

IETF RMCAT Working Group
Internet-Draft
Intended status: Standards Track
Expires: May 1, 2017

Z. Sarker
Ericsson AB
C. Perkins
University of Glasgow
V. Singh
callstats.io
M. Ramalho
Cisco Systems
October 28, 2016

RTP Control Protocol (RTCP) Feedback for Congestion Control
draft-dt-rmcat-feedback-message-01

Abstract

This document describes a feedback message intended to enable congestion control for interactive real-time traffic. The RTP Media Congestion Avoidance Techniques (RMCAT) Working Group formed a design team to analyze feedback requirements from various congestion control algorithms and to design a generic feedback message to help ensure interoperability across those algorithms. The feedback message is designed for a sender-based congestion control, which means the receiver of the media will send necessary feedback to the sender of the media to perform the congestion control at the sender.

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 <http://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 1, 2017.

Copyright Notice

Copyright (c) 2016 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 (<http://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. Feedback Message	3
3.1. RTCP XR Block for Reporting Congestion Control Feedback .	4
3.2. RTP/AVPF Transport Layer Feedback for Congestion Control	6
4. Feedback Frequency and Overhead	7
5. Design Rationale	8
6. Acknowledgements	9
7. IANA Considerations	9
7.1. RTP/AVPF Transport Layer Feedback Message	9
7.2. RTCP XR Report Blocks	9
8. Security Considerations	9
9. References	9
9.1. Normative References	9
9.2. Informative References	10
Authors' Addresses	11

1. Introduction

For interactive real-time traffic the typical protocol choice is Realtime Transport Protocol (RTP) over User Datagram Protocol (UDP). RTP does not provide any guarantee of Quality of Service (QoS), reliable or timely delivery and expects the underlying transport protocol to do so. UDP alone certainly does not meet that expectation. However, RTP Control Protocol (RTCP) provides a mechanism to periodically send transport and media metrics to the media sender which can be utilized and extended for the purposes of RMCAT congestion control. For a congestion control algorithm which operates at the media sender, RTCP messages can be transmitted from the media receiver back to the media sender to enable congestion control. In the absence of standardized messages for this purpose, the congestion control algorithm designers have designed proprietary RTCP messages that convey only those parameters required for their respective designs. As a direct result, the different congestion control (a.k.a. rate adaptation) 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 generic congestion control feedback format.

To help achieve interoperability for unicast RTP congestion control, this memo proposes a common RTCP feedback format that can be used by NADA [I-D.ietf-rmcat-nada], SCReAM [I-D.ietf-rmcat-scream-cc], Google Congestion Control [I-D.ietf-rmcat-gcc] and Shared Bottleneck Detection [I-D.ietf-rmcat-sbd], and hopefully future RTP congestion control algorithms as well.

[Editor's Note: consider removing this part of the section in the later versions] In preparing this memo, we have considered the following:

- o What are the feedback requirements for the proposed RTP congestion control candidate solution?
- o Can we design a feedback message that is future proof, and general enough to meet the needs of algorithms that have yet to be defined?
- o Can we use existing RTCP Extended Report (XR) blocks and/or RTCP Feedback Messages? If not, what is the rationale behind new XR blocks and/or RTCP feedback messages?
- o What will be the wire format of the generic feedback message?

2. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

In addition the terminology defined in [RFC3550], [RFC3551], [RFC3611], [RFC4585], and [RFC5506] applies.

3. Feedback Message

The design team analyzed the feedback requirements from the different proposed candidate in RMCAT WG. The analysis showed some commonalities between the proposed solution candidate and some can be derived from other information. The design team has agreed to have following packet information block in the feedback message to satisfy different requirement analyzed.

- o Packet Identifier : RTP sequence number. The RTP packet header includes an incremental packet sequence number that the sender

needs to correlate packets sent at the sender with packets received at the receiver.

- o Packet Arrival Time : Arrival time stamp at the receiver of the media. The sender requires the arrival time stamp of the respective packet to determine delay and jitter the packet had experienced during transmission. In a sender based congestion control solution the sender requires to keep track of the sent packets - usually packet sequence number, packet size and packet send time. With the packet arrival time the sender can detect the delay and jitter information. Along with packet loss and delay information the sender can estimate the available bandwidth and thus adapt to the situation.
- o Packet Explicit Congestion Notification (ECN) Marking : If ECN [RFC3168] is used, it is necessary to report on the 2-bit ECN mark in received packets, indicating for each packet whether it is marked not-ECT, ECT(0), ECT(1), or ECN-CE. If the path on which the media traffic traversing is ECN capable then the sender can use the Congestion Experienced (ECN-CE) marking information for congestion control. It is important that the receiver sends the ECN-CE marking information of the packet back to the sender to take the advantages of ECN marking. Note that how the receiver gets the ECN marking information at application layer is out of the scope of this design team. Additional information for ECN use with RTP can be found at [RFC6679].

The feedback messages can have one or more of the above information blocks. For RTCP based feedback message the packet information block will be grouped by Synchronization Source (SSRC) identifier.

As a practical matter, we note that host Operating System (OS) process interruptions can occur at inopportune times. Thus, the recording of the sent times at the sender and arrival times at the receiver should be made 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 should be recorded at the latest opportunity prior to outputting the media packet at the sender (e.g., socket or RTP API) and the arrival time at the receiver (e.g., socket or RTP API) should be recorded at the earliest opportunity available to the receiver.

3.1. RTCP XR Block for Reporting Congestion Control Feedback

Congestion control feedback can be sent as part of a scheduled RTCP report, or as RTP/AVPF early feedback. If sent as part of a scheduled RTCP report, the feedback is sent as an XR block, as part of a regular compound RTCP packet. The format of the RTCP XR report

block is as follows (this will be preceded in the compound RTCP packet by an RTCP XR header, described in [RFC3611], that includes the SSRC of the report sender):

```

      0               1               2               3
      0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-----+-----+-----+-----+-----+-----+-----+-----+
|  BT=RC2F   | Report count |   Block Length = TBD   |
+-----+-----+-----+-----+-----+-----+-----+-----+
|                                     Report Timestamp (32bits)
+-----+-----+-----+-----+-----+-----+-----+-----+
|                                     SSRC of 1st media source
+-----+-----+-----+-----+-----+-----+-----+-----+
|          begin_seq          |          end_seq          |
+-----+-----+-----+-----+-----+-----+-----+-----+
|L|ECN|  Arrival time offset  | ...                      |
+-----+-----+-----+-----+-----+-----+-----+-----+
.
.
+-----+-----+-----+-----+-----+-----+-----+-----+
|                                     SSRC of nth media source
+-----+-----+-----+-----+-----+-----+-----+-----+
|          begin_seq          |          end_seq          |
+-----+-----+-----+-----+-----+-----+-----+-----+
|L|ECN|  Arrival time offset  | ...                      |
.
.
+-----+-----+-----+-----+-----+-----+-----+-----+

```

The XR Discard RLE report block uses the same format as specified for the loss and duplicate report blocks in [RFC3611]. The fields "block length", "begin_seq", and "end_seq" have the same semantics and representation as defined in [RFC3611]

Block Type (BT, 8 bits): The RMCAT congestion control feedback Report Block is identified by the constant RC2F. [Note to RFC Editor: Please replace RC2F with the IANA provided RTCP XR block type for this block.]

Report Count (8 bits): field describes the number of SSRCs reported by this report block. The number should at least be 1.

Report Timestamp (RTS, 32 bits): represents the timestamp when this report was generated. The sender of the feedback message decides on the wall-clock. Usually, it should be derived from the same wall-clock that is used for timestamping RTP packets arrival. Consistency in the unit and resolution (10th of millisecond should be good enough

) is important here. In addition, the media sender can ask for a specific resolution it wants.

Each sequence number between the begin_seq and end_seq (both inclusive) is represented by a packet metric block of 16-bits that contains the L, ECN, and ATO metrics. If an odd number of reports are included, i.e., end_seq - begin_seq is odd then it should be rounded up to four (4) bytes boundary. [FIXME : the solution will depend on the compression used (if any), revisit this if packet format is changed later]

L (1 bit): is a boolean to indicate if the packet was received. 0 represents that the packet was not yet received and all the subsequent bits (ECN and ATO) are also set to 0. 1 represent the packet was received and the subsequent bits in the block need to be parsed.

ECN (2 bits): is the echoed ECN mark of the packet (00 if not received or if ECN is not used).

Arrival time offset (ATO, 13 bits): it the relative arrival time of the RTP packets at the receiver before this feedback report was generated measured in milliseconds. It is calculated by subtracting the reception timestamp of the RTP packet denoted by this 16bit block and the timestamp (RTS) of this report.

[FIXME: should reserve 0xFFFF to mean anything greater than 0xFFFE? This needs to wait until we have fixed the packet format]

3.2. RTP/AVPF Transport Layer Feedback for Congestion Control

Congestion control feedback can also be sent in a non-compound RTCP packet [RFC5506] if the RTP/AVPF profile [RFC4585] or the RTP/SAVPF profile [RFC5124] is used. In this case, the congestion control feedback is sent as a Transport Layer FB message (RTCP packet type 205), with FMT=2 (congestion control feedback). The format of this RTCP packet is as follows:

```

      0               1               2               3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-----+-----+-----+-----+-----+-----+-----+-----+
|V=2|P| FMT = 2 |   PT = 205   |           length           |
+-----+-----+-----+-----+-----+-----+-----+-----+
|                               SSRC of packet sender          |
+-----+-----+-----+-----+-----+-----+-----+-----+
|                               SSRC of 1st media source        |
+-----+-----+-----+-----+-----+-----+-----+-----+
|           begin_seq           |           end_seq            |
+-----+-----+-----+-----+-----+-----+-----+-----+
|L|ECN|  Arrival time offset   | ...                          |
+-----+-----+-----+-----+-----+-----+-----+-----+
.                               .                               .
.                               .                               .
+-----+-----+-----+-----+-----+-----+-----+-----+
|                               SSRC of nth media source        |
+-----+-----+-----+-----+-----+-----+-----+-----+
|           begin_seq           |           end_seq            |
+-----+-----+-----+-----+-----+-----+-----+-----+
|L|ECN|  Arrival time offset   | ...                          |
.                               .                               .
.                               .                               .
+-----+-----+-----+-----+-----+-----+-----+-----+
|                               Report Timestamp (32bits)       |
+-----+-----+-----+-----+-----+-----+-----+-----+

```

The first 8 octets are the RTCP header, with PT=205 and FMT=2 specifying the remainder is a congestion control feedback packet, and including the SSRC of the packet sender.

Section 6.1 of [RFC4585] requires this is followed by the SSRC of the media source being reported upon. Accordingly, the format of the report is changed from that of the RTCP XR report block, to move the timestamp to the end. The meaning of all the fields is as described in Section 3.1.

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, and the possible rates of feedback.

It is a general understanding that the congestion control algorithms will work better with more frequent feedback - per packet feedback. However, RTCP bandwidth and transmission rules put some upper limits on how frequently the RTCP feedback messages can be sent from the

media receiver to the media sender. It has been shown [I-D.ietf-rmcat-rtp-cc-feedback] that in most cases a per frame feedback is a reasonable assumption on how frequent the RTCP feedback messages can be transmitted. The design team also have noted that even if a higher frequency of feedback is desired it is not viable if the feedback messages starts to compete against the media traffic on the feedback path during congestion period. Analyzing the feedback interval requirement [feedback-requirements] it can be seen that the candidate algorithms can perform with a feedback interval range of 50-200ms. A value within this range need to be negotiated at session setup.

5. Design Rationale

The primary function of RTCP Sender Report (SR) / Receiver Report (RR) is to report the reception quality of media. The regular SR / RR reports contain information about observed jitter, fractional packet loss and cumulative packet loss. The original intent of this information was to assist flow and congestion control mechanisms. Even though it is possible to do congestion control based on information provided in the SR/RR reports it is not sufficient to design an efficient congestion control algorithm for interactive real-time communication. An efficient congestion control algorithm requires more fine grain information on per packet (see Section 3) to react to the congestion or to avoid funder congestion on the path.

Codec Control Message for AVPF [RFC5104] defines Temporary Maximum Media Bit Rate (TMMBR) message which conveys a temporary maximum bitrate limitation from the receiver of the media to the sender of the media. Even though it is 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 with reduced sized reports [RFC5506]. This requires the receiver of the media to analyze the data reception, detect congestion level and recommend a maximum bitrate suitable for current available bandwidth on the path with an assumption that the sender of the media always honors the TMMBR message. This requirement is completely opposite of the sender based congestion control approach. Hence, TMMBR cannot be as a signaling means for a sender based congestion control mechanism. However, TMMBR should be viewed a complimentary signaling mechanism to establish receiver's current view of acceptable maximum bitrate which a sender based congestion control should honor.

There are number of RTCP eXtended Report (XR) blocks have been defined for reporting of delay, loss and ECN marking. It is possible to combine several XR blocks to report the loss and ECN marking at

the cost of overhead and complexity. However, there is no existing RTCP XR block to report packet arrival time.

Considering the issues discussed here it is rational to design a new congestion control feedback signaling mechanism for sender based congestion control algorithm.

6. Acknowledgements

This document is an outcome of RMCAT design team discussion. We would like to thank all participants specially Xiaoqing Zhu, Stefan Holmer, David, Ingemar Johansson and Randell Jesup for their valuable contribution to the discussions and to the document.

7. IANA Considerations

7.1. RTP/AVPF Transport Layer Feedback Message

TBD

7.2. RTCP XR Report Blocks

TBD

8. Security Considerations

There is a risk of causing congestion if an on-path attacker modifies the feedback messages in such a manner to make available bandwidth greater than it is in reality. [More on security consideration TBD.]

9. References

9.1. Normative References

- [I-D.ietf-rmcat-rtp-cc-feedback]
Perkins, C., "Using RTP Control Protocol (RTCP) Feedback for Unicast Multimedia Congestion Control", draft-ietf-rmcat-rtp-cc-feedback-01 (work in progress), July 2016.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<http://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, <<http://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, <<http://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, <<http://www.rfc-editor.org/info/rfc3551>>.
- [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, <<http://www.rfc-editor.org/info/rfc3611>>.
- [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, <<http://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, <<http://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, <<http://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, <<http://www.rfc-editor.org/info/rfc6679>>.

9.2. Informative References

- [feedback-requirements]
"RMCAT Feedback Requirements",
<[://www.ietf.org/proceedings/95/slides/slides-95-rmcat-1.pdf](http://www.ietf.org/proceedings/95/slides/slides-95-rmcat-1.pdf)>.
- [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-nada]

Zhu, X., Pan, R., Ramalho, M., Cruz, S., Jones, P., Fu, J., D'Aronco, S., and C. Ganzhorn, "NADA: A Unified Congestion Control Scheme for Real-Time Media", draft-ietf-rmcat-nada-03 (work in progress), September 2016.

[I-D.ietf-rmcat-sbd]

Hayes, D., Ferlin, S., Welzl, M., and K. Hiorth, "Shared Bottleneck Detection for Coupled Congestion Control for RTP Media.", draft-ietf-rmcat-sbd-05 (work in progress), September 2016.

