

Network Working Group
Internet-Draft
Intended status: Experimental
Expires: April 21, 2016

S. Holmer
M. Flodman
E. Sprang
Google
October 19, 2015

RTP Extensions for Transport-wide Congestion Control
draft-holmer-rmcat-transport-wide-cc-extensions-01

Abstract

This document proposes an RTP header extension and an RTCP message for use in congestion control algorithms for RTP-based media flows. It adds transport-wide packet sequence numbers and corresponding feedback message so that congestion control can be performed on a transport level at the send-side, while keeping the receiver dumb.

Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

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 April 21, 2016.

Copyright Notice

Copyright (c) 2015 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. Transport-wide Sequence Number	3
2.1. Semantics	3
2.2. RTP header extension format	3
2.3. Signaling of use of this extension	3
3. Transport-wide RTCP Feedback Message	4
3.1. Message format	4
3.1.1. Packet Status Symbols	6
3.1.2. Packet Status Chunks	7
3.1.3. Run Length Chunk	7
3.1.4. Status Vector Chunk	8
3.1.5. Receive Delta	9
4. Overhead discussion	10
5. IANA considerations	10
6. Security Considerations	10
7. Acknowledgements	10
8. References	10
8.1. Normative References	10
8.2. Informative References	10
Appendix A. Change log	11
A.1. First version	11
Authors' Addresses	11

1. Introduction

This document proposes RTP header extension containing a transport-wide packet sequence number and an RTCP feedback message feeding back the arrival times and sequence numbers of the packets received on a connection.

Some of the benefits that these extensions bring are:

- o The congestion control algorithms are easier to maintain and improve as there is less synchronization between sender and receiver versions needed. It should be possible to implement [I-D.ietf-rmcat-gcc], [I-D.ietf-rmcat-nada] and [I-D.ietf-rmcat-scream-cc] with the proposed protocol.

- o More flexibility in what algorithms are used, as long as they are having most of their logic on the send-side. For instance different behavior can be used depending on if the rate produced is application limited or not.

2. Transport-wide Sequence Number

2.1. Semantics

This RTP header extension is added on the transport layer, and uses the same counter for all packets which are sent over the same connection (for instance as defined by bundle).

The benefit with a transport-wide sequence numbers is two-fold:

- o It is a better fit for congestion control as the congestion controller doesn't operate on media streams, but on packet flows.
- o It allows for earlier packet loss detection (and recovery) since a loss in stream A can be detected when a packet from stream B is received, thus we don't have to wait until the next packet of stream A is received.

2.2. RTP header extension format

This document describes a message using the application specific payload type. This is suitable for experimentation; upon standardization, a specific type can be assigned for the purpose.

```

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
+-----+-----+-----+-----+-----+-----+-----+-----+
| 0xBE   | 0xDE   | length=1 |                                     |
+-----+-----+-----+-----+-----+-----+-----+-----+
| ID     | L=1   | transport-wide sequence number | zero padding |
+-----+-----+-----+-----+-----+-----+-----+-----+

```

An RTP header extension with a 16 bits sequence number attached to all packets sent. This sequence number is incremented by 1 for each packet being sent over the same socket.

2.3. Signaling of use of this extension

When signalled in SDP, the standard mechanism for RTP header extensions [RFC5285] is used:

a=extmap:5 <http://www.ietf.org/id/draft-holmer-rmcat-transport-wide-cc-extensions>

3. Transport-wide RTCP Feedback Message

To allow the most freedom possible to the sender, information about each packet delivered is needed. The simplest way of accomplishing that is to have the receiver send back a message containing an arrival timestamp and a packet identifier for each packet received. This way, the receiver is dumb and simply records arrival timestamps (A) of packets. The sender keeps a map of in-flight packets, and upon feedback arrival it looks up the on-wire timestamp (S) of the corresponding packet. From these two timestamps the sender can compute metrics such as:

- o Inter-packet delay variation: $d(i) = A(i) - S(i) - (A(i-1) - S(i-1))$
- o Estimated queueing delay: $q(i) = A(i) - S(i) - \min\{j=i-1..i-w\}(A(j) - S(j))$

Since the sender gets feedback about each packet sent, it will be set to better assess the cost of sending bursts of packets compared to aiming at sending at a constant rate decided by the receiver.

Two down-sides with this approach are:

