changes to CCID3 in the appendix.

* Some rephrasing, as a result of Working Group Last Call.

* Specified the value of the inverted loss event rate when the loss event rate is 0. From a suggestion from David Vos.

* Added that the optional procedure for estimated the RTT at the receiver does not work when the inter-packet sending times are greater than the RTT. From a suggestion by Ladan Gharai.

Changes from draft-ietf-dccp-ccid3-05.txt:

* Added a section on Response to Idle and Application-limited Periods

* Added a paragraph on the sending rate when no feedback is received from the receiver.

* Expanded on the discussion of the packet size s used in the TCP throughput equation.

* Some editing to improve the presentation.

* Added to discussion of response to Data Dropped and Slow Receiver.

* Deleted the optional algorithm given in Section 8.7.1 for receivers to estimate the RTT, and replaced it with one sentence.

* Added a section on Other Possible Changes to TFRC.

Changes from draft-ietf-dccp-ccid3-04.txt:

* Minor editing.

* Said that implementations may check for apps that are manipulating the packet size inappropriately.

* Deletes the maximum packet size of 1500 bytes.

* Added discussion on using the CCVal counter for estimating the round-trip time.

* Changed the option number for the Loss Intervals option.

* Added the Intellectual Property Notice.

Changes from draft-ietf-dccp-ccid3-03.txt:

* Added more text to the section on Congestion Control on Data Packets to make it more readable, and to summarize the key mechanisms specified in the TFRC spec.

* Said that it is OK to use an initial sending rate of 2-4 pkts/RTT, based on RFC 3390. And that in the future an initial sending rate of up to 8 pkts/RTT might be specified, for very small packets.

* Receive Rate is measured in bytes per second, as RFC 3448 specifies.

* New definition of Loss Intervals option, because old definition was 24-bit-sequence-number specific; and add an example.

Changes from draft-ietf-dccp-ccid3-02.txt:

* Added to the section on Application Requirements.

* Added a section on Packet Sizes.

Changes from draft-ietf-dccp-ccid3-01.txt:

* Added "Security Considerations" and "IANA Considerations" sections.

* Store Window Counter in the DCCP header's CCVal field, not a separate option.

* Add to the description of a loss interval in the Loss Intervals option: a loss interval includes at most one round-trip time's worth of possibly-marked packets, and at least one round-trip time's worth of packets in all.

* Added a description of when the loss event rate calculated by the sender could differ from that calculated by the receiver.

* Window counter fixups.

* Add Use Loss Intervals and Use Loss Event Rate features, and explain their interaction.

* Move Elapsed Time option to DCCP's main specification (and simultaneously change its units to tenths of milliseconds). Allow the use of either Elapsed Time or Timestamp Echo.

* Clarify the definition of quiescence.

* Change calculations for determining loss events to take window counter wrapping into account.

Changes from draft-ietf-dccp-ccid3-00.txt:

* Changed the guidelines to say that required acknowledgement packets should include one or more of the following: The Loss Event Rate, Loss Intervals, or the Ack Vector.

* Added a separate section on "The Use of Ack Vectors". This section says that Ack-of-acks must be used when the Ack Vector is used.

* Renamed the "ECN Nonce Option" to the "Loss Intervals" option, and extended this option to include up to eight loss intervals. This is to enable more precise verification by the sender of the receiver's feedback.

* Added a section about "When should Ack Vector or Loss Intervals be used?" In progress.

* Added a section about using the ECN Nonce to verify the receiver's feedback.

* Said that the ECN-Nonce feedback must be returned in every required acknowledgement.

* Added a sentence saying that the TFRC spec "separately specifies the minimum sending rate from rate reductions during an idle period."

Table of Contents

1. Introduction 7
2. Conventions 7
3. Usage 7
  3.1. Relationship with TFRC 8
  3.2. Example Half-Connection 8
4. Connection Establishment 8
5. Congestion Control on Data Packets 9
  5.1. Response to Idle and Application-limited Periods 10
  5.2. Response to Data Dropped and Slow Receiver 10
  5.3. Packet Sizes 11
6. Acknowledgements 11
  6.1. Loss Interval Definition 12
    6.1.1. Loss Interval Lengths 13
  6.2. Congestion Control on Acknowledgements 14
  6.3. Acknowledgements of Acknowledgements 14
  6.4. Quiescence 15
7. Explicit Congestion Notification 15
8. Options and Features 15
  8.1. Window Counter Value 16
  8.2. Elapsed Time Options 17
  8.3. Receive Rate Option 17
  8.4. Send Loss Event Rate Feature. 17
  8.5. Loss Event Rate Option 18
  8.6. Loss Intervals Option 18
    8.6.1. Option Details 19
    8.6.2. Example 19
9. Verifying Congestion Control Compliance With ECN 20
  9.1. Verifying the ECN Nonce Echo 21
  9.2. Verifying the Reported Loss Intervals and Loss Event Rate 21
10. Implementation Issues 22
  10.1. Timestamp Usage 22
  10.2. Determining Loss Events at the Receiver 22
  10.3. Sending Feedback Packets 23
11. Security Considerations. 24
12. IANA Considerations 25
  12.1. Reset Codes 25
  12.2. Option Types 25
  12.3. Feature Numbers 25
13. Thanks 25
A. Appendix: Possible Future Changes to CCID 3 26
Normative References 26
Informative References 27
Authors' Addresses 27
Full Copyright Statement 27
Intellectual Property 28

List of Tables

Table 1: DCCP CCID 3 Options 15
Table 2: DCCP CCID 3 Feature Numbers. 15

1. Introduction

This document contains the profile for Congestion Control Identifier 3, TCP-friendly rate control (TFRC), in the Datagram Congestion Control Protocol (DCCP) [DCCP]. DCCP uses Congestion Control Identifiers, or CCIDs, to specify the congestion control mechanism in use on a half-connection.

TFRC is a receiver-based congestion control mechanism that provides a TCP-friendly sending rate, while minimizing the abrupt rate changes characteristic of TCP or of TCP-like congestion control [RFC 3448]. The sender's allowed sending rate is set in response to the loss event rate, which is typically reported by the receiver to the sender. See Section 3 for more on application requirements.

2. Conventions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119.

All multi-byte numerical quantities in CCID 3, such as arguments to options, are transmitted in network byte order (most significant byte first).

A DCCP half-connection consists of the application data sent by one endpoint and the corresponding acknowledgements sent by the other endpoint. The terms "HC-Sender" and "HC-Receiver" denote the endpoints sending application data and acknowledgements, respectively. Since CCIDs apply at the level of half-connections, we abbreviate HC-Sender to "sender" and HC-Receiver to "receiver" in this document. See DCCP for more discussion.

For simplicity, we say that senders send DCCP-Data packets and receivers send DCCP-Ack packets. Both of these categories are meant to include DCCP-DataAck packets.

The phrases "ECN-marked" and "marked" refer to packets marked ECN Congestion Experienced unless otherwise noted.

This document uses a number of variables from RFC 3448, including:

- X_recv: The receive rate in bytes per second. See [RFC 3448] (Section 3.2.2).
- s: The packet size in bytes. See [RFC 3448] (Section 3.1).
- p: The loss event rate. See [RFC 3448] (Section 3.1).

3. Usage

CCID 3's TFRC congestion control is appropriate for flows that would prefer to minimize abrupt changes in the sending rate, including streaming media applications with small or moderate receiver buffering before playback. TCP-like congestion control, such as that of DCCP's CCID 2 [CCID 2 PROFILE], halves the sending rate in response to each congestion event, and thus cannot provide a relatively smooth sending rate.

