

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

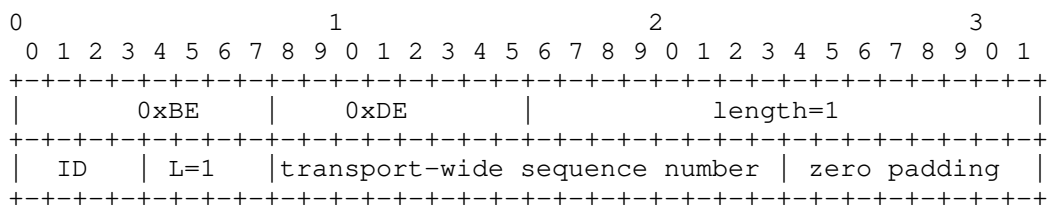
This RTP header extension is added on the transport layer, and uses the same counter for all packets which are sent over the same connection (for instance as defined by bundle).

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.



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:

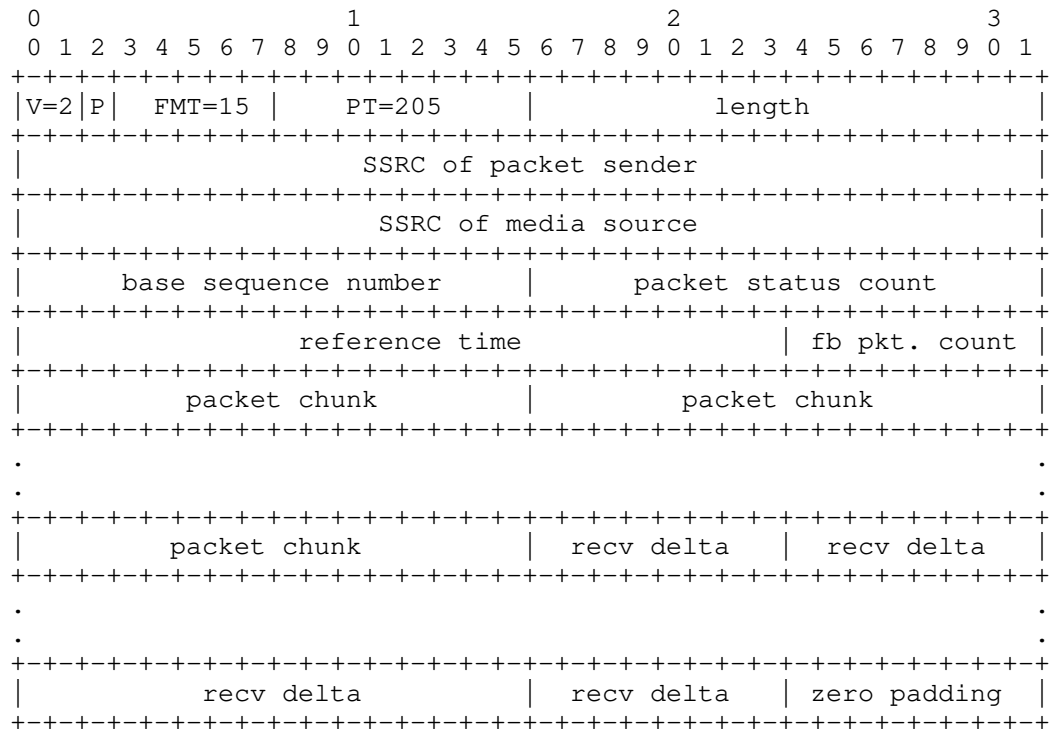
- o It isn't possible to differentiate between lost feedback on the downlink and lost packets on the uplink.
- o Increased feedback rate on the reverse direction.

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 recv 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.

recv 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

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

E. Berger
S. Nandakumar
M. Zanaty
Cisco Systems
October 19, 2015

Frame Marking RTP Header Extension
draft-ietf-avtext-framemarking-00

Abstract

This document describes a Frame Marking RTP header extension used to convey information about video frames that is critical for error recovery and packet forwarding in RTP middleboxes or network nodes. It is most useful when media is encrypted, and essential when the middlebox or node has no access to the media encryption keys. It is also useful for codec-agnostic processing of encrypted or unencrypted media, while it also supports extensions for codec-specific information.

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. Solution	3
2.1. Mandatory Extension	4
2.2. Layer ID Mappings	4
2.2.1. H265 LID Mapping	4
2.2.2. VP9 LID Mapping	5
2.2.3. VP8 LID Mapping	5
2.2.4. H264-SVC LID Mapping	5
2.2.5. H264 (AVC) LID Mapping	5
2.3. Signaling information	6
2.4. Considerations on use	6
3. Security Considerations	6
4. Acknowledgements	6
5. IANA Considerations	7
6. References	7
6.1. Normative References	7
6.2. Informative References	7
Authors' Addresses	8

1. Introduction

Many widely deployed RTP topologies used in modern voice and video conferencing systems include a centralized component that acts as an RTP switch. It receives voice and video streams from each participant, which may be encrypted using SRTP [RFC3711], or extensions that provide participants with private media via end-to-end encryption that excludes the switch. The goal is to provide a set of streams back to the participants which enable them to render the right media content. In a simple video configuration, for example, the goal will be that each participant sees and hears just the active speaker. In that case, the goal of the switch is to receive the voice and video streams from each participant, determine the active speaker based on energy in the voice packets, possibly using the client-to-mixer audio level RTP header extension, and select the corresponding video stream for transmission to participants; see Figure 1.

In this document, an "RTP switch" is used as a common short term for the terms "switching RTP mixer", "source projecting middlebox", "source forwarding unit/middlebox" and "video switching MCU" as discussed in [I-D.ietf-avtcore-rtp-topologies-update].

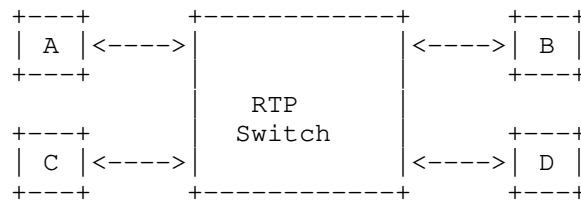


Figure 1: RTP switch

In order to properly support switching of video streams, the RTP switch typically needs some critical information about video frames in order to start and stop forwarding streams.

- o Because of inter-frame dependencies, it should ideally switch video streams at a point where the first frame from the new speaker can be decoded by recipients without prior frames, e.g. switch on an intra-frame.
- o In many cases, the switch may need to drop frames in order to realize congestion control techniques, and needs to know which frames can be dropped with minimal impact to video quality.
- o Furthermore, it is highly desirable to do this in a way which is not specific to the video codec. Nearly all modern video codecs share common concepts around frame types.
- o It is also desirable to be able to do this for SRTP without requiring the video switch to decrypt the packets. SRTP will encrypt the RTP payload format contents and consequently this data is not usable for the switching function without decryption, which may not even be possible in the case of end-to-end encryption of private media.

A comprehensive discussion of SFU considerations around codec agnostic selective forwarding of RTP media is described in [I-D.draft-aboba-avtcore-sfu-rtp]

By providing meta-information about the RTP streams outside the encrypted media payload an RTP switch can do selective forwarding without decrypting the payload. This document provides a solution to this problem.

2. Solution

The solution uses RTP header extensions as defined in [RFC5285]. A subset of meta-information from the video stream is provided as an RTP header extension to allow a RTP switch to do generic video switching handling of video streams encoded with different video codecs.

2.1. Mandatory Extension

The following information are extracted from the media payload:

- o S: Start of Frame (1 bit) - MUST be 1 in the first packet in a frame within a layer; otherwise MUST be 0.
- o E: End of Frame (1 bit) - MUST be 1 in the last packet in a frame within a layer; otherwise MUST be 0.
- o I: Independent Frame (1 bit) - MUST be 1 for frames that can be decoded independent of prior frames, e.g. intra-frame, VPx keyframe, H.264 IDR [RFC6184], H.265 CRA/BLA; otherwise MUST be 0.
- o D: Discardable Frame (1 bit) - MUST be 1 for frames that can be dropped, and still provide a decodable media stream; otherwise MUST be 0.
- o B: Base Layer Sync (1 bit) - MUST be 1 if this frame only depends on the base layer; otherwise MUST be 0.
- o TID: Temporal ID (3 bits) - The base temporal quality starts with 0, and increases with 1 for each temporal layer/sub-layer.
- o LID: Layer ID (8 bits) - Identifies the spatial and quality layer encoded.

NOTE: Given the opaque nature of the LID, consider having the layer structure information as RTCP SDES item (either in the RTCP SDES message or as the RTP SDES Header extension) to map the LIDs to specific resolutions and bitrates thus enabling the RTP Switch to make informed decisions

The values of frame information can be carried as RTP header extensions encoded using the one-byte header as described in [RFC5285] as shown below.

```

      0                               1                               2
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
| ID=2 | L=1 | S | E | I | D | B | TID | LID |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+

```

2.2. Layer ID Mappings

2.2.1. H265 LID Mapping

The following shows H265-LayerID (6 bits) mapped to the generic LID field.

```

      0                               1                               2
      0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
| ID=2 | L=1 | S|E|I|D|B| TID | 0|0| LayerID |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+

```

2.2.2. VP9 LID Mapping

The following shows VP9 Layer encoding information (4 bits for spatial and quality) mapped to the generic LID field.

```

      0                               1                               2
      0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
| ID=2 | L=1 | S|E|I|D|B| TID | 0|0|0|0| RS| RQ|
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+

```

2.2.3. VP8 LID Mapping

The following shows the header extension for VP8 that contains no layer information.

```

      0                               1                               2
      0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
| ID=2 | L=1 | S|E|I|D|B| TID | 0|0|0|0|0|0|0|0|0|0|
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+

```

2.2.4. H264-SVC LID Mapping

The following shows H264-SVC Layer encoding information (3 bits for spatial and 4 bits quality) mapped to the generic LID field.

```

      0                               1                               2
      0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
| ID=2 | L=1 | S|E|I|D|B| TID | 0| DID | QID |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+

```

2.2.5. H264 (AVC) LID Mapping

The following shows the header extension for H264 (AVC) that contains no layer information.


```

      0               1               2
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3
+---+---+---+---+---+---+---+---+---+---+---+---+---+
| ID=2 | L=1 | S|E|I|D|B| TID |0|0|0|0|0|0|0|0|0|
+---+---+---+---+---+---+---+---+---+---+---+---+

```

2.3. Signaling information

The URI for declaring this header extension in an extmap attribute is "urn:ietf:params:rtp-hdext:framemarkinginfo". It does not contain any extension attributes.

An example attribute line in SDP:

```
a=extmap:3 urn:ietf:params:rtp-hdext:framemarkinginfo
```

2.4. Considerations on use

The header extension values MUST represent what is already in the RTP payload.

When a RTP switch needs to discard a received video frame due to congestion control considerations, it is RECOMMENDED that it preferably drop frames marked with the "discardable" bit.

When a RTP switch wants to forward a new video stream to a receiver, it is RECOMMENDED to select the new video stream from the first switching point (I bit set) and forward the same. A RTP switch can request a media source to generate a switching point for H.264 by sending Full Intra Request (RTCP FIR) as defined in [RFC5104], for example.

3. Security Considerations

In the Secure Real-Time Transport Protocol (SRTP) [RFC3711], RTP header extensions are authenticated but not encrypted. When header extensions are used some of the payload type information are exposed and is visible to middle boxes. The encrypted media data is not exposed, so this is not seen as a high risk exposure.

4. Acknowledgements

Many thanks to Bernard Aboba, Jonathan Lennox for their inputs.

5. IANA Considerations

This document defines a new extension URI to the RTP Compact HeaderExtensions sub-registry of the Real-Time Transport Protocol (RTP) Parameters registry, according to the following data:

Extension URI: urn:ietf:params:rtp-hdext:framemarkinginfo
Description: Frame marking information for video streams
Contact: espeberg@cisco.com
Reference: RFC XXXX

Note to RFC Editor: please replace RFC XXXX with the number of this RFC.

