Internet Engineering Task Force
INTERNET-DRAFT
draft-padhye-dcp-ccid3-04.ps

Jitendra Padhye
Microsoft Research
Sally Floyd
Eddie Kohler
ICIR

19 June 2002
Expires: December 2002

Profile for DCCP Congestion Control ID 3: TFRC Congestion Control

Status of this Document

This document is an Internet-Draft and is in full conformance with all provisions of Section 10 of RFC2026.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

The list of current Internet-Drafts can be accessed at http://www.ietf.org/ietf/1id-abstracts.txt

The list of Internet-Draft Shadow Directories can be accessed at http://www.ietf.org/shadow.html.

Abstract

This document contains the profile for Congestion Control Identifier 3, TCP-friendly rate control (TFRC), in the Datagram Congestion Control Protocol (DCCP). DCCP implements a congestion-controlled unreliable datagram flow suitable for use by applications such as streaming media. The TFRC CCID is used by applications that want a TCP-friendly send rate, possibly with Explicit Congestion Notification (ECN), while minimizing abrupt rate changes.

Table of Contents

1. Introduction 3
1.1. Usage Scenario. 3
1.2. Example Half-Connection 3
2. Connection Establishment 4
3. Congestion Control on Data Packets 4
4. Acknowledgments 4
4.1. Congestion Control on Acknowledgments 4
4.2. Quiescence 5
4.3. Acknowledgments of Acknowledgments 5
5. Explicit Congestion Notification 5
6. Relevant Options and Features 5
6.1. Window counter option 5
6.2. Elapsed time option 5
6.3. Loss Event Rate Option 6
6.4. Receive Rate Option 6
6.5. ECN NONCE Option 6
7. Application Requirements 7
8. Design Considerations 7
8.1. Determining Loss Events 7
8.2. Sending Feedback Packets 8
9. Thanks 9
10. References 9
11. Authors' Addresses 10

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 uses Congestion Control Identifiers, or CCIDs, to specify the congestion control mechanism in use on a half-connection. (A half-connection might consist of data packets sent from DCCP A to DCCP B, plus acknowledgments sent from DCCP B to DCCP A. DCCP A is the sending DCCP, and DCCP B the acknowledging DCCP, for this half-connection.)

TFRC is a receiver-based congestion control mechanism that provides a TCP-friendly send rate, while minimizing abrupt rate changes [1].

The basic TFRC protocol is as follows. The sender sends a stream of data packets to the receiver at some rate. The receiver sends a feedback packet to the sender at least once every round-trip time. Based on the information contained in the feedback packets, the sender adjusts its sending rate in accordance with the TCP throughput equation [2], to maintain TCP-friendliness. If no feedback is received from the receiver in several round-trip times (four, in the current TFRC specification), the sender halves its sending rate.

The values of the round-trip time *RTT*, the loss event rate *p* and the base timeout value *TO* are needed by the sender to calculate the send rate using the TCP throughput equation. The sender calculates the values of *RTT* and *TO*, while the receiver calculates the value of *p*.

1.1. Usage Scenario

DCCP with TFRC congestion control is intended to provide congestion control for the flow of data packets from the server to the client for applications that do not require fully reliable data transmission, or that desire to implement reliability on top of DCCP. TFRC congestion control is appropriate for flows that would prefer to minimize abrupt changes in the sending rate.

1.2. Example Half-Connection

This example, taken from the main DCCP draft, is of a half-connection using TFRC Congestion Control specified by CCID 3. The "sender" is the HC-Sender, and the "receiver" is the HC-Receiver.

(1) The sender sends DCCP-Data packets, where the number of packets sent is governed by an allowed transmit rate, as specified in [1]. Each DCCP-Data packet has a sequence number and a window counter option.

One or more of these data packets are DCCP-DataAck packets acknowledging the data packet from the receiver, but for simplicity we will not discuss the half-connection of data from the receiver to the sender in this example.

(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 RTT, as indicated by the TFRC specification [1]. 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, as specified below.

(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 [1].

(4) The sender estimates round-trip times and calculates a TimeOut value *TO* as specified in [1].

(5) If the use of 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 below.

2. Connection Establishment

The connection is initiated by the client using mechanisms described in the DCCP specification [3]. The client and the server MAY negotiate the use of the ACK Vector option. The ACK vector option is described in [3].

3. Congestion Control on Data Packets

The sender sends DCCP-Data packets to the receiver at the rate specified by the TCP throughput equation [2].

Each DCCP-Data packet has a sequence number, and an acknowledgment number that is the sequence number of the most recent acknowledgment packet received from the receiver. Each data packet contains the window counter option. The format of the window counter option is described below.

After each feedback packet is received from the receiver, the sender updates values of *RTT*, *TO* and the sending rate using procedures specified in [1].