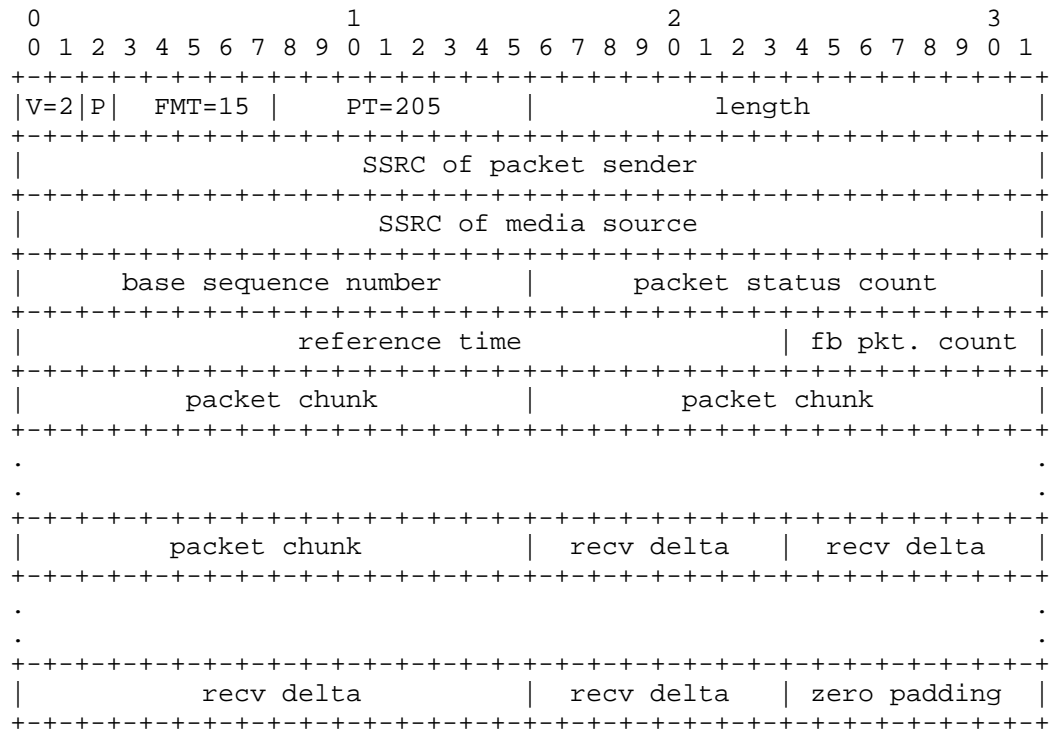
- o It isn't possible to differentiate between lost feedback on the downlink and lost packets on the uplink.
- o Increased feedback rate on the reverse direction.

From a congestion control perspective, lost feedback messages are handled by ignoring packets which would have been reported as lost or received in the lost feedback messages. This behavior is similar to how a lost RTCP receiver report is handled.

It is recommended that a feedback message is sent for every frame received, but in cases of low uplink bandwidth it is acceptable to send them less frequently, e.g., for instance once per RTT, to reduce the overhead.

3.1. Message format

The message is an RTCP message with payload type 206. RFC 3550 [RFC3550] defines the range, RFC 4585 [RFC3550] defines the specific PT value 206 and the FMT value 15.



version (V): 2 bits This field identifies the RTP version. The current version is 2.

padding (P): 1 bit If set, the padding bit indicates that the packet contains additional padding octets at the end that are not part of the control information but are included in the length field.

feedback message type (FMT): 5 bits This field identifies the type of the FB message. It must have the value 15.

payload type (PT): 8 bits This is the RTCP packet type that identifies the packet as being an RTCP FB message. The value must be RTPFB = 205.

SSRC of packet sender: 32 bits The synchronization source identifier for the originator of this packet.

SSRC of media source: 32 bits The synchronization source identifier of the media source that this piece of feedback

information is related to. TODO: This is transport wide, do we just pick any of the media source SSRCS?

base sequence number: 16 bits The transport-wide sequence number of the first packet in this feedback. This number is not necessarily increased for every feedback; in the case of reordering it may be decreased.

packet status count: 16 bits The number of packets this feedback contains status for, starting with the packet identified by the base sequence number.

reference time: 24 bits Signed integer indicating an absolute reference time in some (unknown) time base chosen by the sender of the feedback packets. The value is to be interpreted in multiples of 64ms. The first rcv delta in this packet is relative to the reference time. The reference time makes it possible to calculate the delta between feedbacks even if some feedback packets are lost, since it always uses the same time base.

feedback packet count: 8 bits A counter incremented by one for each feedback packet sent. Used to detect feedback packet losses.

packet chunk: 16 bits A list of packet status chunks. These indicate the status of a number of packets starting with the one identified by base sequence number. See below for details.

rcv delta: 8 bits For each "packet received" status, in the packet status chunks, a receive delta block will follow. See details below.

3.1.1. Packet Status Symbols

The status of a packet is described using a 2-bit symbol:

- 00 Packet not received
- 01 Packet received, small delta
- 10 Packet received, large or negative delta
- 11 [Reserved]

Packets with status "Packet not received" should not necessarily be interpreted as lost. They might just not have arrived yet.

For each packet received with a delta, to the previous received packet, within +/-8191.75ms, a receive delta block is appended to the feedback message.

Note: In the case the base sequence number is decreased, creating a window overlapping the previous feedback messages, the status for any packets previously reported as received must be marked as "Packet not received" and thus no delta included for that symbol.

3.1.2. Packet Status Chunks

Packet status is described in chunks, similar to a Loss RLE Report Block. There are two different kinds of chunks:

- o Run length chunk
- o Status vector chunk

All chunk types are 16 bits in length. The first bit of the chunk identifies whether it is an RLE chunk or a vector chunk.