6. References

6.1. Normative References

[KEYWORDS]

Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

6.2. Informative References

[I-D.ietf-avtcore-rtp-topologies-update]

Westerlund, M. and S. Wenger, "RTP Topologies", draft-ietf-avtcore-rtp-topologies-update (work in progress), April 2013.

[I-D.draft-aboba-avtcore-sfu-rtp]

Aboba, B., "Codec-Independent Selective Forwarding", draft-aboba-avtcore-sfu-rtp-00 (work in progress), July 2015.

[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>>.

[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>>.

[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>>.

- [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>>.
- [RFC6184] Wang, Y., Even, R., Kristensen, T., and R. Jesup, "RTP Payload Format for H.264 Video", RFC 6184, DOI 10.17487/RFC6184, May 2011, <<http://www.rfc-editor.org/info/rfc6184>>.

Authors' Addresses

Espen Berger
Cisco Systems

Phone: +47 98228179
Email: espeberg@cisco.com

Suhas Nandakumar
Cisco Systems
170 West Tasman Drive
San Jose, CA 95134
US

Email: snandaku@cisco.com

Mo Zanaty
Cisco Systems
170 West Tasman Drive
San Jose, CA 95134
US

Email: mzanaty@cisco.com

Payload Working Group
Internet-Draft
Intended status: Standards Track
Expires: April 21, 2016

J. Lennox
D. Hong
Vidyo
J. Uberti
S. Holmer
M. Flodman
Google
October 19, 2015

The Layer Refresh Request (LRR) RTCP Feedback Message
draft-ietf-avtext-lrr-01

Abstract

This memo describes the RTCP Payload-Specific Feedback Message "Layer Refresh Request" (LRR), which can be used to request a state refresh of one or more substreams of a layered media stream. It also defines its use with several scalable media formats.

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. Conventions, Definitions and Acronyms	2
2.1. Terminology	3
3. Layer Refresh Request	5
3.1. Message Format	5
4. Usage with specific codecs	6
4.1. H264 SVC	7
4.2. VP8	8
4.3. H265	8
4.4. VP9	10
5. Usage with different scalability transmission mechanisms . .	10
6. Security Considerations	11
7. IANA Considerations	11
8. References	11
8.1. Normative References	11
8.2. Informative References	12
Authors' Addresses	12

1. Introduction

This memo describes an RTCP [RFC3550] Payload-Specific Feedback Message [RFC4585] "Layer Refresh Request" (LRR). It is designed to allow a receiver of a layered media stream to request that one or more of its substreams be refreshed, such that it can then be decoded by an endpoint which previously was not receiving those layers, without requiring that the entire stream be refreshed (as it would be if the receiver sent a Full Intra Request (FIR) [RFC5104]).

The message is designed to be applicable both to temporally and spatially scaled streams, and to both single-stream and multi-stream scalability modes.

2. Conventions, Definitions and Acronyms

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].

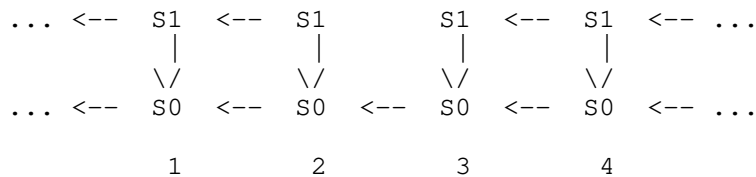
2.1. Terminology

A "Layer Refresh Point" is a point in a scalable stream after which a decoder, which previously had been able to decode only some (possibly none) of the available layers of stream, is able to decode a greater number of the layers.

For spatial (or quality) layers, layer refresh typically requires that a spatial layer be encoded in a way that references only lower-layer subpictures of the current picture, not any earlier pictures of that spatial layer. Additionally, the encoder must promise that no earlier pictures of that spatial layer will be used as reference in the future.

In a layer refresh, however, other layers than the ones requested for refresh may still maintain dependency on earlier content of the stream. This is the difference between a layer refresh and a Full Intra Request [RFC5104]. This minimizes the coding overhead of refresh to only those parts of the stream that actually need to be refreshed at any given time.

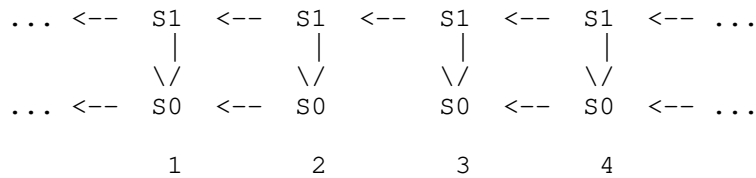
An illustration of spatial layer refresh of an enhancement layer is shown below.



In this illustration, frame 3 is a layer refresh point for spatial layer S1; a decoder which had previously only been decoding spatial layer S0 would be able to decode layer S1 starting at frame 3.

Figure 1

An illustration of spatial layer refresh of a base layer is shown below.

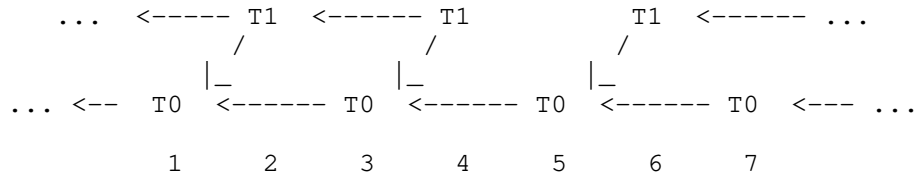


In this illustration, frame 3 is a layer refresh point for spatial layer S0; a decoder which had previously not been decoding the stream at all could decode layer S0 starting at frame 3.

Figure 2

For temporal layers, layer refresh requires that the layer be "temporally nested", i.e. use as reference only earlier frames of a lower temporal layer, not any earlier frames of this temporal layer, and also promise that no future frames of this temporal layer will reference frames of this temporal layer before the refresh point. In many cases, the temporal structure of the stream will mean that all frames are temporally nested, in which case decoders will have no need to send LRR messages for the stream.

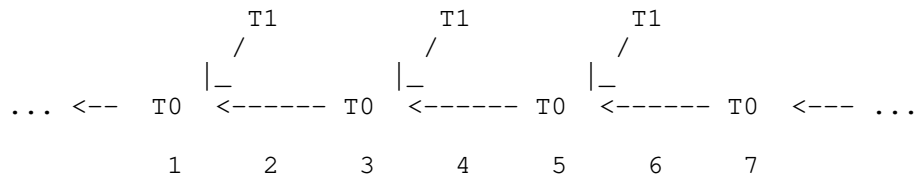
An illustration of temporal layer refresh is shown below.



In this illustration, frame 6 is a layer refresh point for temporal layer T1; a decoder which had previously only been decoding temporal layer T0 would be able to decode layer T1 starting at frame 6.

Figure 3

An illustration of an inherently temporally nested stream is shown below.



In this illustration, the stream is temporally nested in its ordinary structure; a decoder receiving layer T0 can begin decoding layer T1 at any point.

Figure 4

3. Layer Refresh Request

A layer refresh frame can be requested by sending a Layer Refresh Request (LRR), which is an RTCP payload-specific feedback message [RFC4585] asking the encoder to encode a frame which makes it possible to upgrade to a higher layer. The LRR contains one or two tuples, indicating the layer the decoder wants to upgrade to, and (optionally) the currently highest layer the decoder can decode.

The specific format of the tuples, and the mechanism by which a receiver recognizes a refresh frame, is codec-dependent. Usage for several codecs is discussed in Section 4.

LRR follows the model of the Full Intra Request (FIR) [RFC5104] (Section 3.5.1) for its retransmission, reliability, and use in multipoint conferences. TODO: expand these here.

The LRR message is identified by RTCP packet type value PT=PSFB and FMT=TBD. The FCI field MUST contain one or more FIR entries. Each entry applies to a different media sender, identified by its SSRC.

3.1. Message Format

The Feedback Control Information (FCI) for the Layer Refresh Request consists of one or more FCI entries, the content of which is depicted in Figure 5. The length of the LRR feedback message MUST be set to $2+3*N$, where N is the number of FCI entries.

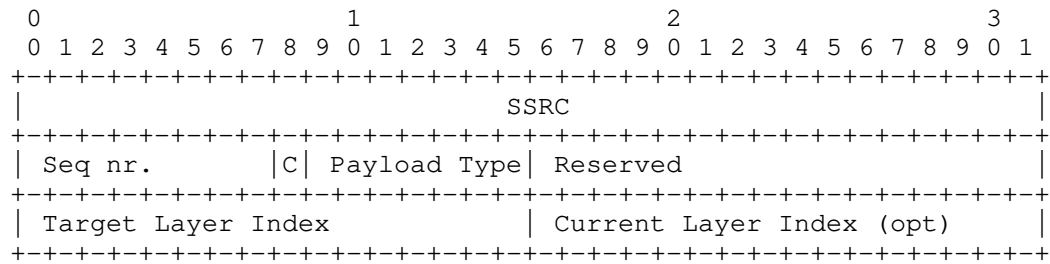


Figure 5

SSRC (32 bits) The SSRC value of the media sender that is requested to send a layer refresh point.

Seq nr. (8 bits) Command sequence number. The sequence number space is unique for each pairing of the SSRC of command source and the SSRC of the command target. The sequence number SHALL be increased by 1 modulo 256 for each new command. A repetition SHALL NOT increase the sequence number. The initial value is arbitrary.

C (1 bit) A flag bit indicating whether the "Current Layer Index" field is present in the FCI. If this bit is false, the sender of the LRR message is requesting refresh of all layers up to and including the target layer.

Payload Type (7 bits) The RTP payload type for which the LRR is being requested. This gives the context in which the target layer index is to be interpreted.

Reserved (16 bits) All bits SHALL be set to 0 by the sender and SHALL be ignored on reception.

Target Layer Index (16 bits) The target layer for which the receiver wishes a refresh point. Its format is dependent on the payload type field.

Current Layer Index (16 bits) If C is 1, the current layer being decoded by the receiver. This message is not requesting refresh of layers at or below this layer. If C is 0, this field SHALL be set to 0 by the sender and SHALL be ignored on reception.

4. Usage with specific codecs

4.1. H264 SVC

H.264 SVC [RFC6190] defines temporal, dependency (spatial), and quality scalability modes.

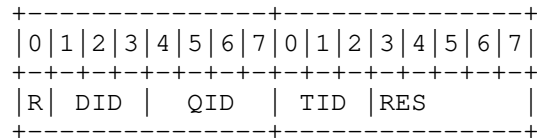


Figure 6

Figure 6 shows the format of the layer index field for H.264 SVC streams. This is designed to follow the same layout as the third and fourth bytes of the H.264 SVC NAL unit extension, which carry the stream's layer information. The "R" and "RES" fields MUST be set to 0 on transmission and ignored on reception. See [RFC6190] Section 1.1.3 for details on the DID, QID, and TID fields.

A dependency or quality layer refresh of a given layer in H.264 SVC can be identified by the "I" bit (idr_flag) in the extended NAL unit header, present in NAL unit types 14 (prefix NAL unit) and 20 (coded scalable slice). Layer refresh of the base layer can also be identified by its NAL unit type of its coded slices, which is "5" rather than "1". A dependency or quality layer refresh is complete once this bit has been seen on all the appropriate layers (in decoding order) above the current layer index (if any, or beginning from the base layer if not) through the target layer index.

Note that as the "I" bit in a PACSI header is set if the corresponding bit is set in any of the aggregated NAL units it describes; thus, it is not sufficient to identify layer refresh when NAL units of multiple dependency or quality layers are aggregated.

In H.264 SVC, temporal layer refresh information can be determined from various Supplemental Encoding Information (SEI) messages in the bitstream.

