

IETF RMCAT Working Group
Internet-Draft
Intended status: Standards Track
Expires: May 6, 2021

Z. Sarker
Ericsson AB
C. Perkins
University of Glasgow
V. Singh
callstats.io
M. Ramalho
November 2, 2020

RTP Control Protocol (RTCP) Feedback for Congestion Control
draft-ietf-avtcore-cc-feedback-message-09

Abstract

An effective RTP congestion control algorithm requires more fine-grained feedback on packet loss, timing, and ECN marks than is provided by the standard RTP Control Protocol (RTCP) Sender Report (SR) and Receiver Report (RR) packets. This document describes an RTCP feedback message intended to enable congestion control for interactive real-time traffic using RTP. The feedback message is designed for use with a sender-based congestion control algorithm, in which the receiver of an RTP flow sends RTCP feedback packets to the sender containing the information the sender needs to perform congestion control.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <https://datatracker.ietf.org/drafts/current/>.

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

This Internet-Draft will expire on May 6, 2021.

Copyright Notice

Copyright (c) 2020 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	2
2. Terminology	3
3. RTCP Feedback for Congestion Control	3
3.1. RTCP Congestion Control Feedback Report	4
4. Feedback Frequency and Overhead	7
5. Response to Loss of Feedback Packets	8
6. SDP Signalling	8
7. Relation to RFC 6679	9
8. Design Rationale	10
9. Acknowledgements	11
10. IANA Considerations	11
11. Security Considerations	12
12. References	13
12.1. Normative References	13
12.2. Informative References	14
Authors' Addresses	15

1. Introduction

For interactive real-time traffic, such as video conferencing flows, the typical protocol choice is the Real-time Transport Protocol (RTP) [RFC3550] running over the User Datagram Protocol (UDP). RTP does not provide any guarantee of Quality of Service (QoS), reliability, or timely delivery, and expects the underlying transport protocol to do so. UDP alone certainly does not meet that expectation. However, the RTP Control Protocol (RTCP) [RFC3550] provides a mechanism by which the receiver of an RTP flow can periodically send transport and media quality metrics to the sender of that RTP flow. This information can be used by the sender to perform congestion control. In the absence of standardized messages for this purpose, designers of congestion control algorithms have developed proprietary RTCP messages that convey only those parameters needed for their respective designs. As a direct result, the different congestion control designs are not interoperable. To enable algorithm evolution as well as interoperability across designs (e.g., different rate

adaptation algorithms), it is highly desirable to have a generic congestion control feedback format.

To help achieve interoperability for unicast RTP congestion control, this memo proposes a common RTCP feedback packet format that can be used by NADA [RFC8698], SReAM [RFC8298], Google Congestion Control [I-D.ietf-rmcat-gcc] and Shared Bottleneck Detection [RFC8382], and hopefully also by future RTP congestion control algorithms.

2. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

In addition the terminology defined in [RFC3550], [RFC4585], and [RFC5506] applies.

3. RTCP Feedback for Congestion Control

Based on an analysis of NADA [RFC8698], SReAM [RFC8298], Google Congestion Control [I-D.ietf-rmcat-gcc] and Shared Bottleneck Detection [RFC8382], the following per-RTP packet congestion control feedback information has been determined to be necessary:

- o RTP sequence number: The receiver of an RTP flow needs to feed the sequence numbers of the received RTP packets back to the sender, so the sender can determine which packets were received and which were lost. Packet loss is used as an indication of congestion by many congestion control algorithms.
- o Packet Arrival Time: The receiver of an RTP flow needs to feed the arrival time of each RTP packet back to the sender. Packet delay and/or delay variation (jitter) is used as a congestion signal by some congestion control algorithms.
- o Packet Explicit Congestion Notification (ECN) Marking: If ECN [RFC3168], [RFC6679] is used, it is necessary to feed back the 2-bit ECN mark in received RTP packets, indicating for each RTP packet whether it is marked not-ECT, ECT(0), ECT(1), or ECN-CE. If the path used by the RTP traffic is ECN capable the sender can use Congestion Experienced (ECN-CE) marking information as a congestion control signal.

Every RTP flow is identified by its Synchronization Source (SSRC) identifier. Accordingly, the RTCP feedback format needs to group its reports by SSRC, sending one report block per received SSRC.

As a practical matter, we note that host operating system (OS) process interruptions can occur at inopportune times. Accordingly, recording RTP packet send times at the sender, and the corresponding RTP packet arrival times at the receiver, needs to be done with deliberate care. This is because the time duration of host OS interruptions can be significant relative to the precision desired in the one-way delay estimates. Specifically, the send time needs to be recorded at the last opportunity prior to transmitting the RTP packet at the sender, and the arrival time at the receiver needs to be recorded at the earliest available opportunity.

3.1. RTCP Congestion Control Feedback Report

Congestion control feedback can be sent as part of a regular scheduled RTCP report, or in an RTP/AVPF early feedback packet. If sent as early feedback, congestion control feedback MAY be sent in a non-compound RTCP packet [RFC5506] if the RTP/AVPF profile [RFC4585] or the RTP/SAVPF profile [RFC5124] is used.

Irrespective of how it is transported, the congestion control feedback is sent as a Transport Layer Feedback Message (RTCP packet type 205). The format of this RTCP packet is shown in Figure 1:

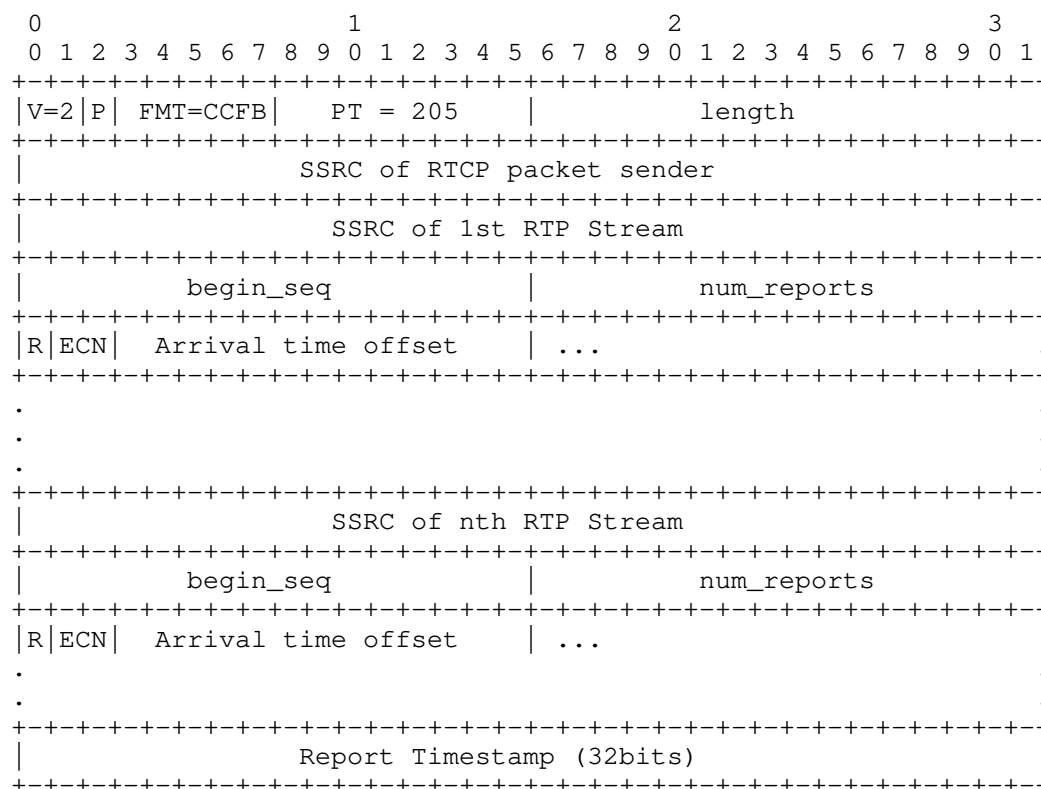


Figure 1: RTCP Congestion Control Feedback Packet Format

The first eight octets comprise a standard RTCP header, with PT=205 and FMT=CCFB indicating that this is a congestion control feedback packet, and with the SSRC set to that of the sender of the RTCP packet. (NOTE TO RFC EDITOR: please replace CCFB here and in the above diagram with the IANA assigned RTCP feedback packet type, and remove this note)

Section 6.1 of [RFC4585] requires the RTCP header to be followed by the SSRC of the RTP flow being reported upon. Accordingly, the RTCP header is followed by a report block for each SSRC from which RTP packets have been received, followed by a Report Timestamp.

Each report block begins with the SSRC of the received RTP Stream on which it is reporting. Following this, the report block contains a 16-bit packet metric block for each RTP packet with sequence number in the range begin_seq to begin_seq+num_reports inclusive (calculated using arithmetic modulo 65536 to account for possible sequence number wrap-around). If the number of 16-bit packet metric blocks included

in the report block is not a multiple of two, then 16 bits of zero padding MUST be added after the last packet metric block, to align the end of the packet metric blocks with the next 32 bit boundary. The value of num_reports MAY be zero, indicating that there are no packet metric blocks included for that SSRC. Each report block MUST NOT include more than 16384 packet metric blocks (i.e., it MUST NOT report on more than one quarter of the sequence number space in a single report).

The contents of each 16-bit packet metric block comprises the R, ECN, and ATO fields as follows:

- o Received (R, 1 bit): is a boolean to indicate if the packet was received. 0 represents that the packet was not yet received and the subsequent 15-bits (ECN and ATO) in this 16-bit packet metric block are also set to 0 and MUST be ignored. 1 represents that the packet was received and the subsequent bits in the block need to be parsed.
- o ECN (2 bits): is the echoed ECN mark of the packet. These are set to 00 if not received, or if ECN is not used.
- o Arrival time offset (ATO, 13 bits): is the arrival time of the RTP packet at the receiver, as an offset before the time represented by the Report Timestamp (RTS) field of this RTCP congestion control feedback report. The ATO field is in units of 1/1024 seconds (this unit is chosen to give exact offsets from the RTS field) so, for example, an ATO value of 512 indicates that the corresponding RTP packet arrived exactly half a second before the time instant represented by the RTS field. If the measured value is greater than 8189/1024 seconds (the value that would be coded as 0x1FFD), the value 0x1FFE MUST be reported to indicate an over-range measurement. If the measurement is unavailable, or if the arrival time of the RTP packet is after the time represented by the RTS field, then an ATO value of 0x1FFF MUST be reported for the packet.

The RTCP congestion control feedback report packet concludes with the Report Timestamp field (RTS, 32 bits). This denotes the time instant on which this packet is reporting, and is the instant from which the arrival time offset values are calculated. The value of RTS field is derived from the same clock used to generate the NTP timestamp field in RTCP Sender Report (SR) packets. It is formatted as the middle 32 bits of an NTP format timestamp, as described in Section 4 of [RFC3550].

RTCP congestion control feedback packets SHOULD include a report block for every active SSRC. The sequence number ranges reported on

in consecutive reports for a given SSRC will generally be contiguous, but overlapping reports MAY be sent (and need to be sent in cases where RTP packet reordering occurs across the boundary between consecutive reports). If an RTP packet was reported as received in one report, that packet MUST also be reported as received in any overlapping reports sent later that cover its sequence number range. If reports covering overlapping sequence number ranges are sent, information in later reports updates that sent in previous reports for RTP packets included in both reports.

RTCP congestion control feedback packets can be large if they are sent infrequently relative to the number of RTP data packets. If an RTCP congestion control feedback packet is too large to fit within the path MTU, its sender SHOULD split it into multiple feedback packets. The RTCP reporting interval SHOULD be chosen such that feedback packets are sent often enough that they are small enough to fit within the path MTU ([I-D.ietf-rmcat-rtp-cc-feedback] discusses how to choose the reporting interval; specifications for RTP congestion control algorithms can also provide guidance).

If duplicate copies of a particular RTP packet are received, then the arrival time of the first copy to arrive MUST be reported. If any of the copies of the duplicated packet are ECN-CE marked, then an ECN-CE mark MUST be reported that for packet; otherwise the ECN mark of the first copy to arrive is reported.

If no packets are received from an SSRC in a reporting interval, a report block MAY be sent with `begin_seq` set to the highest sequence number previously received from that SSRC and `num_reports` set to zero (or, the report can simply be omitted). The corresponding SR/RR packet will have a non-increased extended highest sequence number received field that will inform the sender that no packets have been received, but it can ease processing to have that information available in the congestion control feedback reports too.

A report block indicating that certain RTP packets were lost is not to be interpreted as a request to retransmit the lost packets. The receiver of such a report might choose to retransmit such packets, provided a retransmission payload format has been negotiated, but there is no requirement that it do so.

4. Feedback Frequency and Overhead

There is a trade-off between speed and accuracy of reporting, and the overhead of the reports. [I-D.ietf-rmcat-rtp-cc-feedback] discusses this trade-off, suggests desirable RTCP feedback rates, and provides guidance on how to configure the RTCP bandwidth fraction, etc., to make appropriate use of the reporting block described in this memo.

Specifications for RTP congestion control algorithms can also provide guidance.

It is generally understood that congestion control algorithms work better with more frequent feedback. However, RTCP bandwidth and transmission rules put some upper limits on how frequently the RTCP feedback messages can be sent from an RTP receiver to the RTP sender. In many cases, sending feedback once per frame is an upper bound before the reporting overhead becomes excessive, although this will depend on the media rate and more frequent feedback might be needed with high-rate media flows [I-D.ietf-rmcat-rtp-cc-feedback]. Analysis [feedback-requirements] has also shown that some candidate congestion control algorithms can operate with less frequent feedback, using a feedback interval range of 50-200ms. Applications need to negotiate an appropriate congestion control feedback interval at session setup time, based on the choice of congestion control algorithm, the expected media bit rate, and the acceptable feedback overhead.

