Sally Floyd
ICIR
Eddie Kohler
UCLA
Jitendra Padhye
Microsoft Research
18 July 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-ccid3-05.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

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. (A half-connection consists of data packets sent from DCCP A to DCCP B, plus acknowledgements sent from DCCP B to DCCP A. DCCP A is the HC-Sender, and DCCP B the HC-Receiver, for this half-connection. In this document, we abbreviate HC-Sender and HC-Receiver as "sender" and "receiver", respectively. These terms are defined more fully in [DCCP].)

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.

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 buffering at the receive application before the playback time. CCID 2, TCP-like congestion control [CCID 2 PROFILE], which halves the sending rate in response to a congestion event, cannot satisfy a preference for a relatively smooth sending rate.

As explained in [RFC 3448], 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.

As described in the TFRC specification [RFC 3448], 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 TFRC Congestion Control specified by CCID 3, not including connection initiation and termination. The example is informative, not normative.

1. The sender sends DCCP-Data packets, where the sending rate is governed by the allowed transmit rate, as specified in [RFC 3448]. 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.

   If the use of Explicit Congestion Notification (ECN) has been negotiated, 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]. Each DCCP-Ack packet uses a sequence number and identifies the most recent packet received from the sender. Each DCCP-Ack packet includes feedback about the loss event rate calculated 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]. This update is based upon the loss event rate reported by the receiver and the round-trip time estimated at the sender. If it prefers, the sender can also calculate the loss event rate itself, based on information provided by the receiver.

4. The sender estimates round-trip times and calculates a nofeedback time, as specified in [RFC 3448]. 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, Send Loss Intervals, and Send Loss Event Rate features.

CCID 3 requires CCID-specific feedback from the receiver, and thus MUST NOT masquerade as CCID 1.

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; that RFC should be considered normative except where specifically indicated.

The sender starts in a slow-start phase, roughly doubling its allowed sending rate each round-trip time. After the slow-start phase is ended by the receiver's report of a packet drop or mark, the sender calculates the allowed sending rate based on the round-trip time and on the loss event rate or equivalent information reported by the receiver.

The feedback packets from the receiver contain a Receive Rate option specifying the rate at which data packets were received by the receiver since the last feedback packet. The allowed sending rate can be at most twice the rate that the receiver received in the last round-trip time.

RFC 3448 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. However, 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 [RFC 2581], and RFC 3390 allows an initial TCP window of three or four segments (up to 4380 bytes) [RFC 3390]. Therefore, following RFC 3390, 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 or Timestamp Echo option contained in feedback packets. The sender maintains an average round-trip time heavily weighted on the most recent measurements.

As stated earlier, the slow-start phase ends when the sender receives a report of a packet drop or mark. Each DCCP-Data packet contains a sequence number. Each DCCP-Data packet also contains a Window Counter Value, as described in Section 6.1 below. The Window Counter Value is incremented by one every quarter round-trip time, and is used by the receiver in the calculation of the loss event rate. In particular, the Window Counter Value is used by the receiver as a coarse-grained timestamp to determine when a packet loss should be considered part of an existing loss event.

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, the sender sets a nofeedback timer to at least four round-trip times, or to twice the interval between data packets, whichever is larger. RFC 3448 specifies that 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.

If the sender never receives a feedback packet from the receiver, the sending rate is left at its initialized 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.

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. As specified in RFC 3448, the allowed sending rate is never reduced to less than two packets per round-trip time as the result of an idle period.

Revising this to take into account the larger initial windows allowed by RFC 3390, the allowed sending rate should not be reduced to less than the initial sending rate during the idle period. That is, 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. But 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.

RFC 3448 also specifies that the sender's allowed sending rate is limited to at most twice the receive rate reported by the receiver. As a consequence, 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 round-trip time.
- **Adjusting the receive rate X_recv.** A CCID 3 sender SHOULD also respond to non-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 RFC 3448's throughput calculations. 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 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, defined in [DCCP] as a request that the sender not increase its sending rate for roughly a round-trip time, then X_drop should be set to X_inrecv. Similarly, if the sender receives a Data Dropped option indicating that three packets were dropped with Drop Code 2, then the upper bound on the sending rate will be decreased by three, with the sender setting X_drop to X_inrecv - 3*s, for s 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-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.
- **Exiting slow-start.** The sender MUST also exit slow start whenever it receives a relevant Data Dropped or Slow Receiver option.