Whether an H.264 SVC stream is scalably nested can be determined from the Scalability Information SEI message's temporal_id_nesting flag. If this flag is set in a stream's currently applicable Scalability Information SEI, receivers SHOULD NOT send temporal LRR messages for that stream, as every frame is implicitly a temporal layer refresh point. (The Scalability Information SEI message may also be available in the signaling negotiation of H.264 SVC, as the sprop-scalability-info parameter.)

If a stream's `temporal_id_nesting` flag is not set, the Temporal Level Switching Point SEI message identifies temporal layer switching points. A temporal layer refresh is satisfied when this SEI message is present in a frame with the target layer index, if the message's `delta_frame_num` refer to a frame with the requested current layer index. (Alternately, temporal layer refresh can also be satisfied by a complete state refresh, such as an IDR.) Senders which support receiving LRR for non-scalably-nested streams MUST insert Temporal Level Switching Point SEI messages as appropriate.

4.2. VP8

The VP8 RTP payload format [I-D.ietf-payload-vp8] defines temporal scalability modes. It does not support spatial scalability.

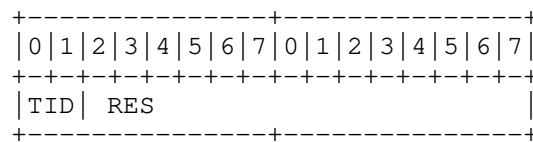


Figure 7

Figure 7 shows the format of the layer index field for VP8 streams. The "RES" fields MUST be set to 0 on transmission and ignored on reception. See [I-D.ietf-payload-vp8] Section 4.2 for details on the TID field.

A VP8 layer refresh point can be identified by the presence of the "Y" bit in the VP8 payload header. When this bit is set, this and all subsequent frames depend only on the current base temporal layer. On receipt of an LRR for a VP8 stream, A sender which supports LRR MUST encode the stream so it can set the Y bit in a packet whose temporal layer is at or below the target layer index.

Note that in VP8, not every layer switch point can be identified by the Y bit, since the Y bit implies layer switch of all layers, not just the layer in which it is sent. Thus the use of LRR with VP8 can result in some inefficiency in transmission. However, this is not expected to be a major issue for temporal structures in normal use.

4.3. H265

The initial version of the H.265 payload format [I-D.ietf-payload-rtp-h265] defines temporal scalability, with protocol elements reserved for spatial or other scalability modes (which are expected to be defined in a future version of the specification).

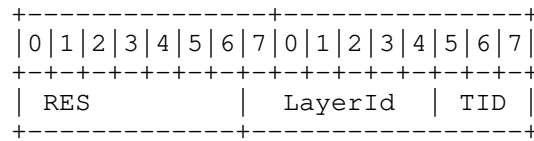


Figure 8

Figure 8 shows the format of the layer index field for H.265 streams. This is designed to follow the same layout as the first and second bytes of the H.265 NAL unit header, which carry the stream's layer information. The "RES" field MUST be set to 0 on transmission and ignored on reception. See [I-D.ietf-payload-rtp-h265] Section 1.1.4 for details on the LayerId and TID fields.

H.265 streams signal whether they are temporally nested, using the `vps_temporal_id_nesting_flag` in the Video Parameter Set (VPS), and the `sps_temporal_id_nesting_flag` in the Sequence Parameter Set (SPS). If this flag is set in a stream's currently applicable VPS or SPS, receivers SHOULD NOT send temporal LRR messages for that stream, as every frame is implicitly a temporal layer refresh point.

If a stream's `sps_temporal_id_nesting_flag` is not set, the NAL unit types 2 to 5 inclusively identify temporal layer switching points. A layer refresh to any higher target temporal layer is satisfied when a NAL unit type of 4 or 5 with TID equal to 1 more than current TID is seen. Alternatively, layer refresh to a target temporal layer can be incrementally satisfied with NAL unit type of 2 or 3. In this case, given current TID = T0 and target TID = TN, layer refresh to TN is satisfied when NAL unit type of 2 or 3 is seen for TID = T1, then TID = T2, all the way up to TID = TN. During this incremental process, layer refresh to TN can be completely satisfied as soon as a NAL unit type of 2 or 3 is seen.

Of course, temporal layer refresh can also be satisfied whenever any Intra Random Access Point (IRAP) NAL unit type (with values 16-23, inclusively) is seen. An IRAP picture is similar to an IDR picture in H.264 (NAL unit type of 5 in H.264) where decoding of the picture can start without any older pictures.

In the (future) H.265 payloads that support spatial scalability, a spatial layer refresh of a specific layer can be identified by NAL units with the requested layer ID and NAL unit types between 16 and 21 inclusive. A dependency or quality layer refresh is complete once NAL units of this type have been seen on all the appropriate layers (in decoding order) above the current layer index (if any, or beginning from the base layer if not) through the target layer index.

4.4. VP9

The RTP payload format for VP9 [I-D.uberti-payload-vp9] defines how it can be used for spatial and temporal scalability.

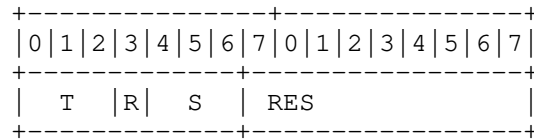


Figure 9

Figure 9 shows the format of the layer index field for VP9 streams. This is designed to follow the same layout as the "L" byte of the VP9 payload header, which carries the stream's layer information. The "R" and "RES" fields MUST be set to 0 on transmission and ignored on reception. See [I-D.uberti-payload-vp9] for details on the T and S fields.

Identification of a layer refresh frame can be derived from the reference IDs of each frame by backtracking the dependency chain until reaching a point where only decodable frames are being referenced. Therefore it's recommended for both the flexible and the non-flexible mode that, when upgrade frames are being encoded in response to a LRR, those packets should contain layer indices and the reference fields so that the decoder or an MCU can make this derivation.

Example:

LRR {1,0}, {2,1} is sent by an MCU when it is currently relaying {1,0} to a receiver and which wants to upgrade to {2,1}. In response the encoder should encode the next frames in layers {1,1} and {2,1} by only referring to frames in {1,0}, or {0,0}.

In the non-flexible mode, periodic upgrade frames can be defined by the layer structure of the SS, thus periodic upgrade frames can be automatically identified by the picture ID.

5. Usage with different scalability transmission mechanisms

Several different mechanisms are defined for how scalable streams can be transmitted in RTP. The RTP Taxonomy [I-D.ietf-avtext-rtp-grouping-taxonomy] Section 3.7 defines three mechanisms: Single RTP Stream on a Single Media Transport (SRST), Multiple RTP Streams on a Single Media Transport (MRST), and Multiple RTP Streams on Multiple Media Transports (MRMT).

The LRR message is applicable to all these mechanisms. For MRST and MRMT mechanisms, the "media source" field of the LRR FCI is set to the SSRC of the RTP stream containing the layer indicated by the Current Layer Index (if "C" is 1), or the stream containing the base encoded stream (if "C" is 0). For MRMT, it is sent on the RTP session on which this stream is sent. On receipt, the sender MUST refresh all the layers requested in the stream, simultaneously in decode order.

Note: arguably, for the MRST and MRMT mechanisms, FIR feedback messages could instead be used to refresh specific individual layers. However, the usage of FIR for MRSR/MRMT is not explicitly specified anywhere, and if FIR is interpreted as refreshing layers, there is no way to request an actual full, synchronized refresh of all the layers of an MRST/MRMT layered source. Thus, the authors feel that interpreting FIR as refreshing the entire source, and using LRR for the individual layers, would be more useful.

6. Security Considerations

All the security considerations of FIR feedback packets [RFC5104] apply to LRR feedback packets as well. Additionally, media senders receiving LRR feedback packets MUST validate that the payload types and layer indices they are receiving are valid for the stream they are currently sending, and discard the requests if not.

7. IANA Considerations

The IANA is requested to register the following values:

- TODO: PSFB value for LRR

8. References

8.1. Normative References

[I-D.ietf-payload-rtp-h265]

Wang, Y., Sanchez, Y., Schierl, T., Wenger, S., and M. Hannuksela, "RTP Payload Format for H.265/HEVC Video", draft-ietf-payload-rtp-h265-14 (work in progress), August 2015.

[I-D.ietf-payload-vp8]

Westin, P., Lundin, H., Glover, M., Uberti, J., and F. Galligan, "RTP Payload Format for VP8 Video", draft-ietf-payload-vp8-17 (work in progress), September 2015.

- [I-D.uberti-payload-vp9]
Uberti, J., Holmer, S., Flodman, M., Lennox, J., and D. Hong, "RTP Payload Format for VP9 Video", draft-uberti-payload-vp9-01 (work in progress), March 2015.
- [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>>.
- [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>>.
- [RFC6190] Wenger, S., Wang, Y., Schierl, T., and A. Eleftheriadis, "RTP Payload Format for Scalable Video Coding", RFC 6190, DOI 10.17487/RFC6190, May 2011, <<http://www.rfc-editor.org/info/rfc6190>>.

8.2. Informative References

- [I-D.ietf-avtext-rtp-grouping-taxonomy]
Lennox, J., Gross, K., Nandakumar, S., Salgueiro, G., and B. Burman, "A Taxonomy of Semantics and Mechanisms for Real-Time Transport Protocol (RTP) Sources", draft-ietf-avtext-rtp-grouping-taxonomy-08 (work in progress), July 2015.
- [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

Jonathan Lennox
Vidyo, Inc.
433 Hackensack Avenue
Seventh Floor
Hackensack, NJ 07601
US

Email: jonathan@vidyo.com

Danny Hong
Vidyo, Inc.
433 Hackensack Avenue
Seventh Floor
Hackensack, NJ 07601
US

Email: danny@vidyo.com

Justin Uberti
Google, Inc.
747 6th Street South
Kirkland, WA 98033
USA

Email: justin@uberti.name

Stefan Holmer
Google, Inc.
Kungsbron 2
Stockholm 111 22
Sweden

Email: holmer@google.com

Magnus Flodman
Google, Inc.
Kungsbron 2
Stockholm 111 22
Sweden

Email: mflodman@google.com

Network Working Group
Internet-Draft
Intended status: Informational
Expires: January 21, 2016

J. Lennox
Vidyo
K. Gross
AVA
S. Nandakumar
G. Salgueiro
Cisco Systems
B. Burman, Ed.
Ericsson
July 20, 2015

A Taxonomy of Semantics and Mechanisms for Real-Time Transport Protocol
(RTP) Sources
draft-ietf-avtext-rtp-grouping-taxonomy-08

Abstract

The terminology about, and associations among, Real-Time Transport Protocol (RTP) sources can be complex and somewhat opaque. This document describes a number of existing and proposed properties and relationships among RTP sources, and defines common terminology for discussing protocol entities and their relationships.