5. Response to Loss of Feedback Packets

Like all RTCP packets, RTCP congestion control feedback packets might be lost. All RTP congestion control algorithms MUST specify how they respond to the loss of feedback packets.

RTCP packets do not contain a sequence number, so loss of feedback packets has to be inferred based on the time since the last feedback packet. If only a single congestion control feedback packet is lost, an appropriate response is to assume that the level of congestion has remained roughly the same as the previous report. However, if multiple consecutive congestion control feedback packets are lost, then the media sender SHOULD rapidly reduce its sending rate as this likely indicates a path failure. The RTP circuit breaker [RFC8083] provides further guidance.

6. SDP Signalling

A new "ack" feedback parameter, "ccfb", is defined for use with the "a=rtcp-fb:" SDP extension to indicate the use of the RTP Congestion Control feedback packet format defined in Section 3. The ABNF definition of this SDP parameter extension is:

```
rtcp-fb-ack-param = <See Section 4.2 of [RFC4585]>
rtcp-fb-ack-param =/ ccfb-par
ccfb-par           = SP "ccfb"
```

The payload type used with "ccfb" feedback MUST be the wildcard type ("*"). This implies that the congestion control feedback is sent for

all payload types in use in the session, including any FEC and retransmission payload types. An example of the resulting SDP attribute is:

```
a=rtcp-fb:* ack ccfb
```

The offer/answer rules for these SDP feedback parameters are specified in Section 4.2 of the RTP/AVPF profile [RFC4585].

An SDP offer might indicate support for both the congestion control feedback mechanism specified in this memo and one or more alternative congestion control feedback mechanisms that offer substantially the same semantics. In this case, the answering party SHOULD include only one of the offered congestion control feedback mechanisms in its answer. If a re-invite offering the same set of congestion control feedback mechanisms is received, the generated answer SHOULD choose the same congestion control feedback mechanism as in the original answer where possible.

When the SDP BUNDLE extension [I-D.ietf-mmusic-sdp-bundle-negotiation] is used for multiplexing, the "a=rtcp-fb:" attribute has multiplexing category IDENTICAL-PER-PT [I-D.ietf-mmusic-sdp-mux-attributes].

7. Relation to RFC 6679

Use of Explicit Congestion Notification (ECN) with RTP is described in [RFC6679]. That specifies how to negotiate the use of ECN with RTP, and defines an RTCP ECN Feedback Packet to carry ECN feedback reports. It uses an SDP "a=ecn-capable-rtp:" attribute to negotiate use of ECN, and the "a=rtcp-fb:" attributes with the "nack" parameter "ecn" to negotiate the use of RTCP ECN Feedback Packets.

The RTCP ECN Feedback Packet is not useful when ECN is used with the RTP Congestion Control Feedback Packet defined in this memo since it provides duplicate information. When congestion control feedback is to be used with RTP and ECN, the SDP offer generated MUST include an "a=ecn-capable-rtp:" attribute to negotiate ECN support, along with an "a=rtcp-fb:" attribute with the "ack" parameter "ccfb" to indicate that the RTP Congestion Control Feedback Packet can be used. The "a=rtcp-fb:" attribute MAY also include the "nack" parameter "ecn", to indicate that the RTCP ECN Feedback Packet is also supported. If an SDP offer signals support for both RTP Congestion Control Feedback Packets and the RTCP ECN Feedback Packet, the answering party SHOULD signal support for one, but not both, formats in its SDP answer to avoid sending duplicate feedback.

When using ECN with RTP, the guidelines in Section 7.2 of [RFC6679] MUST be followed to initiate the use of ECN in an RTP session. The guidelines in Section 7.3 of [RFC6679] MUST also be followed about ongoing use of ECN within an RTP session, with the exception that feedback is sent using the RTCP Congestion Control Feedback Packets described in this memo rather than using RTP ECN Feedback Packets. Similarly, the guidance in Section 7.4 of [RFC6679] around detecting failures MUST be followed, with the exception that the necessary information is retrieved from the RTCP Congestion Control Feedback Packets rather than from RTP ECN Feedback Packets.

8. Design Rationale

The primary function of RTCP SR/RR packets is to report statistics on the reception of RTP packets. The reception report blocks sent in these packets contain information about observed jitter, fractional packet loss, and cumulative packet loss. It was intended that this information could be used to support congestion control algorithms, but experience has shown that it is not sufficient for that purpose. An efficient congestion control algorithm requires more fine-grained information on per-packet reception quality than is provided by SR/RR packets to react effectively. The feedback format defined in this memo provides such fine-grained feedback.

Several other RTCP extensions also provide more detailed feedback than SR/RR packets:

TMMBR: The Codec Control Messages for the RTP/AVPF profile [RFC5104] include a Temporary Maximum Media Bit Rate (TMMBR) message. This is used to convey a temporary maximum bit rate limitation from a receiver of RTP packets to their sender. Even though it was not designed to replace congestion control, TMMBR has been used as a means to do receiver based congestion control where the session bandwidth is high enough to send frequent TMMBR messages, especially when used with non-compound RTCP packets [RFC5506]. This approach requires the receiver of the RTP packets to monitor their reception, determine the level of congestion, and recommend a maximum bit rate suitable for current available bandwidth on the path; it also assumes that the RTP sender can/will respect that bit rate. This is the opposite of the sender-based congestion control approach suggested in this memo, so TMMBR cannot be used to convey the information needed for a sender-based congestion control. TMMBR could, however, be viewed as a complementary mechanism that can inform the sender of the receiver's current view of acceptable maximum bit rate. Mechanisms that convey the receiver's estimate of the maximum available bit-rate provide similar feedback.

RTCP Extended Reports (XR): Numerous RTCP extended report (XR) blocks have been defined to report details of packet loss, arrival times [RFC3611], delay [RFC6843], and ECN marking [RFC6679]. It is possible to combine several such XR blocks into a compound RTCP packet, to report the detailed loss, arrival time, and ECN marking information needed for effective sender-based congestion control. However, the result has high overhead both in terms of bandwidth and complexity, due to the need to stack multiple reports.

Transport-wide Congestion Control: The format defined in this memo provides individual feedback on each SSRC. An alternative is to add a header extension to each RTP packet, containing a single, transport-wide, packet sequence number, then have the receiver send RTCP reports giving feedback on these additional sequence numbers [I-D.holmer-rmcat-transport-wide-cc-extensions]. Such an approach adds the per-packet overhead of the header extension (8 octets per packet in the referenced format), but reduces the size of the feedback packets, and can simplify the rate calculation at the sender if it maintains a single rate limit that applies to all RTP packets sent irrespective of their SSRC. Equally, the use of transport-wide feedback makes it more difficult to adapt the sending rate, or respond to lost packets, based on the reception and/or loss patterns observed on a per-SSRC basis (for example, to perform differential rate control and repair for audio and video flows, based on knowledge of what packets from each flow were lost). Transport-wide feedback is also a less natural fit with the wider RTP framework, which makes extensive use of per-SSRC sequence numbers and feedback.

Considering these issues, we believe it appropriate to design a new RTCP feedback mechanism to convey information for sender-based congestion control algorithms. The new congestion control feedback RTCP packet described in Section 3 provides such a mechanism.

9. Acknowledgements

This document is based on the outcome of a design team discussion in the RTP Media Congestion Avoidance Techniques (RMCAT) working group. The authors would like to thank David Hayes, Stefan Holmer, Randell Jesup, Ingemar Johansson, Jonathan Lennox, Sergio Mena, Nils Ohlmeier, Magnus Westerlund, and Xiaoqing Zhu for their valuable feedback.

10. IANA Considerations

The IANA is requested to register one new RTP/AVPF Transport-Layer Feedback Message in the table for FMT values for RTPFB Payload Types [RFC4585] as defined in Section 3.1:

Name: CCFB
Long name: RTP Congestion Control Feedback
Value: (to be assigned by IANA)
Reference: (RFC number of this document, when published)

The IANA is also requested to register one new SDP "rtcp-fb" attribute "ack" parameter, "ccfb", in the SDP ("ack" and "nack" Attribute Values) registry:

Value name: ccfb
Long name: Congestion Control Feedback
Usable with: ack
Mux: IDENTICAL-PER-PT
Reference: (RFC number of this document, when published)

11. Security Considerations

The security considerations of the RTP specification [RFC3550], the applicable RTP profile (e.g., [RFC3551], [RFC3711], or [RFC4585]), and the RTP congestion control algorithm that is in use (e.g., [RFC8698], [RFC8298], [I-D.ietf-rmcat-gcc], or [RFC8382]) apply.

A receiver that intentionally generates inaccurate RTCP congestion control feedback reports might be able to trick the sender into sending at a greater rate than the path can support, thereby causing congestion on the path. This will negatively impact the quality of experience of that receiver, and potentially cause denial of service to other traffic sharing the path and excessive resource usage at the media sender. Since RTP is an unreliable transport, a sender can intentionally drop a packet, leaving a gap in the RTP sequence number space without causing serious harm, to check that the receiver is correctly reporting losses (this needs to be done with care and some awareness of the media data being sent, to limit impact on the user experience).

An on-path attacker that can modify RTCP congestion control feedback packets can change the reports to trick the sender into sending at either an excessively high or excessively low rate, leading to denial of service. The secure RTCP profile [RFC3711] can be used to authenticate RTCP packets to protect against this attack.

An off-path attacker that can spoof RTCP congestion control feedback packets can similarly trick a sender into sending at an incorrect rate, leading to denial of service. This attack is difficult, since the attacker needs to guess the SSRC and sequence number in addition to the destination transport address. As with on-path attacks, the secure RTCP profile [RFC3711] can be used to authenticate RTCP packets to protect against this attack.

12. References

12.1. Normative References

- [I-D.ietf-mmusic-sdp-bundle-negotiation]
Holmberg, C., Alvestrand, H., and C. Jennings,
"Negotiating Media Multiplexing Using the Session
Description Protocol (SDP)", draft-ietf-mmusic-sdp-bundle-
negotiation-54 (work in progress), December 2018.
- [I-D.ietf-mmusic-sdp-mux-attributes]
Nandakumar, S., "A Framework for SDP Attributes when
Multiplexing", draft-ietf-mmusic-sdp-mux-attributes-19
(work in progress), August 2020.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate
Requirement Levels", BCP 14, RFC 2119,
DOI 10.17487/RFC2119, March 1997,
<<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC3168] Ramakrishnan, K., Floyd, S., and D. Black, "The Addition
of Explicit Congestion Notification (ECN) to IP",
RFC 3168, DOI 10.17487/RFC3168, September 2001,
<<https://www.rfc-editor.org/info/rfc3168>>.
- [RFC3550] Schulzrinne, H., Casner, S., Frederick, R., and V.
Jacobson, "RTP: A Transport Protocol for Real-Time
Applications", STD 64, RFC 3550, DOI 10.17487/RFC3550,
July 2003, <<https://www.rfc-editor.org/info/rfc3550>>.
- [RFC3551] Schulzrinne, H. and S. Casner, "RTP Profile for Audio and
Video Conferences with Minimal Control", STD 65, RFC 3551,
DOI 10.17487/RFC3551, July 2003,
<<https://www.rfc-editor.org/info/rfc3551>>.
- [RFC3711] Baugher, M., McGrew, D., Naslund, M., Carrara, E., and K.
Norrman, "The Secure Real-time Transport Protocol (SRTP)",
RFC 3711, DOI 10.17487/RFC3711, March 2004,
<<https://www.rfc-editor.org/info/rfc3711>>.
- [RFC4585] Ott, J., Wenger, S., Sato, N., Burmeister, C., and J. Rey,
"Extended RTP Profile for Real-time Transport Control
Protocol (RTCP)-Based Feedback (RTP/AVPF)", RFC 4585,
DOI 10.17487/RFC4585, July 2006,
<<https://www.rfc-editor.org/info/rfc4585>>.

- [RFC5124] Ott, J. and E. Carrara, "Extended Secure RTP Profile for Real-time Transport Control Protocol (RTCP)-Based Feedback (RTP/SAVPF)", RFC 5124, DOI 10.17487/RFC5124, February 2008, <<https://www.rfc-editor.org/info/rfc5124>>.
- [RFC5506] Johansson, I. and M. Westerlund, "Support for Reduced-Size Real-Time Transport Control Protocol (RTCP): Opportunities and Consequences", RFC 5506, DOI 10.17487/RFC5506, April 2009, <<https://www.rfc-editor.org/info/rfc5506>>.
- [RFC6679] Westerlund, M., Johansson, I., Perkins, C., O'Hanlon, P., and K. Carlberg, "Explicit Congestion Notification (ECN) for RTP over UDP", RFC 6679, DOI 10.17487/RFC6679, August 2012, <<https://www.rfc-editor.org/info/rfc6679>>.
- [RFC8083] Perkins, C. and V. Singh, "Multimedia Congestion Control: Circuit Breakers for Unicast RTP Sessions", RFC 8083, DOI 10.17487/RFC8083, March 2017, <<https://www.rfc-editor.org/info/rfc8083>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.

12.2. Informative References

- [feedback-requirements]
"RMCAT Feedback Requirements",
<[://www.ietf.org/proceedings/95/slides/slides-95-rmcat-1.pdf](https://www.ietf.org/proceedings/95/slides/slides-95-rmcat-1.pdf)>.
- [I-D.alvestrand-rmcat-remb]
Alvestrand, H., "RTCP message for Receiver Estimated Maximum Bitrate", draft-alvestrand-rmcat-remb-03 (work in progress), October 2013.
- [I-D.holmer-rmcat-transport-wide-cc-extensions]
Holmer, S., Flodman, M., and E. Sprang, "RTP Extensions for Transport-wide Congestion Control", draft-holmer-rmcat-transport-wide-cc-extensions-01 (work in progress), October 2015.
- [I-D.ietf-rmcat-gcc]
Holmer, S., Lundin, H., Carlucci, G., Cicco, L., and S. Mascolo, "A Google Congestion Control Algorithm for Real-Time Communication", draft-ietf-rmcat-gcc-02 (work in progress), July 2016.