[I-D.ietf-rmcat-scream-cc]

Johansson, I. and Z. Sarker, "Self-Clocked Rate Adaptation for Multimedia", draft-ietf-rmcat-scream-cc-06 (work in progress), August 2016.

[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, <<http://www.rfc-editor.org/info/rfc5104>>.

Authors' Addresses

Zaheduzzaman Sarker
Ericsson AB
Luleae
Sweden

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

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

Email: csp@cspcrkins.org

Varun Singh
Nemu Dialogue Systems Oy
Runeberginkatu 4c A 4
Helsinki 00100
Finland

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

Michael A. Ramalho
Cisco Systems, Inc.
6310 Watercrest Way Unit 203
Lakewood Ranch, FL 34202
USA

Phone: +1 919 476 2038
Email: mramalho@cisco.com

IETF RMCAT Working Group
Internet-Draft
Intended status: Standards Track
Expires: April 30, 2018

Z. Sarker
Ericsson AB
C. Perkins
University of Glasgow
V. Singh
callstats.io
M. Ramalho
Cisco Systems
October 27, 2017

RTP Control Protocol (RTCP) Feedback for Congestion Control
draft-dt-rmcat-feedback-message-04

Abstract

This document describes a feedback message intended to enable congestion control for interactive real-time traffic. The RTP Media Congestion Avoidance Techniques (RMCAT) Working Group formed a design team to analyze feedback requirements from various congestion control algorithms and to design a generic feedback message to help ensure interoperability across those algorithms. The feedback message is designed for a sender-based congestion control, which means the receiver of the media will send necessary feedback to the sender of the media to perform the congestion control at the sender.

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 April 30, 2018.

Copyright Notice

Copyright (c) 2017 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. Feedback Message	3
3.1. RTCP Congestion Control Feedback Report	4
4. Feedback Frequency and Overhead	6
5. Design Rationale	7
6. Acknowledgements	7
7. IANA Considerations	8
8. Security Considerations	8
9. References	8
9.1. Normative References	8
9.2. Informative References	9
Authors' Addresses	10

1. Introduction

For interactive real-time traffic the typical protocol choice is Realtime Transport Protocol (RTP) over User Datagram Protocol (UDP). RTP does not provide any guarantee of Quality of Service (QoS), reliable or timely delivery and expects the underlying transport protocol to do so. UDP alone certainly does not meet that expectation. However, RTP Control Protocol (RTCP) provides a mechanism to periodically send transport and media metrics to the media sender which can be utilized and extended for the purposes of RMCAT congestion control. For a congestion control algorithm which operates at the media sender, RTCP messages can be transmitted from the media receiver back to the media sender to enable congestion control. In the absence of standardized messages for this purpose, the congestion control algorithm designers have designed proprietary RTCP messages that convey only those parameters required for their respective designs. As a direct result, the different congestion control (a.k.a. rate adaptation) 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 generic congestion control feedback format.

To help achieve interoperability for unicast RTP congestion control, this memo proposes a common RTCP feedback format that can be used by NADA [I-D.ietf-rmcat-nada], SCReAM [I-D.ietf-rmcat-scream-cc], Google Congestion Control [I-D.ietf-rmcat-gcc] and Shared Bottleneck Detection [I-D.ietf-rmcat-sbd], and hopefully future RTP congestion control algorithms as well.

2. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

In addition the terminology defined in [RFC3550], [RFC3551], [RFC3611], [RFC4585], and [RFC5506] applies.

3. Feedback Message

The design team analyzed the feedback requirements from the different proposed candidate in RMCAT WG. The analysis showed some commonalities between the proposed solution candidate and some can be derived from other information. The design team has agreed to have following packet information block in the feedback message to satisfy different requirement analyzed.

- o Packet Identifier : RTP sequence number. The RTP packet header includes an incremental packet sequence number that the sender needs to correlate packets sent at the sender with packets received at the receiver.
- o Packet Arrival Time : Arrival time stamp at the receiver of the media. The sender requires the arrival time stamp of the respective packet to determine delay and jitter the packet had experienced during transmission. In a sender based congestion control solution the sender requires to keep track of the sent packets - usually packet sequence number, packet size and packet send time. With the packet arrival time the sender can detect the delay and jitter information. Along with packet loss and delay information the sender can estimate the available bandwidth and thus adapt to the situation.
- o Packet Explicit Congestion Notification (ECN) Marking : If ECN [RFC3168] is used, it is necessary to report on the 2-bit ECN mark in received packets, indicating for each packet whether it is marked not-ECT, ECT(0), ECT(1), or ECN-CE. If the path on which the media traffic traversing is ECN capable then the sender can use the Congestion Experienced (ECN-CE) marking information for congestion control. It is important that the receiver sends the

ECN-CE marking information of the packet back to the sender to take the advantages of ECN marking. Note that how the receiver gets the ECN marking information at application layer is out of the scope of this design team. Additional information for ECN use with RTP can be found at [RFC6679].

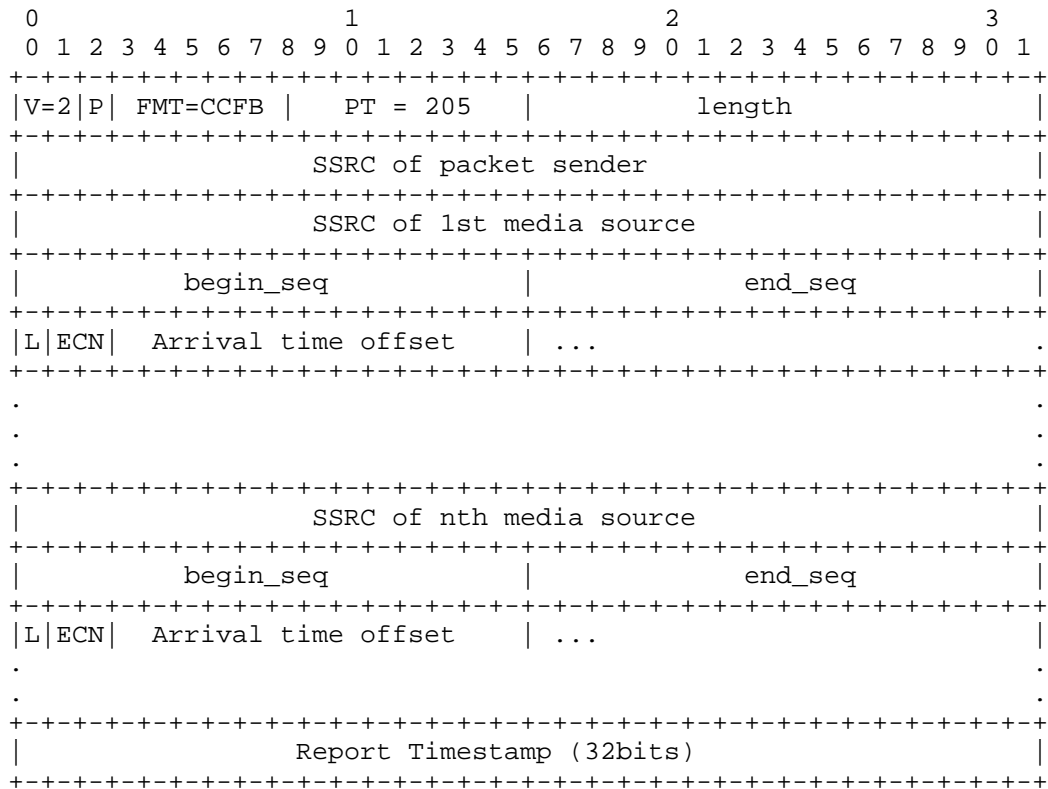
The feedback messages can have one or more of the above information blocks. For RTCP based feedback message the packet information block will be grouped by Synchronization Source (SSRC) identifier.

As a practical matter, we note that host Operating System (OS) process interruptions can occur at inopportune times. Thus, the recording of the sent times at the sender and arrival times at the receiver should be made 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 should be recorded at the latest opportunity prior to outputting the media packet at the sender (e.g., socket or RTP API) and the arrival time at the receiver (e.g., socket or RTP API) should be recorded at the earliest opportunity available to the receiver.

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 as follows:



The first 8 octets are the RTCP header, with PT=205 and FMT=CCFB specifying the remainder is a congestion control feedback packet, and including the SSRC of the packet sender. (NOTE TO RFC EDITOR: please replace CCFB here and in the above diagram with the IANA assigned RTCP feedback packet type)

Section 6.1 of [RFC4585] requires the RTCP header to be followed by the SSRC of the media source being reported upon. Accordingly, the RTCP header is followed by a report for each SSRC received, followed by the Report Timestamp.

The report for each SSRC received starts with the SSRC of that media source. Then, each sequence number between the begin_seq and end_seq (both inclusive) is represented by a packet metric block of 16-bits that contains the L, ECN, and ATO fields. If an odd number of reports are included, i.e., end_seq - begin_seq is odd then 16 bits of zero padding MUST be added after the last report, to align the RTCP packet to a four (4) bytes boundary. The L, ECN, and ATO fields are as follows:

- o L (1 bit): is a boolean to indicate if the packet was received. 0 represents that the packet was not yet received and all the subsequent bits (ECN and ATO) are also set to 0. 1 represent 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): it the relative arrival time of the RTP packets at the receiver before this feedback report was generated measured in milliseconds. It is calculated by subtracting the reception timestamp of the RTP packet denoted by this 16bit block and the timestamp (RTS) of this report. If the measured value is greater than 8.189 seconds (the value that would be coded as 0x1FFD), the value 0x1FFE MUST be reported to indicate an over-range positive measurement. If the measurement is unavailable, the value 0x1FFF MUST be reported.

Report Timestamp (RTS, 32 bits): represents the timestamp when this report was generated. The sender of the feedback message decides on the wall-clock. Usually, it should be derived from the same wall-clock that is used for timestamping RTP packets arrival. Consistency in the unit and resolution (10th of millisecond should be good enough) is important here. In addition, the media sender can ask for a specific resolution it wants.

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, and the possible rates of feedback.

It is a general understanding that the congestion control algorithms will work better with more frequent feedback - per packet feedback. However, RTCP bandwidth and transmission rules put some upper limits on how frequently the RTCP feedback messages can be send from the media receiver to the media sender. It has been shown [I-D.ietf-rmcat-rtp-cc-feedback] that in most cases a per frame feedback is a reasonable assumption on how frequent the RTCP feedback messages can be transmitted. The design team also have noted that even if a higher frequency of feedback is desired it is not viable if the feedback messages starts to compete against the media traffic on the feedback path during congestion period. Analyzing the feedback interval requirement [feedback-requirements] it can be seen that the candidate algorithms can perform with a feedback interval range of 50-200ms. A value within this range need to be negotiated at session setup.

5. Design Rationale

The primary function of RTCP Sender Report (SR) / Receiver Report (RR) is to report the reception quality of media. The regular SR / RR reports contain information about observed jitter, fractional packet loss and cumulative packet loss. The original intent of this information was to assist flow and congestion control mechanisms. Even though it is possible to do congestion control based on information provided in the SR/RR reports it is not sufficient to design an efficient congestion control algorithm for interactive real-time communication. An efficient congestion control algorithm requires more fine grain information on per packet (see Section 3) to react to the congestion or to avoid further congestion on the path.

Codec Control Message for AVPF [RFC5104] defines Temporary Maximum Media Bit Rate (TMMBR) message which conveys a temporary maximum bitrate limitation from the receiver of the media to the sender of the media. Even though it is 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 with reduced sized reports [RFC5506]. This requires the receiver of the media to analyze the data reception, detect congestion level and recommend a maximum bitrate suitable for current available bandwidth on the path with an assumption that the sender of the media always honors the TMMBR message. This requirement is completely opposite of the sender based congestion control approach. Hence, TMMBR cannot be as a signaling means for a sender based congestion control mechanism. However, TMMBR should be viewed a complimentary signaling mechanism to establish receiver's current view of acceptable maximum bitrate which a sender based congestion control should honor.

There are number of RTCP eXtended Report (XR) blocks have been defined for reporting of delay, loss and ECN marking. It is possible to combine several XR blocks to report the loss and ECN marking at the cost of overhead and complexity. However, there is no existing RTCP XR block to report packet arrival time.

Considering the issues discussed here it is rational to design a new congestion control feedback signaling mechanism for sender based congestion control algorithm.

6. Acknowledgements

This document is an outcome of RMCAT design team discussion. We would like to thank all participants specially Xiaoqing Zhu, Stefan Holmer, David, Ingemar Johansson and Randell Jesup for their valuable contribution to the discussions and to the document.

7. IANA Considerations

IANA is requested to assign a new value in the "FMT Values for RTPFB Payload Types" registry for the CCFB transport layer feedback packet described in Section 3.1.

8. Security Considerations

There is a risk of causing congestion if an on-path attacker modifies the feedback messages in such a manner to make available bandwidth greater than it is in reality. [More on security consideration TBD.]

9. References

9.1. Normative References

- [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-03 (work in progress), November 2016.
- [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>>.
- [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>>.

- [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>>.

9.2. 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.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-nada]
Zhu, X., Pan, R., Ramalho, M., Cruz, S., Jones, P., Fu, J., and S. D'Aronco, "NADA: A Unified Congestion Control Scheme for Real-Time Media", draft-ietf-rmcat-nada-05 (work in progress), September 2017.
- [I-D.ietf-rmcat-sbd]
Hayes, D., Ferlin, S., Welzl, M., and K. Hiorth, "Shared Bottleneck Detection for Coupled Congestion Control for RTP Media.", draft-ietf-rmcat-sbd-08 (work in progress), July 2017.

[I-D.ietf-rmcat-scream-cc]

Johansson, I. and Z. Sarker, "Self-Clocked Rate Adaptation for Multimedia", draft-ietf-rmcat-scream-cc-13 (work in progress), October 2017.

[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>>.

Authors' Addresses

Zaheduzzaman Sarker
Ericsson AB
Lulea
Sweden

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

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

Email: csp@csp Perkins.org

Varun Singh
Nemu Dialogue Systems Oy
Runeberginkatu 4c A 4
Helsinki 00100
Finland

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

Michael A. Ramalho
Cisco Systems, Inc.
6310 Watercrest Way Unit 203
Lakewood Ranch, FL 34202
USA

Phone: +1 919 476 2038
Email: mramalho@cisco.com

Network Working Group
Internet-Draft
Intended status: Experimental
Expires: March 22, 2017

X. Zhu
R. Pan
M. Ramalho
S. Mena
P. Jones
J. Fu
Cisco Systems
S. D'Aronco
EPFL
C. Ganzhorn
September 18, 2016

NADA: A Unified Congestion Control Scheme for Real-Time Media
draft-ietf-rmcat-nada-03

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 <http://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 22, 2017.