3.1.3. Run Length Chunk

A run length chunk starts with 0 bit, followed by a packet status symbol and the run length of that symbol.

```

      0                               1
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5
+-----+
|T| S |           Run Length          |
+-----+
```

chunk type (T): 1 bit A zero identifies this as a run length chunk.

packet status symbol (S): 2 bits The symbol repeated in this run.
See above.

run length (L): 13 bits An unsigned integer denoting the run length.

Example 1:

```

      0                               1
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5
+-----+
|0|0 0|0 0 0 0 0 1 1 0 1 1 1 0 1|
+-----+
```

This is a run of the "packet not received" status of length 221.

Example 2:

```

      0                               1
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5
+---+---+---+---+---+---+---+---+---+
|0|1 1|0 0 0 0 0 0 0 0 1 1 0 0 0|
+---+---+---+---+---+---+---+---+

```

This is a run of the "packet received, w/o recv delta" status of length 24.

3.1.4. Status Vector Chunk

A status vector chunk starts with a 1 bit to identify it as a vector chunk, followed by a symbol size bit and then 7 or 14 symbols, depending on the size bit.

```

      0                               1
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5
+---+---+---+---+---+---+---+---+---+
|T|S|           symbol list         |
+---+---+---+---+---+---+---+---+

```

chunk type (T): 1 bit A one identifies this as a status vector chunk.

symbol size (S): 1 bit A zero means this vector contains only "packet received" (0) and "packet not received" (1) symbols. This means we can compress each symbol to just one bit, 14 in total. A one means this vector contains the normal 2-bit symbols, 7 in total.

symbol list: 14 bits A list of packet status symbols, 7 or 14 in total.

Example 1:

```

      0                               1
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5
+---+---+---+---+---+---+---+---+---+
|1|0|0 1 1 1 1 1 0 0 0 1 1 1 0 0|
+---+---+---+---+---+---+---+---+

```

This chunk contains, in order:

1x "packet not received"

5x "packet received"

3x "packet not received"

3x "packet received"

2x "packet not received"

Example 2:

```

      0                               1
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5
+-----+
|1|1|0 0 1 1 0 1 0 1 0 1 0 0 0 0|
+-----+
```

This chunk contains, in order:

1x "packet not received"

1x "packet received, w/o timestamp"

3x "packet received"

2x "packet not received"

3.1.5. Receive Delta

Deltas are represented as multiples of 250us:

- o If the "Packet received, small delta" symbol has been appended to the status list, an 8-bit unsigned receive delta will be appended to rcv delta list, representing a delta in the range [0, 63.75] ms.
- o If the "Packet received, large or negative delta" symbol has been appended to the status list, a 16-bit signed receive delta will be appended to rcv delta list, representing a delta in the range [-8192.0, 8191.75] ms.
- o If the delta exceeds even the larger limits, a new feedback message must be used, where the 24-bit base receive delta can cover very large gaps.

Note that the first receive delta is relative to the reference time indicated by the base receive delta.

TODO: Add examples.

The smaller receive delta upper bound of 63.75 ms means that this is only viable at about $1000/25.5 \approx 16$ packets per second and above. With a packet size of 1200 bytes/packet that amounts to a bitrate of about 150 kbit/s.

The 0.25 ms resolution means that up to 4000 packets per second can be represented. With a 1200 bytes/packet payload, that amounts to 38.4 Mbit/s payload bandwidth.

4. Overhead discussion

TODO: Examples of overhead in various scenarios.

5. IANA considerations

Upon publication of this document as an RFC (if it is decided to publish it), IANA is requested to register the string "goog-remb" in its registry of "rtcp-fb" values in the SDP attribute registry group.

6. Security Considerations

If the RTCP packet is not protected, it is possible to inject fake RTCP packets that can increase or decrease bandwidth. This is not different from security considerations for any other RTCP message.

7. Acknowledgements

8. References

8.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC3550] Schulzrinne, H., Casner, S., Frederick, R., and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications", STD 64, RFC 3550, July 2003.
- [RFC5285] Singer, D. and H. Desineni, "A General Mechanism for RTP Header Extensions", RFC 5285, DOI 10.17487/RFC5285, July 2008, <<http://www.rfc-editor.org/info/rfc5285>>.

8.2. Informative References

[I-D.ietf-rmcat-gcc]

Holmer, S., Marcon, J., Carlucci, G., Cicco, L., and S. Mascolo, "A Google Congestion Control Algorithm for Real-Time Communication", draft-ietf-rmcat-gcc-00 (work in progress), September 2015.

[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-01 (work in progress), October 2015.

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

Johansson, I. and Z. Sarker, "Self-Clocked Rate Adaptation for Multimedia", draft-ietf-rmcat-scream-cc-01 (work in progress), July 2015.

Appendix A. Change log

A.1. First version

Authors' Addresses

Stefan Holmer
Google
Kungsbron 2
Stockholm 11122
Sweden

Email: holmer@google.com

Magnus Flodman
Google
Kungsbron 2
Stockholm 11122
Sweden

Email: mflodman@google.com

Erik Sprang
Google
Kungsbron 2
Stockholm 11122
Sweden

Email: sprang@google.com