- [I-D.ietf-rmcat-rtp-cc-feedback]
Perkins, C., "RTP Control Protocol (RTCP) Feedback for Congestion Control in Interactive Multimedia Conferences", draft-ietf-rmcat-rtp-cc-feedback-05 (work in progress), November 2019.
- [RFC3611] Friedman, T., Ed., Caceres, R., Ed., and A. Clark, Ed., "RTP Control Protocol Extended Reports (RTCP XR)", RFC 3611, DOI 10.17487/RFC3611, November 2003, <<https://www.rfc-editor.org/info/rfc3611>>.
- [RFC5104] Wenger, S., Chandra, U., Westerlund, M., and B. Burman, "Codec Control Messages in the RTP Audio-Visual Profile with Feedback (AVPF)", RFC 5104, DOI 10.17487/RFC5104, February 2008, <<https://www.rfc-editor.org/info/rfc5104>>.
- [RFC6843] Clark, A., Gross, K., and Q. Wu, "RTP Control Protocol (RTCP) Extended Report (XR) Block for Delay Metric Reporting", RFC 6843, DOI 10.17487/RFC6843, January 2013, <<https://www.rfc-editor.org/info/rfc6843>>.
- [RFC8298] Johansson, I. and Z. Sarker, "Self-Clocked Rate Adaptation for Multimedia", RFC 8298, DOI 10.17487/RFC8298, December 2017, <<https://www.rfc-editor.org/info/rfc8298>>.
- [RFC8382] Hayes, D., Ed., Ferlin, S., Welzl, M., and K. Hiorth, "Shared Bottleneck Detection for Coupled Congestion Control for RTP Media", RFC 8382, DOI 10.17487/RFC8382, June 2018, <<https://www.rfc-editor.org/info/rfc8382>>.
- [RFC8698] Zhu, X., Pan, R., Ramalho, M., and S. Mena, "Network-Assisted Dynamic Adaptation (NADA): A Unified Congestion Control Scheme for Real-Time Media", RFC 8698, DOI 10.17487/RFC8698, February 2020, <<https://www.rfc-editor.org/info/rfc8698>>.

Authors' Addresses

Zaheduzzaman Sarker
Ericsson AB
Torshamnsgatan 21
Stockholm 164 40
Sweden

Phone: +46107173743
Email: zaheduzzaman.sarker@ericsson.com

Colin Perkins
University of Glasgow
School of Computing Science
Glasgow G12 8QQ
United Kingdom

Email: csp@csperskins.org

Varun Singh
CALLSTATS I/O Oy
Annankatu 31-33 C 42
Helsinki 00100
Finland

Email: varun.singh@iki.fi
URI: <http://www.callstats.io/>

Michael A. Ramalho
6310 Watercrest Way Unit 203
Lakewood Ranch, FL 34202-5122
USA

Phone: +1 732 832 9723
Email: mar42@cornell.edu
URI: <http://ramalho.webhop.info/>

Network Working Group
Internet-Draft
Intended status: Experimental
Expires: March 8, 2020

X. Zhu
R. Pan
M. Ramalho
S. Mena
Cisco Systems
September 5, 2019

NADA: A Unified Congestion Control Scheme for Real-Time Media
draft-ietf-rmcat-nada-13

Abstract

This document describes NADA (network-assisted dynamic adaptation), a novel congestion control scheme for interactive real-time media applications, such as video conferencing. In the proposed scheme, the sender regulates its sending rate based on either implicit or explicit congestion signaling, in a unified approach. The scheme can benefit from explicit congestion notification (ECN) markings from network nodes. It also maintains consistent sender behavior in the absence of such markings, by reacting to queuing delays and packet losses instead.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <https://datatracker.ietf.org/drafts/current/>.

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

This Internet-Draft will expire on March 8, 2020.

Copyright Notice

Copyright (c) 2019 IETF Trust and the persons identified as the document authors. All rights reserved.

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

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

Table of Contents

1. Introduction	3
2. Terminology	3
3. System Overview	3
4. Core Congestion Control Algorithm	5
4.1. Mathematical Notations	5
4.2. Receiver-Side Algorithm	8
4.3. Sender-Side Algorithm	10
5. Practical Implementation of NADA	13
5.1. Receiver-Side Operation	13
5.1.1. Estimation of one-way delay and queuing delay	13
5.1.2. Estimation of packet loss/marketing ratio	13
5.1.3. Estimation of receiving rate	14
5.2. Sender-Side Operation	14
5.2.1. Rate shaping buffer	15
5.2.2. Adjusting video target rate and sending rate	16
5.3. Feedback Message Requirements	16
6. Discussions and Further Investigations	17
6.1. Choice of delay metrics	17
6.2. Method for delay, loss, and marking ratio estimation	18
6.3. Impact of parameter values	18
6.4. Sender-based vs. receiver-based calculation	20
6.5. Incremental deployment	20
7. Reference Implementations	20
8. Suggested Experiments	21
9. IANA Considerations	22
10. Security Considerations	22
11. Acknowledgments	22
12. Contributors	22
13. References	23
13.1. Normative References	23
13.2. Informative References	24
Appendix A. Network Node Operations	26
A.1. Default behavior of drop tail queues	27
A.2. RED-based ECN marking	27
A.3. Random Early Marking with Virtual Queues	28
Authors' Addresses	28

1. Introduction

Interactive real-time media applications introduce a unique set of challenges for congestion control. Unlike TCP, the mechanism used for real-time media needs to adapt quickly to instantaneous bandwidth changes, accommodate fluctuations in the output of video encoder rate control, and cause low queuing delay over the network. An ideal scheme should also make effective use of all types of congestion signals, including packet loss, queuing delay, and explicit congestion notification (ECN) [RFC3168] markings. The requirements for the congestion control algorithm are outlined in [I-D.ietf-rmcat-cc-requirements]. It highlights that the desired congestion control scheme should avoid flow starvation and attain a reasonable fair share of bandwidth when competing against other flows, adapt quickly, and operate in a stable manner.

This document describes an experimental congestion control scheme called network-assisted dynamic adaptation (NADA). The design of NADA benefits from explicit congestion control signals (e.g., ECN markings) from the network, yet also operates when only implicit congestion indicators (delay and/or loss) are available. Such a unified sender behavior distinguishes NADA from other congestion control schemes for real-time media. In addition, its core congestion control algorithm is designed to guarantee stability for path round-trip-times (RTTs) below a prescribed bound (e.g., 250ms with default parameter choices). It further supports weighted bandwidth sharing among competing video flows with different priorities. The signaling mechanism consists of standard RTP timestamp [RFC3550] and RTCP feedback reports. The definition of the desired RTCP feedback message is described in detail in [I-D.ietf-avtcore-cc-feedback-message] so as to support the successful operation of several congestion control schemes for real-time interactive media.

2. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

3. System Overview

Figure 1 shows the end-to-end system for real-time media transport that NADA operates in. Note that there also exist network nodes along the reverse (potentially uncongested) path that the RTCP

feedback reports traverse. Those network nodes are not shown in the figure for sake of brevity.

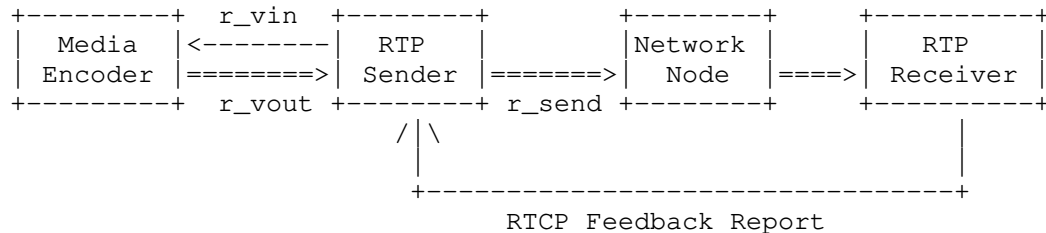


Figure 1: System Overview

- o Media encoder with rate control capabilities. It encodes raw media (audio and video) frames into a compressed bitstream which is later packetized into RTP packets. As discussed in [RFC8593], the actual output rate from the encoder `r_vout` may fluctuate around the target `r_vin`. Furthermore, it is possible that the encoder can only react to bit rate changes at rather coarse time intervals, e.g., once every 0.5 seconds.
- o RTP sender: responsible for calculating the NADA reference rate based on network congestion indicators (delay, loss, or ECN marking reports from the receiver), for updating the video encoder with a new target rate `r_vin`, and for regulating the actual sending rate `r_send` accordingly. The RTP sender also generates a sending timestamp for each outgoing packet.
- o RTP receiver: responsible for measuring and estimating end-to-end delay (based on sender timestamp), packet loss (based on RTP sequence number), ECN marking ratios (based on [RFC6679]), and receiving rate (`r_rcv`) of the flow. It calculates the aggregated congestion signal (`x_curr`) that accounts for queuing delay, ECN markings, and packet losses. The receiver also determines the mode for sender rate adaptation (`rmode`) based on whether the flow has encountered any standing non-zero congestion. The receiver sends periodic RTCP reports back to the sender, containing values of `x_curr`, `rmode`, and `r_rcv`.
- o Network node with several modes of operation. The system can work with the default behavior of a simple drop tail queue. It can also benefit from advanced AQM features such as PIE [RFC8033], FQ-CoDel [RFC8290], ECN marking based on RED [RFC7567], and PCN marking using a token bucket algorithm ([RFC6660]). Note that network node operation is out of control for the design of NADA.

4. Core Congestion Control Algorithm

Like TCP-Friendly Rate Control (TFRC) [Floyd-CCR00] [RFC5348], NADA is a rate-based congestion control algorithm. In its simplest form, the sender reacts to the collection of network congestion indicators in the form of an aggregated congestion signal, and operates in one of two modes:

- o Accelerated ramp-up: when the bottleneck is deemed to be underutilized, the rate increases multiplicatively with respect to the rate of previously successful transmissions. The rate increase multiplier (γ) is calculated based on observed round-trip-time and target feedback interval, so as to limit self-inflicted queuing delay.
- o Gradual rate update: in the presence of non-zero aggregate congestion signal, the sending rate is adjusted in reaction to both its value (x_{curr}) and its change in value (x_{diff}).

This section introduces the list of mathematical notations and describes the core congestion control algorithm at the sender and receiver, respectively. Additional details on recommended practical implementations are described in Section 5.1 and Section 5.2.

4.1. Mathematical Notations

This section summarizes the list of variables and parameters used in the NADA algorithm. Figure 3 also includes the default values for choosing the algorithm parameters either to represent a typical setting in practical applications or based on theoretical and simulation studies. See Section 6.3 for some of the discussions on the impact of parameter values. Additional studies in real-world settings suggested in Section 8 could gather further insight on how to choose and adapt these parameter values in practical deployment.

Notation	Variable Name
t_curr	Current timestamp
t_last	Last time sending/receiving a feedback
delta	Observed interval between current and previous feedback reports: $\text{delta} = \text{t_curr} - \text{t_last}$
r_ref	Reference rate based on network congestion
r_send	Sending rate
r_recv	Receiving rate
r_vin	Target rate for video encoder
r_vout	Output rate from video encoder
d_base	Estimated baseline delay
d_fwd	Measured and filtered one-way delay
d_queue	Estimated queuing delay
d_tilde	Equivalent delay after non-linear warping
p_mark	Estimated packet ECN marking ratio
p_loss	Estimated packet loss ratio
x_curr	Aggregate congestion signal
x_prev	Previous value of aggregate congestion signal
x_diff	Change in aggregate congestion signal w.r.t. its previous value: $\text{x_diff} = \text{x_curr} - \text{x_prev}$
rmode	Rate update mode: (0 = accelerated ramp-up; 1 = gradual update)
gamma	Rate increase multiplier in accelerated ramp-up mode
loss_int	Measured average loss interval in packet count
loss_exp	Threshold value for setting the last observed packet loss to expiration
rtt	Estimated round-trip-time at sender
buffer_len	Rate shaping buffer occupancy measured in bytes

Figure 2: List of variables.

Notation	Parameter Name	Default Value
PRIO	Weight of priority of the flow	1.0
RMIN	Minimum rate of application supported by media encoder	150Kbps
RMAX	Maximum rate of application supported by media encoder	1.5Mbps
XREF	Reference congestion level	10ms
KAPPA	Scaling parameter for gradual rate update calculation	0.5
ETA	Scaling parameter for gradual rate update calculation	2.0

TAU	Upper bound of RTT in gradual rate update calculation	500ms
DELTA	Target feedback interval	100ms
+.....+		
LOGWIN	Observation window in time for calculating packet summary statistics at receiver	500ms
QEPS	Threshold for determining queuing delay build up at receiver	10ms
DFILT	Bound on filtering delay	120ms
GAMMA_MAX	Upper bound on rate increase ratio for accelerated ramp-up	0.5
QBOUND	Upper bound on self-inflicted queuing delay during ramp up	50ms
+.....+		
MULTILOSS	Multiplier for self-scaling the expiration threshold of the last observed loss (loss_exp) based on measured average loss interval (loss_int)	7.0
QTH	Delay threshold for invoking non-linear warping	50ms
LAMBDA	Scaling parameter in the exponent of non-linear warping	0.5
+.....+		
PLRREF	Reference packet loss ratio	0.01
PMRREF	Reference packet marking ratio	0.01
DLOSS	Reference delay penalty for loss when packet loss ratio is at PLRREF	10ms
DMARK	Reference delay penalty for ECN marking when packet marking is at PMRREF	2ms
+.....+		
FPS	Frame rate of incoming video	30
BETA_S	Scaling parameter for modulating outgoing sending rate	0.1
BETA_V	Scaling parameter for modulating video encoder target rate	0.1
ALPHA	Smoothing factor in exponential smoothing of packet loss and marking ratios	0.1
+-----+		

Figure 3: List of algorithm parameters and their default values.