Copyright Notice

Copyright (c) 2016 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 (<http://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	9
5. Practical Implementation of NADA	12
5.1. Receiver-Side Operation	12
5.1.1. Estimation of one-way delay and queuing delay	12
5.1.2. Estimation of packet loss/marketing ratio	12
5.1.3. Estimation of receiving rate	13
5.2. Sender-Side Operation	13
5.2.1. Rate shaping buffer	14
5.2.2. Adjusting video target rate and sending rate	15
5.3. Feedback Message Requirements	15
6. Discussions and Further Investigations	16
6.1. Choice of delay metrics	16
6.2. Method for delay, loss, and marking ratio estimation	16
6.3. Impact of parameter values	17
6.4. Sender-based vs. receiver-based calculation	18
6.5. Incremental deployment	18
7. Implementation Status	18
8. Suggested Experiments	19
9. IANA Considerations	19
10. Acknowledgements	19
11. References	20
11.1. Normative References	20
11.2. Informative References	21
Appendix A. Network Node Operations	22
A.1. Default behavior of drop tail queues	22

A.2. RED-based ECN marking	23
A.3. Random Early Marking with Virtual Queues	23
Authors' Addresses	24

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].

This document describes an experimental congestion control scheme called network-assisted dynamic adaptation (NADA). The NADA design 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 with non-standard messages.

2. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described [RFC2119].

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.

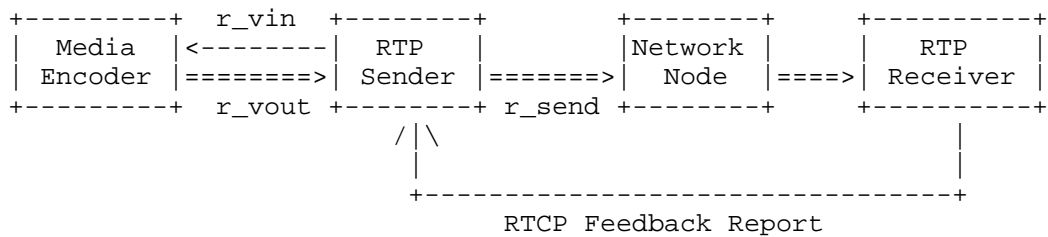


Figure 1: System Overview

- o Media encoder with rate control capabilities. It encodes raw media (audio and video) frames into compressed bitstream which is later packetized into RTP packets. As discussed in [I-D.ietf-rmcat-video-traffic-model], 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_{recv}) 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_{recv} .
- 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, FQ-CoDel, RED-based ECN marking, and PCN marking using a token bucket algorithm. 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.

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 queueing 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
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	150 Kbps
RMAX	Maximum rate of application supported by media encoder	1.5 Mbps
XREF	Reference congestion level	20ms
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
DFILT	Bound on filtering delay	120ms
LOGWIN	Observation window in time for calculating packet summary statistics at receiver	500ms
TEXPLOSS	Expiration time for previously observed packet loss	30s
QEPS	Threshold for determining queuing delay build up at receiver	10ms
QTH	Delay threshold for non-linear warping	50ms
DLOSS	Delay penalty for loss	1.0s
DMARK	Delay penalty for ECN marking	200ms
GAMMA_MAX	Upper bound on rate increase ratio for accelerated ramp-up	50%
QBOUND	Upper bound on self-inflicted queuing delay during ramp up	50ms
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.

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

```

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 ($t_{curr} - t_{last} > \text{DELTA}$):

```

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 RTCP feedback report 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, a moderate queuing delay value below 100ms 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 packet losses are observed within the previous time window of TLOSS, the estimated queuing delay follows a non-linear warping:

$$d_{\text{tilde}} = \begin{cases} d_{\text{queue}}, & \text{if } d_{\text{queue}} < QTH; \\ QTH \exp\left(-\frac{-(d_{\text{queue}} - QTH)}{QTH}\right), & \text{otherwise.} \end{cases} \quad (1)$$

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 aggregate congestion signal is:

$$x_{curr} = d_{tilde} + p_{mark} * DMARK + p_{loss} * DLOSS. \quad (2)$$

Here, DMARK is prescribed delay penalty associated with ECN markings and DLOSS is prescribed delay penalty associated with packet losses. 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 DLOS and DMARK need to be set consistently across all NADA flows for them to 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 [RMIN, 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:

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

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

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

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 $PRIO * XREF$.

At equilibrium, the aggregated congestion signal stabilizes at $x_curr = PRIO * XREF * RMAX / r_ref$. 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 maximum rate ($RMAX$). Values of $RMIN$ and $RMAX$ will be provided by the media codec, as specified in [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 2Mbps for $RMAX$.

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

$$r_ref = \min(r_ref, RMAX) \quad (8)$$

$$r_ref = \max(r_ref, RMIN) \quad (9)$$

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]. Instead of relying on RTP timestamps, the NADA sender generates its own timestamp based on local system clock and embeds 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. All delay estimations are based on sender timestamps with higher granularity than RTP timestamps.

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. Current implementation employs a 15-tap minimum filter over per-packet queuing delay estimates.

5.1.2. Estimation of packet loss/marketing ratio

The receiver detects packet losses via gaps in the RTP sequence numbers of received packets. Packets arriving out-of-order are discarded, and count towards 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) is 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.

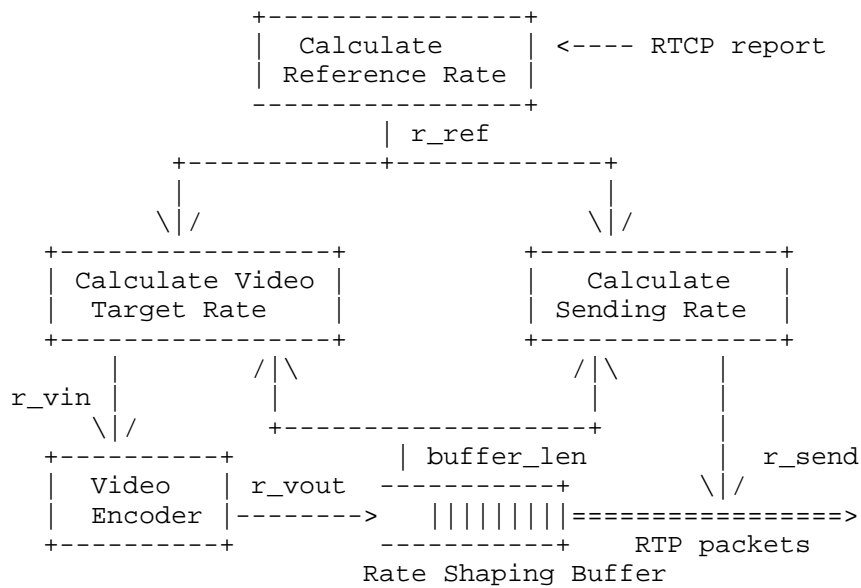


Figure 4: NADA Sender Structure

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

The target rate for the live video encoder deviates from the network congestion control rate r_{ref} based on the level of occupancy in the rate shaping buffer:

$$r_{\text{vin}} = r_{\text{ref}} - \text{BETA_V} * 8 * \text{buffer_len} * \text{FPS}. \quad (11)$$

The actual sending rate r_{send} is regulated in a similar fashion:

$$r_{\text{send}} = r_{\text{ref}} + \text{BETA_S} * 8 * \text{buffer_len} * \text{FPS}. \quad (12)$$

In (11) and (12), the first term indicates the rate calculated from network congestion feedback alone. The second term indicates 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} .

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 $8 * \text{buffer_len} * \text{FPS}$, where FPS stands for the 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.

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 is 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.

The above list of information can be accommodated by 48 bits, or 6 bytes, in total. 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

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.

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 acknowledgements 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 simply applying a 15-tab minimum filter 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 currently under investigation.

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 TAU=500ms, DELTA=100ms and 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 DELTA=100ms is recommended, corresponding to a feedback bandwidth of 16Kbps with 200 bytes per feedback message --- approximately 1.6% overhead for a 1 Mbps flow. Furthermore, both simulation studies and frequency-domain analysis have established that a feedback interval below 250ms 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 and TLOSS (for determining whether to perform the non-linear warping). It is possible to adapt the value of both based on past observed patterns of queuing delay in the presence of packet losses.

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, 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) is under investigation.

[Editor's note: Choice of start value: is this in scope of congestion control, or should this be decided by the application?]

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). Alternatively, one can move the logics of (1) and (2) to the sender. Such an approach requires slightly higher overhead in the feedback messages, which should contain individual fields on queuing delay (`d_queue`), packet loss ratio (`p_loss`), packet marking ratio (`p_mark`), receiving rate (`r_recv`), and recommended rate adaptation mode (`rmode`).

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.

To start off with, the proposed congestion control mechanism can be implemented without any explicit support from the network, and relies solely on observed 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. Implementation Status

The NADA scheme has been implemented in [ns-2] and [ns-3] simulation platforms. Extensive ns-2 simulation evaluations of an earlier version of the draft are documented in [Zhu-PV13]. Evaluation results of the current draft over several test cases in [I-D.ietf-rmcat-eval-test] have been presented at recent IETF

meetings [IETF-90][IETF-91]. Evaluation results of the current draft over several test cases in [I-D.ietf-rmcat-wireless-tests] have been presented at [IETF-93].

The scheme has also been implemented and evaluated in a lab setting as described in [IETF-90]. Preliminary evaluation results of NADA in single-flow and multi-flow scenarios have been presented in [IETF-91].

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 also been evaluated in implementations embedded in video conferencing clients.

Further experiments are suggested for the following scenarios:

- o Experiments with ECN marking capability turned on at the network for existing test cases.
- o Experiments with multiple RMCAT 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).
- o

9. IANA Considerations

This document makes no request of IANA.

10. Acknowledgements

The authors would like to thank Randell Jesup, Luca De Cicco, Piers O'Hanlon, Ingemar Johansson, Stefan Holmer, Cesar Ilharco Magalhaes, Safiqul Islam, Mirja Kuhlewind, and Karen Elisabeth Egede Nielsen for

their valuable questions and comments on earlier versions of this draft.

11. References

11.1. 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, <<http://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, <<http://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, <<http://www.rfc-editor.org/info/rfc3550>>.
- [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, <<http://www.rfc-editor.org/info/rfc6679>>.
- [I-D.ietf-rmcat-eval-test] Sarker, Z., Singh, V., Zhu, X., and D. Ramalho, "Test Cases for Evaluating RMCAT Proposals", draft-ietf-rmcat-eval-test-03 (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-video-traffic-model] Zhu, X., Cruz, S., and Z. Sarker, "Modeling Video Traffic Sources for RMCAT Evaluations", draft-ietf-rmcat-video-traffic-model-01 (work in progress), July 2016.
- [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-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-02 (work in progress), May 2016.

11.2. Informative References

- [RFC2309] Braden, B., Clark, D., Crowcroft, J., Davie, B., Deering, S., Estrin, D., Floyd, S., Jacobson, V., Minshall, G., Partridge, C., Peterson, L., Ramakrishnan, K., Shenker, S., Wroclawski, J., and L. Zhang, "Recommendations on Queue Management and Congestion Avoidance in the Internet", RFC 2309, DOI 10.17487/RFC2309, April 1998, <<http://www.rfc-editor.org/info/rfc2309>>.
- [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, <<http://www.rfc-editor.org/info/rfc5348>>.
- [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, <<http://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, <<http://www.rfc-editor.org/info/rfc6817>>.
- [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.
- [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.

- [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.
- [ns-2] "The Network Simulator - ns-2", <<http://www.isi.edu/nsnam/ns/>>.
- [ns-3] "The Network Simulator - ns-3", <<https://www.nsnam.org/>>.
- [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>>.

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.

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 [RFC2309]. Calculation of the marking probability involves the following steps:

```

on packet arrival:
  update smoothed queue size q_avg as:
    q_avg = w*q + (1-w)*q_avg.

  calculate marking probability p as:

      / 0,                                if q < q_lo;
      |
      |      q_avg - q_lo
      |      p_max*-----, if q_lo <= q < q_hi;
      |      q_hi - q_lo
      |
      \ p = 1,                            if q >= q_hi.

```

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.

```

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

* update token level b_tk;

* calculate the marking probability as:

```

$$p = \begin{cases} / 0, & \text{if } b-b_{tk} < b_{lo}; \\ | \\ p_{max} * \frac{b-b_{tk}-b_{lo}}{b_{hi}-b_{lo}}, & \text{if } b_{lo} \leq b-b_{tk} < b_{hi}; \\ | \\ \backslash 1, & \text{if } b-b_{tk} \geq b_{hi}. \end{cases}$$

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
Cisco Systems
3625 Cisco Way
San Jose, CA 95134
USA

Email: ropan@cisco.com

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

Phone: +1 919 476 2038
Email: mramalho@cisco.com

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

Email: semena@cisco.com

Paul E. Jones
Cisco Systems
7025 Kit Creek Rd.
Research Triangle Park, NC 27709
USA

Email: paulej@packetizer.com

Jiantao Fu
Cisco Systems
707 Tasman Drive
Milpitas, CA 95035
USA

Email: jianfu@cisco.com

Stefano D'Aronco
Ecole Polytechnique Federale de Lausanne
EPFL STI IEL LTS4, ELD 220 (Batiment ELD), Station 11
Lausanne CH-1015
Switzerland

Email: stefano.daronco@epfl.ch

Charles Ganzhorn
7900 International Drive, International Plaza, Suite 400
Bloomington, MN 55425
USA

Email: charles.ganzhorn@gmail.com

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.

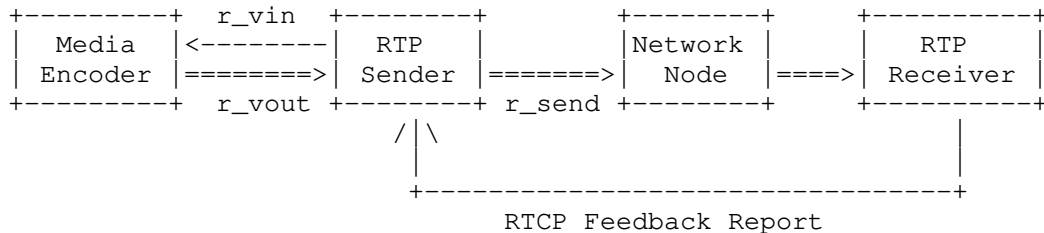


Figure 1: System Overview

- 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:

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

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.

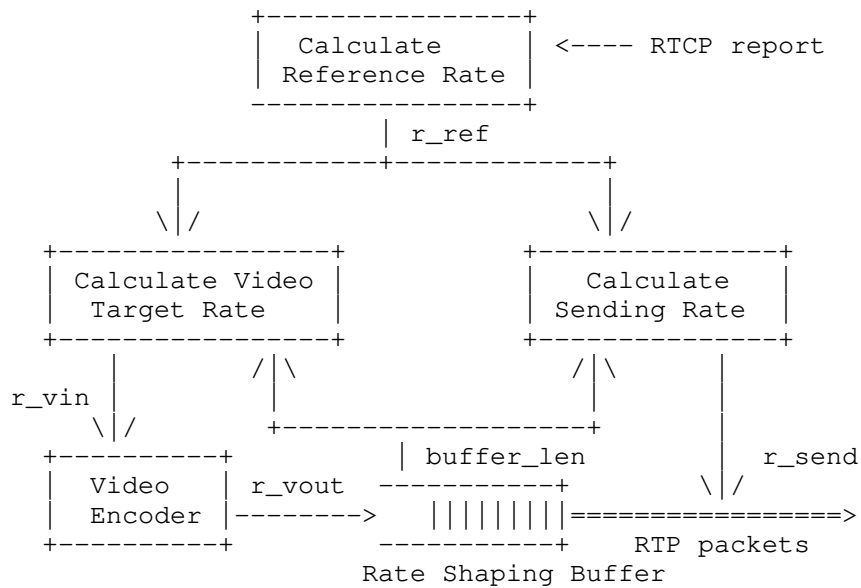


Figure 4: NADA Sender Structure

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
        | p_max* -----, if b_lo<= b-b_tk <b_hi;
        | b_hi-b_lo
        |
        \ 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: May 4, 2017

C. Perkins
University of Glasgow
October 31, 2016

RTP Control Protocol (RTCP) Feedback for Congestion Control in
Interactive Multimedia Conferences
draft-ietf-rmcat-rtp-cc-feedback-02

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 <http://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 4, 2017.