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. For this, a CCID 3 implementation MAY use 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 8.7.1.

Alternately, a CCID 3 implementation MAY use the Maximum Packet Size to derive the packet size "s" is used in the TCP throughput equation. In this case, the packet size "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. In this case, each packet transmitted counts as one MSS, regardless of the actual segment size. In this case, 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. However, 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.

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].

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 receiver received the packet whose sequence number appears in the Acknowledgement Number field. These options are described in Sections 13.2 and 13.1 of [DCCP].
2. A Receive Rate option (Section 8.3) specifying the rate at which the receiver received data since the last DCCP-Ack was sent.
3. One or more options concerning the loss event rate p experienced by the receiver, as described in [RFC 3448]. Relevant options include Loss Event Rate, which gives the loss event rate calculated by the receiver (Section 8.5); Loss Intervals, which specifies the beginning and end of each loss interval, from which the sender can easily calculate and/or verify the loss event rate (Section 8.7); and Ack Vector, which says exactly which packets were lost or marked, again allowing the sender to calculate and/or verify the loss event rate (see Section 11.4 of [DCCP]).

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. Congestion Control on Acknowledgements

The rate and timing for generating acknowledgements is determined by the TFRC algorithm [RFC 3448]. The sending rate for acknowledgements is relatively low, and there is no explicit congestion control on the acknowledgements.

6.2. 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.

When Ack Vector is used, therefore, 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 relevant 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.

6.3. Quiescence

This section refers to quiescence in the DCCP sense (see section 8.1 of [DCCP]): How does a CCID 3 receiver determine that the corresponding sender is not sending any data?

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

Explicit Congestion Notification (ECN) [RFC 3168] MAY be used with CCID 3. If ECN is enabled, then the ECN Nonce will automatically be used following the specification for the ECN Nonce for TCP [RFC 3540]. For the data sub-flow, the sender sets either the ECT[0] or ECT[1] codepoint on DCCP-Data packets.

If ECN is used, then the receiver MUST use at least one of Ack Vector and Loss Intervals to return ECN Nonce information to the sender.

If the Ack Vector option is being used, then it will include the ECN Nonce Sum. The sender can maintain a table with the ECN nonce sum for each packet, and use this information to probabilistically verify the ECN nonce sum returned in each DCCP-Ack packet, as described in Appendix A of [DCCP].

If the Ack Vector option is not being used, the information about the ECN Nonce is returned by the receiver using the Loss Intervals option described below. In this case, an ECN-capable receiver MUST include this option on every required acknowledgement.

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 Capable features. In addition, the following CCID-specific options are defined for use with CCID 3:

Type	Option Length	Meaning	Section Reference
128-191		Reserved	
192	6	Loss Event Rate	8.5
193	6	Loss Intervals	8.7
194	6	Receive Rate	8.3
195-255		Reserved	

The following CCID-specific features are also defined. The Rec'n Rule column defines each feature's reconciliation rule; both are server-priority.

Number	Meaning	Rec'n Rule	Initial Value	Section Reference
128-191	Reserved			
192	Send Loss Event Rate	SP	1	8.4
193	Send Loss Intervals	SP	0	8.6
194-255	Reserved			

Although the use of Ack Vector, Loss Intervals, and Loss Event Rate are controlled by separate features, only some combinations of these features make sense. In particular, if ECN Capable is true, then every required acknowledgement MUST include at least one of Ack Vector and Loss Intervals; otherwise, every required acknowledgement MUST include at least one of Ack Vector, Loss Intervals, and Loss Event Rate. This may impel the receiver to send certain options even when their corresponding Send features are false. A sender that receives several invalid acknowledgements -- that include only Loss Event Rate on an ECN-capable connection, for example -- SHOULD respond by resetting the connection with Reason set to "Option Error".

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]. For reference, the DCCP generic header is as follows (diagram repeated from [DCCP]):

```
 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 |X| Type  |              Sequence Number                  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

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 for determining when multiple losses belong to a single loss event, determining the interval for calculating the receive rate, and determining when to send feedback packets. None of these procedures require an explicit estimate of the round-trip time by the receiver. However, for implementors who wish to have an estimate of the round-trip time at the receiver, we suggest estimating the round-trip time from received data packets as follows: Say that packets arrived as follows:

```
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. When estimating the round-trip time in this way, the receiver MUST limit itself to packet pairs whose CCVals differ by 2, 3, or 4 (representing intervals of 1/2, 3/4, and 1 RTT, respectively); differences of 4 SHOULD be preferred.