4.2. Receiver-Side Algorithm

The receiver-side algorithm can be outlined as below:

On initialization:

- set `d_base` = +INFINITY
- set `p_loss` = 0
- set `p_mark` = 0
- set `r_recv` = 0
- set both `t_last` and `t_curr` as current time in milliseconds

On receiving a media packet:

- obtain current timestamp `t_curr` from system clock
- obtain from packet header sending time stamp `t_sent`
- obtain one-way delay measurement: `d_fwd` = `t_curr` - `t_sent`
- update baseline delay: `d_base` = min(`d_base`, `d_fwd`)
- update queuing delay: `d_queue` = `d_fwd` - `d_base`
- update packet loss ratio estimate `p_loss`
- update packet marking ratio estimate `p_mark`
- update measurement of receiving rate `r_recv`

On time to send a new feedback report (`t_curr` - `t_last` > DELTA):

- calculate non-linear warping of delay `d_tilde` if packet loss exists
- calculate current aggregate congestion signal `x_curr`
- determine mode of rate adaptation for sender: `rmode`
- send feedback containing values of: `rmode`, `x_curr`, and `r_recv`
- update `t_last` = `t_curr`

In order for a delay-based flow to hold its ground when competing against loss-based flows (e.g., loss-based TCP), it is important to distinguish between different levels of observed queuing delay. For instance, over wired connections, a moderate queuing delay value on the order of tens of milliseconds is likely self-inflicted or induced by other delay-based flows, whereas a high queuing delay value of several hundreds of milliseconds may indicate the presence of a loss-based flow that does not refrain from increased delay.

If the last observed packet loss is within the expiration window of `loss_exp` (measured in terms of packet counts), the estimated queuing delay follows a non-linear warping:

$$d_tilde = \begin{cases} d_queue, & \text{if } d_queue < QTH; \\ QTH \exp(-LAMBDA \frac{(d_queue - QTH)}{QTH}), & \text{otherwise.} \end{cases} \quad (1)$$

In (1), the queuing delay value is unchanged when it is below the first threshold QTH ; otherwise it is scaled down following a non-linear curve. This non-linear warping is inspired by the delay-adaptive congestion window backoff policy in [Budzisz-TON11], so as to "gradually nudge" the controller to operate based on loss-induced congestion signals when competing against loss-based flows. The exact form of the non-linear function has been simplified with respect to [Budzisz-TON11]. The value of the threshold QTH should be carefully tuned for different operational environments, so as to avoid potential risks of prematurely discounting the congestion signal level. Typically, a higher value of QTH is required in a noisier environment (e.g., over wireless connections, or where the video stream encounters many time-varying background competing traffic) so as to stay robust against occasional non-congestion-induced delay spikes. Additional insights on how this value can be tuned or auto-tuned should be gathered from carrying out experimental studies in different real-world deployment scenarios.

The value of $loss_exp$ is configured to self-scale with the average packet loss interval $loss_int$ with a multiplier $MULTILOSS$:

$$loss_exp = MULTILOSS * loss_int.$$

Estimation of the average loss interval $loss_int$, in turn, follows Section 5.4 of the TCP Friendly Rate Control (TFRC) protocol [RFC5348].

In practice, it is recommended to linearly interpolate between the warped (d_tilde) and non-warped (d_queue) values of the queuing delay during the transitional period lasting for the duration of $loss_int$.

The aggregate congestion signal is:

$$x_curr = d_tilde + DMARK * \frac{p_mark^2}{PMRREF} + DLOSS * \frac{p_loss^2}{PLRREF}. \quad (2)$$

Here, DMARK is prescribed reference delay penalty associated with ECN markings at the reference marking ratio of PMRREF; DLOSS is prescribed reference delay penalty associated with packet losses at the reference packet loss ratio of PLRREF. The value of DLOSS and DMARK does not depend on configurations at the network node. Since ECN-enabled active queue management schemes typically mark a packet before dropping it, the value of DLOSS SHOULD be higher than that of DMARK. Furthermore, the values of DLOSS and DMARK need to be set consistently across all NADA flows sharing the same bottleneck link, so that they can compete fairly.

In the absence of packet marking and losses, the value of x_{curr} reduces to the observed queuing delay d_{queue} . In that case the NADA algorithm operates in the regime of delay-based adaptation.

Given observed per-packet delay and loss information, the receiver is also in a good position to determine whether the network is underutilized and recommend the corresponding rate adaptation mode for the sender. The criteria for operating in accelerated ramp-up mode are:

- o No recent packet losses within the observation window LOGWIN; and
- o No build-up of queuing delay: $d_{fwd} - d_{base} < QEPS$ for all previous delay samples within the observation window LOGWIN.

Otherwise the algorithm operates in graduate update mode.

4.3. Sender-Side Algorithm

The sender-side algorithm is outlined as follows:

```

on initialization:
    set r_ref = RMIN
    set rtt = 0
    set x_prev = 0
    set t_last and t_curr as current system clock time

on receiving feedback report:
    obtain current timestamp from system clock: t_curr
    obtain values of rmode, x_curr, and r_recv from feedback report
    update estimation of rtt
    measure feedback interval: delta = t_curr - t_last
    if rmode == 0:
        update r_ref following accelerated ramp-up rules
    else:
        update r_ref following gradual update rules

    clip rate r_ref within the range of minimum rate (RMIN)
    and maximum rate (RMAX).
    x_prev = x_curr
    t_last = t_curr

```

In accelerated ramp-up mode, the rate r_ref is updated as follows:

$$\gamma = \min(\text{GAMMA_MAX}, \frac{\text{QBOUND}}{\text{rtt} + \text{DELTA} + \text{DFILT}}) \quad (3)$$

$$r_ref = \max(r_ref, (1 + \gamma) r_recv) \quad (4)$$

The rate increase multiplier γ is calculated as a function of upper bound of self-inflicted queuing delay (QBOUND), round-trip-time (rtt), target feedback interval (DELTA) and bound on filtering delay for calculating d_queue (DFILT). It has a maximum value of GAMMA_MAX. The rationale behind (3)-(4) is that the longer it takes for the sender to observe self-inflicted queuing delay build-up, the more conservative the sender should be in increasing its rate, hence the smaller the rate increase multiplier.

In gradual update mode, the rate r_ref is updated as:

$$x_offset = x_curr - \text{PRIO} * \text{XREF} * \text{RMAX} / r_ref \quad (5)$$

$$x_diff = x_curr - x_prev \quad (6)$$

$$r_ref = r_ref - \text{KAPPA} * \frac{\text{delta}}{\text{TAU}} * \frac{x_offset}{\text{TAU}} * r_ref - \text{KAPPA} * \text{ETA} * \frac{x_diff}{\text{TAU}} * r_ref \quad (7)$$

The rate changes in proportion to the previous rate decision. It is affected by two terms: offset of the aggregate congestion signal from its value at equilibrium (x_offset) and its change (x_diff). Calculation of x_offset depends on maximum rate of the flow (RMAX), its weight of priority (PRIO), as well as a reference congestion signal (XREF). The value of XREF is chosen so that the maximum rate of RMAX can be achieved when the observed congestion signal level is below $\text{PRIO} * \text{XREF}$.

At equilibrium, the aggregated congestion signal stabilizes at $x_curr = \text{PRIO} * \text{XREF} * \text{RMAX} / r_ref$. This ensures that when multiple flows share the same bottleneck and observe a common value of x_curr , their rates at equilibrium will be proportional to their respective priority levels (PRIO) and the range between minimum and maximum rate. Values of the minimum rate (RMIN) and maximum rate (RMAX) will be provided by the media codec, for instance, as outlined by [I-D.ietf-rmcat-cc-codec-interactions]. In the absence of such information, NADA sender will choose a default value of 0 for RMIN , and 3Mbps for RMAX .

As mentioned in the sender-side algorithm, the final rate is always clipped within the dynamic range specified by the application:

$$r_ref = \min(r_ref, \text{RMAX}) \quad (8)$$

$$r_ref = \max(r_ref, \text{RMIN}) \quad (9)$$

The above operations ignore many practical issues such as clock synchronization between sender and receiver, filtering of noise in delay measurements, and base delay expiration. These will be addressed in Section 5.

5. Practical Implementation of NADA

5.1. Receiver-Side Operation

The receiver continuously monitors end-to-end per-packet statistics in terms of delay, loss, and/or ECN marking ratios. It then aggregates all forms of congestion indicators into the form of an equivalent delay and periodically reports this back to the sender. In addition, the receiver tracks the receiving rate of the flow and includes that in the feedback message.

5.1.1. Estimation of one-way delay and queuing delay

The delay estimation process in NADA follows a similar approach as in earlier delay-based congestion control schemes, such as LEDBAT [RFC6817]. For experimental implementations, instead of relying on RTP timestamps and the transmission time offset RTP header extension [RFC5450], the NADA sender can generate its own timestamp based on local system clock and embed that information in the transport packet header. The NADA receiver estimates the forward delay as having a constant base delay component plus a time varying queuing delay component. The base delay is estimated as the minimum value of one-way delay observed over a relatively long period (e.g., tens of minutes), whereas the individual queuing delay value is taken to be the difference between one-way delay and base delay. By re-estimating the base delay periodically, one can avoid the potential issue of base delay expiration, whereby an earlier measured base delay value is no longer valid due to underlying route changes or cumulative timing difference introduced by the clock rate skew between sender and receiver. All delay estimations are based on sender timestamps with a recommended granularity of 100 microseconds or finer.

The individual sample values of queuing delay should be further filtered against various non-congestion-induced noise, such as spikes due to processing "hiccup" at the network nodes. Therefore, in addition to calculating the value of queuing delay using $d_{\text{queue}} = d_{\text{fwd}} - d_{\text{base}}$, as expressed in Section 5.1, current implementation further employs a minimum filter with a window size of 15 samples over per-packet queuing delay values.

5.1.2. Estimation of packet loss/marketing ratio

The receiver detects packet losses via gaps in the RTP sequence numbers of received packets. For interactive real-time media application with stringent latency constraint (e.g., video conferencing), the receiver avoids the packet re-ordering delay by treating out-of-order packets as losses. The instantaneous packet

loss ratio p_{inst} is estimated as the ratio between the number of missing packets over the number of total transmitted packets within the recent observation window LOGWIN. The packet loss ratio p_{loss} is obtained after exponential smoothing:

$$p_{loss} = \text{ALPHA} * p_{inst} + (1 - \text{ALPHA}) * p_{loss}. \quad (10)$$

The filtered result is reported back to the sender as the observed packet loss ratio p_{loss} .

Estimation of packet marking ratio p_{mark} follows the same procedure as above. It is assumed that ECN marking information at the IP header can be passed to the receiving endpoint, e.g., by following the mechanism described in [RFC6679].

5.1.3. Estimation of receiving rate

It is fairly straightforward to estimate the receiving rate r_{recv} . NADA maintains a recent observation window with time span of LOGWIN, and simply divides the total size of packets arriving during that window over the time span. The receiving rate (r_{recv}) can be calculated at either the sender side based on the per-packet feedback from the receiver, or included as part of the feedback report.

5.2. Sender-Side Operation

Figure 4 provides a detailed view of the NADA sender. Upon receipt of an RTCP feedback report from the receiver, the NADA sender calculates the reference rate r_{ref} as specified in Section 4.3. It further adjusts both the target rate for the live video encoder r_{vin} and the sending rate r_{send} over the network based on the updated value of r_{ref} and rate shaping buffer occupancy $buffer_{len}$.

The NADA sender behavior stays the same in the presence of all types of congestion indicators: delay, loss, and ECN marking. This unified approach allows a graceful transition of the scheme as the network shifts dynamically between light and heavy congestion levels.

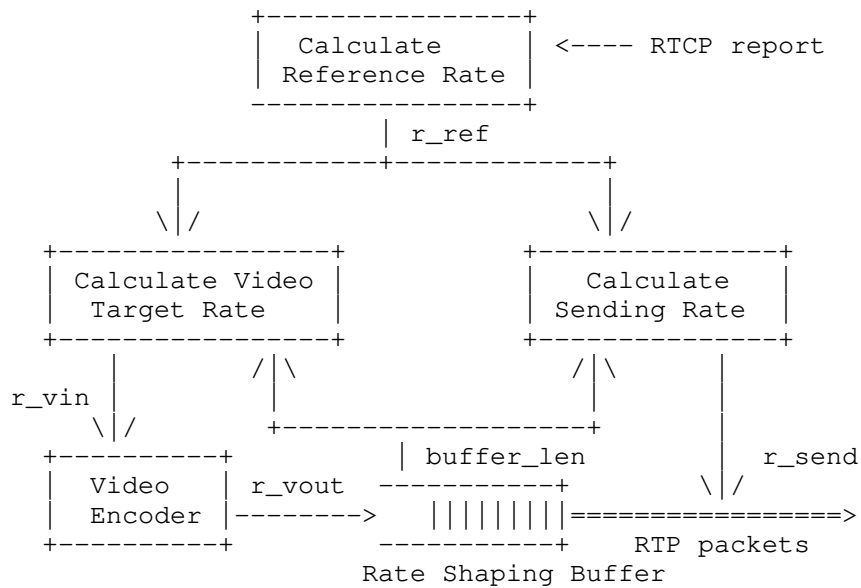


Figure 4: NADA Sender Structure

5.2.1. Rate shaping buffer

The operation of the live video encoder is out of the scope of the design for the congestion control scheme in NADA. Instead, its behavior is treated as a black box.

A rate shaping buffer is employed to absorb any instantaneous mismatch between encoder rate output r_{vout} and regulated sending rate r_{send} . Its current level of occupancy is measured in bytes and is denoted as $buffer_len$.

A large rate shaping buffer contributes to higher end-to-end delay, which may harm the performance of real-time media communications. Therefore, the sender has a strong incentive to prevent the rate shaping buffer from building up. The mechanisms adopted are:

- o To deplete the rate shaping buffer faster by increasing the sending rate r_{send} ; and
- o To limit incoming packets of the rate shaping buffer by reducing the video encoder target rate r_{vin} .