Copyright Notice

Copyright (c) 2016 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 (<http://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	2
3. What Feedback is Achievable With RTCP?	4
3.1. Scenario 1: Voice Telephony	4
3.2. Scenario 2: Point-to-Point Video Conference	6
3.3. Scenario 3: Group Video Conference	10
3.4. Scenario 4: Screen Sharing	10
4. Discussion and Conclusions	10
5. Security Considerations	10
6. IANA Considerations	11
7. Acknowledgements	11
8. Informative References	11
Author's Address	12

1. Introduction

The coming deployment of WebRTC systems raises the prospect that high quality video conferencing will see extremely wide use. To ensure the stability of the network in the face of this use, WebRTC systems will need to use some form of congestion control for their RTP-based media traffic. 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 the 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 [I-D.ietf-rtcweb-rtp-usage] systems. Nothing in this memo is specific to the secure version of the profile, or to WebRTC, however.

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? Traditional congestion control protocols, such as TCP, send acknowledgements with every packet (or, at least, every couple of packets). That is straight-forward and 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. It becomes a problem, however, when there is no return traffic on which to piggyback acknowledgements, and when acknowledgements are similar in size to data packets; this can be the case for some forms of media traffic, especially for voice over IP (VoIP) flows, but less so for video.

When considering multimedia traffic, it might make sense to consider less frequent feedback. 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).

The amount of overhead due to congestion control feedback that is considered acceptable has to be determined. RTCP data 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? Or is there a desire to either see 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?

3. What Feedback is Achievable With RTCP?

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. A compound RTCP packet is sent once for every N_{nc} non-compound RTCP packets.

Compound RTCP packets contain a Sender Report (SR) packet and a Source Description (SDES) packet, and an RTP Congestion Control Feedback (RC2F) packet [I-D.dt-rmcat-feedback-message]. Non-compound RTCP packets contain only the RC2F packet. Since each participant sends only a single media stream, the extensions for RTCP report aggregation [I-D.ietf-avtcore-rtp-multi-stream] and reporting group optimisation [I-D.ietf-avtcore-rtp-multi-stream-optimisation] 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 SDES 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 RC2F packets contains an XR block header and SSRC (8 octets), a block type and timestamp (8 octets), the SSRC, beginning and ending sequence numbers (8 octets), and $2 * N_r$ octets of reports, for a total of $24 + 2 * N_r$ octets. 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 $Sc = 132 + 2*Nr$ octets.

The non-compound RTCP packets will comprise just the RC2F packet with a UDP/IP header. It can be seen that these packets will be $Snc = 52 + 2*Nr$ 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 ($n=2$ in this scenario).

To ensure a report is sent every Nr frames, 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 for 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 expression gives the needed RTCP bandwidth $Brtcp = 53.1kbps$. Increasing the frame duration, or reducing the frequency of reports, reduces the RTCP bandwidth, as shown below:

Tf (seconds)	Nr (frames)	rtcp_bw (kbps)
20ms	2	53.1
20ms	4	27.3
20ms	8	14.5
20ms	16	8.01
60ms	2	17.7
60ms	4	9.1
60ms	8	4.8
60ms	16	2.66

Table 1: Required RTCP bandwidth for VoIP feedback

The final row of the table (60ms frames, report every 16 frames) sends RTCP reports once per second, giving an RTCP bandwidth of 2.66kbps.

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:

Tf (seconds)	Nr (frames)	rtcp_bw (kbps)
20ms	2	37.5
20ms	4	19.5
20ms	8	10.5
20ms	16	6.1
60ms	2	12.5
60ms	4	6.5
60ms	8	3.5
60ms	16	2.01

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 2.01kbps. 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, and 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 co-located audio and video SSRCs at each end point are aggregated [I-D.ietf-avtcore-rtp-multi-stream]; the optimisations in [I-D.ietf-avtcore-rtp-multi-stream-optimisation] are used; and congestion control feedback is sent [I-D.dt-rmcat-feedback-message].

When all members are senders, the RTCP timing rules in Section 6.2 and 6.3 of [RFC3550] and [RFC4585] reduce to:

$$\text{rtcp_interval} = \text{avg_rtcp_size} * n / \text{rtcp_bw}$$

where n is the number of members in the session, the `avg_rtcp_size` is measured in octets, and the `rtcp_bw` is the bandwidth available for RTCP, measured in octets per second (this will typically be 5% of the session bandwidth).

The average RTCP size will depend on the amount of feedback that is 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. Since the RTCP reporting group extensions are used, one of these SSRCs will be a reporting SSRC, and the other will delegate its reports to that.

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 [I-D.ietf-rtcweb-rtp-usage] 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 XR 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 [I-D.ietf-rtcweb-rtp-usage] 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 XR congestion control feedback report comprises an 8 octet XR header, an 8 octet RC2F header, then 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 $8 + 8 + 8$

+ (2 * Nv) + 8 + (2 * Na) where Nv is the number of video packets per report, and Na 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 SRTP authentication tag, and a UDP/IPv4 header. The size of this RTCP packet is therefore: 156 + (2 * Nv) + (2 * Na) octets. Since the aggregate RTCP packet contains reports from two SSRCs, the RTCP packet size is halved before use [I-D.ietf-avtcore-rtp-multi-stream]. Accordingly, we define $Sc = (156 + (2 * Nv) + (2 * Na))/2$ for this scenario.

How many packets does the RTCP XR congestion control feedback packet report on? This is obviously highly dependent on the choice of codec and encoding parameters, and might be quite bursty if the codec sends I-frames from which later frames are predicted. For now though, assume constant rate media with an MTU around 1500 octets, with reports for both audio and video being aggregated and sent to align with video frames. This gives the following, assuming $Nr = 1$ and $Nnc = 0$ (i.e., send a compound RTCP packet for each video frame, and no non-compound packets), and using the calculation from Scenario 1: $Brtcp = (n * (Sc + Nnc * Snc)) / (Nr * Tf * (1 + Nnc))$

Data Rate (kbps)	Video Frame Rate	Video Packets per Report: Nv	Audio Packets per Report: Na	Required RTCP bandwidth: Brtcp (kbps)
100	8	1	6	21 (21%)
200	16	1	3	41 (21%)
350	30	1	2	76 (21%)
700	30	2	2	77 (11%)
700	60	1	1	150 (21%)
1024	30	3	2	78 (8%)
1400	60	2	1	152 (11%)
2048	30	6	2	81 (4%)
2048	60	3	1	154 (8%)
4096	30	12	2	86 (2%)
4096	60	6	1	159 (4%)

Table 3: Required RTCP bandwidth, reporting on every frame

The RTCP bandwidth needed scales inversely with Nr . That is, it is halved if $Nr=2$ (report on every second packet), is reduced to one-third if $Nr=3$ (report on every third packet), and so on.

The needed RTCP bandwidth scales as a percentage of the data rate following the ratio of the frame rate to the data rate. As can be

seen from the table above, the RTCP bandwidth needed is a significant fraction of the media rate, if reporting on every frame for low rate video. This can be solved by reporting less often at lower rates. For example, to report on every frame of 100kbps/8fps video requires the RTCP bandwidth to be 21% of the media rate; reporting every fourth frame (i.e., twice per second) reduces this overhead to 5%.

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 UDP/IP header. That is, $S_{nc} = (96 + 2*N_v + 2*N_a)/2$. Repeating the analysis above, but alternating compound and non-compound reports, i.e., setting $N_{nc} = 1$, gives:

Data Rate (kbps)	Video Frame Rate	Video Packets per Report: N_v	Audio Packets per Report: N_a	Required RTCP bandwidth: Brtcp (kbps)
100	8	1	6	18 (18%)
200	16	1	3	33 (17%)
350	30	1	2	62 (18%)
700	30	2	2	62 (9%)
700	60	1	1	121 (17%)
1024	30	3	2	64 (6%)
1400	60	2	1	123 (9%)
2048	30	6	2	66 (3%)
2048	60	3	1	125 (6%)
4096	30	12	2	72 (2%)
4096	60	6	1	131 (3%)

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.

3.3. Scenario 3: Group Video Conference

(tbd)

3.4. Scenario 4: Screen Sharing

(tbd)

4. Discussion and Conclusions

RTCP as it is currently specified cannot be used to send per-packet congestion feedback. 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 it's current form for congestion feedback in WebRTC, the multimedia congestion control algorithm needs be designed to work with feedback sent every few frames, since that fits within the limitations of RTCP. That feedback can be a little more complex than just an acknowledgement, provided care is taken to consider the impact of the extra feedback on the overhead, possibly allowing for a degree of semantic feedback, meaningful to the codec layer as well as the congestion control algorithm.

The format described in [I-D.dt-rmcat-feedback-message] 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.

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 and the members of the RMCAT feedback design team for their feedback.

8. Informative References

- [I-D.dt-rmcat-feedback-message]
Sarker, Z., Perkins, C., Singh, V., and M. Ramalho, "RTP Control Protocol (RTCP) Feedback for Congestion Control", draft-dt-rmcat-feedback-message-01 (work in progress), October 2016.
- [I-D.ietf-avtcore-rtp-multi-stream]
Lennox, J., Westerlund, M., Wu, Q., and C. Perkins, "Sending Multiple RTP Streams in a Single RTP Session", draft-ietf-avtcore-rtp-multi-stream-11 (work in progress), December 2015.
- [I-D.ietf-avtcore-rtp-multi-stream-optimisation]
Lennox, J., Westerlund, M., Wu, Q., and C. Perkins, "Sending Multiple RTP Streams in a Single RTP Session: Grouping RTCP Reception Statistics and Other Feedback", draft-ietf-avtcore-rtp-multi-stream-optimisation-12 (work in progress), March 2016.
- [I-D.ietf-rtcweb-rtp-usage]
Perkins, C., Westerlund, M., and J. Ott, "Web Real-Time Communication (WebRTC): Media Transport and Use of RTP", draft-ietf-rtcweb-rtp-usage-26 (work in progress), March 2016.
- [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, <<http://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, <<http://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, <<http://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, <<http://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, <<http://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, <<http://www.rfc-editor.org/info/rfc5506>>.
- [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, <<http://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, <<http://www.rfc-editor.org/info/rfc7201>>.

Author's Address

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

Email: csp@csp Perkins.org

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:

$$T_{rtcp} = n * S_{rtcp} / B_{rtcp}$$

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.org

Network Working Group
Internet-Draft
Intended status: Informational
Expires: January 9, 2017

X. Zhu
S. Mena
Cisco Systems
Z. Sarker
Ericsson AB
July 8, 2016

Modeling Video Traffic Sources for RMCAT Evaluations
draft-ietf-rmcat-video-traffic-model-01

Abstract

This document describes two reference video traffic source models for evaluating RMCAT candidate algorithms. The first model statistically characterizes the behavior of a live video encoder in response to changing requests on target video rate. The second model is trace-driven, and emulates the encoder output by scaling the pre-encoded video frame sizes from a widely used video test sequence. Both models are designed to strike a balance between simplicity, repeatability, and authenticity in modeling the interactions between a video traffic source and the congestion control module.

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 <http://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 January 9, 2017.

Copyright Notice

Copyright (c) 2016 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 (<http://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. Desired Behavior of A Synthetic Video Traffic Model	3
4. Interactions Between Synthetic Video Traffic Source and Other Components at the Sender	4
5. A Statistical Reference Model	6
5.1. Time-damped response to target rate update	7
5.2. Temporary burst/oscillation during transient	7
5.3. Output rate fluctuation at steady state	8
5.4. Rate range limit imposed by video content	8
6. A Trace-Driven Model	8
6.1. Choosing the video sequence and generating the traces	9
6.2. Using the traces in the syntethic codec	10
6.2.1. Main algorithm	10
6.2.2. Notes to the main algorithm	12
6.3. Varying frame rate and resolution	12
7. Combining The Two Models	13
8. Implementation Status	14
9. IANA Considerations	14
10. References	15
10.1. Normative References	15
10.2. Informative References	15
Authors' Addresses	15

1. Introduction

When evaluating candidate congestion control algorithms designed for real-time interactive media, it is important to account for the characteristics of traffic patterns generated from a live video encoder. Unlike synthetic traffic sources that can conform perfectly to the rate changing requests from the congestion control module, a live video encoder can be sluggish in reacting to such changes. Output rate of a live video encoder also typically deviates from the target rate due to uncertainties in the encoder rate control process. Consequently, end-to-end delay and loss performance of a real-time media flow can be further impacted by rate variations introduced by the live encoder.

On the other hand, evaluation results of a candidate RMCAT algorithm should mostly reflect performance of the congestion control module, and somewhat decouple from peculiarities of any specific video codec. It is also desirable that evaluation tests are repeatable, and be easily duplicated across different candidate algorithms.

One way to strike a balance between the above considerations is to evaluate RMCAT algorithms using a synthetic video traffic source model that captures key characteristics of the behavior of a live video encoder. To this end, this draft presents two reference models. The first is based on statistical modelling; the second is trace-driven. The draft also discusses the pros and cons of each approach, as well as the how both approaches can be combined.

2. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described RFC2119 [RFC2119].

3. Desired Behavior of A Synthetic Video Traffic Model

A live video encoder employs encoder rate control to meet a target rate by varying its encoding parameters, such as quantization step size, frame rate, and picture resolution, based on its estimate of the video content (e.g., motion and scene complexity). In practice, however, several factors prevent the output video rate from perfectly conforming to the input target rate.

Due to uncertainties in the captured video scene, the output rate typically deviates from the specified target. In the presence of a significant change in target rate, it sometimes takes several frames before the encoder output rate converges to the new target. Finally, while most of the frames in a live session are encoded in predictive mode, the encoder can occasionally generate a large intra-coded frame (or a frame partially containing intra-coded blocks) in an attempt to recover from losses, to re-sync with the receiver, or during the transient period of responding to target rate or spatial resolution changes.

Hence, a synthetic video source should have the following capabilities:

- o To change bitrate. This includes ability to change framerate and/or spatial resolution, or to skip frames when required.
- o To fluctuate around the target bitrate specified by the congestion control module.

- o To delay in convergence to the target bitrate.
- o To generate intra-coded or repair frames on demand.