As explained in RFC 3448 (Section 1), the penalty of having smoother throughput than TCP while competing fairly for bandwidth is that the TFRC mechanism in CCID 3 responds slower than TCP or TCP-like mechanisms to changes in available bandwidth. Thus, CCID 3 should only be used for applications with a requirement for smooth throughput, in particular avoiding TCP's halving of the sending rate in response to a single packet drop. For applications that simply need to transfer as much data as possible in as short a time as possible, we recommend using TCP-like congestion control, such as CCID 2.

CCID 3 should also not be used by applications that change their sending rate by varying the packet size, rather than varying the rate at which packets are sent. A new CCID will be required for these applications.

3.1. Relationship with TFRC

The congestion control mechanisms described here follow the TFRC mechanism standardized by the IETF [RFC 3448]. Conformant CCID 3 implementations MAY track updates to the TCP throughput equation directly, as updates are standardized in the IETF, rather than waiting for revisions of this document. However, conformant implementations SHOULD wait for explicit updates to CCID 3 before implementing other changes to TFRC congestion control.

3.2. Example Half-Connection

This example shows the typical progress of a half-connection using CCID 3's TFRC Congestion Control, not including connection initiation and termination. The example is informative, not normative.

1. The sender transmits DCCP-Data packets, where the sending rate is governed by the allowed transmit rate as specified in RFC 3448 (Section 3.2). Each DCCP-Data packet has a sequence number, and the DCCP header's CCVal field contains the window counter value, used by the receiver in determining when multiple losses belong in a single loss event.

   In the typical case of an ECN-capable half-connection, each DCCP-Data and DCCP-DataAck packet is sent as ECN-Capable, with either the ECT(0) or the ECT(1) codepoint set. The use of the ECN Nonce with TFRC is described in Section 9.

2. The receiver sends DCCP-Ack packets at least once per round-trip time acknowledging the data packets, unless the sender is sending at a rate of less than one packet per round-trip time, as indicated by the TFRC specification RFC 3448 (Section 6). Each DCCP-Ack packet uses a sequence number, identifies the most recent packet received from the sender, and includes feedback about the recent loss intervals experienced by the receiver.

3. The sender continues sending DCCP-Data packets as controlled by the allowed transmit rate. Upon receiving DCCP-Ack packets, the sender updates its allowed transmit rate as specified in RFC 3448 (Section 4.3). This update is based upon a loss event rate calculated by the sender, based on the receiver's loss intervals feedback. If it prefers, the sender can also use a loss event rate calculated and reported by the receiver.

4. The sender estimates round-trip times and calculates a nofeedback time, as specified in RFC 3448 (Section 4.4). If no feedback is received from the receiver in that time (at least four round-trip times), the sender halves its sending rate.

4. Connection Establishment

The connection is initiated by the client using mechanisms described in the DCCP specification [DCCP]. During or after CCID 3 negotiation, the client and/or server may want to negotiate the values of the Send Ack Vector and Send Loss Event Rate features.

5. Congestion Control on Data Packets

CCID 3 uses the congestion control mechanisms of TFRC [RFC 3448]. The following discussion summarizes information from RFC 3448, which should be considered normative except where specifically indicated.

Loss Event Rate

The basic operation of CCID 3 centers around the calculation of a loss event rate: the number of loss events as a fraction of the number of packets transmitted, weighted over the last several loss intervals. This loss event rate, a round-trip time estimate, and the average packet size are plugged into the TCP throughput equation, as specified in RFC 3448 (Section 3.1). The result is a fair transmit rate, close to what a modern TCP would achieve in the same conditions. CCID 3 senders are limited to this fair rate.

The loss event rate itself is calculated in CCID 3 using recent loss interval lengths reported by the receiver. Loss intervals are precisely defined in Section 6.1. In summary, a loss interval is up to 1 RTT of possibly lost or ECN-marked data packets, followed by an arbitrary number of non-dropped, non-marked data packets. Thus, long loss intervals represent low congestion rates. The CCID 3 Loss Intervals option is used to report loss interval lengths; see Section 8.6.

Other Congestion Control Mechanisms

The sender starts in a slow-start phase, roughly doubling its allowed sending rate each round-trip time. The slow-start phase is ended by the receiver's report of a data packet drop or mark, after which the sender uses the loss event rate to calculate its allowed sending rate.

RFC 3448 (Section 4) specifies an initial sending rate of one packet per RTT (Round-Trip Time) as follows: The sender initializes the allowed sending rate to one packet per second. As soon as a feedback packet is received from the receiver, the sender has a measurement of the round-trip time, and then sets the initial allowed sending rate to one packet per RTT. However, while the initial TCP window used to be one segment, RFC 2581 allows an initial TCP window of two segments, and RFC 3390 allows an initial TCP window of three or four segments (up to 4380 bytes). RFC 3390 gives an upper bound on the initial window of

    min(4*MSS, max(2*MSS, 4380 bytes)).

Translating this to the packet-based congestion control of CCID 3, the initial CCID 3 sending rate is allowed to be at least two packets per RTT, and at most four packets per RTT, depending on the packet size. The initial rate is only allowed to be three or four packets per RTT when, in terms of segment size, that translates to at most 4380 bytes per RTT.

The sender's measurement of the round-trip time uses the Elapsed Time and/or Timestamp Echo option contained in feedback packets, as described in Section 8.2. The Elapsed Time option is required, while the Timestamp Echo option is not required. The sender maintains an average round-trip time heavily weighted on the most recent measurements.

Each DCCP-Data packet contains a sequence number. Each DCCP-Data packet also contains a window counter value, as described in Section 8.1 below. The window counter is incremented by one every quarter round-trip time. The receiver uses it as a coarse-grained timestamp to determine when a packet loss should be considered part of an existing loss interval, or must begin a new loss interval.

Because TFRC is rate-based instead of window-based, and because feedback packets can be dropped in the network, the sender needs some mechanism for reducing its sending rate in the absence of positive feedback from the receiver. As described in Section 6, the receiver sends feedback packets roughly once per round-trip time. As specified in RFC 3448 (Section 4.3), the sender sets a nofeedback timer to at least four round-trip times, or to twice the interval between data packets, whichever is larger; if the sender hasn't received a feedback packet from the receiver when the nofeedback timer expires, then the sender halves its allowed sending rate. The allowed sending rate is never reduced below one packet per 64 seconds. Note that not all acknowledgements are considered feedback packets, since feedback packets must contain valid Loss Intervals, Elapsed Time, and Receive Rate options.

If the sender never receives a feedback packet from the receiver, and as a consequence never gets to set the allowed sending rate to one packet per RTT, then the sending rate is left at its initial rate of one packet per second, with the nofeedback timer expiring after two seconds. The allowed sending rate is halved each time the nofeedback timer expires. Thus, if no feedback is received from the receiver, the allowed sending rate is never above one packet per second, and is quickly reduced below one packet per second.

The feedback packets from the receiver contain a Receive Rate option specifying the rate at which data packets arrived at the receiver since the last feedback packet. The allowed sending rate can be at most twice the rate received at the receiver in the last round-trip time. This may be less than the nominal fair rate if, for example, the application is sending less than its fair share.

5.1. Response to Idle and Application-limited Periods

One consequence of the nofeedback timer is that the sender reduces the allowed sending rate when the sender has been idle for a significant period of time. In RFC 3448 (Section 4.4), the allowed sending rate is never reduced to less than two packets per round-trip time as the result of an idle period. In CCID 3, we revise this to take into account the larger initial windows allowed by RFC 3390. That is, the allowed sending rate is never reduced to less than the RFC 3390 initial sending rate as the result of an idle period. If the allowed sending rate is less than the initial sending rate upon entry to the idle period, then it will still be less than the initial sending rate when exiting the idle period. However, the allowed sending rate should not be reduced to below the initial sending rate because of reductions of the allowed sending rate during the idle period itself.