5.2.2. Adjusting video target rate and sending rate

If the level of occupancy in the rate shaping buffer is accessible at the sender, such information can be leveraged to further adjust the target rate of the live video encoder r_{vin} as well as the actual sending rate r_{send} . The purpose of such adjustments is to mitigate the additional latencies introduced by the rate shaping buffer. The amount of rate adjustment can be calculated as follows:

$$r_{diff_v} = \min(0.05 \cdot r_{ref}, BETA_V \cdot 8 \cdot buffer_len \cdot FPS). \quad (11)$$

$$r_{diff_s} = \min(0.05 \cdot r_{ref}, BETA_S \cdot 8 \cdot buffer_len \cdot FPS). \quad (12)$$

$$r_{vin} = \max(RMIN, r_{ref} - r_{diff_v}). \quad (13)$$

$$r_{send} = \min(RMAX, r_{ref} + r_{diff_s}). \quad (14)$$

In (11) and (12), the amount of adjustment is calculated as proportional to the size of the rate shaping buffer but is bounded by 5% of the reference rate r_{ref} calculated from network congestion feedback alone. This ensures that the adjustment introduced by the rate shaping buffer will not counteract with the core congestion control process. Equations (13) and (14) indicate the influence of the rate shaping buffer. A large rate shaping buffer nudges the encoder target rate slightly below -- and the sending rate slightly above -- the reference rate r_{ref} . The final video target rate (r_{vin}) and sending rate (r_{send}) are further bounded within the original range of $[RMIN, RMAX]$.

Intuitively, the amount of extra rate offset needed to completely drain the rate shaping buffer within the duration of a single video frame is given by $8 \cdot buffer_len \cdot FPS$, where FPS stands for the reference frame rate of the video. The scaling parameters $BETA_V$ and $BETA_S$ can be tuned to balance between the competing goals of maintaining a small rate shaping buffer and deviating from the reference rate point. Empirical observations show that the rate shaping buffer for a responsive live video encoder typically stays empty and only occasionally holds a large frame (e.g., when an intra-frame is produced) in transit. Therefore, the rate adjustment introduced by this mechanism is expected to be minor. For instance, a rate shaping buffer of 2000 Bytes will lead to a rate adjustment of 48Kbps given the recommended scaling parameters of $BETA_V = 0.1$ and $BETA_S = 0.1$ and reference frame rate of $FPS = 30$.

5.3. Feedback Message Requirements

The following list of information is required for NADA congestion control to function properly:

- o Recommended rate adaptation mode (rmode): a 1-bit flag indicating whether the sender should operate in accelerated ramp-up mode (rmode=0) or gradual update mode (rmode=1).
- o Aggregated congestion signal (x_curr): the most recently updated value, calculated by the receiver according to Section 4.2. This information can be expressed with a unit of 100 microsecond (i.e., 1/10 of a millisecond) in 15 bits. This allows a maximum value of x_curr at approximately 3.27 second.
- o Receiving rate (r_recv): the most recently measured receiving rate according to Section 5.1.3. This information is expressed with a unit of bits per second (bps) in 32 bits (unsigned int). This allows a maximum rate of approximately 4.3Gbps, approximately 1000 times of the streaming rate of a typical high-definition (HD) video conferencing session today. This field can be expanded further by a few more bytes, in case an even higher rate need to be specified.

The above list of information can be accommodated by 48 bits, or 6 bytes, in total. They can be either included in the feedback report from the receiver, or, in the case where all receiver-side calculations are moved to the sender, derived from per-packet information from the feedback message as defined in [I-D.ietf-avtcore-cc-feedback-message]. Choice of the feedback message interval DELTA is discussed in Section 6.3. A target feedback interval of DELTA=100ms is recommended.

6. Discussions and Further Investigations

This section discussed the various design choices made by NADA, potential alternative variants of its implementation, and guidelines on how the key algorithm parameters can be chosen. Section 8 recommends additional experimental setups to further explore these topics.

6.1. Choice of delay metrics

The current design works with relative one-way-delay (OWD) as the main indication of congestion. The value of the relative OWD is obtained by maintaining the minimum value of observed OWD over a relatively long time horizon and subtract that out from the observed absolute OWD value. Such an approach cancels out the fixed difference between the sender and receiver clocks. It has been widely adopted by other delay-based congestion control approaches such as [RFC6817]. As discussed in [RFC6817], the time horizon for tracking the minimum OWD needs to be chosen with care: it must be long enough for an opportunity to observe the minimum OWD with zero

standing queue along the path, and sufficiently short so as to timely reflect "true" changes in minimum OWD introduced by route changes and other rare events and to mitigate the cumulative impact of clock rate skew over time.

The potential drawback in relying on relative OWD as the congestion signal is that when multiple flows share the same bottleneck, the flow arriving late at the network experiencing a non-empty queue may mistakenly consider the standing queuing delay as part of the fixed path propagation delay. This will lead to slightly unfair bandwidth sharing among the flows.

Alternatively, one could move the per-packet statistical handling to the sender instead and use relative round-trip-time (RTT) in lieu of relative OWD, assuming that per-packet acknowledgments are available. The main drawback of RTT-based approach is the noise in the measured delay in the reverse direction.

Note that the choice of either delay metric (relative OWD vs. RTT) involves no change in the proposed rate adaptation algorithm. Therefore, comparing the pros and cons regarding which delay metric to adopt can be kept as an orthogonal direction of investigation.

6.2. Method for delay, loss, and marking ratio estimation

Like other delay-based congestion control schemes, performance of NADA depends on the accuracy of its delay measurement and estimation module. Appendix A in [RFC6817] provides an extensive discussion on this aspect.

The current recommended practice of applying minimum filter with a window size of 15 samples suffices in guarding against processing delay outliers observed in wired connections. For wireless connections with a higher packet delay variation (PDV), more sophisticated techniques on de-noising, outlier rejection, and trend analysis may be needed.

More sophisticated methods in packet loss ratio calculation, such as that adopted by [Floyd-CCR00], will likely be beneficial. These alternatives are part of the experiments this document proposes.

6.3. Impact of parameter values

In the gradual rate update mode, the parameter TAU indicates the upper bound of round-trip-time (RTT) in feedback control loop. Typically, the observed feedback interval delta is close to the target feedback interval DELTA, and the relative ratio of delta/TAU versus ETA dictates the relative strength of influence from the

aggregate congestion signal offset term (x_{offset}) versus its recent change (x_{diff}), respectively. These two terms are analogous to the integral and proportional terms in a proportional-integral (PI) controller. The recommended choice of $\text{TAU}=500\text{ms}$, $\text{DELTA}=100\text{ms}$ and $\text{ETA}=2.0$ corresponds to a relative ratio of 1:10 between the gains of the integral and proportional terms. Consequently, the rate adaptation is mostly driven by the change in the congestion signal with a long-term shift towards its equilibrium value driven by the offset term. Finally, the scaling parameter KAPPA determines the overall speed of the adaptation and needs to strike a balance between responsiveness and stability.

The choice of the target feedback interval DELTA needs to strike the right balance between timely feedback and low RTCP feedback message counts. A target feedback interval of $\text{DELTA}=100\text{ms}$ is recommended, corresponding to a feedback bandwidth of 16Kbps with 200 bytes per feedback message --- approximately 1.6% overhead for a 1Mbps flow. Furthermore, both simulation studies and frequency-domain analysis in [IETF-95] have established that a feedback interval below 250ms (i.e., more frequently than 4 feedback messages per second) will not break up the feedback control loop of NADA congestion control.

In calculating the non-linear warping of delay in (1), the current design uses fixed values of QTH for determining whether to perform the non-linear warping). Its value should be carefully tuned for different operational environments (e.g., over wired vs. wireless connections), so as to avoid the potential risk of prematurely discounting the congestion signal level. It is possible to adapt its value based on past observed patterns of queuing delay in the presence of packet losses. It needs to be noted that the non-linear warping mechanism may lead to multiple NADA streams stuck in loss-based mode when competing against each other.

In calculating the aggregate congestion signal x_{curr} , the choice of DMARK and DLOSS influence the steady-state packet loss/marketing ratio experienced by the flow at a given available bandwidth. Higher values of DMARK and DLOSS result in lower steady-state loss/marketing ratios, but are more susceptible to the impact of individual packet loss/marketing events. While the value of DMARK and DLOSS are fixed and predetermined in the current design, this document also encourages further explorations of a scheme for automatically tuning these values based on desired bandwidth sharing behavior in the presence of other competing loss-based flows (e.g., loss-based TCP).

6.4. Sender-based vs. receiver-based calculation

In the current design, the aggregated congestion signal `x_curr` is calculated at the receiver, keeping the sender operation completely independent of the form of actual network congestion indications (delay, loss, or marking) in use.

Alternatively, one can shift receiver-side calculations to the sender, whereby the receiver simply reports on per-packet information via periodic feedback messages as defined in [I-D.ietf-avtcore-cc-feedback-message]. Such an approach enables interoperability amongst senders operating on different congestion control schemes, but requires slightly higher overhead in the feedback messages. See additional discussions in [I-D.ietf-avtcore-cc-feedback-message] regarding the desired format of the feedback messages and the recommended feedback intervals.

6.5. Incremental deployment

One nice property of NADA is the consistent video endpoint behavior irrespective of network node variations. This facilitates gradual, incremental adoption of the scheme.

Initially, the proposed congestion control mechanism can be implemented without any explicit support from the network, and relies solely on observed relative one-way delay measurements and packet loss ratios as implicit congestion signals.

When ECN is enabled at the network nodes with RED-based marking, the receiver can fold its observations of ECN markings into the calculation of the equivalent delay. The sender can react to these explicit congestion signals without any modification.

Ultimately, networks equipped with proactive marking based on token bucket level metering can reap the additional benefits of zero standing queues and lower end-to-end delay and work seamlessly with existing senders and receivers.

7. Reference Implementations

The NADA scheme has been implemented in both [ns-2] and [ns-3] simulation platforms. The implementation in ns-2 hosts the calculations as described in Section 4.2 at the receiver side, whereas the implementation in ns-3 hosts these receiver-side calculations at the sender for the sake of interoperability. Extensive ns-2 simulation evaluations of an earlier version of the draft are documented in [Zhu-PV13]. An open source implementation of NADA as part of a ns-3 module is available at [ns3-rmcat].

Evaluation results of the current draft based on ns-3 are presented in [IETF-90] and [IETF-91] for wired test cases as documented in [I-D.ietf-rmcat-eval-test]. Evaluation results of NADA over WiFi-based test cases as defined in [I-D.ietf-rmcat-wireless-tests] are presented in [IETF-93]. These simulation-based evaluations have shown that NADA flows can obtain their fair share of bandwidth when competing against each other. They typically adapt fast in reaction to the arrival and departure of other flows, and can sustain a reasonable throughput when competing against loss-based TCP flows.

[IETF-90] describes the implementation and evaluation of NADA in a lab setting. Preliminary evaluation results of NADA in single-flow and multi-flow test scenarios have been presented in [IETF-91].

A reference implementation of NADA has been carried out by modifying the WebRTC module embedded in the Mozilla open source browser. Presentations from [IETF-103] and [IETF-105] document real-world evaluations of the modified browser driven by NADA. The experimental setting involve remote connections with endpoints over either home or enterprise wireless networks. These evaluations validate the effectiveness of NADA flows in recovering quickly from throughput drops caused by intermittent delay spikes over the last-hop wireless connections.

8. Suggested Experiments

NADA has been extensively evaluated under various test scenarios, including the collection of test cases specified by [I-D.ietf-rmcat-eval-test] and the subset of WiFi-based test cases in [I-D.ietf-rmcat-wireless-tests]. Additional evaluations have been carried out to characterize how NADA interacts with various active queue management (AQM) schemes such as RED, CoDel, and PIE. Most of these evaluations have been carried out in simulators. A few key test cases have been evaluated in lab environments with implementations embedded in video conferencing clients. It is strongly recommended to carry out implementation and experimentation of NADA in real-world settings. Such exercise will provide insights on how to choose or automatically adapt the values of the key algorithm parameters (see list in Figure 3) as discussed in Section 6.

Additional experiments are suggested for the following scenarios and preferably over real-world networks:

- o Experiments reflecting the setup of a typical WAN connection.
- o Experiments with ECN marking capability turned on at the network for existing test cases.

- o Experiments with multiple NADA streams bearing different user-specified priorities.
- o Experiments with additional access technologies, especially over cellular networks such as 3G/LTE.
- o Experiments with various media source contents, including audio only, audio and video, and application content sharing (e.g., slide shows).

9. IANA Considerations

This document makes no request of IANA.

10. Security Considerations

The rate adaptation mechanism in NADA relies on feedback from the receiver. As such, it is vulnerable to attacks where feedback messages are hijacked, replaced, or intentionally injected with misleading information resulting in denial of service, similar to those that can affect TCP. It is therefore RECOMMENDED that the RTCP feedback message is at least integrity checked. In addition, [I-D.ietf-avtcore-cc-feedback-message] discusses the potential risk of a receiver providing misleading congestion feedback information and the mechanisms for mitigating such risks.

The modification of sending rate based on send-side rate shaping buffer may lead to temporary excessive congestion over the network in the presence of a unresponsive video encoder. However, this effect can be mitigated by limiting the amount of rate modification introduced by the rate shaping buffer, bounding the size of the rate shaping buffer at the sender, and maintaining a maximum allowed sending rate by NADA.

11. Acknowledgments

The authors would like to thank Randell Jesup, Luca De Cicco, Piers O'Hanlon, Ingemar Johansson, Stefan Holmer, Cesar Ilharco Magalhaes, Safiqul Islam, Michael Welzl, Mirja Kuhlewind, Karen Elisabeth Egede Nielsen, Julius Flohr, Roland Bless, Andreas Smas, and Martin Stiernerling for their valuable review comments and helpful input to this specification.