While there exist many different approaches in developing a synthetic video traffic model, it is desirable that the outcome follows a few common characteristics, as outlined below.

- o Low computational complexity: The model should be computationally lightweight, otherwise it defeats the whole purpose of serving as a substitute for a live video encoder.
- o Temporal pattern similarity: The individual traffic trace instances generated by the model should mimic the temporal pattern of those from a real video encoder.
- o Statistical resemblance: The synthetic traffic should match the outcome of the real video encoder in terms of statistical characteristics, such as the mean, variance, peak, and autocorrelation coefficients of the bitrate. It is also important that the statistical resemblance should hold across different time scales, ranging from tens of milliseconds to sub-seconds.
- o Wide range of coverage: The model should be easily configurable to cover a wide range of codec behaviors (e.g., with either fast or slow reaction time in live encoder rate control) and video content variations (e.g, ranging from high-motion to low-motion).

These distinct behavior features can be characterized via simple statistical models, or a trace-driven approach. We present an example of each in Section 5 and Section 6

4. Interactions Between Synthetic Video Traffic Source and Other Components at the Sender

Figure 1 depicts the interactions of the synthetic video encoder with other components at the sender, such as the application, the congestion control module, the media packet transport module, etc. Both reference models, as described later in Section 5 and Section 6, follow the same set of interactions.

The synthetic video encoder takes in raw video frames captured by the camera and then dynamically generates a sequence of encoded video frames with varying size and interval. These encoded frames are processed by other modules in order to transmit the video stream over the network. During the lifetime of a video transmission session, the synthetic video encoder will typically be required to adapt its

encoding bitrate, and sometimes the spatial resolution and frame rate.

In our model, the synthetic video encoder module has a group of incoming and outgoing interface calls that allow for interaction with other modules. The following are some of the possible incoming interface calls --- marked as (a) in Figure 1 --- that the synthetic video encoder may accept. The list is not exhaustive and can be complemented by other interface calls if deemed necessary.

- o Target rate $R_v(t)$: requested at time t , typically from the congestion control module. Depending on the congestion control algorithm in use, the update requests can either be periodic (e.g., once per second), or on-demand (e.g., only when a drastic bandwidth change over the network is observed).
- o Target frame rate $FPS(t)$: the instantaneous frame rate measured in frames-per-second at time t . This depends on the native camera capture frame rate as well as the target/preferred frame rate configured by the application or user.
- o Frame resolution $XY(t)$: the 2-dimensional vector indicating the preferred frame resolution in pixels at time t . Several factors govern the resolution requested to the synthetic video encoder over time. Examples of such factors are the capturing resolution of the native camera; or the current target rate $R_v(t)$, since very small resolutions do not make sense with very high bitrates, and vice-versa.
- o Instant frame skipping: the request to skip the encoding of one or several captured video frames, for instance when a drastic decrease in available network bandwidth is detected.
- o On-demand generation of intra (I) frame: the request to encode another I frame to avoid further error propagation at the receiver, if severe packet losses are observed. This request typically comes from the error control module.

An example of outgoing interface call --- marked as (b) in Figure 1 --- is the rate range, that is, the dynamic range of the video encoder's output rate for the current video contents: $[R_{min}, R_{max}]$. Here, R_{min} and R_{max} are meant to capture the dynamic rate range the encoder is capable of outputting. This typically depends on the

video content complexity and/or display type (e.g., higher R_{\max} for video contents with higher motion complexity, or for displays of higher resolution). Therefore, these values will not change with R_v , but may change over time if the content is changing.

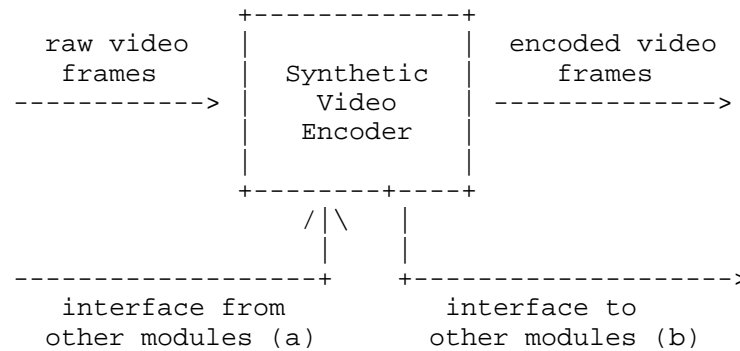


Figure 1: Interaction between synthetic video encoder and other modules at the sender

5. A Statistical Reference Model

In this section, we describe one simple statistical model of the live video encoder traffic source. Figure 2 summarizes the list of tunable parameters in this statistical model. A more comprehensive survey of popular methods for modelling video traffic source behavior can be found in [Tanwir2013].

Notation	Parameter Name	Example Value
$R_v(t)$	Target rate request at time t	1 Mbps
$R_o(t)$	Output rate at time t	1.2 Mbps
τ_v	Encoder reaction latency	0.2 s
K_d	Burst duration during transient	5 frames
K_r	Burst size during transient	5:1
$R_e(t)$	Error in output rate at time t	0.2 Mbps
SIGMA	standard deviation of normally distributed relative rate error	0.1
DELTA	upper and lower bound (+/-) of uniformly distributed relative rate error	0.1
R_{min}	minimum rate supported by video encoder or content activity	150 Kbps
R_{max}	maximum rate supported by video encoder or content activity	1.5Mbps

Figure 2: List of tunable parameters in a statistical video traffic source model.

5.1. Time-damped response to target rate update

While the congestion control module can update its target rate request $R_v(t)$ at any time, our model dictates that the encoder will only react to such changes after τ_v seconds from a previous rate transition. In other words, when the encoder has reacted to a rate change request at time t , it will simply ignore all subsequent rate change requests until time $t+\tau_v$.

5.2. Temporary burst/oscillation during transient

The output rate R_o during the period $[t, t+\tau_v]$ is considered to be in transient. Based on observations from video encoder output data, we model the transient behavior of an encoder upon reacting to a new target rate request in the form of largely varying output sizes. It is assumed that the overall average output rate R_o during this period matches the target rate R_v . Consequently, the occasional burst of large frames are followed by smaller-than average encoded frames.

This temporary burst is characterized by two parameters:

- o burst duration K_d : number frames in the burst event; and

- o burst size K_r : ratio of a burst frame and average frame size at steady state.

It can be noted that these burst parameters can also be used to mimic the insertion of a large on-demand I frame in the presence of severe packet losses. The values of K_d and K_r are fitted to reflect the typical ratio between I and P frames for a given video content.

5.3. Output rate fluctuation at steady state

We model output rate R_o as randomly fluctuating around the target rate R_v after convergence. There are two variants in modeling the random fluctuation $R_e = R_o - R_v$:

- o As normal distribution: with a mean of zero and a standard deviation $SIGMA$ specified in terms of percentage of the target rate. A typical value of $SIGMA$ is 10 percent of target rate.
- o As uniform distribution bounded between $-DELTA$ and $DELTA$. A typical value of $DELTA$ is 10 percent of target rate.

The distribution type (normal or uniform) and model parameters ($SIGMA$ or $DELTA$) can be learned from data samples gathered from a live encoder output.

5.4. Rate range limit imposed by video content

The output rate R_o is further clipped within the dynamic range $[R_{min}, R_{max}]$, which in reality are dictated by scene and motion complexity of the captured video content. In our model, these parameters are specified by the application.

6. A Trace-Driven Model

We now present the second approach to model a video traffic source. This approach is based on running an actual live video encoder offline on a set of chosen raw video sequences and using the encoder's output traces for constructing a synthetic live encoder. With this approach, the recorded video traces naturally exhibit temporal fluctuations around a given target rate request $R_v(t)$ from the congestion control module.

The following list summarizes this approach's main steps:

- 1) Choose one or more representative raw video sequences.

- 2) Using an actual live video encoder, encode the sequences at various bitrates. Keep just the sequences of frame sizes for each bitrate.
- 3) Construct a data structure that contains the output of the previous step. The data structure should allow for easy bitrate lookup.
- 4) Upon a target bitrate request $R_v(t)$ from the controller, look up the closest bitrates among those previously stored. Use the frame size sequences stored for those bitrates to approximate the frame sizes to output.
- 5) The output of the synthetic encoder contains "encoded" frames with zeros as contents but with realistic sizes.

Section 6.1 explains steps 1), 2), and 3), Section 6.2 elaborates on steps 4) and 5). Finally, Section 6.3 briefly discusses the possibility to extend the model for supporting variable frame rate and/or variable frame resolution.

6.1. Choosing the video sequence and generating the traces

The first step we need to perform is a careful choice of a set of video sequences that are representative of the use cases we want to model. Our use case here is video conferencing, so we must choose a low-motion sequence that resembles a "talking head", for instance a news broadcast or a video capture of an actual conference call.

The length of the chosen video sequence is a tradeoff. If it is too long, it will be difficult to manage the data structures containing the traces. If it is too short, there will be an obvious periodic pattern in the output frame sizes, leading to biased results when evaluating congestion controller performance. In our experience, a one-minute-long sequence is a fair tradeoff.

Once we have chosen the raw video sequence, denoted S , we use a live encoder, e.g. [H264] or [HEVC] to produce a set of encoded sequences. As discussed in Section 3, a live encoder's output bitrate can be tuned by varying three input parameters, namely, quantization step size, frame rate, and picture resolution. In order to simplify the choice of these parameters for a given target rate, we assume a fixed frame rate (e.g. 25 fps) and a fixed resolution (e.g., 480p). See section 6.3 for a discussion on how to relax these assumptions.

Following these simplifications, we run the chosen encoder by setting a constant target bitrate at the beginning, then letting the encoder

vary the quantization step size internally while encoding the input video sequence. Besides, we assume that the first frame is encoded as an I-frame and the rest are P-frames. We further assume that the encoder algorithm does not use knowledge of frames in the future so as to encode a given frame.

We define R_{\min} and R_{\max} as the minimum and maximum bitrate at which the synthetic codec is to operate. We divide the bitrate range between R_{\min} and R_{\max} in $n_s + 1$ bitrate steps of length $l = (R_{\max} - R_{\min}) / n_s$. We then use the following simple algorithm to encode the raw video sequence.

```
r = R_min
while r <= R_max do
    Traces[r] = encode_sequence(S, r, e)
    r = r + l
```

where function `encode_sequence` takes as parameters, respectively, a raw video sequence, a constant target rate, and an encoder algorithm; it returns a vector with the sizes of frames in the order they were encoded. The output vector is stored in a map structure called `Traces`, whose keys are bitrates and values are frame size vectors.

The choice of a value for n_s is important, as it determines the number of frame size vectors stored in map `Traces`. The minimum value one can choose for n_s is 1, and its maximum value depends on the amount of memory available for holding the map `Traces`. A reasonable value for n_s is one that makes the steps' length $l = 200$ kbps. We will further discuss step length l in the next section.

6.2. Using the traces in the syntethic codec

The main idea behind the trace-driven synthetic codec is that it mimics a real live codec's rate adaptation when the congestion controller updates the target rate $R_v(t)$. It does so by switching to a different frame size vector stored in the map `Traces` when needed.

6.2.1. Main algorithm

We maintain two variables r_{current} and t_{current} :

* r_{current} points to one of the keys of the map `Traces`. Upon a change in the value of $R_v(t)$, typically because the congestion controller detects that the network conditions have changed, r_{current} is updated to the greatest key in `Traces` that is less than or equal to the new value of $R_v(t)$. For the moment, we assume the value of $R_v(t)$ to be clipped in the range $[R_{\min}, R_{\max}]$.

```

r_current = r
such that
  ( r in keys(Traces) and
    r <= R_v(t) and
    (not(exists) r' in keys(Traces) such that r < r' <= R_v(t)) )

```

* $t_current$ is an index to the frame size vector stored in $Traces[r_current]$. It is updated every time a new frame is due. We assume all vectors stored in $Traces$ to have the same size, denoted $size_traces$. The following equation governs the update of $t_current$:

```

if t_current < SkipFrames then
  t_current = t_current + 1
else
  t_current = ((t_current+1-SkipFrames) % (size_traces- SkipFrames))
              + SkipFrames

```

where operator $\%$ denotes modulo, and $SkipFrames$ is a predefined constant that denotes the number of frames to be skipped at the beginning of frame size vectors after $t_current$ has wrapped around. The point of constant $SkipFrames$ is avoiding the effect of periodically sending a (big) I-frame followed by several smaller-than-normal P-frames. We typically set $SkipFrames$ to 20, although it could be set to 0 if we are interested in studying the effect of sending I-frames periodically.

We initialize $r_current$ to R_min , and $t_current$ to 0.

When a new frame is due, we need to calculate its size. There are three cases:

- a) $R_min \leq R_v(t) < R_{max}$: In this case we use linear interpolation of the frame sizes appearing in $Traces[r_current]$ and $Traces[r_current + 1]$. The interpolation is done as follows:

```

size_lo = Traces[r_current][t_current]
size_hi = Traces[r_current + 1][t_current]
distance_lo = ( R_v(t) - r_current ) / 1
framesize = size_hi * distance_lo + size_lo * (1 - distance_lo)

```

- b) $R_v(t) < R_min$: In this case, we scale the trace sequence with the lowest bitrate, in the following way:

```

factor = R_v(t) / R_min
framesize = max(1, factor * Traces[R_min][t_current])

```

- c) $R_v(t) \geq R_{\max}$: We also use scaling for this case. We use the trace sequence with the greatest bitrate:

```
factor = R_v(t) / R_max
framesize = factor * Traces[R_max][t_current]
```

In case b), we set the minimum to 1 byte, since the value of factor can be arbitrarily close to 0.

6.2.2. Notes to the main algorithm

* Reacting to changes in target bitrate. Similarly to the statistical model presented in Section 5, the trace-driven synthetic codec can have a time bound, τ_v , to reacting to target bitrate changes. If the codec has reacted to an update in $R_v(t)$ at time t , it will delay any further update to $R_v(t)$ to time $t + \tau_v$. Note that, in any case, the value of τ_v cannot be chosen shorter than the time between frames, i.e. the inverse of the frame rate.

* I-frames on demand. The synthetic codec could be extended to simulate the sending of I-frames on demand, e.g., as a reaction to losses. To implement this extension, the codec's API is augmented with a new function to request a new I-frame. Upon calling such function, t_{current} is reset to 0.

* Variable length l of steps defined between R_{\min} and R_{\max} . In the main algorithm's description, the step length l is fixed. However, if the range $[R_{\min}, R_{\max}]$ is very wide, it is also possible to define a set of steps with a non-constant length. The idea behind this modification is that the difference between 400 kbps and 600 kbps as bitrate is much more important than the difference between 4400 kbps and 4600 kbps. For example, one could define steps of length 200 Kbps under 1 Mbps, then length 300 kbps between 1 Mbps and 2 Mbps, 400 kbps between 2 Mbps and 3 Mbps, and so on.

6.3. Varying frame rate and resolution

The trace-driven synthetic codec model explained in this section is relatively simple because we have fixed the frame rate and the frame resolution. The model could be extended to have variable frame rate, variable spatial resolution, or both.