The sender's allowed sending rate is limited to at most twice the receive rate reported by the receiver. Thus, after an application-limited period, the sender can at most double its sending rate from one round-trip time to the next, until it reaches the allowed sending rate determined by the loss event rate.

5.2. Response to Data Dropped and Slow Receiver

A CCID 3 sender responds to packets acknowledged as Data Dropped as described in [DCCP], with the following further clarifications.

- **Drop Code 2 ("receive buffer drop").** The allowed sending rate is reduced by one packet per RTT for each packet newly acknowledged as Drop Code 2, except that it is never reduced below one packet per RTT as a result of Drop Code 2.
- **Adjusting the receive rate X_recv.** A CCID 3 sender SHOULD also respond to non-network-congestion events, such as those implied by Data Dropped and Slow Receiver options, by adjusting X_recv, the receive rate reported by the receiver in Receive Rate options (see Section 8.3). The CCID 3 sender's allowed sending rate is limited to at most twice the receive rate reported by the receiver, via the "min(..., 2*X_recv)" clause

in TFRC's throughput calculations [RFC 3448] (Section 4.3). When the sender receives one or more Data Dropped and Slow Receiver options, the sender SHOULD adjust X_recv as follows:

1. Let X_inrecv equal the Receive Rate in bytes per second reported by the receiver in the most recent acknowledgement.
2. Let X_drop equal the upper bound on the sending rate implied by Data Dropped and Slow Receiver options. If the sender receives a Slow Receiver option, which requests that the sender not increase its sending rate for roughly a round-trip time [DCCP], then X_drop should be set to X_inrecv. Similarly, if the sender receives a Data Dropped option indicating, for example, that three packets were dropped with Drop Code 2, then the upper bound on the sending rate will be decreased by at most three packets per RTT, by the sender setting X_drop to
   max(X_inrecv - 3*s/RTT, min(X_inrecv, s/RTT)).
   Again, s is the packet size in bytes.
3. Set X_recv := min(X_inrecv, X_drop/2).

As a result, the next round-trip time's sending rate will be limited to at most 2*(X_drop/2) = X_drop. The effects of the Slow Receiver and Data Dropped options on X_recv will mostly vanish by the round-trip time after that, which is appropriate for this non-network-congestion feedback. This procedure MUST only be used for those Drop Codes not related to corruption (see [DCCP]). Currently, this is limited to Drop Codes 0, 1, and 2.

5.3. Packet Sizes

CCID 3 is intended for applications that use a fixed packet size, and that vary their sending rate in packets per second in response to congestion. CCID 3 is not appropriate for applications that require a fixed interval of time between packets, and vary their packet size instead of their packet rate in response to congestion. However, some attention might be required for applications using CCID 3 that vary their packet size not in response to congestion, but in response to other application-level requirements.

The packet size s is used in the TCP throughput equation. A CCID 3 implementation MAY calculate s as the segment size averaged over multiple round trip times -- for example, over the most recent four loss intervals, for loss intervals as defined in Section 6.1. Alternately, a CCID 3 implementation MAY use the Maximum Packet Size to derive s. In this case, s is set to the Maximum Segment Size (MSS), the maximum size in bytes for the data segment, not including the default DCCP and IP packet headers. Each packet transmitted then counts as one MSS, regardless of the actual segment size, and the TCP throughput equation can be interpreted as specifying the sending rate in packets per second.

CCID 3 implementations MAY check for applications that appear to be manipulating the packet size inappropriately. For example, an application might send small packets for a while, building up a fast rate, then switch to large packets to take advantage of the fast rate. (Preliminary simulations indicate that applications may not be able to increase their overall transfer rates this way, so it is not clear this manipulation will occur in practice [V03].)

6. Acknowledgements

The receiver sends an acknowledgement to the sender roughly once per round-trip time, if the sender is sending packets that frequently. This rate is determined by the TFRC protocol, specified in RFC 3448 (Section 6).

As specified in [DCCP], the acknowledgement number acknowledges the greatest valid sequence number received so far on this connection. ("Greatest" is, of course, measured in circular sequence space.) Each acknowledgement required by TFRC also includes at least the following options:

1. An Elapsed Time and/or Timestamp Echo option specifying the amount of time elapsed since the arrival at the receiver of the packet whose sequence number appears in the Acknowledgement Number field. These options are described in [DCCP] (Sections 13.2 and 13.1).
2. A Receive Rate option, defined in Section 8.3, specifying the rate at which data was received since the last DCCP-Ack was sent.
3. A Loss Intervals option, defined in Section 8.6, specifying the most recent loss intervals experienced by the receiver. (The definition of a loss interval is provided below.) From Loss Intervals, the sender can easily calculate the loss event rate *p* using the procedure described in RFC 3448 (Section 5.4).

Acknowledgements not containing at least these three options are not considered feedback packets.

The receiver MAY also include other options concerning the loss event rate, including Loss Event Rate, which gives the loss event rate calculated by the receiver, defined in Section 8.5, and DCCP's generic Ack Vector option, which reports the specific sequence numbers of any lost or marked packets [DCCP] (Section 11.4). Ack Vector is not required by CCID 3's congestion control mechanisms: the Loss Intervals option provides all the information needed to manage the transmit rate and probabilistically verify receiver feedback. However, Ack Vector may be useful for applications that need to determine exactly which packets were lost.

If the HC-Receiver is also sending data packets to the HC-Sender, then it MAY piggyback acknowledgement information on those data packets more frequently than TFRC's specified acknowledgement rate allows.

6.1. Loss Interval Definition

As described in RFC 3448 (Section 5.2), a loss interval begins with a lost or ECN-marked data packet; continues with at most one round trip time's worth of packets that may or may not be lost or marked; and completes with an arbitrarily-long series of non-dropped, non-marked data packets. For example, here is a single loss interval, assuming that sequence numbers increase as you move right:

```
  Lossy Part
   <= 1 RTT   __________ Lossless Part __________
 /         \/                                    \
 *----*--*--*-------------------------------------
 ^    ^  ^  ^
losses or marks
```

Note that a loss interval's lossless part might be empty, as in the first interval below:

```
        Lossy Part   Lossy Part
         <= 1 RTT     <= 1 RTT   _____ Lossless Part _____
       /          \/           \/                         \
       *----*--*--***--------*-*----------------------------
       ^    ^  ^  ^^^        ^ ^
       \_ Int. 1 _/\_____________ Interval 2 _____________/
```

As in RFC 3448 (Section 5.2), the length of the lossy part MUST be <= 1 RTT. CCID 3 uses window counter values, not receive times, to determine whether multiple packets occurred in the same RTT, and thus belong to the same loss event; see Section 10.2. A loss interval whose lossy part lasts for more than 1 RTT, or whose lossless part contains a dropped or marked data packet, is invalid.

A missing data packet doesn't begin a new loss interval until NDUPACK packets have been seen after the "hole", where NDUPACK = 3. Thus, up to NDUPACK of the most recent sequence numbers (including the sequence numbers of any holes) might temporarily not be part of any loss interval, while the implementation waits to see whether a hole will be filled. See RFC 3448 (Section 5.1) and RFC 2581 (Section 3.2) for further discussion of NDUPACK.

As specified by RFC 3448 (Section 5), all loss intervals except the first begin with a lost or marked data packet, and all loss intervals are as long as possible, subject to the validity constraints above.