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 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	4
2. Concepts	4
2.1. Media Chain	5
2.1.1. Physical Stimulus	9
2.1.2. Media Capture	9
2.1.3. Raw Stream	9
2.1.4. Media Source	10
2.1.5. Source Stream	10
2.1.6. Media Encoder	11
2.1.7. Encoded Stream	12
2.1.8. Dependent Stream	12
2.1.9. Media Packetizer	12
2.1.10. RTP Stream	13
2.1.11. RTP-based Redundancy	13
2.1.12. Redundancy RTP Stream	14
2.1.13. RTP-based Security	14
2.1.14. Secured RTP Stream	15
2.1.15. Media Transport	15
2.1.16. Media Transport Sender	16
2.1.17. Sent RTP Stream	17
2.1.18. Network Transport	17
2.1.19. Transported RTP Stream	17
2.1.20. Media Transport Receiver	17
2.1.21. Received Secured RTP Stream	18
2.1.22. RTP-based Validation	18
2.1.23. Received RTP Stream	18
2.1.24. Received Redundancy RTP Stream	18
2.1.25. RTP-based Repair	18
2.1.26. Repaired RTP Stream	18
2.1.27. Media Depacketizer	19
2.1.28. Received Encoded Stream	19
2.1.29. Media Decoder	19
2.1.30. Received Source Stream	19
2.1.31. Media Sink	19
2.1.32. Received Raw Stream	20
2.1.33. Media Render	20
2.2. Communication Entities	20
2.2.1. Endpoint	22

2.2.2.	RTP Session	22
2.2.3.	Participant	23
2.2.4.	Multimedia Session	23
2.2.5.	Communication Session	24
3.	Concepts of Inter-Relations	24
3.1.	Synchronization Context	24
3.1.1.	RTCP CNAME	25
3.1.2.	Clock Source Signaling	25
3.1.3.	Implicitly via RtcMediaStream	25
3.1.4.	Explicitly via SDP Mechanisms	25
3.2.	Endpoint	25
3.3.	Participant	26
3.4.	RtcMediaStream	26
3.5.	Multi-Channel Audio	26
3.6.	Simulcast	27
3.7.	Layered Multi-Stream	28
3.8.	RTP Stream Duplication	29
3.9.	Redundancy Format	30
3.10.	RTP Retransmission	31
3.11.	Forward Error Correction	33
3.12.	RTP Stream Separation	34
3.13.	Multiple RTP Sessions over one Media Transport	35
4.	Mapping from Existing Terms	35
4.1.	Telepresence Terms	35
4.1.1.	Audio Capture	35
4.1.2.	Capture Device	35
4.1.3.	Capture Encoding	36
4.1.4.	Capture Scene	36
4.1.5.	Endpoint	36
4.1.6.	Individual Encoding	36
4.1.7.	Media Capture	36
4.1.8.	Media Consumer	36
4.1.9.	Media Provider	37
4.1.10.	Stream	37
4.1.11.	Video Capture	37
4.2.	Media Description	37
4.3.	Media Stream	37
4.4.	Multimedia Conference	37
4.5.	Multimedia Session	38
4.6.	Multipoint Control Unit (MCU)	38
4.7.	Multi-Session Transmission (MST)	38
4.8.	Recording Device	39
4.9.	RtcMediaStream	39
4.10.	RtcMediaStreamTrack	39
4.11.	RTP Sender	39
4.12.	RTP Session	39
4.13.	Single Session Transmission (SST)	39
4.14.	SSRC	39

5. Security Considerations	40
6. Acknowledgement	40
7. Contributors	40
8. IANA Considerations	41
9. Informative References	41
Appendix A. Changes From Earlier Versions	44
A.1. Modifications Between WG Version -07 and -08	44
A.2. Modifications Between WG Version -06 and -07	45
A.3. Modifications Between WG Version -05 and -06	45
A.4. Modifications Between WG Version -04 and -05	46
A.5. Modifications Between WG Version -03 and -04	46
A.6. Modifications Between WG Version -02 and -03	47
A.7. Modifications Between WG Version -01 and -02	47
A.8. Modifications Between WG Version -00 and -01	48
A.9. Modifications Between Version -02 and -03	48
A.10. Modifications Between Version -01 and -02	48
A.11. Modifications Between Version -00 and -01	48
Authors' Addresses	49

1. Introduction

The existing taxonomy of sources in the Real-Time Transport Protocol (RTP) [RFC3550] has previously been regarded as confusing and inconsistent. Consequently, a deep understanding of how the different terms relate to each other becomes a real challenge. Frequently cited examples of this confusion are (1) how different protocols that make use of RTP use the same terms to signify different things and (2) how the complexities addressed at one layer are often glossed over or ignored at another.

This document improves clarity by reviewing the semantics of various aspects of sources in RTP. As an organizing mechanism, it approaches this by describing various ways that RTP sources are transformed on their way between sender and receiver, and how they can be grouped and associated together.

All non-specific references to ControLling mUltiple streams for tElepresence (CLUE) in this document map to [I-D.ietf-clue-framework] and all references to Web Real-Time Communications (WebRTC) map to [I-D.ietf-rtcweb-overview].

2. Concepts

This section defines concepts that serve to identify and name various transformations and streams in a given RTP usage. For each concept, alternate definitions and usages that co-exist today are listed along with various characteristics that further describes the concept. These concepts are divided into two categories, one related to the

chain of streams and transformations that media can be subject to, the other for entities involved in the communication.

2.1. Media Chain

In the context of this document, Media is a sequence of synthetic or Physical Stimuli (Section 2.1.1) (sound waves, photons, key-strokes), represented in digital form. Synthesized Media is typically generated directly in the digital domain.

This section contains the concepts that can be involved in taking Media at a sender side and transporting it to a receiver, which may recover a sequence of physical stimuli. This chain of concepts is of two main types, streams and transformations. Streams are time-based sequences of samples of the physical stimulus in various representations, while transformations changes the representation of the streams in some way.

The below examples are basic ones and it is important to keep in mind that this conceptual model enables more complex usages. Some will be further discussed in later sections of this document. In general the following applies to this model:

- o A transformation may have zero or more inputs and one or more outputs.
- o A stream is of some type, such as audio, video, real-time text, etc.
- o A stream has one source transformation and one or more sink transformations (with the exception of Physical Stimulus (Section 2.1.1) that may lack source or sink transformation).
- o Streams can be forwarded from a transformation output to any number of inputs on other transformations that support that type.
- o If the output of a transformation is sent to multiple transformations, those streams will be identical; it takes a transformation to make them different.
- o There are no formal limitations on how streams are connected to transformations.

It is also important to remember that this is a conceptual model. Thus real-world implementations may look different and have different structure.

To provide a basic understanding of the relationships in the chain we first introduce the concepts for the sender side (Figure 1). This covers physical stimuli until media packets are emitted onto the network.

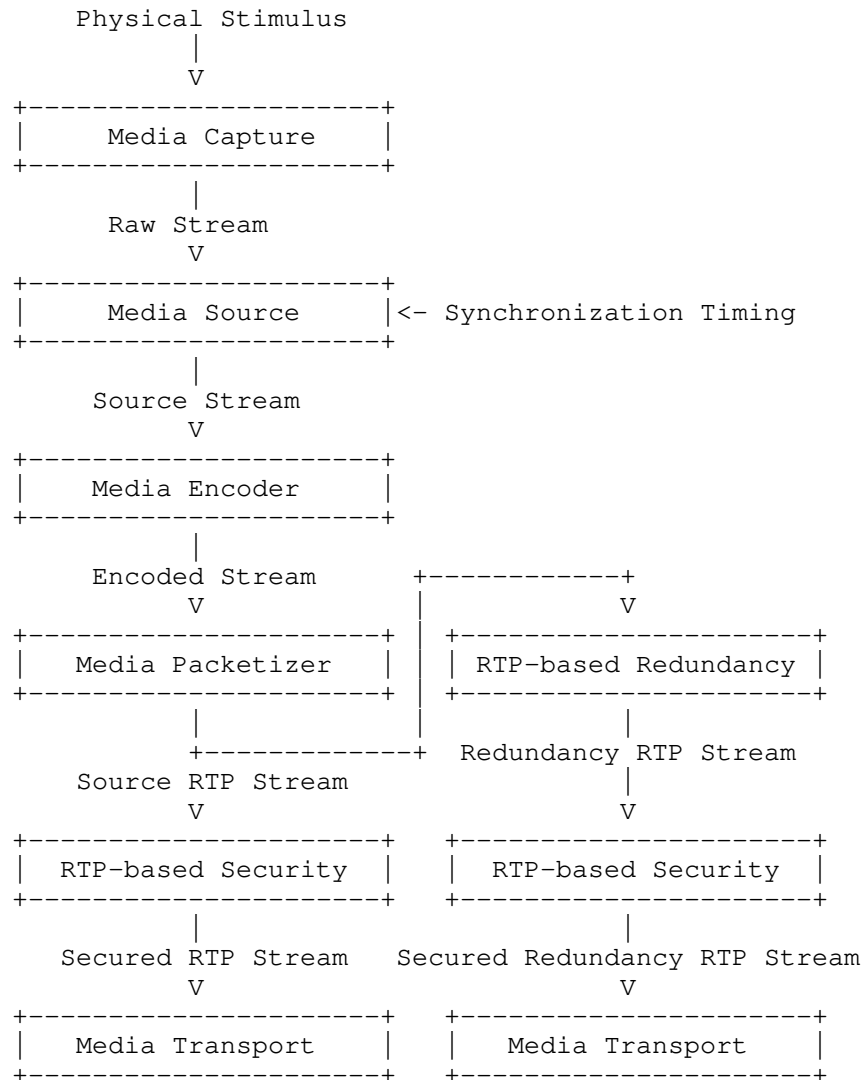


Figure 1: Sender Side Concepts in the Media Chain

In Figure 1 we have included a branched chain to cover the concepts for using redundancy to improve the reliability of the transport.

The Media Transport concept is an aggregate that is decomposed in Section 2.1.15.

In Figure 2 we review a receiver media chain matching the sender side, to look at the inverse transformations and their attempts to recover identical streams as in the sender chain, subject to what may be lossy compression and imperfect Media Transport. Note that the streams out of a reverse transformation, like the Source Stream out the Media Decoder are in many cases not the same as the corresponding ones on the sender side, thus they are prefixed with a "Received" to denote a potentially modified version. The reason for not being the same lies in the transformations that can be of irreversible type. For example, lossy source coding in the Media Encoder prevents the Source Stream out of the Media Decoder to be the same as the one fed into the Media Encoder. Other reasons include packet loss or late loss in the Media Transport transformation that even RTP-based Repair, if used, fails to repair. However, some transformations are not always present, like RTP-based Repair that cannot operate without Redundancy RTP Streams.