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. The elapsed time value is included in one of 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. Otherwise, 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. The Receive Rate is calculated as the number of bytes received in the most recent t seconds, divided by t, where t is the larger of the following: the time since the last Receive Rate Option was sent, and the estimated round-trip time. The receiver can use the Window Counter Value in received data packets to determine if an interval of t seconds corresponds to at least a round-trip time.

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 one.

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 packets per loss interval. See [RFC 3448] for a normative calculation of loss event rate.

8.6. Send Loss Intervals Feature

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

Send Loss Intervals has feature number 193, and is server-priority. It takes one-byte Boolean values. DCCP B MUST send Loss Intervals options on its acknowledgements when Send Loss Intervals/B is one, although it MAY send Loss Intervals options even when Send Loss Intervals/B is zero. Values of two or more are reserved. A CCID 3 half-connection starts with Send Loss Intervals equal to zero.

8.7. Loss Intervals Option

```
                              ___ Loss Interval ___
                             /                     \
+--------+--------+--------+----...----+----...----+--------+---
|11000001| Length |  Skip  | Lossless  |E|  Loss   | Up to 7 Loss
|        |        | Length |  Length   | | Length  | Intervals...
+--------+--------+--------+----...----+----...----+--------+---
 Type=193                     3 bytes     3 bytes
```

This option MAY be set by the data receiver on acknowledgements. (If ECN is enabled and Ack Vector is off, or if the Send Loss Intervals feature is true, it MUST be sent with every required acknowledgement.) The option reports up to 8 loss intervals seen by the receiver, allowing the sender to calculate a loss event rate and to probabilistically verify the receiver's ECN Nonce Echo.

8.7.1. Loss Interval Definition

As described in [RFC 3448] (Section 5.2), a loss interval begins with a lost or ECN-marked 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 packets. Call these the lossy part and the lossless part of the loss interval. 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
```

The Loss Event Rate, reported by option 192, is the weighted average of the last 8 loss interval lengths, inverted. Note that a loss interval's lossless part might be empty.

The length of the lossy part must be <= 1 RTT; however, if the packet that starts a loss interval was actually lost, the receiver cannot know its receive time. Section 5.2 of RFC 3448 gives a calculation whereby the receiver interpolates a likely receive time for each lost packet. CCID 3 implementations SHOULD use this calculation.

As a slightly simpler alternative, the receiver MAY instead calculate loss intervals using the receive time for the preceding data packet as an estimate for the receive time for a lost packet.

Note that a missing packet doesn't begin a new loss interval until 3 packets have been seen after the "hole" (see Section 5.1 of [RFC 3448]). Thus, up to three 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.

8.7.2. Option Details

The Loss Intervals option contains information about between one and eight consecutive loss intervals, always including the most recent loss interval. Intervals are listed in reverse chronological order. The option MUST contain information about the most recent 8 loss intervals unless (1) there have not yet been 8 loss intervals, in which case the receiver SHOULD send information about all the loss intervals it has experienced; or (2) the receiver knows, because of acknowledgements from the sender, that information about older loss intervals has been received by the sender, in which case the receiver MUST send at least information about the loss intervals the sender has not acknowledged. In any case, the Loss Intervals option MUST contain the most recent loss interval.

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 three.

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

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 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 nonces received over the loss interval's lossless part (which is Lossless Length packets long). If Lossless Length is 0, or if the receiver is ECN-incapable, the ECN Nonce Echo MUST be reported as 0.

The Loss Intervals option serves several purposes.

- The sender can use the Loss Intervals option to easily calculate the Loss Event Rate, perhaps using a later version of the TFRC algorithm than that deployed at the receiver.
- 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.

8.7.3. 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           L4
```

A Loss Intervals option sent to acknowledge this set of loss intervals, on a packet with Acknowledgement Number 44, might contain the bytes 193,33,2, 0,0,10, 128,0,1, 0,0,8, 0,0,5, 0,0,8, 0,0,1, 0,0,5, 128,0,3, 0,0,2, 128,0,0. This option is interpreted as follows.

193 The Loss Intervals option number.

33 The length of the option, including option type and length bytes. This option contains information about (33 - 3)/6 = 5 loss intervals.