Lost and ECN-marked non-data packets may occur freely in the lossless part of a loss interval. (Non-data packets consist of those packet types that cannot carry application data, namely DCCP-Ack, DCCP-Close, DCCP-CloseReq, DCCP-Reset, DCCP-Sync, and DCCP-SyncAck.) In the absence of better information, a receiver MUST conservatively assume that every lost packet was a data packet, and thus must occur in some lossy part. DCCP's NDP Count option can help the receiver determine whether a particular packet contained data; see [DCCP] (Section 7.7).

6.1.1. Loss Interval Lengths

RFC 3448 defines the TFRC congestion control mechanism in terms of a one-way transfer of data, with data packets going from the sender to the receiver and feedback packets going from the receiver back to the sender. However, CCID 3 applies in a context of two half-connections, with DCCP-Data and and DCCP-DataAck packets from one half-connection sharing sequence number space with DCCP-Ack packets from the other half-connection. For the purposes of CCID 3 congestion control, loss interval lengths should only include data packets, and exclude the acknowledgement packets from the reverse half-connection; but it's also useful to report the total number of packets in each loss interval (for example, to facilitate ECN Nonce verification).

CCID 3's Loss Intervals option thus reports two lengths for each loss interval. An interval's sequence length is the total number of packets the sender transmitted during the interval, and is easily calculated in DCCP as the greatest packet sequence number in the interval minus the greatest packet sequence number in the preceding interval (or, if there is no preceding interval, the initial sequence number in the CCID 3 half-connection). An interval's data length is the number used in TFRC's loss event rate calculation, as defined in RFC 3448 (Section 5), and is calculated as follows.

For all loss intervals except the first, the data length equals the sequence length minus the number of non-data packets the sender transmitted during the loss interval, except that the minimum data length is one packet. In the absence of better information, an endpoint MUST conservatively assume that the loss interval contained only data packets, in which case the data length equals the sequence length. To achieve greater precision, the sender can calculate the exact number of non-data packets in an interval by remembering which sent packets contained data; the receiver can count non-data packets received or received ECN-marked, and for packets that were not received, it may be able to discriminate between lost data packets and lost non-data packets using DCCP's NDP Count option.

For the first loss interval, the data length is undefined until the first loss event. RFC 3448 (Section 6.3.1) specifies how the first loss interval's data length is calculated once the first loss event has occurred; this calculation uses X_recv, the most recent receive rate, as input. Until this first loss event, the loss event rate is zero, as is the data length reported for the interval in the Loss Intervals option.

The first loss interval's data length might be less than, equal to, or even greater than its sequence length. Any other loss interval's data length must be less than or equal to its sequence length.

A sender MAY use the loss event rate or loss interval data lengths as reported by the receiver, or it MAY recalculate loss event rate and/or loss interval data lengths based on receiver feedback and additional information. For example, assume the network drops a DCCP-Ack packet with sequence number 50. The receiver might then report a loss interval beginning at sequence number 50. If the sender determined that this loss interval actually contained no lost or ECN-marked data packets, then it might coalesce the loss interval with the previous loss interval, resulting in a larger allowed transmit rate.

6.2. Congestion Control on Acknowledgements

The rate and timing for generating acknowledgements is determined by the TFRC algorithm [RFC 3448] (Section 6). The sending rate for acknowledgements is relatively low -- roughly once per round-trip time -- so there is no need for explicit congestion control on acknowledgements.

6.3. Acknowledgements of Acknowledgements

TFRC acknowledgements don't generally need to be reliable, so the sender generally need not acknowledge the receiver's acknowledgements. When Ack Vector is used, however, the sender, DCCP A, MUST occasionally acknowledge the receiver's acknowledgements so that the receiver can free up Ack Vector state. When both half-connections are active, the necessary acknowledgements will be contained in A's acknowledgements to B's data. If the B-to-A half-connection goes quiescent, however, DCCP A must send an acknowledgement proactively.

Thus, when Ack Vector is used, an active sender MUST acknowledge the receiver's acknowledgements approximately once per round-trip time, within a factor of two or three, probably by sending a DCCP-DataAck packet. No acknowledgement options are necessary, just the Acknowledgement Number in the DCCP-DataAck header.

The sender MAY choose to acknowledge the receiver's acknowledgements even if they do not contain Ack Vectors. For instance, regular acknowledgements can shrink the size of the Loss Intervals option. Unlike the Ack Vector, however, the Loss Intervals option is bounded in size (and receiver state), so acks-of-acks are not required.

6.4. Quiescence

This section describes how a CCID 3 receiver determines that the corresponding sender is not sending any data, and therefore has gone quiescent. See [DCCP] (Section 11.1) for general information on quiescence.

Let T equal the greater of 0.2 seconds and two round-trip times. (A CCID 3 receiver has a rough measure of the round-trip time, so that it can pace its acknowledgements.) The receiver detects that the sender has gone quiescent after T seconds have passed without receiving any additional data from the sender.

7. Explicit Congestion Notification

CCID 3 supports Explicit Congestion Notification (ECN) [RFC 3168]. In the typical case of an ECN-capable half-connection (where the receiver's ECN Incapable feature is set to zero), the sender will use the ECN Nonce for its data packets, as specified in [DCCP] (Section 12.2). Information about the ECN Nonce MUST be returned by the receiver using the Loss Intervals option, and any Ack Vector options MUST include the ECN Nonce Sum. The sender MAY maintain a table with the ECN nonce sum for each packet, and use this information to probabilistically verify the ECN nonce sums returned in Loss Intervals or Ack Vector options. Section 9 describes this further.

8. Options and Features

CCID 3 can make use of DCCP's Ack Vector, Timestamp, Timestamp Echo, and Elapsed Time options, and its Send Ack Vector and ECN Incapable features. In addition, the following CCID-specific options are defined for use with CCID 3.

Type	Option Length	Meaning	DCCP-Data?	Section Reference
128-191		Reserved		
192	6	Loss Event Rate	N	8.5
193	variable	Loss Intervals	N	8.6
194	6	Receive Rate	N	8.3
195-255		Reserved		

Table 1: DCCP CCID 3 Options

The "DCCP-Data?" column indicates that all currently defined CCID 3-specific options MUST be ignored when they occur on DCCP-Data packets.

The following CCID-specific feature is also defined.

Number	Meaning	Rec'n Rule	Initial Value	Req'd	Section Reference
128-191	Reserved				
192	Send Loss Event Rate	SP	0	N	8.4
193-255	Reserved				

Table 2: DCCP CCID 3 Feature Numbers

The column meanings are described in [DCCP] (Table 4). "Rec'n Rule" defines the feature's reconciliation rule, where "SP" means server-priority. "Req'd" specifies whether every CCID 3 implementation MUST understand a feature; Send Loss Event Rate is optional, in that it behaves like an extension [DCCP] (Section 15).

8.1. Window Counter Value

The data sender stores a 4-bit window counter value in the DCCP generic header's CCVal field on every data packet it sends. This value is set to 0 at the beginning of the transmission, and generally increased by 1 every quarter of a round-trip time, as described in RFC 3448 (Section 3.2.1). For reference, the DCCP generic header is as follows (diagram repeated from [DCCP], which also shows the generic header with a 24-bit Sequence Number field).