If no feedback packet is received from the receiver after an interval specified in [1], the sending rate is halved. However, the sending rate is never reduced below one packet per 64 seconds. See [1] for more details.

4. Acknowledgments

The receiver sends a DCCP-Ack packet to the sender roughly once per round-trip time, if the sender is sending packets that frequently. This rate is determined by details of the TFRC protocol, as specified in [1].

The acknowledgment number in the DCCP-Ack packet acknowledges the most recent packet received from the sender. Each DCCP-Ack packet from the receiver includes the following options:

1. An option specifying the amount of time elapsed between since the receiver received the packet whose sequence number appears in the acknowledgment field.

2. An option specifying the loss event rate p calculated by the receiver as described in [1].

3. An option specifying the rate at which the receiver received data since the last DCCP-Ack was sent.

The format of these options is described below.

4.1. Congestion Control on Acknowledgments

The rate and timing for generating acknowledgments is determined by the TFRC algorithm [1]. The sending rate for acknowledgements is relatively low, and there is no explicit congestion control on the acknowledgements.

4.2. Quiescence

This section refers to quiescence in the DCCP sense (see section 6.1 of [3]): How does a CCID 3 receiver determine that the corresponding sender is not sending any data?

The receiver detects that the sender has gone quiescent after two round-trip times have passed without receiving any additional data. Since ACKs are not required to be reliable, the receiver needs to do nothing special in this case, unlike CCID 2 [5].

4.3. Acknowledgments of Acknowledgments

Acknowledgments in TFRC are entirely unreliable -- TFRC works even if every acknowledgment is dropped -- and it is never necessary for the sender to acknowledge an acknowledgment.

5. Explicit Congestion Notification

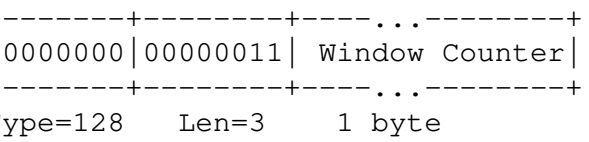
ECN [6] MAY be used with CCID 3. If ECN is used, then the ECN Nonce will automatically be used for the data packets, following the specification for the ECN Nonce [4] for TCP. For the data sub-flow, the sender sets either the ECT[0] or ECT[1] codepoint on DCCP-Data packets.

If the ACK vector option is being used, the ECN-NONCE information is returned via the ACK vector.

If the ACK vector option is not being used, the information about the ECN-NONCE is returned by the receiver using the ECN-NONCE option described below. In this case the receiver MUST return this option if it is reporting a lower packet loss rate than the one it reported in the previous acknowledgment.

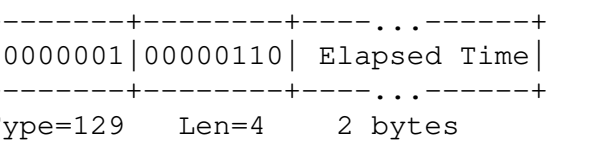
6. Relevant Options and Features

6.1. Window counter option



This option is set by the data sender on all data packets. The first byte gives the option type and the second gives the option length. The last byte gives the value of a counter which the sender sets to 0 at the beginning of the transmission, and increases by 1 every quarter of round trip time as described in [1].

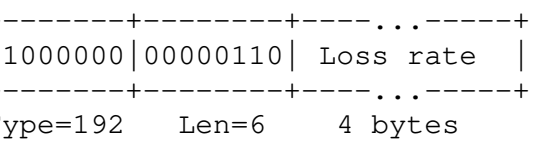
6.2. Elapsed time option



This option is set by the data receiver on all acknowledgment packets. The first byte gives

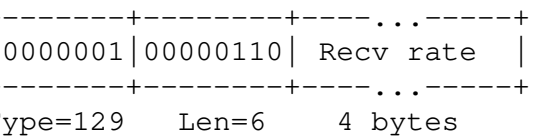
the option type and the second gives the option length. The last two bytes indicate the amount of time (in milliseconds) elapsed since the packet being acknowledged was received.

6.3. Loss Event Rate Option



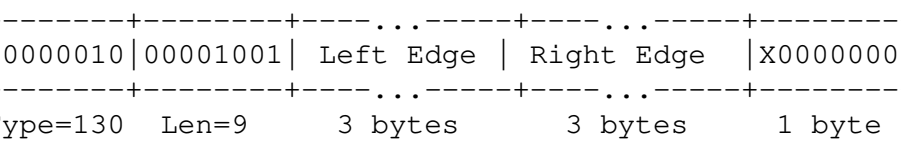
This option is set by the data receiver on all acknowledgment packets. The first byte gives the option type and the second gives the option length. The last four bytes indicate the inverse of the loss event rate, rounded UP, as calculated by the receiver.