When the encoded picture quality at a given bitrate is low, one can potentially decrease the frame rate (if the video sequence is currently in low motion) or the spatial resolution in order to improve quality-of-experience (QoE) in the overall encoded video. On the other hand, if target bitrate increases to a point where there is no longer a perceptible improvement in the picture quality of

individual frames, then one might afford to increase the spatial resolution or the frame rate (useful if the video is currently in high motion).

Many techniques have been proposed to choose over time the best combination of encoder quantization step size, frame rate, and spatial resolution in order to maximize the quality of live video codecs [Ozer2011][Hu2010]. Future work may consider extending the trace-driven codec to accommodate variable frame rate and/or resolution.

From the perspective of congestion control, varying the spatial resolution typically requires a new intra-coded frame to be generated, thereby incurring a temporary burst in the output traffic pattern. The impact of frame rate change tends to be more subtle: reducing frame rate from high to low leads to sparsely spaced larger encoded packets instead of many densely spaced smaller packets. Such difference in traffic profiles may still affect the performance of congestion control, especially when outgoing packets are not paced at the transport module. We leave the investigation of varying frame rate to future work.

7. Combining The Two Models

It is worthwhile noting that the statistical and trace-driven models each has its own advantages and drawbacks. While both models are fairly simple to implement, it takes significantly greater effort to fit the parameters of a statistical model to actual encoder output data whereas it is straightforward for a trace-driven model to obtain encoded frame size data. On the other hand, once validated, the statistical model is more flexible in mimicking a wide range of encoder/content behaviors by simply varying the corresponding parameters in the model. In this regard, a trace-driven model relies -- by definition -- on additional data collection efforts for accommodating new codecs or video contents.

In general, trace-driven model is more realistic for mimicking ongoing, steady-state behavior of a video traffic source whereas statistical model is more versatile for simulating transient events (e.g., when target rate changes from A to B with temporary bursts during the transition). It is also possible to combine both models into a hybrid approach, using traces during steady-state and statistical model during transients.

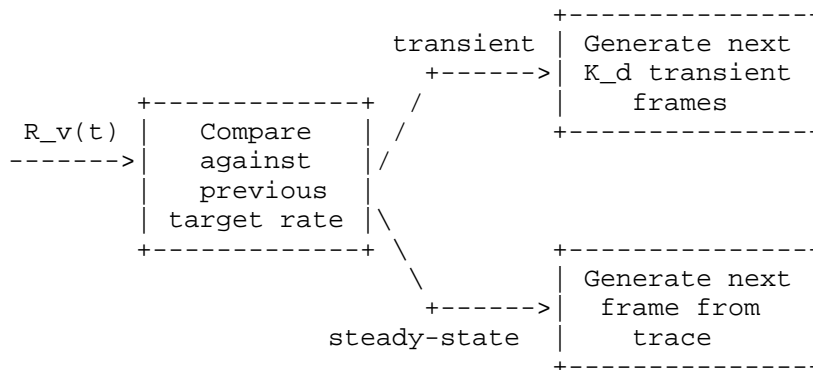


Figure 3: Hybrid approach for modeling video traffic

As shown in Figure 3, the video traffic model operates in transient state if the requested target rate $R_v(t)$ is substantially higher than the previous target, or else it operates in steady state. During transient state, a total of K_d frames are generated by the statistical model, resulting in 1 big burst frame (on average K_r times larger than average frame size at the target rate) followed by K_d-1 small frames. When operating in steady-state, the video traffic model simply generates a frame according to the trace-driven model given the target rate. One example criteria for determining whether the traffic model should operate in transient state is whether the rate increase exceeds 20% of previous target rate.

8. Implementation Status

The statistical model has been implemented as a traffic generator module within the [ns-2] network simulation platform.

More recently, both the statistical and trace-driven models have been implemented as a stand-alone traffic source module. This can be easily integrated into network simulation platforms such as [ns-2] and [ns-3], as well as testbeds using a real network. The stand-alone traffic source module is available as an open source implementation at [Syncodecs].

9. IANA Considerations

There are no IANA impacts in this memo.

10. References

10.1. 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, <<http://www.rfc-editor.org/info/rfc2119>>.
- [H264] ITU-T Recommendation H.264, "Advanced video coding for generic audiovisual services", 2003, <<http://www.itu.int/rec/T-REC-H.264-201304-I>>.
- [HEVC] ITU-T Recommendation H.265, "High efficiency video coding", 2015.

10.2. Informative References

- [Hu2010] Hu, H., Ma, Z., and Y. Wang, "Optimization of Spatial, Temporal and Amplitude Resolution for Rate-Constrained Video Coding and Scalable Video Adaptation", in Proc. 19th IEEE International Conference on Image Processing, (ICIP'12), September 2012.
- [Ozer2011] Ozer, J., "Video Compression for Flash, Apple Devices and HTML5", ISBN 13:978-0976259503, 2011.
- [Tanwir2013] Tanwir, S. and H. Perros, "A Survey of VBR Video Traffic Models", IEEE Communications Surveys and Tutorials, vol. 15, no. 5, pp. 1778-1802., October 2013.
- [ns-2] "The Network Simulator - ns-2", <<http://www.isi.edu/nsnam/ns/>>.
- [ns-3] "The Network Simulator - ns-3", <<https://www.nsnam.org/>>.
- [Syncodecs] Mena, S., D'Aronco, S., and X. Zhu, "Syncodecs: Synthetic codecs for evaluation of RMCAT work", <<https://github.com/cisco/syncodecs>>.

Authors' Addresses

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

Email: xiaoqzhu@cisco.com

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

Email: semena@cisco.com

Zaheduzzaman Sarker
Ericsson AB
Luleae, SE 977 53
Sweden

Phone: +46 10 717 37 43
Email: zaheduzzaman.sarker@ericsson.com

Network Working Group
Internet-Draft
Intended status: Informational
Expires: August 23, 2019

X. Zhu
S. Mena
Cisco Systems
Z. Sarker
Ericsson AB
February 19, 2019

Video Traffic Models for RTP Congestion Control Evaluations
draft-ietf-rmcat-video-traffic-model-07

Abstract

This document describes two reference video traffic models for evaluating RTP congestion control algorithms. The first model statistically characterizes the behavior of a live video encoder in response to changing requests on the target video rate. The second model is trace-driven and emulates the output of actual encoded video frame sizes from a high-resolution test sequence. Both models are designed to strike a balance between simplicity, repeatability, and authenticity in modeling the interactions between a live video traffic source and the congestion control module. Finally, the document describes how both approaches can be combined into a hybrid model.

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 August 23, 2019.

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	2
2. Terminology	3
3. Desired Behavior of A Synthetic Video Traffic Model	3
4. Interactions Between Synthetic Video Traffic Source and Other Components at the Sender	5
5. A Statistical Reference Model	6
5.1. Time-damped response to target rate update	7
5.2. Temporary burst and oscillation during the transient period	8
5.3. Output rate fluctuation at steady state	8
5.4. Rate range limit imposed by video content	9
6. A Trace-Driven Model	9
6.1. Choosing the video sequence and generating the traces	10
6.2. Using the traces in the synthetic codec	11
6.2.1. Main algorithm	11
6.2.2. Notes to the main algorithm	13
6.3. Varying frame rate and resolution	14
7. Combining The Two Models	14
8. Implementation Status	16
9. IANA Considerations	16
10. Security Considerations	16
11. References	16
11.1. Normative References	16
11.2. Informative References	16
Authors' Addresses	17

1. Introduction

When evaluating candidate congestion control algorithms designed for real-time interactive media, it is important to account for the characteristics of traffic patterns generated from a live video encoder. Unlike synthetic traffic sources that can conform perfectly to the rate changing requests from the congestion control module, a live video encoder can be sluggish in reacting to such changes. The output rate of a live video encoder also typically deviates from the target rate due to uncertainties in the encoder rate control process.

Consequently, end-to-end delay and loss performance of a real-time media flow can be further impacted by rate variations introduced by the live encoder.

On the other hand, evaluation results of a candidate RTP congestion control algorithm should mostly reflect the performance of the congestion control module and somewhat decouple from peculiarities of any specific video codec. It is also desirable that evaluation tests are repeatable, and be easily duplicated across different candidate algorithms.

One way to strike a balance between the above considerations is to evaluate congestion control algorithms using a synthetic video traffic source model that captures key characteristics of the behavior of a live video encoder. The synthetic traffic model should also contain tunable parameters so that it can be flexibly adjusted to reflect the wide variations in real-world live video encoder behaviors. To this end, this draft presents two reference models. The first is based on statistical modeling. The second is driven by frame size and interval traces recorded from a real-world encoder. The draft also discusses the pros and cons of each approach, as well as how both approaches can be combined into a hybrid model.

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. Desired Behavior of A Synthetic Video Traffic Model

A live video encoder employs encoder rate control to meet a target rate by varying its encoding parameters, such as quantization step size, frame rate, and picture resolution, based on its estimate of the video content (e.g., motion and scene complexity). In practice, however, several factors prevent the output video rate from perfectly conforming to the input target rate.

Due to uncertainties in the captured video scene, the output rate typically deviates from the specified target. In the presence of a significant change in target rate, the encoder's output frame sizes sometimes fluctuate for a short, transient period of time before the output rate converges to the new target. Finally, while most of the frames in a live session are encoded in predictive mode (i.e., P-frames in [H264]), the encoder can occasionally generate a large intra-coded frame (i.e., I-frame as defined in [H264]) or a frame

partially containing intra-coded blocks in an attempt to recover from losses, to re-sync with the receiver, or during the transient period of responding to target rate or spatial resolution changes.

Hence, a synthetic video source should have the following capabilities:

- o To change bitrate. This includes the ability to change framerate and/or spatial resolution or to skip frames upon request.
- o To fluctuate around the target bitrate specified by the congestion control module.
- o To show a delay in convergence to the target bitrate.
- o To generate intra-coded or repair frames on demand.

While there exist many different approaches in developing a synthetic video traffic model, it is desirable that the outcome follows a few common characteristics, as outlined below.

- o Low computational complexity: The model should be computationally lightweight, otherwise it defeats the whole purpose of serving as a substitute for a live video encoder.
- o Temporal pattern similarity: The individual traffic trace instances generated by the model should mimic the temporal pattern of those from a real video encoder.
- o Statistical resemblance: The synthetic traffic source should match the outcome of the real video encoder in terms of statistical characteristics, such as the mean, variance, peak, and autocorrelation coefficients of the bitrate. It is also important that the statistical resemblance should hold across different time scales, ranging from tens of milliseconds to sub-seconds.
- o A wide range of coverage: The model should be easily configurable to cover a wide range of codec behaviors (e.g., with either fast or slow reaction time in live encoder rate control) and video content variations (e.g., ranging from high to low motion).

These distinct behavior features can be characterized via simple statistical modeling or a trace-driven approach. Section 5 and Section 6 provide an example of each approach, respectively. Section 7 discusses how both models can be combined together.

4. Interactions Between Synthetic Video Traffic Source and Other Components at the Sender

Figure 1 depicts the interactions of the synthetic video traffic source with other components at the sender, such as the application, the congestion control module, the media packet transport module, etc. Both reference models --- as described later in Section 5 and Section 6 --- follow the same set of interactions.

The synthetic video source dynamically generates a sequence of dummy video frames with varying size and interval. These dummy frames are processed by other modules in order to transmit the video stream over the network. During the lifetime of a video transmission session, the synthetic video source will typically be required to adapt its encoding bitrate, and sometimes the spatial resolution and frame rate.

In this model, the synthetic video source module has a group of incoming and outgoing interface calls that allow for interaction with other modules. The following are some of the possible incoming interface calls --- marked as (a) in Figure 1 --- that the synthetic video traffic source may accept. The list is not exhaustive and can be complemented by other interface calls if necessary.

- o Target bitrate R_v : target bitrate request measured in bits per second (bps). Typically, the congestion control module calculates the target bitrate and updates it dynamically over time. Depending on the congestion control algorithm in use, the update requests can either be periodic (e.g., once per second), or on-demand (e.g., only when a drastic bandwidth change over the network is observed).
- o Target frame rate FPS: the instantaneous frame rate measured in frames-per-second at a given time. This depends on the native camera capture frame rate as well as the target/preferred frame rate configured by the application or user.
- o Target frame resolution XY: the 2-dimensional vector indicating the preferred frame resolution in pixels. Several factors govern the resolution requested to the synthetic video source over time. Examples of such factors include the capturing resolution of the native camera and the display size of the destination screen. The target frame resolution also depends on the current target bitrate R_v , since it does not make sense to pair very low spatial resolutions with very high bitrates, and vice-versa.

- o Instant frame skipping: the request to skip the encoding of one or several captured video frames, for instance when a drastic decrease in available network bandwidth is detected.
- o On-demand generation of intra (I) frame: the request to encode another I frame to avoid further error propagation at the receiver when severe packet losses are observed. This request typically comes from the error control module. It can be initiated either by the sender or by the receiver via Full Intra Request (FIR) messages as defined in [RFC5104].

An example of outgoing interface call --- marked as (b) in Figure 1 --- is the rate range $[R_{min}, R_{max}]$. Here, R_{min} and R_{max} are meant to capture the dynamic rate range and actual live video encoder is capable of generating given the input video content. This typically depends on the video content complexity and/or display type (e.g., higher R_{max} for video contents with higher motion complexity, or for displays of higher resolution). Therefore, these values will not change with R_v but may change over time if the content is changing.

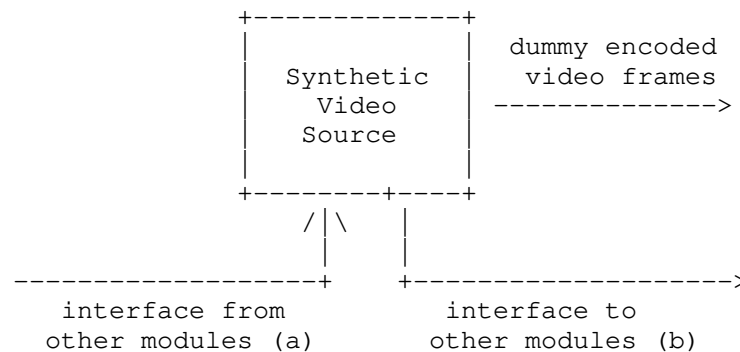


Figure 1: Interaction between synthetic video encoder and other modules at the sender

5. A Statistical Reference Model

This section describes one simple statistical model of the live video encoder traffic source. Figure 2 summarizes the list of tunable parameters in this statistical model. A more comprehensive survey of popular methods for modeling video traffic source behavior can be found in [Tanwir2013].

Notation	Parameter Name	Example Value
R_v	Target bitrate request	1 Mbps
FPS	Target frame rate	30 Hz
tau_v	Encoder reaction latency	0.2 s
K_d	Burst duration of the transient period	8 frames
K_B	Burst frame size during the transient period	13.5 KBytes*
t0	Reference frame interval 1/FPS	33 ms
B0	Reference frame size R_v/8/FPS	4.17 KBytes
SCALE_t	Scaling parameter of the zero-mean Laplacian distribution describing deviations in normalized frame interval $(t-t_0)/t_0$	0.15
SCALE_B	Scaling parameter of the zero-mean Laplacian distribution describing deviations in normalized frame size $(B-B_0)/B_0$	0.15
R_min	minimum rate supported by video encoder type or content activity	150 Kbps
R_max	maximum rate supported by video encoder type or content activity	1.5 Mbps

* Example value of K_B for a video stream encoded at 720p and 30 frames per second, using H.264/AVC encoder.