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|          Source Port          |           Dest Port           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Data Offset  | CCVal | CsCov |           Checksum            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Res | Type  |1|   Reserved    |  Sequence Number (high bits)  .
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
.                  Sequence Number (low bits)                   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The CCVal field has enough space to express 4 round-trip times at quarter-RTT granularity. The sender MUST avoid wrapping CCVal on adjacent packets, as might happen, for example, if two data-carrying packets were sent 4 round-trip times apart with no packets intervening. Therefore, the sender SHOULD use the following algorithm for setting CCVal. The algorithm uses three variables: "last_WC" holds the last window counter value sent, "last_WC_time" is the time at which the first packet with window counter value "last_WC" was sent, and "RTT" is the current round-trip time estimate. last_WC is initialized to zero, and last_WC_time to the time of the first packet sent. Then, before sending a new packet, proceed like this:

```
Let quarter_RTTs = floor((current_time - last_WC_time) / (RTT/4)).
If quarter_RTTs > 0, then:
    Set last_WC := (last_WC + min(quarter_RTTs, 5)) mod 16, and
    Set last_WC_time := current_time.
Set the packet header's CCVal field to last_WC.
```

When this algorithm is used, adjacent data-carrying packets' CCVal counters never differ by more than five, modulo 16.

The window counter value may also change as feedback packets arrive. In particular, after receiving an acknowledgement for a packet sent with window counter WC, the sender SHOULD increase its window counter, if necessary, so that subsequent packets have window counter value at least (WC + 4) mod 16.

The CCVal counters are used by the receiver to determine whether multiple losses belong to a single loss event, to determine the interval to use for calculating the receive rate, and to determine when to send feedback packets. None of these procedures require the receiver to maintain an explicit estimate of the round-trip time. However, implementors who wish to keep such an RTT estimate may do so using CCVal. Let T(I) be the arrival time of the earliest valid received packet with CCVal = I. (Of course, when the window counter value wraps around to the same value mod 16, we must recalculate T(I).) Let D = 2, 3, or 4, and say that T(K) and T(K+D) both exist (packets were received with window counters K and K+D). Then the value (T(K+D) - T(K)) * 4/D MAY serve as an estimate of the round-trip time. Values of D = 4 SHOULD be preferred for RTT estimation. Concretely, say that the following packets arrived:

```
Time:        T1   T2   T3 T4   T5             T6   T7    T8   T9
       ------*---*---*-*----*------------*---*----*--*---->
CCVal:       K-1 K-1   K K    K+1            K+3 K+4   K+3 K+4
```

Then T7 - T3, the difference between the receive times of the first packet received with window counter K+4 and the first packet received with window counter K, is a reasonable round-trip time estimate. Because of the necessary constraint that measurements can only come from packet pairs whose CCVals differ by at most 4, this procedure does not work when the inter-packet sending times are significantly greater than the RTT, resulting in packet pairs whose CCVals differ by 5. Explicit RTT measurement techniques, such as Timestamp and Timestamp Echo, should be used in that case.

8.2. Elapsed Time Options

The data receiver MUST include an elapsed time value on every required acknowledgement. This helps the sender distinguish between network round-trip time, which it must include in its rate equations, and delay at the receiver due to TFRC's infrequent acknowledgement rate, which it need not include. The elapsed time value is included in one, or possibly two, ways:

1. If at least one recent data packet (i.e., a packet received after the previous DCCP-Ack was sent) included a Timestamp option, then the receiver SHOULD include the corresponding Timestamp Echo option, with Elapsed Time value.
2. In any case, the receiver MUST include an Elapsed Time option.

All these option types are defined in the main DCCP specification [DCCP].

8.3. Receive Rate Option

```
+--------+--------+--------+--------+--------+--------+
|11000010|00000110|            Receive Rate           |
+--------+--------+--------+--------+--------+--------+
 Type=194   Len=6
```

This option MUST be sent by the data receiver on all required acknowledgements. Its four data bytes indicate the rate at which the receiver has received data since it last sent an acknowledgement, in bytes per second. To calculate this receive rate, the receiver sets t to the larger of the estimated round-trip time and the time since the last Receive Rate option was sent. (Received data packets' window counters can be used to produce a suitable RTT estimate, as described in Section 8.1.) The receive rate then equals the number of data bytes received in the most recent t seconds, divided by t.

Receive Rate options MUST NOT be sent on DCCP-Data packets, and any Receive Rate options on received DCCP-Data packets MUST be ignored.

8.4. Send Loss Event Rate Feature

The Send Loss Event Rate feature lets CCID 3 endpoints negotiate whether the receiver MUST provide Loss Event Rate options on its acknowledgements. DCCP A sends a "Change R(Send Loss Event Rate, 1)" option to ask DCCP B to send Loss Event Rate options as part of its acknowledgement traffic.

Send Loss Event Rate has feature number 192, and is server-priority. It takes one-byte Boolean values. DCCP B MUST send Loss Event Rate options on its acknowledgements when Set Loss Event Rate/B is one, although it MAY send Loss Event Rate options even when Send Loss Event Rate/B is zero. Values of two or more are reserved. A CCID 3 half-connection starts with Send Loss Event Rate equal to zero.

8.5. Loss Event Rate Option

```
+--------+--------+--------+--------+--------+--------+
|11000000|00000110|           Loss Event Rate         |
+--------+--------+--------+--------+--------+--------+
 Type=192   Len=6
```

The option value indicates the inverse of the loss event rate, rounded UP, as calculated by the receiver. Its units are data packets per loss interval. Thus, if the Loss Event Rate option value is 100, then the loss event rate is 0.01 loss events per data packet (and the average loss interval contains 100 data packets). When each loss event has exactly one data packet loss, the loss event rate is the same as the data packet drop rate.

See [RFC 3448] (Section 5) for a normative calculation of loss event rate. Before any losses have occurred, when the loss event rate is zero, the Loss Event Rate option value is set to "11111111111111111111111111111111" in binary (or equivalently, to 2^32 - 1). The loss event rate calculation uses loss interval data lengths, as defined in Section 6.1.1.

Loss Event Rate options MUST NOT be sent on DCCP-Data packets, and any Loss Event Rate options on received DCCP-Data packets MUST be ignored.

8.6. Loss Intervals Option

```
+--------+--------+--------+--------...--------+--------+---
|11000001| Length |  Skip  |    Loss Interval    | More Loss
|        |        | Length |                     | Intervals...
+--------+--------+--------+--------...--------+--------+---
 Type=193                            9 bytes
```

Each 9-byte Loss Interval contains three fields, as follows:

```
  ____________________ Loss Interval _____________________
 /                                                        \
+--------...-------+--------...--------+--------...--------+
| Lossless Length  |E|   Loss Length   |    Data Length    |
+--------...-------+--------...--------+--------...--------+
       3 bytes            3 bytes               3 bytes
```

The receiver reports its observed loss intervals using a Loss Intervals option. (Section 6.1 defines loss intervals.) This option MUST be sent by the data receiver on all required acknowledgements. The option reports up to 28 loss intervals seen by the receiver (although TFRC currently uses at most the latest 9 of these). This lets the sender calculate a loss event rate and probabilistically verify the receiver's ECN Nonce Echo.

The Loss Intervals option serves several purposes.

- The sender can use the Loss Intervals option to calculate the Loss Event Rate.

- Loss Intervals information is easily checked for consistency against previous Loss Intervals options, and against any Loss Event Rate calculated by the receiver.
- The sender can probabilistically verify the ECN Nonce Echo for each Loss Interval, reducing the likelihood of misbehavior.

Loss Intervals options MUST NOT be sent on DCCP-Data packets, and any Loss Intervals options on received DCCP-Data packets MUST be ignored.