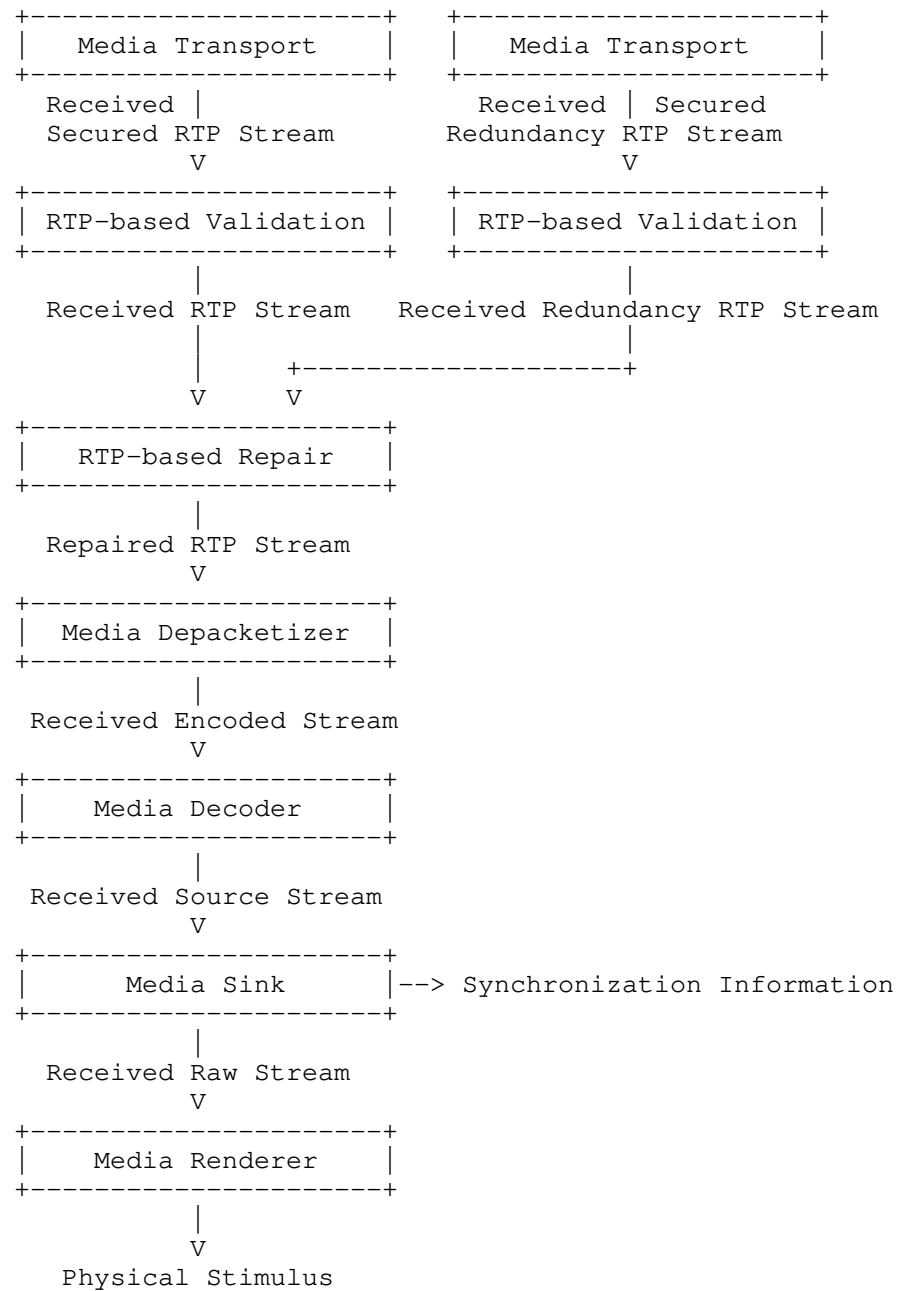


Figure 2: Receiver Side Concepts of the Media Chain

2.1.1. Physical Stimulus

The Physical Stimulus is a physical event in the analog domain that can be sampled and converted to digital form by an appropriate sensor or transducer. This include sound waves making up audio, photons in a light field, or other excitations or interactions with sensors, like keystrokes on a keyboard.

2.1.2. Media Capture

Media Capture is the process of transforming the analog Physical Stimulus (Section 2.1.1) into digital Media using an appropriate sensor or transducer. The Media Capture performs a digital sampling of the physical stimulus, usually periodically, and outputs this in some representation as a Raw Stream (Section 2.1.3). This data is considered "Media", because it includes data that is periodically sampled, or made up of a set of timed asynchronous events. The Media Capture is normally instantiated in some type of device, i.e. media capture device. Examples of different types of media capturing devices are digital cameras, microphones connected to A/D converters, or keyboards.

Characteristics:

- o A Media Capture is identified either by hardware/manufacture ID or via a session-scoped device identifier as mandated by the application usage.
- o A Media Capture can generate an Encoded Stream (Section 2.1.7) if the capture device supports such a configuration.
- o The nature of the Media Capture may impose constraints on the clock handling in some of the subsequent steps. For example, many audio or video capture devices are not completely free in selecting the sample rate.

2.1.3. Raw Stream

A Raw Stream is the time progressing stream of digitally sampled information, usually periodically sampled and provided by a Media Capture (Section 2.1.2). A Raw Stream can also contain synthesized Media that may not require any explicit Media Capture, since it is already in an appropriate digital form.

2.1.4. Media Source

A Media Source is the logical source of a time progressing digital media stream synchronized to a reference clock. This stream is called a Source Stream (Section 2.1.5). This transformation takes one or more Raw Streams (Section 2.1.3) and provides a Source Stream as output. The output is synchronized with a reference clock (Section 3.1), which can be as simple as a system local wall clock or as complex as an NTP synchronized clock.

The output can be of different types. One type is directly associated with a particular Media Capture's Raw Stream. Others are more conceptual sources, like an audio mix of multiple Source Streams (Figure 3). Mixing multiple streams typically requires that the input streams are possible to relate in time, meaning that they have to be Source Streams (Section 2.1.5) rather than Raw Streams. In Figure 3, the generated Source Stream is a mix of the three input Source Streams.

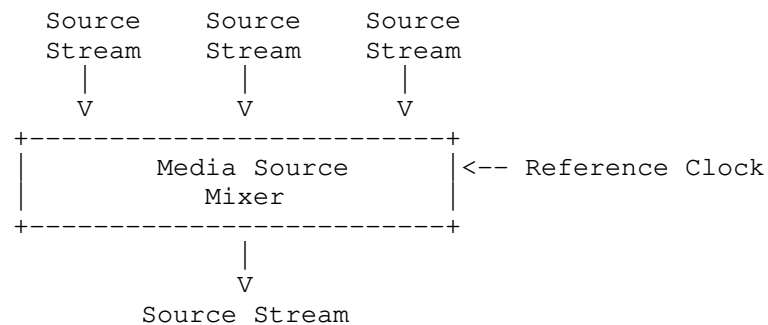


Figure 3: Conceptual Media Source in form of Audio Mixer

Another possible example of a conceptual Media Source is a video surveillance switch, where the input is multiple Source Streams from different cameras, and the output is one of those Source Streams based on some selection criteria, like a round-robin or based on some video activity measure.

2.1.5. Source Stream

A Source Stream is a stream of digital samples that has been synchronized with a reference clock and comes from particular Media Source (Section 2.1.4).

2.1.6. Media Encoder

A Media Encoder is a transform that is responsible for encoding the media data from a Source Stream (Section 2.1.5) into another representation, usually more compact, that is output as an Encoded Stream (Section 2.1.7).

The Media Encoder step commonly includes pre-encoding transformations, such as scaling, resampling etc. The Media Encoder can have a significant number of configuration options that affects the properties of the Encoded Stream. This include properties such as codec, bit-rate, start points for decoding, resolution, bandwidth or other fidelity affecting properties.

Scalable Media Encoders need special attention as they produce multiple outputs that are potentially of different types. As shown in Figure 4, a scalable Media Encoder takes one input Source Stream and encodes it into multiple output streams of two different types; at least one Encoded Stream that is independently decodable and one or more Dependent Streams (Section 2.1.8). Decoding requires at least one Encoded Stream and zero or more Dependent Streams. A Dependent Stream's dependency is one of the grouping relations this document discusses further in Section 3.7.

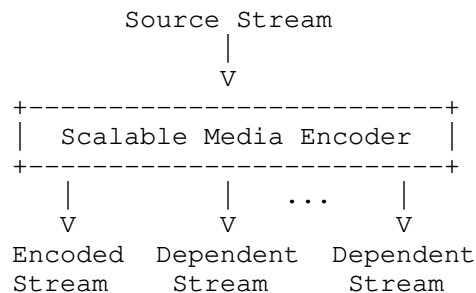


Figure 4: Scalable Media Encoder Input and Outputs

There are also other variants of encoders, like so-called Multiple Description Coding (MDC). Such Media Encoders produce multiple independent and thus individually decodable Encoded Streams. However, (logically) combining multiple of these Encoded Streams into a single Received Source Stream during decoding leads to an improvement in perceptual reproduced quality when compared to decoding a single Encoded Stream.

Creating multiple Encoded Streams from the same Source Stream, where the Encoded Streams are neither in a scalable nor in an MDC

relationship is commonly utilized in Simulcast [I-D.ietf-mmusic-sdp-simulcast] environments.

2.1.7. Encoded Stream

A stream of time synchronized encoded media that can be independently decoded.

Due to temporal dependencies, an Encoded Stream may have limitations in where decoding can be started. These entry points, for example Intra frames from a video encoder, may require identification and their generation may be event based or configured to occur periodically.

2.1.8. Dependent Stream

A stream of time synchronized encoded media fragments that are dependent on one or more Encoded Streams (Section 2.1.7) and zero or more Dependent Streams to be possible to decode.

Each Dependent Stream has a set of dependencies. These dependencies must be understood by the parties in a Multimedia Session that intend to use a Dependent Stream.

2.1.9. Media Packetizer

The transformation of taking one or more Encoded (Section 2.1.7) or Dependent Streams (Section 2.1.8) and putting their content into one or more sequences of packets, normally RTP packets, and output Source RTP Streams (Section 2.1.10). This step includes both generating RTP payloads as well as RTP packets. The Media Packetizer then selects which Synchronization source(s) (SSRC) [RFC3550] and RTP Sessions to use.

The Media Packetizer can combine multiple Encoded or Dependent Streams into one or more RTP Streams:

- o The Media Packetizer can use multiple inputs when producing a single RTP Stream. One such example is SRST packetization when using Scalable Video Coding (SVC) (Section 3.7).
- o The Media Packetizer can also produce multiple RTP Streams, for example when Encoded and/or Dependent Streams are distributed over multiple RTP Streams. One example of this is MRMT packetization when using SVC (Section 3.7).

2.1.10. RTP Stream

An RTP Stream is a stream of RTP packets containing media data, source or redundant. The RTP Stream is identified by an SSRC belonging to a particular RTP Session. The RTP Session is identified as discussed in Section 2.2.2.

A Source RTP Stream is an RTP Stream directly related to an Encoded Stream (Section 2.1.7), targeted for transport over RTP without any additional RTP-based Redundancy (Section 2.1.11) applied.

Characteristics:

- o Each RTP Stream is identified by a Synchronization source (SSRC) [RFC3550] that is carried in every RTP and RTP Control Protocol (RTCP) packet header. The SSRC is unique in a specific RTP Session context.
- o At any given point in time, a RTP Stream can have one and only one SSRC, but SSRCs for a given RTP Stream can change over time. SSRC collision and clock rate change [RFC7160] are examples of valid reasons to change SSRC for an RTP Stream. In those cases, the RTP Stream itself is not changed in any significant way, only the identifying SSRC number.
- o Each SSRC defines a unique RTP sequence numbering and timing space.
- o Several RTP Streams, each with their own SSRC, may represent a single Media Source.
- o Several RTP Streams, each with their own SSRC, can be carried in a single RTP Session.

2.1.11. RTP-based Redundancy

RTP-based Redundancy is defined here as a transformation that generates redundant or repair packets sent out as a Redundancy RTP Stream (Section 2.1.12) to mitigate network transport impairments, like packet loss and delay. Note that this excludes the type of redundancy that most suitable Media Encoders (Section 2.1.6) may add to the media format of the Encoded Stream (Section 2.1.7) that makes it cope better with inevitable RTP packet losses.

The RTP-based Redundancy exists in many flavors; they may be generating independent Repair Streams that are used in addition to the Source Stream (like RTP Retransmission (Section 3.10) and some special types of Forward Error Correction, like RTP stream

duplication (Section 3.8)), they may generate a new Source Stream by combining redundancy information with source information (Using XOR FEC (Section 3.11) as a redundancy payload (Section 3.9)), or completely replace the source information with only redundancy packets.

2.1.12. Redundancy RTP Stream

A Redundancy RTP Stream is an RTP Stream (Section 2.1.10) that contains no original source data, only redundant data, which may either be used standalone or be combined with one or more Received RTP Streams (Section 2.1.23) to produce Repaired RTP Streams (Section 2.1.26).

2.1.13. RTP-based Security

The optional RTP-based Security transformation applies security services such as authentication, integrity protection and confidentiality to an input RTP Stream, like what is specified in The Secure Real-time Transport Protocol (SRTP) [RFC3711], producing a Secured RTP Stream (Section 2.1.14). Either an RTP Stream (Section 2.1.10) or a Redundancy RTP Stream (Section 2.1.12) can be used as input to this transformation.

In SRTP and the related Secure RTCP (SRTCP), all of the above mentioned security services are optional, except for integrity protection of SRTCP, which is mandatory. Also confidentiality (encryption) is effectively optional in SRTP, since it is possible to use a NULL encryption algorithm. As described in [RFC7201], the strength of SRTP data origin authentication depends on the cryptographic transform and key management used, for example in group communication where it is sometimes possible to authenticate group membership but not the actual RTP Stream sender.