12. Contributors

The following individuals have contributed to the implementation and evaluation of the proposed scheme, and therefore have helped to validate and substantially improve this specification.

Paul E. Jones <paulej@packetizer.com> of Cisco Systems implemented an early version of the NADA congestion control scheme and helped with its lab-based testbed evaluations.

Jiantao Fu <jianfu@cisco.com> of Cisco Systems helped with the implementation and extensive evaluation of NADA both in Mozilla web browsers and in earlier simulation-based evaluation efforts.

Stefano D'Aronco <stefano.daronco@geod.baug.ethz.ch> of ETH Zurich (previously at Ecole Polytechnique Federale de Lausanne when contributing to this work) helped with implementation and evaluation of an early version of NADA in [ns-3].

Charles Ganzhorn <charles.ganzhorn@gmail.com> contributed to the testbed-based evaluation of NADA during an early stage of its development.

13. References

13.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC3168] Ramakrishnan, K., Floyd, S., and D. Black, "The Addition of Explicit Congestion Notification (ECN) to IP", RFC 3168, DOI 10.17487/RFC3168, September 2001, <<https://www.rfc-editor.org/info/rfc3168>>.
- [RFC3550] Schulzrinne, H., Casner, S., Frederick, R., and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications", STD 64, RFC 3550, DOI 10.17487/RFC3550, July 2003, <<https://www.rfc-editor.org/info/rfc3550>>.
- [RFC5348] Floyd, S., Handley, M., Padhye, J., and J. Widmer, "TCP Friendly Rate Control (TFRC): Protocol Specification", RFC 5348, DOI 10.17487/RFC5348, September 2008, <<https://www.rfc-editor.org/info/rfc5348>>.
- [RFC6679] Westerlund, M., Johansson, I., Perkins, C., O'Hanlon, P., and K. Carlberg, "Explicit Congestion Notification (ECN) for RTP over UDP", RFC 6679, DOI 10.17487/RFC6679, August 2012, <<https://www.rfc-editor.org/info/rfc6679>>.

- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.

13.2. Informative References

- [Budzisz-TON11]
Budzisz, L., Stanojevic, R., Schlote, A., Baker, F., and R. Shorten, "On the Fair Coexistence of Loss- and Delay-Based TCP", IEEE/ACM Transactions on Networking vol. 19, no. 6, pp. 1811-1824, December 2011.
- [Floyd-CCR00]
Floyd, S., Handley, M., Padhye, J., and J. Widmer, "Equation-based Congestion Control for Unicast Applications", ACM SIGCOMM Computer Communications Review vol. 30, no. 4, pp. 43-56, October 2000.
- [I-D.ietf-avtcore-cc-feedback-message]
Sarker, Z., Perkins, C., Singh, V., and M. Ramalho, "RTP Control Protocol (RTCP) Feedback for Congestion Control", draft-ietf-avtcore-cc-feedback-message-04 (work in progress), July 2019.
- [I-D.ietf-rmcat-cc-codec-interactions]
Zanaty, M., Singh, V., Nandakumar, S., and Z. Sarker, "Congestion Control and Codec interactions in RTP Applications", draft-ietf-rmcat-cc-codec-interactions-02 (work in progress), March 2016.
- [I-D.ietf-rmcat-cc-requirements]
Jesup, R. and Z. Sarker, "Congestion Control Requirements for Interactive Real-Time Media", draft-ietf-rmcat-cc-requirements-09 (work in progress), December 2014.
- [I-D.ietf-rmcat-eval-test]
Sarker, Z., Singh, V., Zhu, X., and M. Ramalho, "Test Cases for Evaluating RMCAT Proposals", draft-ietf-rmcat-eval-test-10 (work in progress), May 2019.
- [I-D.ietf-rmcat-wireless-tests]
Sarker, Z., Johansson, I., Zhu, X., Fu, J., Tan, W., and M. Ramalho, "Evaluation Test Cases for Interactive Real-Time Media over Wireless Networks", draft-ietf-rmcat-wireless-tests-08 (work in progress), July 2019.

- [IETF-103] Zhu, X., Pan, R., Ramalho, M., Mena, S., Jones, P., Fu, J., and S. D'Aronco, "NADA Implementation in Mozilla Browser", November 2018, <<https://datatracker.ietf.org/meeting/103/materials/slides-103-rmcat-nada-implementation-in-mozilla-browser-00>>.
- [IETF-105] Zhu, X., Pan, R., Ramalho, M., Mena, S., Jones, P., Fu, J., and S. D'Aronco, "NADA Implementation in Mozilla Browser and Draft Update", July 2019, <<https://datatracker.ietf.org/meeting/105/materials/slides-105-rmcat-nada-update-02.pdf>>.
- [IETF-90] Zhu, X., Ramalho, M., Ganzhorn, C., Jones, P., and R. Pan, "NADA Update: Algorithm, Implementation, and Test Case Evaluation Results", July 2014, <<https://tools.ietf.org/agenda/90/slides/slides-90-rmcat-6.pdf>>.
- [IETF-91] Zhu, X., Pan, R., Ramalho, M., Mena, S., Ganzhorn, C., Jones, P., and S. D'Aronco, "NADA Algorithm Update and Test Case Evaluations", November 2014, <<http://www.ietf.org/proceedings/interim/2014/11/09/rmcat/slides/slides-interim-2014-rmcat-1-2.pdf>>.
- [IETF-93] Zhu, X., Pan, R., Ramalho, M., Mena, S., Ganzhorn, C., Jones, P., D'Aronco, S., and J. Fu, "Updates on NADA", July 2015, <<https://www.ietf.org/proceedings/93/slides/slides-93-rmcat-0.pdf>>.
- [IETF-95] Zhu, X., Pan, R., Ramalho, M., Mena, S., Jones, P., Fu, J., D'Aronco, S., and C. Ganzhorn, "Updates on NADA: Stability Analysis and Impact of Feedback Intervals", April 2016, <<https://www.ietf.org/proceedings/95/slides/slides-95-rmcat-5.pdf>>.
- [ns-2] "The Network Simulator - ns-2", <<http://www.isi.edu/nsnam/ns/>>.
- [ns-3] "The Network Simulator - ns-3", <<https://www.nsnam.org/>>.
- [ns3-rmcat] Fu, J., Mena, S., and X. Zhu, "NS3 open source module of IETF RMCAT congestion control protocols", November 2017, <<https://github.com/cisco/ns3-rmcat>>.

- [RFC5450] Singer, D. and H. Desineni, "Transmission Time Offsets in RTP Streams", RFC 5450, DOI 10.17487/RFC5450, March 2009, <<https://www.rfc-editor.org/info/rfc5450>>.
- [RFC6660] Briscoe, B., Moncaster, T., and M. Menth, "Encoding Three Pre-Congestion Notification (PCN) States in the IP Header Using a Single Diffserv Codepoint (DSCP)", RFC 6660, DOI 10.17487/RFC6660, July 2012, <<https://www.rfc-editor.org/info/rfc6660>>.
- [RFC6817] Shalunov, S., Hazel, G., Iyengar, J., and M. Kuehlewind, "Low Extra Delay Background Transport (LEDBAT)", RFC 6817, DOI 10.17487/RFC6817, December 2012, <<https://www.rfc-editor.org/info/rfc6817>>.
- [RFC7567] Baker, F., Ed. and G. Fairhurst, Ed., "IETF Recommendations Regarding Active Queue Management", BCP 197, RFC 7567, DOI 10.17487/RFC7567, July 2015, <<https://www.rfc-editor.org/info/rfc7567>>.
- [RFC8033] Pan, R., Natarajan, P., Baker, F., and G. White, "Proportional Integral Controller Enhanced (PIE): A Lightweight Control Scheme to Address the Bufferbloat Problem", RFC 8033, DOI 10.17487/RFC8033, February 2017, <<https://www.rfc-editor.org/info/rfc8033>>.
- [RFC8290] Hoeiland-Joergensen, T., McKeeney, P., Taht, D., Gettys, J., and E. Dumazet, "The Flow Queue CoDel Packet Scheduler and Active Queue Management Algorithm", RFC 8290, DOI 10.17487/RFC8290, January 2018, <<https://www.rfc-editor.org/info/rfc8290>>.
- [RFC8593] Zhu, X., Mena, S., and Z. Sarker, "Video Traffic Models for RTP Congestion Control Evaluations", RFC 8593, DOI 10.17487/RFC8593, May 2019, <<https://www.rfc-editor.org/info/rfc8593>>.
- [Zhu-PV13] Zhu, X. and R. Pan, "NADA: A Unified Congestion Control Scheme for Low-Latency Interactive Video", in Proc. IEEE International Packet Video Workshop (PV'13) San Jose, CA, USA, December 2013.

Appendix A. Network Node Operations

NADA can work with different network queue management schemes and does not assume any specific network node operation. As an example, this appendix describes three variants of queue management behavior

at the network node, leading to either implicit or explicit congestion signals. It needs to be acknowledged that NADA has not yet been tested with non-probabilistic ECN marking behaviors.

In all three flavors described below, the network queue operates with the simple first-in-first-out (FIFO) principle. There is no need to maintain per-flow state. The system can scale easily with a large number of video flows and at high link capacity.

A.1. Default behavior of drop tail queues

In a conventional network with drop tail or RED queues, congestion is inferred from the estimation of end-to-end delay and/or packet loss. Packet drops at the queue are detected at the receiver, and contributes to the calculation of the aggregated congestion signal x_{curr} . No special action is required at network node.

A.2. RED-based ECN marking

In this mode, the network node randomly marks the ECN field in the IP packet header following the Random Early Detection (RED) algorithm [RFC7567]. Calculation of the marking probability involves the following steps:

on packet arrival:

update smoothed queue size q_{avg} as:

$$q_{avg} = w \cdot q + (1-w) \cdot q_{avg}.$$

calculate marking probability p as:

$$p = \begin{cases} 0, & \text{if } q < q_{lo}; \\ p_{max} \cdot \frac{q_{avg} - q_{lo}}{q_{hi} - q_{lo}}, & \text{if } q_{lo} \leq q < q_{hi}; \\ 1, & \text{if } q \geq q_{hi}. \end{cases}$$

Here, q_{lo} and q_{hi} corresponds to the low and high thresholds of queue occupancy. The maximum marking probability is p_{max} .

The ECN markings events will contribute to the calculation of an equivalent delay x_{curr} at the receiver. No changes are required at the sender.

A.3. Random Early Marking with Virtual Queues

Advanced network nodes may support random early marking based on a token bucket algorithm originally designed for Pre-Congestion Notification (PCN) [RFC6660]. The early congestion notification (ECN) bit in the IP header of packets are marked randomly. The marking probability is calculated based on a token-bucket algorithm originally designed for the Pre-Congestion Notification (PCN) [RFC6660]. The target link utilization is set as 90%; the marking probability is designed to grow linearly with the token bucket size when it varies between 1/3 and 2/3 of the full token bucket limit.

Calculation of the marking probability involves the following steps:

```

upon packet arrival:
    meter packet against token bucket (r,b);

    update token level b_tk;

    calculate the marking probability as:

        / 0,                                if b-b_tk < b_lo;
        |
        |      b-b_tk-b_lo
        |      -----, if b_lo<= b-b_tk <b_hi;
        |      b_hi-b_lo
p = <
        |
        \ 1,                                if b-b_tk>=b_hi.

```

Here, the token bucket lower and upper limits are denoted by b_{lo} and b_{hi} , respectively. The parameter b indicates the size of the token bucket. The parameter r is chosen to be below capacity, resulting in slight under-utilization of the link. The maximum marking probability is p_{max} .

The ECN markings events will contribute to the calculation of an equivalent delay x_{curr} at the receiver. No changes are required at the sender. The virtual queuing mechanism from the PCN-based marking algorithm will lead to additional benefits such as zero standing queues.

Authors' Addresses

Xiaoqing Zhu
Cisco Systems
12515 Research Blvd., Building 4
Austin, TX 78759
USA

Email: xiaoqzhu@cisco.com

Rong Pan *
* Pending affiliation change.

Email: rong.pan@gmail.com

Michael A. Ramalho
Cisco Systems, Inc.
8000 Hawkins Road
Sarasota, FL 34241
USA

Phone: +1 919 476 2038
Email: mar42@cornell.edu

Sergio Mena de la Cruz
Cisco Systems
EPFL, Quartier de l'Innovation, Batiment E
Ecublens, Vaud 1015
Switzerland

Email: semena@cisco.com

Network Working Group
Internet-Draft
Intended status: Informational
Expires: July 1, 2022

C. Perkins
University of Glasgow
December 28, 2021

Sending RTP Control Protocol (RTCP) Feedback for Congestion Control in
Interactive Multimedia Conferences
draft-ietf-rmcat-rtp-cc-feedback-08

Abstract

This memo discusses the types of congestion control feedback that it is possible to send using the RTP Control Protocol (RTCP), and their suitability of use in implementing congestion control for unicast multimedia applications.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <https://datatracker.ietf.org/drafts/current/>.

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

This Internet-Draft will expire on July 1, 2022.

Copyright Notice

Copyright (c) 2021 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	2
2. Possible Models for RTCP Feedback	3
3. What Feedback is Achievable With RTCP?	5
3.1. Scenario 1: Voice Telephony	5
3.2. Scenario 2: Point-to-Point Video Conference	8
4. Discussion and Conclusions	11
5. Security Considerations	12
6. IANA Considerations	12
7. Acknowledgements	12
8. Informative References	12
Author's Address	15

1. Introduction

The deployment of WebRTC systems [RFC8825] has resulted in high-quality video conferencing seeing extremely wide use. To ensure the stability of the network in the face of this use, WebRTC systems need to use some form of congestion control for their RTP-based media traffic [RFC2914], [RFC8085], [RFC8083], [RFC8834], allowing them to adapt and adjust the media data they send to match changes in the available network capacity. In addition to ensuring the stable operation of the network, such adaptation is critical to ensuring a good user experience, since it allows the sender to match the media to the network capacity, rather than forcing the receiver to compensate for uncontrolled packet loss when the available capacity is exceeded.