8.6.1. Option Details

The Loss Intervals option contains information about between one and 28 consecutive loss intervals, always including the most recent loss interval. Intervals are listed in reverse chronological order. Should more than 28 loss intervals need to be reported, then multiple Loss Intervals options can be sent; the second option begins where the first left off, and so forth. The options MUST contain information about at least the most recent NINTERVAL + 1 = 9 loss intervals unless (1) there have not yet been NINTERVAL + 1 loss intervals, or (2) the receiver knows, because of the sender's acknowledgements, that some previously-transmitted loss interval information has been received. In this second case, the receiver need not send loss intervals that the sender already knows about, except that it MUST transmit at least one loss interval regardless. The NINTERVAL parameter is equal to "n" as defined in RFC 3448 (Section 5.4).

Loss interval sequence numbers are delta-encoded starting from the Acknowledgement Number. Therefore, Loss Intervals options MUST NOT be sent on packets without an Acknowledgement Number.

The first byte of option data is Skip Length, which indicates the number of packets up to and including the Acknowledgement Number that are not part of any Loss Interval. As discussed above, Skip Length must be less than or equal to NDUPACK = 3.

Loss Interval structures follow Skip Length. Each Loss Interval consists of a Lossless Length, a Loss Length, an ECN Nonce Echo (E), and a Data Length.

Lossless Length, a 24-bit number, specifies the number of packets in the loss interval's lossless part.

Loss Length, a 23-bit number, specifies the number of packets in the loss interval's lossy part. The sum of the Lossless Length and the Loss Length equals the loss interval's sequence length. Receivers SHOULD report the minimum valid Loss Length for each loss interval, making the first and last sequence numbers in each lossy part correspond to lost or marked data packets.

The ECN Nonce Echo, stored in the high-order bit of the 3-byte field containing Loss Length, equals the one-bit sum (exclusive-or, or parity) of data packet nonces received over the loss interval's lossless part (which is Lossless Length packets long). If Lossless Length is 0, the receiver is ECN-incapable, or the Lossless Length contained no data packets, then the ECN Nonce Echo MUST be reported as 0.

Finally, Data Length, a 24-bit number, specifies the loss interval's data length, as defined in Section 6.1.1.

8.6.2. Example

Consider the following sequence of packets, where "-" represents a safely delivered packet and "*" represents a lost or marked packet.

```
Sequence
 Numbers: 0         10        20        30        40  44
          |         |         |         |         |   |
          ----------*--------***-*--------*----------*-
```

Assuming that packet 43 was lost, not marked, this sequence might be divided into loss intervals as follows:

```
          0         10        20        30        40  44
          |         |         |         |         |   |
          ----------*--------***-*--------*----------*-
          \________/\_______/\___________/\_________/
              L0        L1         L2          L3
```

A Loss Intervals option sent to acknowledge this set of loss intervals, on a packet with Acknowledgement Number 44, might contain the bytes 193,39,2, 0,0,10, 128,0,1, 0,0,10, 0,0,8, 0,0,5, 0,0,10, 0,0,8, 0,0,1, 0,0,8, 0,0,10, 128,0,0, 0,0,15. This option is interpreted as follows.

193 The Loss Intervals option number.

39 The length of the option, including option type and length bytes. This option contains information about (39 - 3)/9 = 4 loss intervals.

2 The Skip Length is 2 packets. Thus, the most recent loss interval, L3, ends immediately before sequence number 44 - 2 + 1 = 43.

0,0,10, 128,0,1, 0,0,10
These bytes define L3. L3 consists of a 10-packet lossless part (0,0,10), preceded by a 1-packet lossy part. Continuing to subtract, the lossless part begins with sequence number 43 - 10 = 33, and the lossy part begins with sequence number 33 - 1 = 32. The ECN Nonce Echo for the lossless part, namely packets 33 through 42, inclusive, equals 1. The interval's data length is 10, so the receiver believes that the interval contained exactly one non-data packet.

0,0,8, 0,0,5, 0,0,10
This defines L2, whose lossless part begins with sequence number 32 - 8 = 24; whose lossy part begins with sequence number 24 - 5 = 19; whose ECN Nonce Echo (for packets [24,31]) equals 0; and whose data length is 10.

0,0,8, 0,0,1, 0,0,8
L1's lossless part begins with sequence number 11, its lossy part begins with sequence number 10, its ECN Nonce Echo (for packets [11,18]) equals 0, and its data length is 8.

0,0,10, 128,0,0, 0,0,15
L0's lossless part begins with sequence number 0, it has no lossy part, its ECN Nonce Echo (for packets [0,1]) equals 1, and its data length is 15. (This must be the first loss interval in the connection; otherwise, a data length greater than the sequence length would be invalid.)

9. Verifying Congestion Control Compliance With ECN

The sender can use Loss Intervals options' ECN Nonce Echoes (and possibly any Ack Vectors' ECN Nonce Echoes) to probabilistically verify that the receiver is correctly reporting all dropped or marked packets. Even if ECN is not used (the receiver's ECN Incapable feature is set to one), the sender could still check on the receiver by occasionally not sending a packet, or sending a packet out-of-order, to catch the receiver in an error in Loss Intervals or Ack Vector information. This is not as robust or as non-intrusive as the verification provided by the ECN Nonce, however.

9.1. Verifying the ECN Nonce Echo

To verify the ECN Nonce Echo included with a Loss Intervals option, the sender maintains a table with the ECN nonce sum for each data packet. As defined in RFC 3540, the nonce sum for sequence number S is the one-bit sum (exclusive-or, or parity) of data packet nonces over the sequence number range [I,S], where I is the initial sequence number. Let NonceSum(S) represent this nonce sum for sequence number S, and let NonceSum(I - 1) equal 0. Then the Nonce Echo for a loss interval [Left Edge, Left Edge + Offset) should equal the following one-bit sum:

```
NonceSum(Left Edge - 1) + NonceSum(Left Edge + Offset - 1).
```

Since an ECN Nonce Echo is returned for the lossless part of each Loss Interval, a misbehaving receiver -- meaning a receiver that reports a lost or marked data packet as "received non-marked", to avoid rate reductions -- has only a 50% chance of guessing the correct Nonce Echo for each loss interval.

To verify the ECN Nonce Echo included with an Ack Vector option, the sender maintains a table with the ECN nonce value sent for each packet. The Ack Vector option explicitly says which packets were received non-marked; the sender just adds up the nonces for those packets using a one-bit sum, and compares the result to the Nonce Echo encoded in the Ack Vector's option type. Again, a misbehaving receiver has only a 50% chance of guessing an Ack Vector's correct Nonce Echo. [DCCP] (Appendix A) describes this further. Alternatively, an Ack Vector's ECN Nonce Echo may also be calculated from a table of ECN nonce sums, rather than ECN nonces. If the Ack Vector contains many long runs of non-marked, non-dropped packets, the nonce sum-based calculation will probably be faster than a straightforward nonce-based calculation.

Note that Ack Vector's ECN Nonce Echo is measured over both data packets and non-data packets, while the Loss Intervals option reports ECN Nonce Echoes for data packets only.

9.2. Verifying the Reported Loss Intervals and Loss Event Rate

Besides probabilistically verifying the ECN Nonce Echoes reported by the receiver, the sender may also verify the loss intervals and any loss event rate reported by the receiver, if it so desires. Specifically, the Loss Intervals option explicitly reports the size of each loss interval as seen by the receiver; the sender can verify that the receiver is not falsely combining two loss events into one reported Loss Interval by using saved window counter information. The sender can also compare any Loss Event Rate option to the loss event rate it calculates using the Loss Intervals option.

We note that in some cases the loss event rate calculated by the sender could differ from an explicit Loss Event Rate option sent by the receiver. In particular, when a number of successive packets are dropped, the receiver does not know the sending times for these packets, and interprets these losses as a single loss event. In contrast, if the sender has saved the sending times or window counter information for these packets, then the sender can determine if these losses constitute a single loss event, or several successive loss events. Thus, with its knowledge of the sending times of dropped packets, the sender is able to make a more accurate calculation of the loss event rate. These kinds of differences SHOULD NOT be misinterpreted as attempted receiver misbehavior.