RTP-based Security and RTP-based Redundancy can be combined in a few different ways. One way is depicted in Figure 1, where an RTP Stream and its corresponding Redundancy RTP Stream are protected by separate RTP-based Security transforms. In other cases, like when a Media Translator is adding FEC in Section 3.2.1.3 of [I-D.ietf-avtcore-rtp-topologies-update], a middlebox can apply RTP-based Redundancy to an already Secured RTP Stream instead of a Source RTP Stream. One example of that is depicted in Figure 5 below.

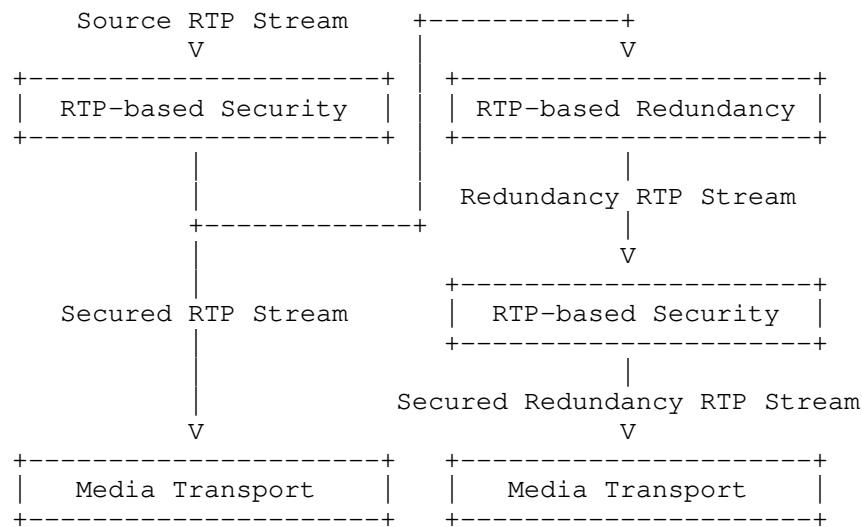


Figure 5: Adding Redundancy to a Secured RTP Stream

In this case, the Redundancy RTP Stream may already have been secured for confidentiality (encrypted) by the first RTP-based Security, and it may therefore not be necessary to apply additional confidentiality protection in the second RTP-based Security. To avoid attacks and negative impact on RTP-based Repair (Section 2.1.25) and the resulting Repaired RTP Stream (Section 2.1.26), it is however still necessary to have this second RTP-based Security apply both authentication and integrity protection to the Redundancy RTP Stream.

2.1.14. Secured RTP Stream

A Secured RTP Stream is a Source or Redundancy RTP Stream that is protected through RTP-based Security (Section 2.1.13) by one or more of the confidentiality, integrity, or authentication security services.

2.1.15. Media Transport

A Media Transport defines the transformation that the RTP Streams (Section 2.1.10) are subjected to by the end-to-end transport from one RTP sender to one specific RTP receiver (an RTP Session (Section 2.2.2) may contain multiple RTP receivers per sender). Each Media Transport is defined by a transport association that is normally identified by a 5-tuple (source address, source port, destination address, destination port, transport protocol), but a proposal exists for sending multiple transport associations on a single 5-tuple [I-D.westerlund-avtcore-transport-multiplexing].

Characteristics:

- o Media Transport transmits RTP Streams of RTP Packets from a source transport address to a destination transport address.
- o Each Media Transport contains only a single RTP Session.
- o A single RTP Session can span multiple Media Transports.

The Media Transport concept sometimes needs to be decomposed into more steps to enable discussion of what a sender emits that gets transformed by the network before it is received by the receiver. Thus we provide also this Media Transport decomposition (Figure 6).

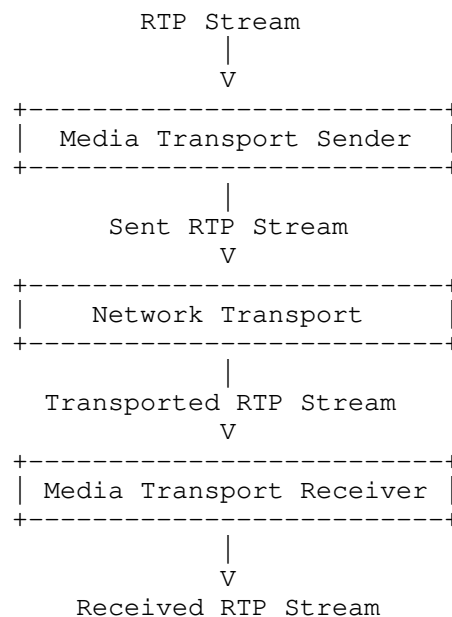


Figure 6: Decomposition of Media Transport

2.1.16. Media Transport Sender

The first transformation within the Media Transport (Section 2.1.15) is the Media Transport Sender. The sending Endpoint (Section 2.2.1) takes an RTP Stream and emits the packets onto the network using the transport association established for this Media Transport, thereby creating a Sent RTP Stream (Section 2.1.17). In the process, it transforms the RTP Stream in several ways. First, it generates the necessary protocol headers for the transport association, for example IP and UDP headers, thus forming IP/UDP/RTP packets. In addition,

the Media Transport Sender may queue, intentionally pace or otherwise affect how the packets are emitted onto the network, thereby potentially introducing delay and delay variations [RFC5481] that characterize the Sent RTP Stream.

2.1.17. Sent RTP Stream

The Sent RTP Stream is the RTP Stream as entering the first hop of the network path to its destination. The Sent RTP Stream is identified using network transport addresses, like for IP/UDP the 5-tuple (source IP address, source port, destination IP address, destination port, and protocol (UDP)).

2.1.18. Network Transport

Network Transport is the transformation that subjects the Sent RTP Stream (Section 2.1.17) to traveling from the source to the destination through the network. This transformation can result in loss of some packets, delay and delay variation on a per packet basis, packet duplication, and packet header or data corruption. This transformation produces a Transported RTP Stream (Section 2.1.19) at the exit of the network path.

2.1.19. Transported RTP Stream

The Transported RTP Stream is the RTP Stream that is emitted out of the network path at the destination, subjected to the Network Transport's transformation (Section 2.1.18).

2.1.20. Media Transport Receiver

The Media Transport Receiver is the receiver Endpoint's (Section 2.2.1) transformation of the Transported RTP Stream (Section 2.1.19) by its reception process, which results in the Received RTP Stream (Section 2.1.23). This transformation includes transport checksums being verified. Sensible system designs typically either discard packets with mis-matching checksums, or pass them on while somehow marking them in the resulting Received RTP Stream so to alert subsequent transformations about the possible corrupt state. In this context it is worth noting that there is typically some probability for corrupt packets to pass through undetected (with a seemingly correct checksum). Other transformations can compensate for delay variations in receiving a packet on the network interface and providing it to the application (de-jitter buffer).

2.1.21. Received Secured RTP Stream

This is the Secured RTP Stream (Section 2.1.14) resulting from the Media Transport (Section 2.1.15) aggregate transformation.

2.1.22. RTP-based Validation

RTP-based Validation is the reverse transformation of RTP-based Security (Section 2.1.13). If this transformation fails, the result is either not usable and must be discarded, or may be usable but cannot be trusted. If the transformation succeeds, the result can be a Received RTP Stream (Section 2.1.23) or a Received Redundancy RTP Stream (Section 2.1.24), depending on what was input to the corresponding RTP-based Security transformation, but can also be a Received Secured RTP Stream (Section 2.1.21) in case several RTP-based Security transformations were applied.

2.1.23. Received RTP Stream

The Received RTP Stream is the RTP Stream (Section 2.1.10) resulting from the Media Transport's aggregate transformation (Section 2.1.15), i.e. subjected to packet loss, packet corruption, packet duplication, delay, and delay variation from sender to receiver.

2.1.24. Received Redundancy RTP Stream

The Received Redundancy RTP Stream is the Redundancy RTP Stream (Section 2.1.12) resulting from the Media Transport transformation, i.e. subjected to packet loss, packet corruption, delay, and delay variation from sender to receiver.

2.1.25. RTP-based Repair

RTP-based Repair is a Transformation that takes as input zero or more Received RTP Streams (Section 2.1.23) and one or more Received Redundancy RTP Streams (Section 2.1.24), and produces one or more Repaired RTP Streams (Section 2.1.26) that are as close to the corresponding sent Source RTP Streams (Section 2.1.10) as possible, using different RTP-based repair methods, for example the ones referred in RTP-based Redundancy (Section 2.1.11).

2.1.26. Repaired RTP Stream

A Repaired RTP Stream is a Received RTP Stream (Section 2.1.23) for which Received Redundancy RTP Stream (Section 2.1.24) information has been used to try to recover the Source RTP Stream (Section 2.1.10) as it was before Media Transport (Section 2.1.15).

2.1.27. Media Depacketizer

A Media Depacketizer takes one or more RTP Streams (Section 2.1.10), depacketizes them, and attempts to reconstitute the Encoded Streams (Section 2.1.7) or Dependent Streams (Section 2.1.8) present in those RTP Streams.

In practical implementations, the Media Depacketizer and the Media Decoder may be tightly coupled and share information to improve or optimize the overall decoding and error concealment process. It is, however, not expected that there would be any benefit in defining a taxonomy for those detailed (and likely very implementation-dependent) steps.

2.1.28. Received Encoded Stream

The Received Encoded Stream is the received version of an Encoded Stream (Section 2.1.7).

2.1.29. Media Decoder

A Media Decoder is a transformation that is responsible for decoding Encoded Streams (Section 2.1.7) and any Dependent Streams (Section 2.1.8) into a Source Stream (Section 2.1.5).

In practical implementations, the Media Decoder and the Media Depacketizer may be tightly coupled and share information to improve or optimize the overall decoding process in various ways. It is however not expected that there would be any benefit in defining a taxonomy for those detailed (and likely very implementation-dependent) steps.

A Media Decoder has to deal with any errors in the Encoded Streams that resulted from corruption or failure to repair packet losses. Therefore, it commonly is robust to error and losses, and includes concealment methods.

2.1.30. Received Source Stream

The Received Source Stream is the received version of a Source Stream (Section 2.1.5).

2.1.31. Media Sink

The Media Sink receives a Source Stream (Section 2.1.5) that contains, usually periodically, sampled media data together with associated synchronization information. Depending on application, this Source Stream then needs to be transformed into a Raw Stream

(Section 2.1.3) that is conveyed to the Media Render (Section 2.1.33), synchronized with the output from other Media Sinks. The Media Sink may also be connected with a Media Source (Section 2.1.4) and be used as part of a conceptual Media Source.

The Media Sink can further transform the Source Stream into a representation that is suitable for rendering on the Media Render as defined by the application or system-wide configuration. This include sample scaling, level adjustments etc.

2.1.32. Received Raw Stream

The Received Raw Stream is the received version of a Raw Stream (Section 2.1.3).

2.1.33. Media Render

A Media Render takes a Raw Stream (Section 2.1.3) and converts it into Physical Stimulus (Section 2.1.1) that a human user can perceive. Examples of such devices are screens, and D/A converters connected to amplifiers and loudspeakers.

An Endpoint can potentially have multiple Media Renders for each media type.

2.2. Communication Entities

This section contains concepts for entities involved in the communication.