To develop such congestion control, it is necessary to understand the sort of congestion feedback that can be provided within the framework of RTP [RFC3550] and the RTP Control Protocol (RTCP). It then becomes possible to determine if this is sufficient for congestion control, or if some form of RTP extension is needed.

This memo considers unicast congestion feedback that can be sent using RTCP under the RTP/SAVPF profile [RFC5124] (the secure version of the RTP/AVPF profile [RFC4585]). This profile was chosen as it forms the basis for media transport in WebRTC [RFC8834] systems. Nothing in this memo is specific to the secure version of the profile, or to WebRTC, however. It is also assumed that the congestion control feedback mechanism described in [RFC8888], and common RTCP extensions for efficient feedback [RFC5506], [RFC8108], [RFC8861], [RFC8861], and [RFC8872] are available.

2. Possible Models for RTCP Feedback

Several questions need to be answered when providing RTCP reception quality feedback for congestion control purposes. These include:

- o How often is feedback needed?
- o How much overhead is acceptable?
- o How much, and what, data does each report contain?

The key question is how often does the receiver need to send feedback on the reception quality it is experiencing, and hence the congestion state of the network?

Widely used transport protocols, such as TCP, send acknowledgements frequently. For example, a TCP receiver will send an acknowledgement at least once every 0.5 seconds or when new data equal to twice the maximum segment size has been received [I-D.ietf-tcpm-rfc793bis]). That has relatively low overhead when traffic is bidirectional and acknowledgements can be piggybacked onto return path data packets. It can also be acceptable, and can have reasonable overhead, to send separate acknowledgement packets when those packets are much smaller than data packets.

Frequent acknowledgements can become a problem, however, when there is no return traffic on which to piggyback feedback, or if separate feedback and data packets are sent and the feedback is similar in size to the data being acknowledged. This can be the case for some forms of media traffic, especially for voice over IP flows, leading to high overhead when using a transport protocol that sends frequent feedback. Approaches like in-network filtering of acknowledgements can reduce the feedback frequency and overhead in some cases, but this so-called "stretch-ACK" behaviour is non-standard and not guaranteed.

Accordingly, when implementing congestion control for RTP-based multimedia traffic, it might make sense to give the option of sending congestion feedback less often than does TCP. For example, it might be possible to send a feedback packet once per video frame, or every few frames, or once per network round trip time (RTT). This could still give sufficiently frequent feedback for the congestion control loop to be stable and responsive while keeping the overhead reasonable when the feedback cannot be piggybacked onto returning data. In this case, it is important to note that RTCP can send much more detailed feedback than simple acknowledgements. For example, if it were useful, it could be possible to use an RTCP extended report (XR) packet [RFC3611] to send feedback once per RTT comprising a

bitmap of lost and received packets, with reception times, over that RTT. As long as feedback is sent frequently enough that the control loop is stable, and the sender is kept informed when data leaves the network (to provide an equivalent to ACK clocking in TCP), it is not necessary to report on every packet at the instant it is received (indeed, it is unlikely that a video codec can react instantly to a rate change anyway, and there is little point in providing feedback more often than the codec can adapt).

Reducing the feedback frequency compared to TCP will reduce feedback overhead but will lead multimedia flows to adapt to congestion more slowly than TCP, raising concerns about inter-flow fairness. Similar concerns are noted in [RFC5348], and accordingly the congestion control algorithm described therein aims for "reasonable" fairness and a sending rate that is "generally within a factor of two" of that TCP would achieve under the same conditions. It is to be noted, however, that TCP exhibits inter-flow unfairness when flows with differing round-trip times compete, and stretch acknowledgements due to in-network traffic manipulation are not uncommon and also raise fairness concerns. Implementations need to balance potential unfairness against feedback overhead.

Generating and processing feedback consumes resources at the sender and receiver. The feedback packets also incur forwarding costs, contribute to link utilization, and can affect the timing of other traffic on the network. This can affect performance on some types of network, that can be impacted by the rate, timing, and size of feedback packets, as well as by the overall volume of feedback bytes.

The amount of overhead due to congestion control feedback that is considered acceptable has to be determined. RTCP feedback is sent in separate packets to RTP data, and this has some cost in terms of additional header overhead compared to protocols that piggyback feedback on return path data packets. The RTP standards have long said that a 5% overhead for RTCP traffic generally acceptable, while providing the ability to change this fraction. Is this still the case for congestion control feedback? Is there a desire to provide more responsive feedback and congestion control, possibility with a higher overhead? Or is lower overhead wanted, accepting that this might reduce responsiveness of the congestion control algorithm?

Finally, the details of how much, and what, data is to be sent in each report will affect the frequency and/or overhead of feedback. There is a fundamental trade-off that the more frequently feedback packets are sent, the less data can be included in each packet to keep the overhead constant. Does the congestion control need high rate but simple feedback (e.g., like TCP acknowledgements), or is it acceptable to send more complex feedback less often? Is it useful

for the congestion control to receive frequent feedback, perhaps to provide more accurate round-trip time estimates, or to provide robustness in case feedback packets are lost, even if the media sending rate cannot quickly be changed? Or is low-rate feedback, resulting in slowly responsive changes the sending rate, acceptable? Different combinations of congestion control algorithm and media codec might require different trade-offs, and the correct trade-off for interactive, self-paced, real-time multimedia traffic might not be the same as that for TCP congestion control.

3. What Feedback is Achievable With RTCP?

The following sections illustrate how the RTCP congestion control feedback report [RFC8888] can be used in different scenarios, and illustrate the overheads of this approach.

3.1. Scenario 1: Voice Telephony

In many ways, point-to-point voice telephony is the simplest scenario for congestion control, since there is only a single media stream to control. It's complicated, however, by severe bandwidth constraints on the feedback, to keep the overhead manageable.

Assume a two-party point-to-point voice-over-IP call, using RTP over UDP/IP. A rate adaptive speech codec, such as Opus, is used, encoded into RTP packets in frames of duration T_f seconds ($T_f = 20\text{ms}$ in many cases, but values up to 60ms are not uncommon). The congestion control algorithm requires feedback every N_r frames, i.e., every $N_r * T_f$ seconds, to ensure effective control. Both parties in the call send speech data or comfort noise with sufficient frequency that they are counted as senders for the purpose of the RTCP reporting interval calculation.

RTCP feedback packets can be full, compound, RTCP feedback packets, or non-compound RTCP packets [RFC5506]. A compound RTCP packet is sent once for every N_{nc} non-compound RTCP packets.

Compound RTCP packets contain a Sender Report (SR) packet, a Source Description (SDS) packet, and an RTP Congestion Control Feedback (CCFB) packet [RFC8888]. Non-compound RTCP packets contain only the CCFB packet. Since each participant sends only a single RTP media stream, the extensions for RTCP report aggregation [RFC8108] and reporting group optimisation [RFC8861] are not used.

Within each compound RTCP packet, the SR packet will contain a sender information block (28 octets) and a single reception report block (24 octets), for a total of 52 octets. A minimal SDS packet will contain a header (4 octets) and a single chunk containing an SSRC (4

octets) and a CNAME item, and if the recommendations for choosing the CNAME [RFC7022] are followed, the CNAME item will comprise a 2 octet header, 16 octets of data, and 2 octets of padding, for a total SDES packet size of 28 octets. The CCFB packets contains an RTCP header and SSRC (8 octets), a report timestamp (4 octets), the SSRC, beginning and ending sequence numbers (8 octets), and $2*Nr$ octets of reports, for a total of $20 + 2*Nr$ octets. The compound Secure RTCP packet will include 4 octets of trailer followed by an 80 bit (10 octet) authentication tag if HMAC-SHA1 authentication is used. If IPv4 is used, with no IP options, the UDP/IP header will be 28 octets in size. This gives a total compound RTCP packet size of $Sc = 142 + 2*Nr$ octets.

The non-compound RTCP packets will comprise just the CCFB packet, SRTCP trailer and authentication tag, and a UDP/IP header. It can be seen that these packets will be $Snc = 62 + 2*Nr$ octets in size.

The RTCP reporting interval calculation ([RFC3550], Section 6.2) for a two-party session where both participants are senders, reduces to:

$$Trtcp = n * Srtcp / Brtcp$$

where $Srtcp = (Sc + Nnc * Snc) / (1 + Nnc)$ is the average RTCP packet size in octets, $Brtcp$ is the bandwidth allocated to RTCP in octets per second, and n is the number of participants in the RTP session (in this scenario, $n = 2$).

To ensure an RTCP report containing congestion control feedback is sent after every Nr frames of audio, it is necessary to set the RTCP reporting interval $Trtcp = Nr * Tf$, which when substituted into the previous gives $Nr * Tf = n * Srtcp / Brtcp$. Solving this to give the RTCP bandwidth, $Brtcp$, and expanding the definition of $Srtcp$ gives:

$$Brtcp = (n * (Sc + Nnc * Snc)) / (Nr * Tf * (1 + Nnc)).$$

If we assume every report is a compound RTCP packet (i.e., $Nnc = 0$), the frame duration $Tf = 20ms$, and an RTCP report is sent for every second frame (i.e., 25 RTCP reports per second), this gives an RTCP feedback bandwidth, $Brtcp = 57kbps$. Increasing the frame duration, or reducing the frequency of reports, will reduce the RTCP bandwidth as shown in Table 1.

Tf (seconds)	Nr (frames)	rtcp_bw (kbps)
0.020	2	57.0
0.020	4	29.3
0.020	8	15.4
0.020	16	8.5
0.060	2	19.0
0.060	4	9.8
0.060	8	5.1
0.060	16	2.8

Table 1: RTCP bandwidth needed for VoIP feedback

The final row of Table 1 (60ms frames, report every 16 frames) sends RTCP reports once per second, giving an RTCP bandwidth overhead of 2.8kbps.

The overhead can be reduced by sending some reports in non-compound RTCP packets [RFC5506]. For example, if we alternate compound and non-compound RTCP packets, i.e., $N_{nc} = 1$, the calculation gives the results shown in Table 2.

Tf (seconds)	Nr (frames)	rtcp_bw (kbps)
0.020	2	41.4
0.020	4	21.5
0.020	8	11.5
0.020	16	6.5
0.060	2	13.8
0.060	4	7.2
0.060	8	3.8
0.060	16	2.2

Table 2: Required RTCP bandwidth for VoIP feedback (alternating compound and non-compound reports)

The RTCP bandwidth needed for 60ms frames, reporting every 16 frames (once per second), can be seen to drop to 2.2kbps. This calculation can be repeated for other patterns of compound and non-compound RTCP packets, feedback frequency, and frame duration, as needed.

Note: To achieve the RTCP transmission intervals above the RTP/SAVPF profile with $T_{rr_interval}=0$ is used, since even when using the reduced minimal transmission interval, the RTP/SAVP profile would

only allow sending RTCP at most every 0.11s (every third frame of video). Using RTP/SAVPF with `T_rr_interval=0` however is capable of fully utilizing the configured 5% RTCP bandwidth fraction.

3.2. Scenario 2: Point-to-Point Video Conference

Consider a point-to-point video call between two end systems. There will be four RTP flows in this scenario, two audio and two video, with all four flows being active for essentially all the time (the audio flows will likely use voice activity detection and comfort noise to reduce the packet rate during silent periods, but this does not cause the transmissions to stop).

Assume all four flows are sent in a single RTP session, each using a separate SSRC. The RTCP reports from the co-located audio and video SSRCs at each end point are aggregated [RFC8108], the optimisations in [RFC8861] are used, and RTCP congestion control feedback is sent [RFC8888].

When all members are senders, the RTCP reporting interval calculation in Section 6.2 and 6.3 of [RFC3550] and [RFC4585] reduces to:

$$Trtcp = n * Srtcp / Brtcp$$

where `n` is the number of members in the session, `Srtcp` is the average RTCP packet size in octets, and the `Brtcp` is the RTCP bandwidth in octets per second.

The average RTCP packet size, `Srtcp`, depends on the amount of feedback sent in each RTCP packet, on the number of members in the session, on the size of source description (RTCP SDES) information sent, and on the amount of congestion control feedback sent in each packet.

As a baseline, each RTCP packet will be a compound RTCP packet that contains an aggregate of a compound RTCP packet generated by the video SSRC and a compound RTCP packet generated by the audio SSRC. When the RTCP reporting group extensions are used, one of these SSRCs will be a reporting SSRC, to which the other SSRC will have delegated its reports. No non-compound RTCP packets are sent.

The aggregated compound RTCP packet from the non-reporting SSRC will contain an RTCP SR packet, an RTCP SDES packet, and an RTCP RGRS packet. The RTCP SR packet contains the 28 octet header and sender information, but no report blocks (since the reporting is delegated). The RTCP SDES packet will comprise a header (4 octets), originating SSRC (4 octets), a CNAME chunk, a terminating chunk, and any padding. If the CNAME follows [RFC7022] and [RFC8834] it will be 18 octets in

size, and will need 1 octet of padding, making the SDES packet 28 octets in size. The RTCP RGRS packet will be 12 octets in size. This gives a total of $28 + 28 + 12 = 68$ octets.

The aggregated compound RTCP packet from the reporting SSRC will contain an RTCP SR packet, an RTCP SDES packet, and an RTCP congestion control feedback packet. The RTCP SR packet will contain two report blocks, one for each of the remote SSRCs (the report for the other local SSRC is suppressed by the reporting group extension), for a total of $28 + (2 * 24) = 76$ octets. The RTCP SDES packet will comprise a header (4 octets), originating SSRC (4 octets), a CNAME chunk, an RGRP chunk, a terminating chunk, and any padding. If the CNAME follows [RFC7022] and [RFC8834] it will be 18 octets in size. The RGRP chunk similarly comprises 18 octets, and 3 octets of padding are needed, for a total of 48 octets. The RTCP congestion control feedback (CCFB) report comprises an 8 octet RTCP header and SSRC, a 4 octet report timestamp, and for each of the remote audio and video SSRCs, an 8 octet report header, and 2 octets per packet reported upon, and padding to a 4 octet boundary if needed; that is $8 + 4 + 8 + (2 * N_v) + 8 + (2 * N_a)$ where N_v is the number of video packets per report, and N_a is the number of audio packets per report.