2 The Skip Length is 2 packets. Thus, the most recent loss interval, L4, ends immediately before sequence number 44 - 2 + 1 = 43.

0,0,10, 128,0,1
These bytes define L4. L4 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.

0,0,8, 0,0,5
This defines L3, whose lossless part begins with sequence number 32 - 8 = 24; whose lossy part begins with sequence number 24 - 5 = 19; and whose ECN Nonce Echo (for packets [24,31]) equals 0.

0,0,8, 0,0,1
L2's lossless part begins with sequence number 11, its lossy part begins with sequence number 10, and its ECN Nonce Echo (for packets [11,18]) equals 0.

0,0,5, 128,0,3
L1's lossless part begins with sequence number 5, its lossy part begins with sequence number 2, and its ECN Nonce Echo (for packets [5,9]) equals 1.

0,0,2, 128,0,0
L0's lossless part begins with sequence number 0, it has no lossy part, and its ECN Nonce Echo (for packets [0,1]) equals 1.

9. Verifying Congestion Control Compliance With ECN

If ECN is used, the sender can use Ack Vector or the Loss Intervals option to probabilistically verify that the receiver is not lying in reporting packets received undropped and unmarked. The sender could then use the information in acknowledgement packets to roughly verify the Loss Event Rate reported by the receiver, if it so desired.

We note that if ECN is not used, 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 Ack Vector or Loss Intervals information. Similarly, the sender would still use the Ack Vector or Loss Intervals information to verify the loss event rate reported by the receiver. However, this is not as robust or as non-intrusive as the verification provided by the ECN Nonce.

9.1. Verifying the ECN Nonce Echo

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 (exclusive-or, or parity), and compares the result to the Nonce Echo encoded in the Ack Vector's option type.