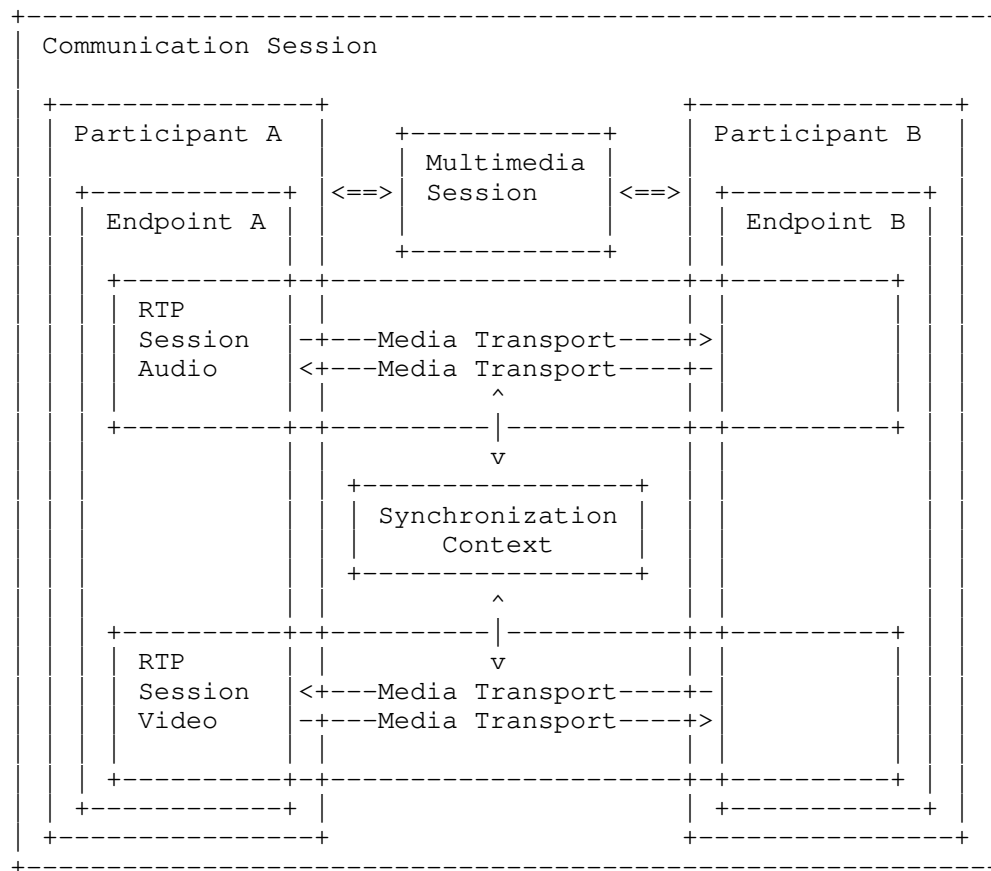


Figure 7: Example Point to Point Communication Session with two RTP Sessions

Figure 7 shows a high-level example representation of a very basic point-to-point Communication Session between Participants A and B. It uses two different audio and video RTP Sessions between A's and B's Endpoints, where each RTP Session is a group communications channel that can potentially carry a number of RTP Streams. It is using separate Media Transports for those RTP Sessions. The Multimedia Session shared by the Participants can, for example, be established using SIP (i.e., there is a SIP Dialog between A and B). The terms used in Figure 7 are further elaborated in the sub-sections below.

2.2.1. Endpoint

An Endpoint is a single addressable entity sending or receiving RTP packets. It may be decomposed into several functional blocks, but as long as it behaves as a single RTP stack entity it is classified as a single "Endpoint".

Characteristics:

- o Endpoints can be identified in several different ways. While RTCP Canonical Names (CNAMEs) [RFC3550] provide a globally unique and stable identification mechanism for the duration of the Communication Session (see Section 2.2.5), their validity applies exclusively within a Synchronization Context (Section 3.1). Thus one Endpoint can handle multiple CNAMEs, each of which can be shared among a set of Endpoints belonging to the same Participant (Section 2.2.3). Therefore, mechanisms outside the scope of RTP, such as application defined mechanisms, must be used to provide Endpoint identification when outside this Synchronization Context.
- o An Endpoint can be associated with at most one Participant (Section 2.2.3) at any single point in time.
- o In some contexts, an Endpoint would typically correspond to a single "host", for example a computer using a single network interface and being used by a single human user. In other contexts, a single "host" can serve multiple Participants, in which case each Participant's Endpoint may share properties, for example the IP address part of a transport address.

2.2.2. RTP Session

An RTP Session is an association among a group of Participants communicating with RTP. It is a group communications channel which can potentially carry a number of RTP Streams. Within an RTP Session, every Participant can find meta-data and control information (over RTCP) about all the RTP Streams in the RTP Session. The bandwidth of the RTCP control channel is shared between all Participants within an RTP Session.

Characteristics:

- o An RTP Session can carry one or more RTP Streams.
- o An RTP Session shares a single SSRC space as defined in RFC3550 [RFC3550]. That is, the Endpoints participating in an RTP Session can see an SSRC identifier transmitted by any of the other Endpoints. An Endpoint can receive an SSRC either as SSRC or as a

Contributing source (CSRC) in RTP and RTCP packets, as defined by the Endpoints' network interconnection topology.

- o An RTP Session uses at least two Media Transports (Section 2.1.15), one for sending and one for receiving. Commonly, the receiving Media Transport is the reverse direction of the Media Transport used for sending. An RTP Session may use many Media Transports and these define the session's network interconnection topology.
- o A single Media Transport always carries a single RTP Session.
- o Multiple RTP Sessions can be conceptually related, for example originating from or targeted for the same Participant (Section 2.2.3) or Endpoint (Section 2.2.1), or by containing RTP Streams that are somehow related (Section 3).

2.2.3. Participant

A Participant is an entity reachable by a single signaling address, and is thus related more to the signaling context than to the media context.

Characteristics:

- o A single signaling-addressable entity, using an application-specific signaling address space, for example a SIP URI.
- o A Participant can participate in several Multimedia Sessions (Section 2.2.4).
- o A Participant can be comprised of several associated Endpoints (Section 2.2.1).

2.2.4. Multimedia Session

A Multimedia Session is an association among a group of Participants (Section 2.2.3) engaged in the communication via one or more RTP Sessions (Section 2.2.2). It defines logical relationships among Media Sources (Section 2.1.4) that appear in multiple RTP Sessions.

Characteristics:

- o A Multimedia Session can be composed of several RTP Sessions with potentially multiple RTP Streams per RTP Session.
- o Each Participant in a Multimedia Session can have a multitude of Media Captures and Media Rendering devices.

- o A single Multimedia Session can contain media from one or more Synchronization Contexts (Section 3.1). An example of that is a Multimedia Session containing one set of audio and video for communication purposes belonging to one Synchronization Context, and another set of audio and video for presentation purposes (like playing a video file) with a separate Synchronization Context that has no strong timing relationship and need not be strictly synchronized with the audio and video used for communication.

2.2.5. Communication Session

A Communication Session is an association among two or more Participants (Section 2.2.3) communicating with each other via one or more Multimedia Sessions (Section 2.2.4).

Characteristics:

- o Each Participant in a Communication Session is identified via an application-specific signaling address.
- o A Communication Session is composed of Participants that share at least one Multimedia Session, involving one or more parallel RTP Sessions with potentially multiple RTP Streams per RTP Session.

For example, in a full mesh communication, the Communication Session consists of a set of separate Multimedia Sessions between each pair of Participants. Another example is a centralized conference, where the Communication Session consists of a set of Multimedia Sessions between each Participant and the conference handler.

3. Concepts of Inter-Relations

This section uses the concepts from previous sections, and looks at different types of relationships among them. These relationships occur at different abstraction levels and for different purposes, but the reason for the needed relationship at a certain step in the media handling chain may exist at another step. For example, the use of Simulcast (Section 3.6) implies a need to determine relations at RTP Stream level, but the underlying reason is that multiple Media Encoders use the same Media Source, i.e. to be able to identify a common Media Source.

3.1. Synchronization Context

A Synchronization Context defines a requirement on a strong timing relationship between the Media Sources, typically requiring alignment of clock sources. Such a relationship can be identified in multiple ways as listed below. A single Media Source can only belong to a

single Synchronization Context, since it is assumed that a single Media Source can only have a single media clock and requiring alignment to several Synchronization Contexts (and thus reference clocks) will effectively merge those into a single Synchronization Context.

3.1.1. RTCP CNAME

RFC3550 [RFC3550] describes Inter-media synchronization between RTP Sessions based on RTCP CNAME, RTP and Network Time Protocol (NTP) [RFC5905] formatted timestamps of a reference clock. As indicated in [RFC7273], despite using NTP format timestamps, it is not required that the clock be synchronized to an NTP source.

3.1.2. Clock Source Signaling

[RFC7273] provides a mechanism to signal the clock source in Session Description Protocol (SDP) [RFC4566] both for the reference clock as well as the media clock, thus allowing a Synchronization Context to be defined beyond the one defined by the usage of CNAME source descriptions.

3.1.3. Implicitly via RtcMediaStream

WebRTC defines "RtcMediaStream" with one or more "RtcMediaStreamTracks". All tracks in a "RtcMediaStream" are intended to be synchronized when rendered, implying that they must be generated such that synchronization is possible.

3.1.4. Explicitly via SDP Mechanisms

The SDP Grouping Framework [RFC5888] defines an m= line (Section 4.2) grouping mechanism called "Lip Synchronization" (with LS identification-tag) for establishing the synchronization requirement across m= lines when they map to individual sources.

Source-Specific Media Attributes in SDP [RFC5576] extends the above mechanism when multiple Media Sources are described by a single m= line.

3.2. Endpoint

Some applications requires knowledge of what Media Sources originate from a particular Endpoint (Section 2.2.1). This can include such decisions as packet routing between parts of the topology, knowing the Endpoint origin of the RTP Streams.

In RTP, this identification has been overloaded with the Synchronization Context (Section 3.1) through the usage of the RTCP source description CNAME (Section 3.1.1). This works for some usages, but in others it breaks down. For example, if an Endpoint has two sets of Media Sources that have different Synchronization Contexts, like the audio and video of the human Participant as well as a set of Media Sources of audio and video for a shared movie, CNAME would not be an appropriate identification for that Endpoint. Therefore, an Endpoint may have multiple CNAMEs. The CNAMEs or the Media Sources themselves can be related to the Endpoint.

3.3. Participant

In communication scenarios, it is commonly needed to know which Media Sources originate from which Participant (Section 2.2.3). One reason is, for example, to enable the application to display Participant Identity information correctly associated with the Media Sources. This association is handled through the signaling solution to point at a specific Multimedia Session where the Media Sources may be explicitly or implicitly tied to a particular Endpoint.

Participant information becomes more problematic due to Media Sources that are generated through mixing or other conceptual processing of Raw Streams or Source Streams that originate from different Participants. This type of Media Sources can thus have a dynamically varying set of origins and Participants. RTP contains the concept of CSRC that carry information about the previous step origin of the included media content on RTP level.

3.4. RtcMediaStream

An RtcMediaStream in WebRTC is an explicit grouping of a set of Media Sources (RtcMediaStreamTracks) that share a common identifier and a single Synchronization Context (Section 3.1).

3.5. Multi-Channel Audio

There exist a number of RTP payload formats that can carry multi-channel audio, despite the codec being a single-channel (mono) encoder. Multi-channel audio can be viewed as multiple Media Sources sharing a common Synchronization Context. These are independently encoded by a Media Encoder and the different Encoded Streams are packetized together in a time synchronized way into a single Source RTP Stream, using the used codec's RTP Payload format. Examples of codecs that support multi-channel audio are PCMA and PCMU [RFC3551], AMR [RFC4867], and G.719 [RFC5404].

3.6. Simulcast

A Media Source represented as multiple independent Encoded Streams constitutes a Simulcast [I-D.ietf-mmusic-sdp-simulcast] or MDC of that Media Source. Figure 8 shows an example of a Media Source that is encoded into three separate Simulcast streams, that are in turn sent on the same Media Transport flow. When using Simulcast, the RTP Streams may be sharing RTP Session and Media Transport, or be separated on different RTP Sessions and Media Transports, or any combination of these two. One major reason to use separate Media Transports is to make use of different Quality of Service for the different Source RTP Streams. Some considerations on separating related RTP Streams are discussed in Section 3.12.