Figure 2: List of tunable parameters in a statistical video traffic source model.

5.1. Time-damped response to target rate update

While the congestion control module can update its target bitrate request R_v at any time, the statistical model dictates that the encoder will only react to such changes tau_v seconds after a

previous rate transition. In other words, when the encoder has reacted to a rate change request at time t , it will simply ignore all subsequent rate change requests until time $t+\tau_v$.

5.2. Temporary burst and oscillation during the transient period

The output bitrate R_o during the period $[t, t+\tau_v]$ is considered to be in a transient state when reacting to abrupt changes in target rate. Based on observations from video encoder output data, the encoder reaction to a new target bitrate request can be characterized by high variations in output frame sizes. It is assumed in the model that the overall average output bitrate R_o during this transient period matches the target bitrate R_v . Consequently, the occasional burst of large frames is followed by smaller-than-average encoded frames.

This temporary burst is characterized by two parameters:

- o burst duration K_d : number of frames in the burst event; and
- o burst frame size K_B : size of the initial burst frame which is typically significantly larger than average frame size at steady state.

It can be noted that these burst parameters can also be used to mimic the insertion of a large on-demand I frame in the presence of severe packet losses. The values of K_d and K_B typically depend on the type of video codec, spatial and temporal resolution of the encoded stream, as well as the video content activity level.

5.3. Output rate fluctuation at steady state

The output bitrate R_o during steady state is modeled as randomly fluctuating around the target bitrate R_v . The output traffic can be characterized as the combination of two random processes denoting the frame interval t and output frame size B over time, as the two major sources of variations in the encoder output. For simplicity, the deviations of t and B from their respective reference levels are modeled as independent and identically distributed (i.i.d) random variables following the Laplacian distribution [Papoulis]. More specifically:

- o Fluctuations in frame interval: the intervals between adjacent frames have been observed to fluctuate around the reference interval of $t_0 = 1/\text{FPS}$. Deviations in normalized frame interval $\text{DELTA}_t = (t-t_0)/t_0$ can be modeled by a zero-mean Laplacian distribution with scaling parameter SCALE_t . The value of SCALE_t dictates the "width" of the Laplacian distribution and therefore

the amount of fluctuation in actual frame intervals (t) with respect to the reference frame interval t_0 .

- o Fluctuations in frame size: the output encoded frame sizes also tend to fluctuate around the reference frame size $B_0 = R_v/8/\text{FPS}$. Likewise, deviations in the normalized frame size $\text{DELTA}_B = (B - B_0)/B_0$ can be modeled by a zero-mean Laplacian distribution with scaling parameter SCALE_B . The value of SCALE_B dictates the "width" of this second Laplacian distribution and correspondingly the amount of fluctuations in output frame sizes (B) with respect to the reference target B_0 .

Both values of SCALE_t and SCALE_B can be obtained via parameter fitting from empirical data captured for a given video encoder. Example values are listed in Figure 2 based on empirical data presented in [IETF-Interim].

5.4. Rate range limit imposed by video content

The output bitrate R_o is further clipped within the dynamic range $[R_{\min}, R_{\max}]$, which in reality are dictated by scene and motion complexity of the captured video content. In the proposed statistical model, these parameters are specified by the application.

6. A Trace-Driven Model

The second approach for modeling a video traffic source is trace-driven. This can be achieved by running an actual live video encoder on a set of chosen raw video sequences and using the encoder's output traces for constructing a synthetic video source. With this approach, the recorded video traces naturally exhibit temporal fluctuations around a given target bitrate request R_v from the congestion control module.

The following list summarizes the main steps of this approach:

1. Choose one or more representative raw video sequences.
2. Encode the sequence(s) using an actual live video encoder. Repeat the process for a number of bitrates. Keep only the sequence of frame sizes for each bitrate.
3. Construct a data structure that contains the output of the previous step. The data structure should allow for easy bitrate lookup.
4. Upon a target bitrate request R_v from the controller, look up the closest bitrates among those previously stored. Use the

frame size sequences stored for those bitrates to approximate the frame sizes to output.

5. The output of the synthetic video traffic source contains "encoded" frames with dummy contents but with realistic sizes.

In the following, Section 6.1 explains the first three steps (1-3), Section 6.2 elaborates on the remaining two steps (4-5). Finally, Section 6.3 briefly discusses the possibility to extend the trace-driven model for supporting time-varying frame rate and/or time-varying frame resolution.

6.1. Choosing the video sequence and generating the traces

The first step is a careful choice of a set of video sequences that are representative of the target use cases for the video traffic model. For the example use case of interactive video conferencing, it is recommended to choose a sequence with content that resembles a "talking head", e.g. from a news broadcast or recording of an actual video conferencing call.

The length of the chosen video sequence is a tradeoff. If it is too long, it will be difficult to manage the data structures containing the traces. If it is too short, there will be an obvious periodic pattern in the output frame sizes, leading to biased results when evaluating congestion control performance. It has been empirically determined that a sequence 2 to 4 minutes in length sufficiently avoids the periodic pattern.

Given the chosen raw video sequence, denoted *S*, one can use a live encoder, e.g. some implementation of [H264] or [HEVC], to produce a set of encoded sequences. As discussed in Section 3, the output bitrate of the live encoder can be achieved by tuning three input parameters: quantization step size, frame rate, and picture resolution. In order to simplify the choice of these parameters for a given target rate, one can typically assume a fixed frame rate (e.g. 30 fps) and a fixed resolution (e.g., 720p) when configuring the live encoder. See Section 6.3 for a discussion on how to relax these assumptions.

Following these simplifications, the chosen encoder can be configured to start at a constant target bitrate, then vary the quantization step size (internally via the video encoder rate controller) to meet various externally specified target rates. It can be further assumed the first frame is encoded as an I-frame and the rest are P-frames (see, e.g., [H264] for definitions of I- and P-frames). For live encoding, the encoder rate control algorithm typically does not use knowledge of frames in the future when encoding a given frame.

Given the minimum and maximum bitrates at which the synthetic codec is to operate (denoted as R_{\min} and R_{\max} , see Section 4), the entire range of target bitrates can be divided into n_s steps. This leads to an encoding bitrate ladder of $(n_s + 1)$ choices equally spaced apart by the step length $l = (R_{\max} - R_{\min})/n_s$. The following simple algorithm is used to encode the raw video sequence.

```
r = R_min
while r <= R_max do
    Traces[r] = encode_sequence(S, r, e)
    r = r + l
```

The function `encode_sequence` takes as input parameters, respectively, a raw video sequence (S), a constant target rate (r), and an encoder rate control algorithm (e); it returns a vector with the sizes of frames in the order they were encoded. The output vector is stored in a map structure called `Traces`, whose keys are bitrates and whose values are vectors of frame sizes.

The choice of a value for the number of bitrate steps n_s is important, since it determines the number of vectors of frame sizes stored in the map `Traces`. The minimum value one can choose for n_s is 1; the maximum value depends on the amount of memory available for holding the map `Traces`. A reasonable value for n_s is one that results in steps of length $l = 200$ kbps. The next section will discuss further the choice of step length l .

Finally, note that, as mentioned in previous sections, R_{\min} and R_{\max} may be modified after the initial sequences are encoded. Henceforth, for notational clarity, we refer to the bitrate range of the trace file as $[Rf_{\min}, Rf_{\max}]$. The algorithm described in the next section also covers the cases when the current target bitrate is less than Rf_{\min} , or greater than Rf_{\max} .

6.2. Using the traces in the synthetic codec

The main idea behind the trace-driven synthetic codec is that it mimics the rate adaptation behavior of a real live codec upon dynamic updates of the target bitrate request R_v by the congestion control module. It does so by switching to a different frame size vector stored in the map `Traces` when needed.

6.2.1. Main algorithm

The main algorithm for rate adaptation in the synthetic codec maintains two variables: r_{current} and t_{current} .

- o The variable `r_current` points to one of the keys of map `Traces`. Upon a change in the value of `R_v`, typically because the congestion controller detects that the network conditions have changed, `r_current` is updated based on `R_v` as follows:

```

R_ref = min (Rf_max, max(Rf_min, R_v))

r_current = r
such that
  (r in keys(Traces) and
   r <= R_ref and
   (not(exists) r' in keys(Traces) such that r < r' <= R_ref))

```

- o The variable `t_current` is an index to the frame size vector stored in `Traces[r_current]`. It is updated every time a new frame is due. It is assumed that all vectors stored in `Traces` have the same size, denoted as `size_traces`. The following equation governs the update of `t_current`:

```

if t_current < SkipFrames then
  t_current = t_current + 1
else
  t_current = ((t_current + 1 - SkipFrames)
               % (size_traces-SkipFrames)) + SkipFrames

```

where operator `%` denotes modulo, and `SkipFrames` is a predefined constant that denotes the number of frames to be skipped at the beginning of frame size vectors after `t_current` has wrapped around. The point of constant `SkipFrames` is avoiding the effect of periodically sending a large I-frame followed by several smaller-than-average P-frames. A typical value of `SkipFrames` is 20, although it could be set to 0 if one is interested in studying the effect of sending I-frames periodically.

The initial value of `r_current` is set to `R_min`, and the initial value of `t_current` is set to 0.

When a new frame is due, its size can be calculated following one of the three cases below:

- a) `Rf_min <= R_v < Rf_max`: the output frame size is calculated via linear interpolation of the frame sizes appearing in `Traces[r_current]` and `Traces[r_current + 1]`. The interpolation is done as follows:

```

size_lo = Traces[r_current][t_current]
size_hi = Traces[r_current + 1][t_current]
distance_lo = (R_v - r_current) / 1
framesize = size_hi*distance_lo + size_lo*(1-distance_lo)

```

- b) $R_v < R_{f_min}$: the output frame size is calculated via scaling with respect to the lowest bitrate R_{f_min} in the trace file, as follows:

```

w = R_v / R_{f\_min}
framesize = max(fs_min, factor * Traces[R_{f\_min}][t_current])

```

- c) $R_v \geq R_{f_max}$: the output frame size is calculated by scaling with respect to the highest bitrate R_{f_max} in the trace file, as follows:

```

w = R_v / R_{f\_max}
framesize = min(fs_max, w * Traces[R_{f\_max}][t_current])

```

In cases b) and c), floating-point arithmetic is used for computing the scaling factor w . The resulting value of the instantaneous frame size ($framesize$) is further clipped within a reasonable range between fs_min (e.g., 10 bytes) and fs_max (e.g., 1MB).

6.2.2. Notes to the main algorithm

Note that the main algorithm as described above can be further extended to mimic some additional typical behaviors of a live video encoder. Two examples are given below:

- o I-frames on demand: The synthetic codec can be extended to simulate the sending of I-frames on demand, e.g., as a reaction to losses. To implement this extension, the codec's incoming interface (see (a) in Figure 1) is augmented with a new function to request a new I-frame. Upon calling such function, $t_current$ is reset to 0.
- o Variable step length l between R_min and R_max : In the main algorithm, the step length l is fixed for ease of explanation. However, if the range $[R_min, R_max]$ is very wide, it is also possible to define a set of intermediate encoding rates with variable step length. The rationale behind this modification is that the difference between 400 kbps and 600 kbps as target bitrate is much more significant than the difference between 4400 kbps and 4600 kbps. For example, one could define steps of length 200 Kbps under 1 Mbps, then steps of length 300 Kbps between 1 Mbps and 2 Mbps; 400 Kbps between 2 Mbps and 3 Mbps, and so on.

6.3. Varying frame rate and resolution

The trace-driven synthetic codec model explained in this section is relatively simple due to the choice of fixed frame rate and frame resolution. The model can be extended further to accommodate variable frame rate and/or variable spatial resolution.

When the encoded picture quality at a given bitrate is low, one can potentially decrease either the frame rate (if the video sequence is currently in low motion) or the spatial resolution in order to improve quality-of-experience (QoE) in the overall encoded video. On the other hand, if target bitrate increases to a point where there is no longer a perceptible improvement in the picture quality of individual frames, then one might afford to increase the spatial resolution or the frame rate (useful if the video is currently in high motion).

Many techniques have been proposed to choose over time the best combination of encoder quantization step size, frame rate, and spatial resolution in order to maximize the quality of live video codecs [Ozer2011][Hu2010]. Future work may consider extending the trace-driven codec to accommodate variable frame rate and/or resolution.

From the perspective of congestion control, varying the spatial resolution typically requires a new intra-coded frame to be generated, thereby incurring a temporary burst in the output traffic pattern. The impact of frame rate change tends to be more subtle: reducing frame rate from high to low leads to sparsely spaced larger encoded packets instead of many densely spaced smaller packets. Such difference in traffic profiles may still affect the performance of congestion control, especially when outgoing packets are not paced by the media transport module. Investigation of varying frame rate and resolution are left for future work.

7. Combining The Two Models

It is worthwhile noting that the statistical and trace-driven models each have their own advantages and drawbacks. Both models are fairly simple to implement. It takes significantly greater effort to fit the parameters of a statistical model to actual encoder output data. In contrast, it is straightforward for a trace-driven model to obtain encoded frame size data. Once validated, the statistical model is more flexible in mimicking a wide range of encoder/content behaviors by simply varying the corresponding parameters in the model. In this regard, a trace-driven model relies -- by definition -- on additional data collection efforts for accommodating new codecs or video contents.

In general, the trace-driven model is more realistic for mimicking the ongoing, steady-state behavior of a video traffic source with fluctuations around a constant target rate. In contrast, the statistical model is more versatile for simulating the behavior of a video stream in transient, such as when encountering sudden rate changes. It is also possible to combine both methods into a hybrid model. In this case, the steady-state behavior is driven by traces during steady state and the transient-state behavior is driven by the statistical model.

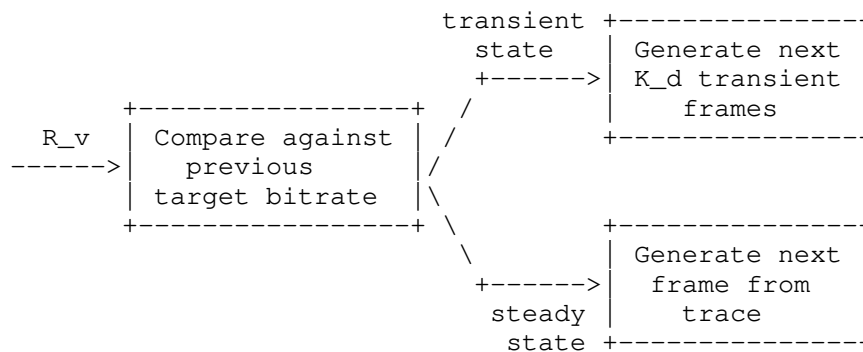


Figure 3: A hybrid video traffic model

As shown in Figure 3, the video traffic model operates in a transient state if the requested target rate R_v is substantially different from the previous target, or else it operates in steady state. During the transient state, a total of K_d frames are generated by the statistical model, resulting in one (1) big burst frame with size K_B followed by K_d-1 smaller frames. When operating at steady state, the video traffic model simply generates a frame according to the trace-driven model given the target rate, while modulating the frame interval according to the distribution specified by the statistical model. One example criterion for determining whether the traffic model should operate in a transient state is whether the rate change exceeds 10% of the previous target rate. Finally, as this model follows transient-state behavior dictated by the statistical model, upon a substantial rate change, the model will follow the time-damping mechanism as defined in Section 5.1, which is governed by parameter τ_v .