10. Implementation Issues

10.1. Timestamp Usage

CCID 3 data packets need not carry Timestamp options. The sender can store the times at which recent packets were sent; the Acknowledgement Number and Elapsed Time option contained on each required acknowledgement then provide sufficient information to compute the round trip time. Alternatively, the sender MAY include Timestamp options on a limited subset of its data packets. The receiver will respond with Timestamp Echo options including Elapsed Times, allowing the sender to calculate round-trip times without storing timestamps at all.

10.2. Determining Loss Events at the Receiver

The window counter is used by the receiver to determine if multiple lost packets belong to the same loss event. The sender increases the window counter by one every quarter round-trip time. This section describes in detail the procedure for using the window counter to determine when two lost packets belong to the same loss event.

Section 3.2.1 of RFC 3448 specifies that each data packet contains a timestamp, and gives as an alternative implementation a "timestamp" that is incremented every quarter of an RTT, as is the window counter in CCID 3. However, Section 5.2 in RFC 3448 on "Translation from Loss History to Loss Events" is written in terms of timestamps, not in terms of window counters. In this section, we give an procedure for the translation from loss history to loss events that is explicitly in terms of window counters.

To determine whether two lost packets with sequence numbers X and Y belong to different loss events, the receiver proceeds as follows. Assume Y > X in circular sequence space.

- Let X_prev be the greatest valid sequence number received with X_prev < X.
- Let Y_prev be the greatest valid sequence number received with Y_prev < Y.
- Given a sequence number N, let C(N) be the window counter value associated with that packet.
- Packets X and Y belong to different loss events if there exists a packet with sequence number S so that X_prev < S <= Y_prev, and the distance from C(X_prev) to C(S) is greater than 4. (The distance is the number D so that C(X_prev) + D = C(S) (mod WCTRMAX), where WCTRMAX is the maximum value for the window counter -- in our case, 16.)

  That is, the receiver only considers losses X and Y as separate loss events if there exists some packet S received between X and Y, with the distance from C(X_prev) to C(S) greater than 4. This complex calculation is necessary to handle the case where window counter space wrapped completely between X and Y. Generally, the receiver can simply check whether the distance from C(X_prev) to C(Y_prev) is greater than 4; if so, then X and Y belong to separate loss events.

Window counters can help the receiver to disambiguate multiple losses after a sudden decrease in the actual round-trip time. When the sender receives an acknowledgement acknowledging a data packet with window counter *i*, the sender increases its window counter, if necessary, so that subsequent data packets are sent with window counter values of at least *i*+4. This can help minimize errors on the part of the receiver of incorrectly interpreting multiple loss events as a single loss event.

We note that if all of the packets between X and Y are lost in the network, then X_prev and Y_prev are both set to X-1, and the series of consecutive losses is treated by the receiver as a single loss event. However, the sender will receive no DCCP-Ack packets during a period of consecutive losses, and the sender will reduce its sending rate accordingly.

As an alternative to the window counter, the sender could have sent its estimate of the round-trip time to the receiver directly in a round-trip time option; the receiver would use the sender's round-trip time estimate to infer when multiple lost or marked packets belong in the same loss event. In some respects, a round-trip time option would give a more precise encoding of the sender's round-trip time estimate than does the window counter. However, the window counter conveys information about the relative *sending* times for packets, while the receiver could only use the round-trip time option to distinguish between the relative *receive* times (in the absence of timestamps). That is, the window counter will give more robust performance when there is a large variation in delay for packets sent within a window of data. Slightly more speculatively, a round-trip time option might possibly be used more easily by middleboxes attempting to verify that a flow was using conformant end-to-end congestion control.

10.3. Sending Feedback Packets

In CCID 3, the window counter is used by the receiver to decide when to send feedback packets. RFC 3448 (Sections 6.1 and 6.2) specifies that the TFRC receiver sends a feedback packet when the new loss event rate p is less that the old value. This rule is followed by CCID 3.

In addition, RFC 3448 (Section 6.2) specifies that the receiver uses a feedback timer to decide when to send additional feedback packets. If the feedback timer expires, and data packets have been received since the previous feedback was sent, then the receiver sends a feedback packet. When the feedback timer expires, the receiver resets the timer to expire after R_m seconds, where R_m is the most recent estimate of the round-trip time received from the sender. This section describes how CCID 3 uses the window counter instead of the feedback timer to determine when to send additional feedback packets.

Whenever the receiver sends a feedback message, the receiver sets a local variable last_counter to the greatest received value of the window counter since the last feedback message was sent, if any data packets have been received since the last feedback message was sent. If the receiver receives a data packet with a window counter value greater than or equal to last_counter + 4, then the receiver sends a new feedback packet. ("Greater" and "greatest" are measured in circular window counter space.)

This procedure ensures that when the sender is sending less than one packet per round-trip time, then the receiver sends a feedback packet after each data packet. Similarly, this procedure ensures that when the sender is sending several packets per round-trip time, then the receiver will send a feedback packet each time that a data packet arrives with a window counter more than four greater than the window counter when the last feedback packet was sent. Thus, the feedback timer is not necessary when the window counter is used.

However, the feedback timer still could be useful in some rare cases to prevent the sender from unnecessarily halving its sending rate. In particular, one could construct scenarios where the use of the feedback timer at the receiver would prevent the unnecessary expiration of the nofeedback timer at the sender. Consider the case below, in which a feedback packet is sent when a data packet arrives with a window counter of K.

```
Window
Counters: K   K+1 K+2 K+3 K+4 K+5 K+6  ...  K+15 K+16 K+17 ...
          |   |   |   |   |   |   |         |    |    |
Data      |   |   |   |   |   |   |         |    |    |
Packets   |   |   |   |   |   |   |         |    |    |
Received:   - -  ---  -                ...   - - -- -  -- --  -
            |                |               |    |    |        |
            |                |               |    |    |        |
Events:     1:               2:              3:   4:   5:       6:
           "A"                              "B"  Timer "B"
           sent                             sent       received

     1:  Feedback message A is sent.
     2:  A feedback message would have been sent if feedback timers
         had been used.
     3:  Feedback message B is sent.
     4:  Sender's nofeedback timer expires.
     5:  Feedback message B is received at the sender.
     6:  Sender's nofeedback timer would have expired if feedback
         timers had been used, and the feedback message at 2 had
         been sent.
```

The receiver receives data after the feedback packet has been sent, but has received no data packets with a window counter between K+4 and K+14. A data packet with a window counter of K+4 or larger would have triggered sending a new feedback packet, but no feedback packet is sent until time 3.

The TFRC protocol specifies that after a feedback packet is received, the sender sets a nofeedback timer to at least four times the round-trip time estimate. If the sender doesn't receive any feedback packets before the nofeedback timer expires, then the sender halves its sending rate. In the figure, the sender receives feedback message A (time 1), then sets the nofeedback timer to expire roughly four round-trip times later (time 4). The sender starts sending again just before the nofeedback timer expires, but doesn't receive the resulting feedback message until after its expiration, resulting in an unnecessary halving of the sending rate. If the connection had used feedback timers, the receiver would have sent a feedback message when the feedback timer expired at time 2, and the halving of the sending rate would have been avoided.

For implementors who wish to implement a feedback timer for the data receiver, we suggest estimating the round-trip time from the most recent data packet as described in Section 8.1. We note that this procedure does not work when the inter-packet sending times are greater than the RTT.