6.4. Receive Rate Option



This option is set by the data receiver on all acknowledgment packets. The first byte gives the option type and the second gives the option length. The last four bytes indicate the rate at which the receiver has received data since it last sent an acknowledgment, in bits per second.

6.5. ECN NONCE Option



If ECN is used without the ACK vector option, then the ECN Nonce option is set by the data receiver on any acknowledgment packet that reports a loss rate lower than the loss rate reported in the previous acknowledgment packet. The first byte gives the option type and the second gives the option length. The right edge (RE) and the left edge (LE) are sequence numbers of data packets, such that:

- Let LastAck be the sequence number of the data packet acknowledged by the previous acknowledgment.

- If (LastAck + 1) was a dropped or marked packet, then RE should be the highest non-dropped and non-marked packet before (LastAck + 1).

- If (LastAck + 1) was not a dropped or marked packet, the RE should be the greatest sequence number such that all data packets between (LastAck + 1) and RE, inclusive, were received and not ECN-marked. Clearly (RE >= LastAck + 1).

- LE should be the smallest sequence number such that all data packets between LE and RE, inclusive, were received and not ECN-marked. Clearly (LE <= RE).

The first bit of the final byte is the Nonce Echo. It equals the base-2 modulus of the number of received ECN Nonce packets between LE and RE, both included.

Note that the interval [LE, RE] would be the largest non-loss interval containing the first packet received since the last report, or, if that was a dropped packet, containing the run before this drop. That is, [LE, RE] would continue to grow during non-drop and non-mark periods. Thus, for every loss event, the receiver reports the Nonce Echo for the consecutive sequence of packets received before the beginning of that loss event.

7. Application Requirements

As described in the TFRC specifications [1], this CCID should 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.

As it is presently specified, this CCID should only be used by senders that are willing to trust the receiver to report the correct loss event rate. If ECN is used, the ECN Nonce Option allows the sender to probabilistically verify the loss rate reported by the receiver. However, we have not specified such a verification procedure in this document.

8. Design Considerations

The data packets do not carry timestamps. The sender can store the times at which recent packets were sent. When an acknowledgement arrives, the acknowledgement number and the elapsed time option provide sufficient information to compute the round trip time.

8.1. Determining Loss Events

The window counter option 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), belong to different loss events, the receiver proceeds as follows:

- Let X_prev be the highest sequence number which was received with X_prev < X.

- Let Y_prev be the highest sequence number which was received with Y_prev < Y.

- Let CX_prev and CY_prev be the window counters associated with packets X_prev and Y_prev respectively. Clearly, CY_prev >= CX_prev.

- Packets X and Y belong to different loss events if (CY_prev - CX_prev) > 4

The use of the window counter option 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.

As an alternative to the window counter option, 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 option. However, the window counter option 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 option 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, the 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.

8.2. Sending Feedback Packets

The window counter option 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 highest 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 last_counter + 4, then the receiver sends a new feedback packet.

The TFRC protocol [1] 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 option is used, the receiver can use information from the window counter option 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. We have not considered this in detail. 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 follows: Let K be the window counter from the most recent data packet, and let T_k be the time that that packet was received. Let J be the highest window counter received that was less than K-4, and let T_j be the most recent time that such a packet was received. Then the round-trip time can be very roughly estimated as 4 (T_k-T_j)/(K-J).

9. Thanks

We thank Mark Handley for his help in defining CCID 3. We thank Sara Karlberg and Yufei Wang for feedback on an earlier version of this document.

10. References

[1] M. Handley, J. Padhye, and S. Floyd. TCP Friendly Rate Control (TFRC): Protocol Specification, draft-ietf-tsvwg-tfrc-04.txt, work in progress, April 2002.

[2] J. Padhye, V. Firoiu, D. Towsley, and J. Kurose. Modeling TCP Throughput: A Simple Model and its Empirical Validation. Proc ACM SIGCOMM 1998.

[3] E. Kohler, M. Handley, S. Floyd, and J. Padhye. Datagram Congestion Control Protocol, draft-kohler-dcp-02.txt, work in progress, March 2002.

[4] Neil Spring, David Wetherall, and David Ely. Robust ECN Signaling with Nonces, draft-ietf-tsvwg-tcp-nonce-03.txt, work in progress, April 2002.

[5] S. Floyd, E. Kohler. Profile for DCCP Congestion Control ID 2: TCP-like Congestion Control, draft-floyd-dcp-ccid2-03.txt, work in progress, May 2002.

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

11. Authors' Addresses

Jitendra Padhye <padhye@microsoft.com>

Microsoft Research
One Microsoft Way
Redmond, WA 98052 USA

Sally Floyd <floyd@icir.org>
Eddie Kohler <kohler@icir.org>

ICSI Center for Internet Research
1947 Center Street, Suite 600
Berkeley, CA 94704 USA