To verify the ECN Nonce Echo included with a Loss Intervals option, the sender maintains a table with the ECN nonce *sum* for each packet. As defined in [RFC 3540], the nonce sum for sequence number S is the one-bit sum of 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).
```

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.

In either of these cases, a misbehaving receiver -- meaning a receiver that reports a lost or marked packet as "received non-marked", to avoid rate reductions -- has only a 50% chance of guessing the correct Nonce Echo.

9.2. Verifying the Reported Loss Event Rate

Once the sender has probabilistically verified the ECN Nonce Echoes reported by the receiver, the sender can calculate for itself the number of packets in each loss interval, to roughly verify the loss event rate reported by the receiver, if it so desires. We note that DCCP's Loss Event Rate Option reports the average loss interval size, which is the inverse of the loss event rate.

If the Ack Vector is used, the sender can identify the packet that begins each new loss interval from the Ack Vector in each DCCP-Ack packet. If the sender saves information about the window counter for each data packet, then the sender also can tell when two lost or marked packets would have been interpreted by the receiver as separate loss events.

The Loss Intervals option explicitly reports the size of each loss interval, as seen by the receiver. The sender can, using saved information about window counters, verify that the receiver is not falsely combining two loss events into one reported loss interval.

Once the sender has reconstructed or verified Loss Intervals, it can easily calculate the expected loss event rate, and compare against the receiver's reported loss event rate.

We note that in some cases the loss event rate calculated by the sender could differ from that calculated 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 the 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.

10. Design Considerations

CCID 3 data packets need not carry Timestamp options. The sender can store the times at which recent packets were sent. Then the Acknowledgement Number and Elapsed Time option contained on each required acknowledgement 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.

10.1. Possible Changes to the Initial Window

In the future, it is possible that an initial sending rate of up to eight small packets per RTT would be allowed, when the total packet size, including headers, was 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.

10.2. Other Possible Changes to TFRC

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 receiver's sending of an acknowledgement for every data packet received, when the receiver receives less than one packet per round-trip time.
2. 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).
3. The limitation of halving the allowed sending rate after an idle period of four round-trip times (possibly down to a sending rate of two packets per round-trip time).

For each of these limitations, research and engineering will be needed to investigate the pros and cons of modifying these limitations in order 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.

Another change that is needed is to modify the response function used in [RFC 3448], to match more closely the behavior of TCP in environments with high packet drop rates [RFC 3714].

10.3. 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 1 every quarter round trip time. To determine whether two lost packets, with sequence numbers X and Y (Y > X in circular sequence space), belong to different loss events, the receiver proceeds as follows:

- Let X_prev be the greatest sequence number which was received with X_prev < X.
- Let Y_prev be the greatest sequence number which was 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, and the receiver should 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 gives 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 in some cases when there is a large variation in delay for packets sent within a window of data. As a slightly more speculative consideration, a round-trip time option could possibly be used more easily by middleboxes attempting to verify that a flow was using conformant end-to-end congestion control.

10.4. Sending Feedback Packets

The window counter is also used by the receiver to decide when to send feedback packets. Feedback packets should normally be sent at least once per round-trip time, if the sender is sending at least one data packet per round-trip time. 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.)

RFC 3448 specifies that the receiver uses a feedback timer to decide when to send feedback packets. In the TFRC protocol, 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 by the receiver from the sender. However, when the window counter is used, the receiver can use its information in deciding when to send feedback packets.

When the sender is sending less than one packet per round-trip time, then the receiver sends a feedback packet after each data packet, and the feedback timer is not required. Similarly, 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, and again the feedback counter is not required. Similarly, the receiver always sends a feedback packet after the detection of a loss event. Thus, the feedback timer is not absolutely 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. Consider the case when the receiver receives data soon after the most recent feedback packet has been sent, but has received no data packets with a window counter sufficiently large to trigger sending a new feedback packet. 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. One could construct scenarios where the use of a feedback timer at the receiver would prevent the unnecessary expiration of the nofeedback timer at the sender.

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.

10.5. When Should Ack Vector And Loss Intervals Be Used?

If the use of ECN has not been negotiated, then the receiver is not required to use either Ack Vector or Loss Intervals. Essentially, in this case the sender is completely relying on the Loss Event Rate reported by the receiver. However, if Ack Vector or Loss Intervals is used in the non-ECN-Capable case, then the sender could test that the receiver is correctly reporting dropped and marked packets by conducting a test and skipping a packet in its transmissions.

In the common case, it is assumed that the use of ECN will be negotiated with CCID 3. However, it is possible that either the sender or the receiver will want to negotiate the use of CCID 3 without ECN, e.g., if there happens to be a known broken middlebox along the path that blocks the use of ECN in the IP packet header. (Note: we are not currently aware of the existence of any middleboxes blocking the use of ECN in the IP header.)

If ECN is used, then the receiver is required to use either Ack Vector or Loss Intervals to return ECN Nonce information to the sender. The Ack Vector returns more information about which packets were lost or marked during a loss event. The sender uses more computation and state for verifying receiver feedback with the Ack Vector than with Loss Intervals, because then it must reconstruct loss intervals from the Ack Vector. The Ack Vector also requires that the sender occasionally acknowledge the receiver's acknowledgements; this is optional with Loss Intervals.

11. Security Considerations

Security considerations for DCCP have been discussed in [DCCP], and security considerations for TFRC have been discussed in [RFC 3448]. 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.

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 the following three sets of numbers whose values should be allocated by IANA. Following the policies outlined in [RFC 2434], these sets of numbers are allocated through an IETF Consensus action, with the specified exceptions for experimental and testing use [RFC 3692].

- CCID 3-specific option numbers 128-183, 191, 195-247, and 255 are allocated through an IETF Consensus action. Option numbers 184-190 and 248-254 are reserved for experimental and testing use. This document assigns option numbers 192-194.
- CCID 3-specific feature numbers 128-183, 191, 194-247, and 255 are allocated through an IETF Consensus action. Feature numbers 184-190 and 248-254 are reserved for experimental and testing use. This document assigns feature numbers 192-193.
- CCID 3-specific Reset Codes 128-183, 191-247, and 255 are allocated through an IETF Consensus action. Reset Codes 184-190 and 248-254 are reserved for experimental and testing use.

13. Thanks

We thank Mark Handley for his help in defining CCID 3. We also thank Sara Karlberg, Greg Minshall, Arun Venkataramani, Yufei Wang, and Magnus Westerlund for feedback on earlier versions of this document.

Normative References

[DCCP] E. Kohler, M. Handley, and S. Floyd. Datagram Congestion Control Protocol, draft-ietf-dccp-spec-07.txt, work in progress, July 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 3168] K.K. Ramakrishnan, S. Floyd, and D. Black. The Addition of Explicit Congestion Notification (ECN) to IP. RFC 3168. September 2001.

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

[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-ccid2-06.txt, work in progress, July 2004.

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

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

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

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

ipr@ietf.org.