8. Implementation Status

The statistical, trace-driven, and hybrid models as described in this draft have been implemented as a stand-alone, platform-independent synthetic traffic source module. It can be easily integrated into network simulation platforms such as [ns-2] and [ns-3], as well as testbeds using a real network. The stand-alone traffic source module is available as an open source implementation at [Syncodecs].

9. IANA Considerations

There are no IANA impacts in this memo.

10. Security Considerations

The synthetic video traffic models as described in this draft do not impose any security threats. They are designed to mimic realistic traffic patterns for evaluating candidate RTP-based congestion control algorithms, so as to ensure stable operations of the network. It is RECOMMENDED that candidate algorithms be tested using the video traffic models presented in this draft before wide deployment over the Internet. If the generated synthetic traffic flows are sent over the Internet, they also need to be congestion controlled.

11. References

11.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.

11.2. Informative References

- [H264] ITU-T Recommendation H.264, "Advanced video coding for generic audiovisual services", May 2003, <<https://www.itu.int/rec/T-REC-H.264>>.
- [HEVC] ITU-T Recommendation H.265, "High efficiency video coding", April 2013, <<https://www.itu.int/rec/T-REC-H.265>>.

- [Hu2010] Hu, H., Ma, Z., and Y. Wang, "Optimization of Spatial, Temporal and Amplitude Resolution for Rate-Constrained Video Coding and Scalable Video Adaptation", in Proc. 19th IEEE International Conference on Image Processing, (ICIP'12), September 2012.
- [IETF-Interim] Zhu, X., Mena, S., and Z. Sarker, "Update on RMCAT Video Traffic Model: Trace Analysis and Model Update", April 2017, <<https://www.ietf.org/proceedings/interim-2017-rmcat-01/slides/slides-interim-2017-rmcat-01-sessa-update-on-video-traffic-model-draft-00.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/>>.
- [Ozer2011] Ozer, J., "Video Compression for Flash, Apple Devices and HTML5", ISBN 13:978-0976259503, 2011.
- [Papoulis] Papoulis, A., "Probability, Random Variables and Stochastic Processes", 2002.
- [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>>.
- [Syncodecs] Mena, S., D'Aronco, S., and X. Zhu, "Syncodecs: Synthetic codecs for evaluation of RMCAT work", <<https://github.com/cisco/syncodecs>>.
- [Tanwir2013] Tanwir, S. and H. Perros, "A Survey of VBR Video Traffic Models", IEEE Communications Surveys and Tutorials, vol. 15, no. 5, pp. 1778-1802., October 2013.

Authors' Addresses

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

Email: xiaoqzhu@cisco.com

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

Email: semena@cisco.com

Zaheduzzaman Sarker
Ericsson AB
Luleae, SE 977 53
Sweden

Phone: +46 10 717 37 43
Email: zaheduzzaman.sarker@ericsson.com

RMCAT WG
Internet-Draft
Intended status: Informational
Expires: January 8, 2017

X. Zhu
Cisco Systems
Z. Sarker
Ericsson AB
July 7, 2016

Framework for Real-time Media Congestion Avoidance Techniques
draft-zhu-rmcat-framework-00

Abstract

Congestion control is an essential element in ensuring fair bandwidth usage and preventing congestion collapse for traffic sharing the Internet. For interactive real-time media traffic such as video conferencing, design of congestion control solution also needs to account for many other factors such as the requirement for low latency packet delivery and interactions with live video encoder. This document describes a common framework with the core functional building blocks for a real-time media congestion solutions.

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 <http://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 January 8, 2017.

Copyright Notice

Copyright (c) 2016 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 (<http://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. Key Words for Requirements	2
3. Functional Modules	3
4. Example Configurations	4
4.1. Example Configurations for a Single Stream	4
4.2. Example Configurations for Multiple Streams	6
5. Acknowledgements	7
6. IANA Considerations	7
7. Security Considerations	7
8. References	7
8.1. Normative References	7
8.2. Informative References	8
Authors' Addresses	8

1. Introduction

Given increasing amount of interactive real-time media traffic over the Internet, such as video conferencing, it is important that these applications employ proper congestion control mechanisms to avoid congestion collapse. [I-D.ietf-rmcat-cc-requirements] specifies the list of requirements of a viable solution.

This document outlines a common framework for designing a congestion control mechanism for real-time interactive communication, so that individual drafts on specific solutions follow a consistent set of terminologies in describing their respective components. The next section (Section 3) describes common functional modules in this framework, whereas Section 4 provides examples on how these modules build together to support single and multiple media streams.

[Editor's note : This document does not describe the interaction between application, codec and congestion control system. The interaction among application, codec and congestion control system are defined in other documents. There is a possibility to merge all the documents into one single document.]

2. Key Words for Requirements

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

3. Functional Modules

A viable solution for real-time media congestion control needs to comprise of several common modules. This section provides a brief description of them and their respective functionalities. A congestion control solution for real-time media should comprise of the described functional modules. This should help understanding different congestion control solutions.

- o Network Congestion Controller : this is the core module for estimating available bandwidth over the network based on periodic RTCP feedback reports [RFC3550] from the receiver. This module contains key functions and calculations required to detect congestion and estimate available bandwidth on the transmission path based on the reception quality of the media traffic. Different congestion control solutions employ different algorithms in detecting congestion and estimating available bandwidth for its media flow. It also possible that multiple media streams are multiplexed over a single transport, hence share a common congestion control module in aggregation.
- o Transmission Queue : this module is needed to absorb the instantaneous mismatch between output from a live video encoder and regulated outgoing media flow. The transmission queue schedules outgoing traffic according to sending rate recommended by the rate controller module. It reports back its occupancy level to the rate controller module to assist future rate control decisions on target video rate, sending rate, and probing rate.
- o Rate Controller : this module takes the estimated available bandwidth from the network congestion controller, shared states of other flows, as well as occupancy level of the transmission queue as input. It makes holistic decisions on: a) target video rate for the live video encoder; b) sending rate for regulating outgoing media flow(s) for the transmission queue; and c) rate of probing packets when needed. In the case where multiple media streams share a single transport and a common network congestion controller (for estimating available bandwidth in aggregation), the rate controller is also responsible for distributing available bandwidth amongst different media streams according to their relative priorities as well as share state information. When losses occur over the network and some previous media packets need to be retransmitted, the rate controller should also account for the bandwidth needed for retransmission.
- o Network Probe Generator: A congestion control solution can actively probe to estimate the available bandwidth on the media transmission path by sending more than what the live video encoder

produces. Such an approach can be especially effective during the ramp up period of media and transmission rates, when no congestion has been observed over the network yet. The network probe generator is responsible for generating probing packets according to the probing rate specified by the rate controller. It can employ different techniques in doing so -- for example by generating simple dummy packets with unknown payload type or by generating Forward Error Correction (FEC) packets. While this document does not specify what probing technique to use or how those packets should be generated, a complete congestion control solution needs should specify total rate of the probe packets via the rate controller module.

- o Live Video Encoder : the sender typically also contains a live video encoder, which adjusts the its encoding parameters according to the target video rate set by the rate controller. The output rate from the video encoder may deviate from this target due to uncertainty in the captured video content characteristics and the encoder rate control process. The output encoded media packets are fed to the transmission queue. Note that internal operations of the live video encoder (i.e., how video encoder rate control works) is out of scope for this document.
- o Shared State: In the case of multiple media streams sharing a common sender hence a common network congestion controller, the sender should also contain a shared state module for storage and exchange of congestion control states [Editor's Note from Xiaoqing: examples of congestion control states??] amongst the multiple flows.

4. Example Configurations

4.1. Example Configurations for a Single Stream

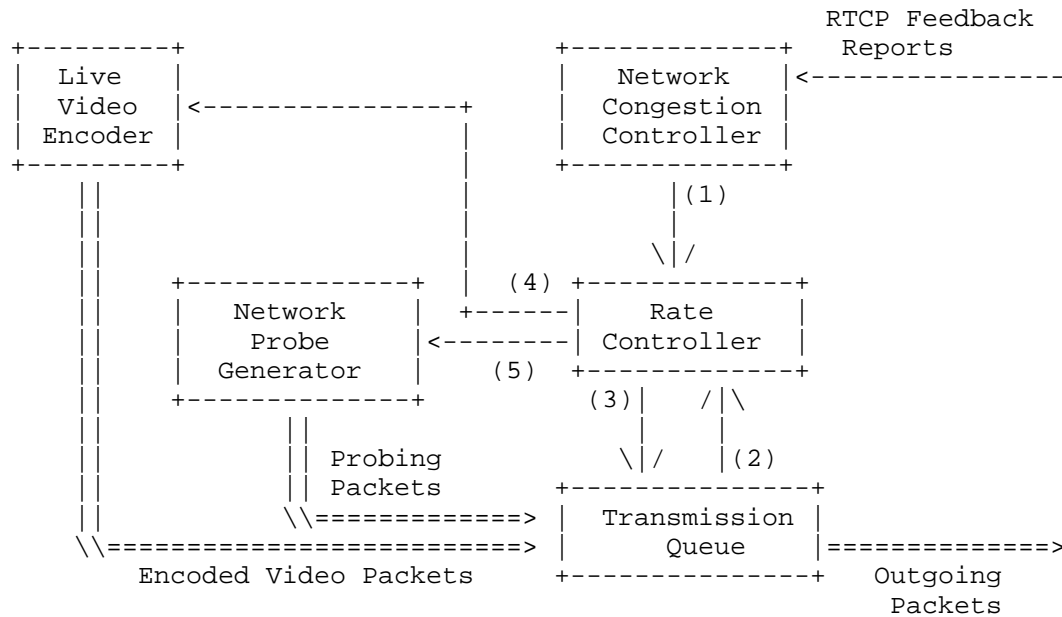


Figure 1: RMCAT Solution Framework at the Sender: Single Stream

Figure 1 shows an example configuration at the sender for supporting a single media stream. The Network Congestion Controller estimates available bandwidth based on periodic RTCP feedback reports. The Rate Controller takes as input the estimated available bandwidth (1) and the current occupancy level of the Transmission Queue (2). It calculates as output sending rate (3) for the Transmission Queue, video target rate (4) for the Live Video Encoder, and probing rate (5) -- if they are needed -- for the Network Probe Generator. The Transmission Queue holds packets generated by both the Live Video Encoder and the Network Probe Generator; it paces transmission of its outgoing packets according to the sending rate (3) specified by Rate Controller.

Obviously, it is possible for a congestion control solution to contain alternative configurations between these functional modules. [TODO: add one quick example on alternative wiring.] It is required that the candidate solution draft specify how their internal functional modules align to this framework.

4.2. Example Configurations for Multiple Streams

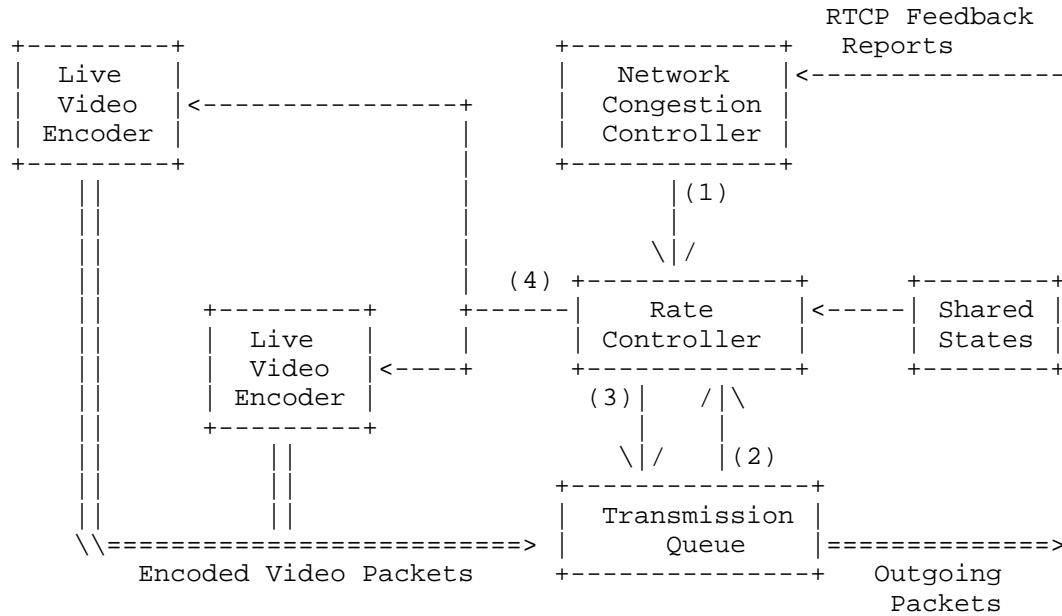


Figure 2: RMCAT Solution Framework at the Sender: Multiple Streams

Figure 2 shows an example configuration for multiple video streams sharing a common Network Congestion Controller. The Network Congestion Controller calculates an aggregated estimated available bandwidth (1) based on periodic RTCP feedback reports. The Rate Controller divides up the aggregate estimated bandwidth (1) from the Network Congestion Controller amongst sub-streams based on their relative priority levels, Shared States, as well as current occupancy level of the Transmission Queue. It subsequently determines the per-flow sending rate (3) as regulated by the Transmission Queue and target video rate (4) for each flow.

In this specific example, the transmission queue is envisioned as a logical entity. For instance, this transmission queue can be implemented priority-based scheduling and one physical queue per stream. For sake of simplicity the role of Network Probe Generator is omitted in the above figure.

5. Acknowledgements

The RMCAT design team discussions contributed to this memo.

6. IANA Considerations

This memo includes no request to IANA.

7. Security Considerations

TBD

8. References

8.1. Normative References

- [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.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<http://www.rfc-editor.org/info/rfc2119>>.
- [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, <<http://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, <<http://www.rfc-editor.org/info/rfc3551>>.
- [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, <<http://www.rfc-editor.org/info/rfc4585>>.
- [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, <<http://www.rfc-editor.org/info/rfc5104>>.

8.2. Informative References

[RFC0768] Postel, J., "User Datagram Protocol", STD 6, RFC 768,
DOI 10.17487/RFC0768, August 1980,
<<http://www.rfc-editor.org/info/rfc768>>.

Authors' Addresses

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

Email: xiaoqzhu@cisco.com

Zaheduzzaman Sarker
Ericsson AB
Luleae
Sweden

Phone: 00 46 10 717 37 43
Email: zaheduzzaman.sarker@ericsson.com