The complete compound RTCP packet contains the RTCP packets from both the reporting and non-reporting SSRCs, an SRTCP trailer and authentication tag, and a UDP/IPv4 header. The size of this RTCP packet is therefore: $262 + (2 * N_v) + (2 * N_a)$ octets. Since the aggregate RTCP packet contains reports from two SSRCs, the RTCP packet size is halved before use [RFC8108]. Accordingly, the size of the RTCP packets is:

$$Srtcp = (262 + (2 * N_v) + (2 * N_a)) / 2$$

How many RTP packets does the RTCP XR congestion control feedback packet included in these compound RTCP packets report on? That is, what are the values of N_v and N_a ? This depends on the RTCP reporting interval, $Trtcp$, the video bit rate and frame rate, R_f , the audio bit rate and framing interval, and whether the receiver chooses to send congestion control feedback in each RTCP packet it sends.

To simplify the calculation, assume it is desired to send one RTCP report for each frame of video received (i.e., $Trtcp = 1 / R_f$) and to include a congestion control feedback packet in each report. Assume that video has constant bit rate and frame rate, and that each frame of packet has to fit into a 1500 octet MTU. Further, assume that the audio takes negligible bandwidth, and that the audio framing interval can be varied within reasonable bounds, so that an integral number of audio frames align with video frame boundaries.

Table 3 shows the resulting values of N_v and N_a , the number of video and audio packets covered by each congestion control feedback report, for a range of data rates and video frame rates, assuming congestion control feedback is sent once per video frame. The table also shows the result of inverting the RTCP reporting interval calculation to find the corresponding RTCP bandwidth, B_{rtcp} . The RTCP bandwidth is given in kbps and as a fraction of the data rate.

It can be seen that, for example, with a data rate of 1024 kbps and video sent at 30 frames-per-second, the RTCP congestion control feedback report sent for each video frame will include reports on 3 video packets and 2 audio packets. The RTCP bandwidth needed to sustain this reporting rate is 127.5 kbps (12% of the data rate). This assumes an audio framing interval of 16.67ms, so that two audio packets are sent for each video frame.

Data Rate (kbps)	Video Frame Rate: R_f	Video Packets per Report: N_v	Audio Packets per Report: N_a	Required RTCP bandwidth: B_{rtcp} (kbps)
100	8	1	6	34.5 (34%)
200	16	1	3	67.5 (33%)
350	30	1	2	125.6 (35%)
700	30	2	2	126.6 (18%)
700	60	1	1	249.4 (35%)
1024	30	3	2	127.5 (12%)
1400	60	2	1	251.2 (17%)
2048	30	6	2	130.3 (6%)
2048	60	3	1	253.1 (12%)
4096	30	12	2	135.9 (3%)
4096	60	6	1	258.8 (6%)

Table 3: Required RTCP bandwidth, reporting on every frame

Use of reduced size RTCP [RFC5506] would allow the SR and SDES packets to be omitted from some reports. These "non-compound" (actually, compound but reduced size in this case) RTCP packets would contain an RTCP RGRS packet from the non-reporting SSRC, and an RTCP SDES RGRP packet and a congestion control feedback packet from the reporting SSRC. This will be $12 + 28 + 12 + 8 + 2*N_v + 8 + 2*N_a$ octets, plus the SRTCP trailer and authentication tag, and a UDP/IP header. That is, the size of the non-compound packets would be $(110 + 2*N_v + 2*N_a)/2$ octets. Repeating the analysis above, but alternating compound and non-compound reports gives results as shown in Table 4.

Data Rate (kbps)	Video Frame Rate: Rf	Video Packets per Report: Nv	Audio Packets per Report: Na	Required RTCP bandwidth: Brtcp (kbps)
100	8	1	6	24.1 (24%)
200	16	1	3	46.8 (23%)
350	30	1	2	86.7 (24%)
700	30	2	2	87.7 (12%)
700	60	1	1	171.6 (24%)
1024	30	3	2	88.6 (8%)
1400	60	2	1	173.4 (12%)
2048	30	6	2	91.4 (4%)
2048	60	3	1	175.3 (8%)
4096	30	12	2	97.0 (2%)
4096	60	6	1	180.9 (4%)

Table 4: Required RTCP bandwidth, reporting on every frame, with reduced-size reports

The use of reduced-size RTCP gives a noticeable reduction in the needed RTCP bandwidth, and can be combined with reporting every few frames rather than every frames. Overall, it is clear that the RTCP overhead can be reasonable across the range of data and frame rates, if RTCP is configured carefully.

4. Discussion and Conclusions

Practical systems will generally send some non-media traffic on the same path as the media traffic. This can include STUN/TURN packets to keep-alive NAT bindings [RFC8445], WebRTC Data Channel packets [RFC8831], etc. Such traffic also needs congestion control, but the means by which this is achieved is out of scope of this memo.

RTCP as it is currently specified cannot be used to send per-packet congestion feedback with reasonable overhead.

RTCP can, however, be used to send congestion feedback on each frame of video sent, provided the session bandwidth exceeds a couple of megabits per second (the exact rate depending on the number of session participants, the RTCP bandwidth fraction, and what RTCP extensions are enabled, and how much detail of feedback is needed). For lower rate sessions, the overhead of reporting on every frame becomes high, but can be reduced to something reasonable by sending reports once per N frames (e.g., every second frame), or by sending non-compound RTCP reports in between the regular reports.

If it is desired to use RTCP in something close to its current form for congestion feedback in WebRTC, the multimedia congestion control algorithm needs to be designed to work with feedback sent every few frames, since that fits within the limitations of RTCP. The provided feedback will be more detailed than just an acknowledgement, however, and will provide a loss bitmap, relative arrival time, and received ECN marks, for each packet sent. This will allow congestion control that is effective, if slowly responsive, to be implemented (there is guidance on providing effective congestion control in Section 3.1 of [RFC8085]).

The format described in [RFC8888] seems sufficient for the needs of congestion control feedback. There is little point optimising this format: the main overhead comes from the UDP/IP headers and the other RTCP packets included in the compound packets, and can be lowered by using the [RFC5506] extensions and sending reports less frequently. The use of header compression [RFC2508], [RFC3545], [RFC5795] can also be beneficial.

Further study of the scenarios of interest is needed, to ensure that the analysis presented is applicable to other media topologies, and to sessions with different data rates and sizes of membership.

5. Security Considerations

An attacker that can modify or spoof RTCP congestion control feedback packets can manipulate the sender behaviour to cause denial of service. This can be prevented by authentication and integrity protection of RTCP packets, for example using the secure RTP profile [RFC3711][RFC5124], or by other means as discussed in [RFC7201].

6. IANA Considerations

There are no actions for IANA.

7. Acknowledgements

Thanks to Magnus Westerlund, Ingemar Johansson, Goran Fairhurst, and the members of the RMCAT feedback design team for their feedback.

8. Informative References

- [I-D.ietf-tcpm-rfc793bis]
Eddy, W., "Transmission Control Protocol (TCP) Specification", draft-ietf-tcpm-rfc793bis-20 (work in progress), January 2021.

- [RFC2508] Casner, S. and V. Jacobson, "Compressing IP/UDP/RTP Headers for Low-Speed Serial Links", RFC 2508, DOI 10.17487/RFC2508, February 1999, <<https://www.rfc-editor.org/info/rfc2508>>.
- [RFC2914] Floyd, S., "Congestion Control Principles", BCP 41, RFC 2914, DOI 10.17487/RFC2914, September 2000, <<https://www.rfc-editor.org/info/rfc2914>>.
- [RFC3545] Koren, T., Casner, S., Geevarghese, J., Thompson, B., and P. Ruddy, "Enhanced Compressed RTP (CRTP) for Links with High Delay, Packet Loss and Reordering", RFC 3545, DOI 10.17487/RFC3545, July 2003, <<https://www.rfc-editor.org/info/rfc3545>>.
- [RFC3550] Schulzrinne, H., Casner, S., Frederick, R., and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications", STD 64, RFC 3550, DOI 10.17487/RFC3550, July 2003, <<https://www.rfc-editor.org/info/rfc3550>>.
- [RFC3611] Friedman, T., Ed., Caceres, R., Ed., and A. Clark, Ed., "RTP Control Protocol Extended Reports (RTCP XR)", RFC 3611, DOI 10.17487/RFC3611, November 2003, <<https://www.rfc-editor.org/info/rfc3611>>.
- [RFC3711] Baugher, M., McGrew, D., Naslund, M., Carrara, E., and K. Norrman, "The Secure Real-time Transport Protocol (SRTP)", RFC 3711, DOI 10.17487/RFC3711, March 2004, <<https://www.rfc-editor.org/info/rfc3711>>.
- [RFC4585] Ott, J., Wenger, S., Sato, N., Burmeister, C., and J. Rey, "Extended RTP Profile for Real-time Transport Control Protocol (RTCP)-Based Feedback (RTP/AVPF)", RFC 4585, DOI 10.17487/RFC4585, July 2006, <<https://www.rfc-editor.org/info/rfc4585>>.
- [RFC5124] Ott, J. and E. Carrara, "Extended Secure RTP Profile for Real-time Transport Control Protocol (RTCP)-Based Feedback (RTP/SAVPF)", RFC 5124, DOI 10.17487/RFC5124, February 2008, <<https://www.rfc-editor.org/info/rfc5124>>.
- [RFC5348] Floyd, S., Handley, M., Padhye, J., and J. Widmer, "TCP Friendly Rate Control (TFRC): Protocol Specification", RFC 5348, DOI 10.17487/RFC5348, September 2008, <<https://www.rfc-editor.org/info/rfc5348>>.

- [RFC5506] Johansson, I. and M. Westerlund, "Support for Reduced-Size Real-Time Transport Control Protocol (RTCP): Opportunities and Consequences", RFC 5506, DOI 10.17487/RFC5506, April 2009, <<https://www.rfc-editor.org/info/rfc5506>>.
- [RFC5795] Sandlund, K., Pelletier, G., and L-E. Jonsson, "The RObust Header Compression (ROHC) Framework", RFC 5795, DOI 10.17487/RFC5795, March 2010, <<https://www.rfc-editor.org/info/rfc5795>>.
- [RFC7022] Begen, A., Perkins, C., Wing, D., and E. Rescorla, "Guidelines for Choosing RTP Control Protocol (RTCP) Canonical Names (CNAMES)", RFC 7022, DOI 10.17487/RFC7022, September 2013, <<https://www.rfc-editor.org/info/rfc7022>>.
- [RFC7201] Westerlund, M. and C. Perkins, "Options for Securing RTP Sessions", RFC 7201, DOI 10.17487/RFC7201, April 2014, <<https://www.rfc-editor.org/info/rfc7201>>.
- [RFC8083] Perkins, C. and V. Singh, "Multimedia Congestion Control: Circuit Breakers for Unicast RTP Sessions", RFC 8083, DOI 10.17487/RFC8083, March 2017, <<https://www.rfc-editor.org/info/rfc8083>>.
- [RFC8085] Eggert, L., Fairhurst, G., and G. Shepherd, "UDP Usage Guidelines", BCP 145, RFC 8085, DOI 10.17487/RFC8085, March 2017, <<https://www.rfc-editor.org/info/rfc8085>>.
- [RFC8108] Lennox, J., Westerlund, M., Wu, Q., and C. Perkins, "Sending Multiple RTP Streams in a Single RTP Session", RFC 8108, DOI 10.17487/RFC8108, March 2017, <<https://www.rfc-editor.org/info/rfc8108>>.
- [RFC8445] Keranen, A., Holmberg, C., and J. Rosenberg, "Interactive Connectivity Establishment (ICE): A Protocol for Network Address Translator (NAT) Traversal", RFC 8445, DOI 10.17487/RFC8445, July 2018, <<https://www.rfc-editor.org/info/rfc8445>>.
- [RFC8825] Alvestrand, H., "Overview: Real-Time Protocols for Browser-Based Applications", RFC 8825, DOI 10.17487/RFC8825, January 2021, <<https://www.rfc-editor.org/info/rfc8825>>.
- [RFC8831] Jesup, R., Loreto, S., and M. Tuexen, "WebRTC Data Channels", RFC 8831, DOI 10.17487/RFC8831, January 2021, <<https://www.rfc-editor.org/info/rfc8831>>.

- [RFC8834] Perkins, C., Westerlund, M., and J. Ott, "Media Transport and Use of RTP in WebRTC", RFC 8834, DOI 10.17487/RFC8834, January 2021, <<https://www.rfc-editor.org/info/rfc8834>>.
- [RFC8861] Lennox, J., Westerlund, M., Wu, Q., and C. Perkins, "Sending Multiple RTP Streams in a Single RTP Session: Grouping RTP Control Protocol (RTCP) Reception Statistics and Other Feedback", RFC 8861, DOI 10.17487/RFC8861, January 2021, <<https://www.rfc-editor.org/info/rfc8861>>.
- [RFC8872] Westerlund, M., Burman, B., Perkins, C., Alvestrand, H., and R. Even, "Guidelines for Using the Multiplexing Features of RTP to Support Multiple Media Streams", RFC 8872, DOI 10.17487/RFC8872, January 2021, <<https://www.rfc-editor.org/info/rfc8872>>.
- [RFC8888] Sarker, Z., Perkins, C., Singh, V., and M. Ramalho, "RTP Control Protocol (RTCP) Feedback for Congestion Control", RFC 8888, DOI 10.17487/RFC8888, January 2021, <<https://www.rfc-editor.org/info/rfc8888>>.

Author's Address

Colin Perkins
University of Glasgow
School of Computing Science
Glasgow G12 8QQ
United Kingdom

Email: csp@csp Perkins