11. Security Considerations

Security considerations for DCCP have been discussed in [DCCP], and security considerations for TFRC have been discussed in RFC 3448 (Section 9). The security considerations for TFRC include the need to protect against spoofed feedback, and the need for protection mechanisms to protect the congestion control mechanisms against incorrect information from the receiver.

In this document we have extensively discussed the mechanisms the sender can use to verify the information sent by the receiver. As the document described, ECN may be used with CCID 3. When ECN is used, the receiver must use either Ack Vector or Loss Intervals to return ECN Nonce information to the sender. When ECN is not used, then, as Section 9 shows, the sender could still use various techniques that might catch the receiver in an error in reporting congestion, but this is not as robust or as non-intrusive as the verification provided by the ECN Nonce.

12. IANA Considerations

This specification defines the value 3 in the DCCP CCID namespace managed by IANA. This assignment is also mentioned in [DCCP]. CCID 3 also introduces three sets of numbers whose values should be allocated by IANA. We refer to allocation policies, such as Standards Action, outlined in RFC 2434, and most registries reserve some values for experimental and testing use [RFC 3692].

12.1. Reset Codes

Each entry in the DCCP CCID 3 Reset Code registry contains a CCID 3-specific Reset Code, which is a number in the range 128-255; a short description of the Reset Code; and a reference to the RFC defining the Reset Code. Reset Codes 184-190 and 248-254 are permanently reserved for experimental and testing use. The remaining Reset Codes -- 128-183, 191-247, and 255 -- are currently reserved, and should be allocated with the IETF Consensus policy, which requires RFC publication (not necessarily standards-track).

12.2. Option Types

Each entry in the DCCP CCID 3 option type registry contains a CCID 3-specific option type, which is a number in the range 128-255; the name of the option, such as "Loss Intervals"; and a reference to the RFC defining the option type. The registry is initially populated using the values in Table 1, in Section 8. This document allocates option types 192-194, and option types 184-190 and 248-254 are permanently reserved for experimental and testing use. The remaining option types -- 128-183, 191, 195-247, and 255 -- are currently reserved, and should be allocated with the IETF Consensus policy, which requires RFC publication (not necessarily standards-track).

12.3. Feature Numbers

Each entry in the DCCP CCID 3 feature number registry contains a CCID 3-specific feature number, which is a number in the range 128-255; the name of the feature, such as "Send Loss Event Rate"; and a reference to the RFC defining the feature number. The registry is initially populated using the values in Table 2, in Section 8. This document allocates feature number 192, and feature numbers 184-190 and 248-254 are permanently reserved for experimental and testing use. The remaining feature numbers -- 128-183, 191, 193-247, and 255 -- are currently reserved, and should be allocated with the IETF Consensus policy, which requires RFC publication (not necessarily standards-track).

13. Thanks

We thank Mark Handley for his help in defining CCID 3. We also thank Mark Allman, Aaron Falk, Ladan Gharai, Sara Karlberg, Greg Minshall, Arun Venkataramani, David Vos, Yufei Wang, Magnus Westerlund, and members of the DCCP Working Group for feedback on versions of this document.

A. Appendix: Possible Future Changes to CCID 3

There are a number of cases where the behavior of TFRC as specified in RFC 3448 does not match the desires of possible users of DCCP. These include the following:

1. The initial sending rate of at most four packets per RTT, as specified in RFC 3390.
2. The receiver's sending of an acknowledgement for every data packet received, when the receiver receives less than one packet per round-trip time.
3. The sender's limitation of at most doubling the sending rate from one round-trip time to the next (or more specifically, of limiting the sending rate to at most twice the reported receive rate over the previous round-trip time).
4. The limitation of halving the allowed sending rate after an idle period of four round-trip times (possibly down to the initial sending rate of two to four packets per round-trip time).
5. Another change that is needed is to modify the response function used in RFC 3448 (Section 3.1) to match more closely the behavior of TCP in environments with high packet drop rates [RFC 3714].

One suggestion for higher initial sending rates is that of an initial sending rate of up to eight small packets per RTT, when the total packet size, including headers, is at most 4380 bytes. Because the packets would be rate-paced out over a round-trip time, instead of sent back-to-back as they would be in TCP, an initial sending rate of eight small packets per RTT with TFRC-based congestion control would be considerably milder than the impact of an initial window of eight small packets sent back-to-back in TCP. As Section 5.1 describes, the initial sending rate also serves as a lower bound for reductions of the allowed sending rate during an idle period.

We note that with CCID 3, the sender is in slow-start in the beginning, and responds promptly to the report of a packet loss or mark. However, in the absence of feedback from the receiver, the sender can maintain its old sending rate for up to four round-trip times. One possibility would be that for an initial window of eight small packets, the initial nofeedback timer would be set to two round-trip times instead of four, so that the sending rate would be reduced after two round-trips without feedback.

Research and engineering will be needed to investigate the pros and cons of modifying these limitations in order to allow larger initial sending rates, to send fewer acknowledgements when the data sending rate is low, to allow more abrupt changes in the sending rate, or to allow a higher sending rate after an idle period.

Normative References

[DCCP] E. Kohler, M. Handley, and S. Floyd. Datagram Congestion Control Protocol, draft-ietf-dccp-spec-09.txt, work in progress, November 2004.

[RFC 2119] S. Bradner. Key Words For Use in RFCs to Indicate Requirement Levels. RFC 2119.

[RFC 2434] T. Narten and H. Alvestrand. Guidelines for Writing an IANA Considerations Section in RFCs. RFC 2434.

[RFC 2581] M. Allman, V. Paxson, and W. Stevens. TCP Congestion Control. RFC 2581.

[RFC 3168] K.K. Ramakrishnan, S. Floyd, and D. Black. The Addition of Explicit Congestion Notification (ECN) to IP. RFC 3168. September 2001.

[RFC 3390] M. Allman, S. Floyd, and C. Partridge. Increasing TCP's Initial Window. RFC 3390.

[RFC 3448] M. Handley, S. Floyd, J. Padhye, and J. Widmer, TCP Friendly Rate Control (TFRC): Protocol Specification, RFC 3448, Proposed Standard, January 2003.

[RFC 3692] T. Narten. Assigning Experimental and Testing Numbers Considered Useful. RFC 3692.

Informative References

[CCID 2 PROFILE] S. Floyd and E. Kohler. Profile for DCCP Congestion Control ID 2: TCP-like Congestion Control, draft-ietf-dccp-ccid2-08.txt, work in progress, November 2004.

[MAF04] A. Medina, M. Allman, and S. Floyd. Measuring Interactions Between Transport Protocols and Middleboxes. ACM SIGCOMM/USENIX Internet Measurement Conference, Sicily, Italy, October 2004. URL "http://www.icir.org/tbit/".

[RFC 3540] N. Spring, D. Wetherall, and D. Ely. Robust Explicit Congestion Notification (ECN) Signaling with Nonces. RFC 3540.

[RFC 3714] S. Floyd and J. Kempf, Editors. IAB Concerns Regarding Congestion Control for Voice Traffic in the Internet. RFC 3714.

[V03] Arun Venkataramani, August 2003. Citation for acknowledgement purposes only.

Authors' Addresses

Sally Floyd <floyd@icir.org>
ICSI Center for Internet Research
1947 Center Street, Suite 600
Berkeley, CA 94704
USA

Eddie Kohler <kohler@cs.ucla.edu>
4531C Boelter Hall
UCLA Computer Science Department
Los Angeles, CA 90095
USA

Jitendra Padhye <padhye@microsoft.com>
Microsoft Research
One Microsoft Way
Redmond, WA 98052
USA

Full Copyright Statement

Intellectual Property