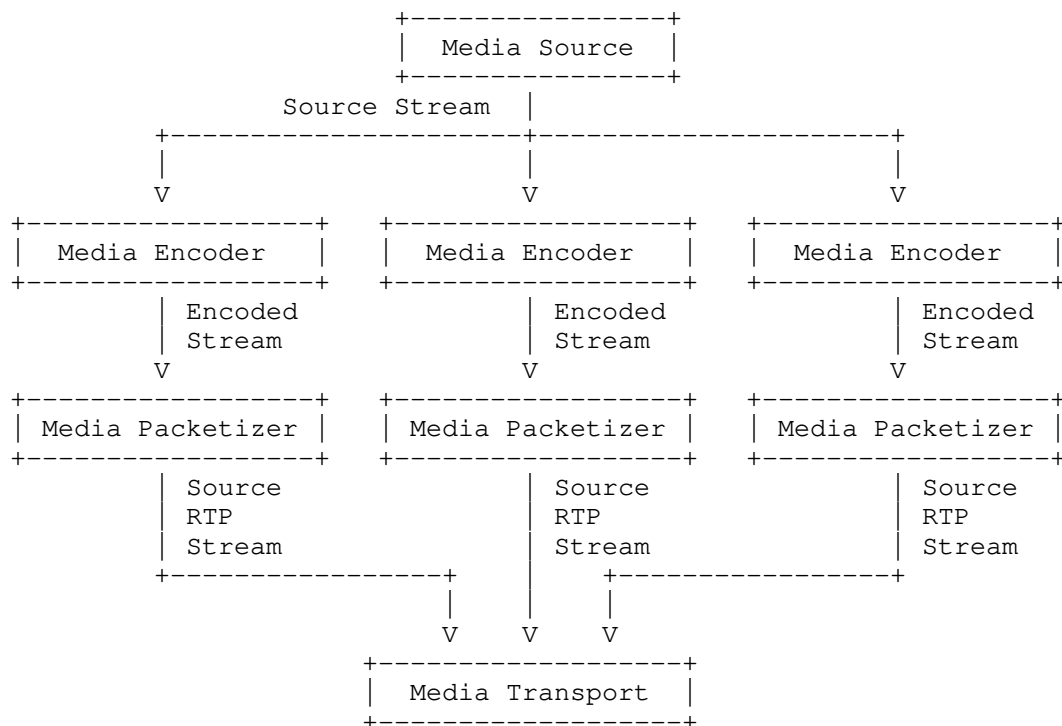


Figure 8: Example of Media Source Simulcast

The Simulcast relation between the RTP Streams is the common Media Source. In addition, to be able to identify the common Media Source, a receiver of the RTP Stream may need to know which configuration or encoding goals that lay behind the produced Encoded Stream and its properties. This enables selection of the stream that is most useful in the application at that moment.

3.7. Layered Multi-Stream

Layered Multi-Stream (LMS) is a mechanism by which different portions of a layered or scalable encoding of a Source Stream are sent using separate RTP Streams (sometimes in separate RTP Sessions). LMSs are useful for receiver control of layered media.

A Media Source represented as an Encoded Stream and multiple Dependent Streams constitutes a Media Source that has layered dependencies. Figure 9 represents an example of a Media Source that is encoded into three dependent layers, where two layers are sent on the same Media Transport using different RTP Streams, i.e. SSRCs, and the third layer is sent on a separate Media Transport.

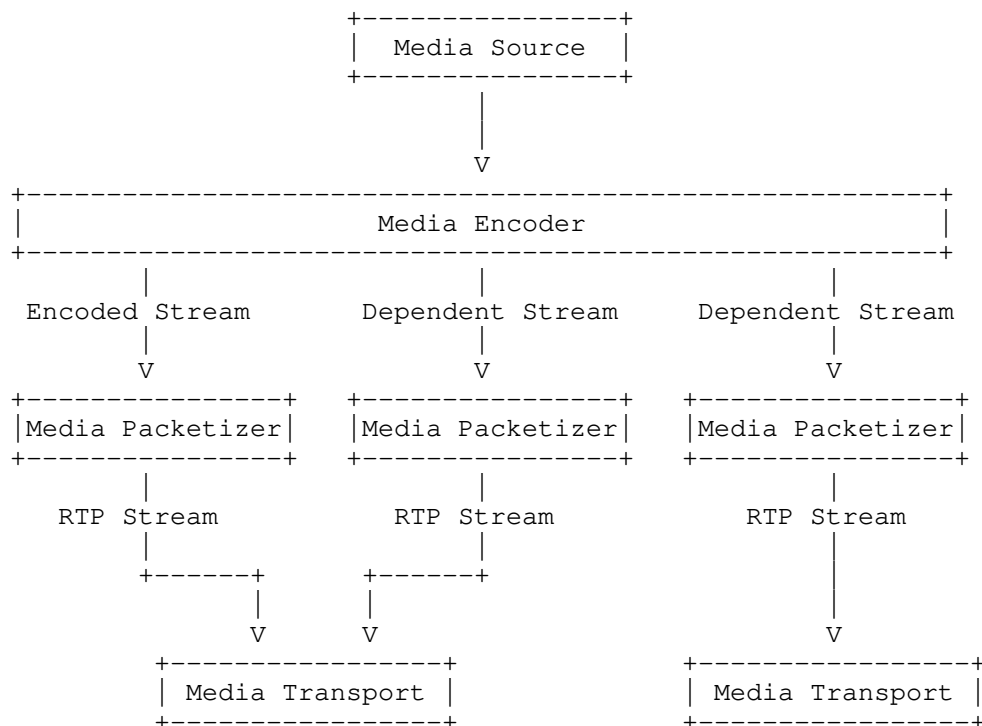


Figure 9: Example of Media Source Layered Dependency

It is sometimes useful to make a distinction between using a single Media Transport or multiple separate Media Transports when (in both cases) using multiple RTP Streams to carry Encoded Streams and Dependent Streams for a Media Source. Therefore, the following new terminology is defined here:

SRST: Single RTP Stream on a Single Media Transport

MRST: Multiple RTP Streams on a Single Media Transport

MRMT: Multiple RTP Streams on Multiple Media Transports

MRST and MRMT relations needs to identify the common Media Encoder origin for the Encoded and Dependent Streams. When using different RTP Sessions (MRMT), a single RTP Stream per Media Encoder, and a single Media Source in each RTP Session, common SSRC and CNAMEs can be used to identify the common Media Source. When multiple RTP Streams are sent from one Media Encoder in the same RTP Session (MRST), then CNAME is the only currently specified RTP identifier that can be used. In cases where multiple Media Encoders use multiple Media Sources sharing Synchronization Context, and thus having a common CNAME, additional heuristics or identification need to be applied to create the MRST or MRMT relationships between the RTP Streams.

3.8. RTP Stream Duplication

RTP Stream Duplication [RFC7198], using the same or different Media Transports, and optionally also delaying the duplicate [RFC7197], offers a simple way to protect media flows from packet loss in some cases (see Figure 10). This is a specific type of redundancy. All but one Source RTP Stream (Section 2.1.10) are effectively Redundancy RTP Streams (Section 2.1.12), but since both Source and Redundant RTP Streams are the same, it does not matter which one is which. This can also be seen as a specific type of Simulcast (Section 3.6) that transmits the same Encoded Stream (Section 2.1.7) multiple times.

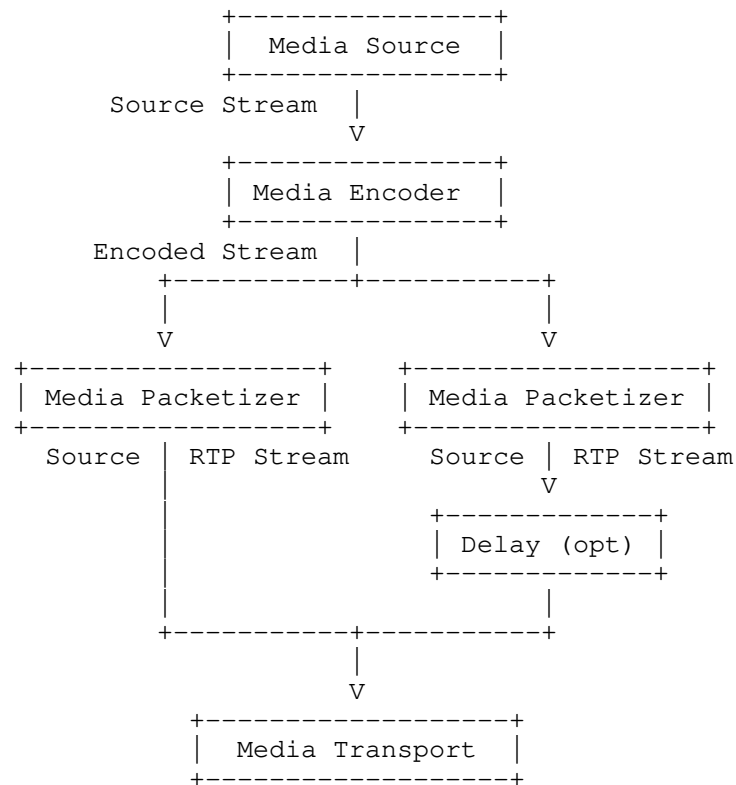


Figure 10: Example of RTP Stream Duplication

3.9. Redundancy Format

The RTP Payload for Redundant Audio Data [RFC2198] defines a transport for redundant audio data together with primary data in the same RTP payload. The redundant data can be a time delayed version of the primary or another time delayed Encoded Stream using a different Media Encoder to encode the same Media Source as the primary, as depicted in Figure 11.

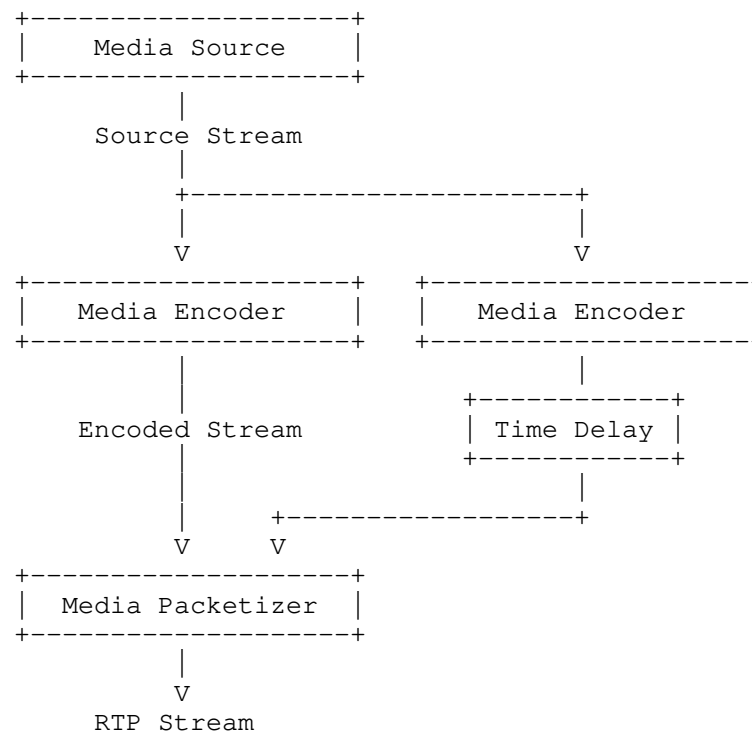


Figure 11: Concept for usage of Audio Redundancy with different Media Encoders

The Redundancy format is thus providing the necessary meta information to correctly relate different parts of the same Encoded Stream. The case depicted above (Figure 11) relates the Received Source Stream fragments coming out of different Media Decoders, to be able to combine them together into a less erroneous Source Stream.

3.10. RTP Retransmission

Figure 12 shows an example where a Media Source's Source RTP Stream is protected by a retransmission (RTX) flow [RFC4588]. In this example the Source RTP Stream and the Redundancy RTP Stream share the same Media Transport.

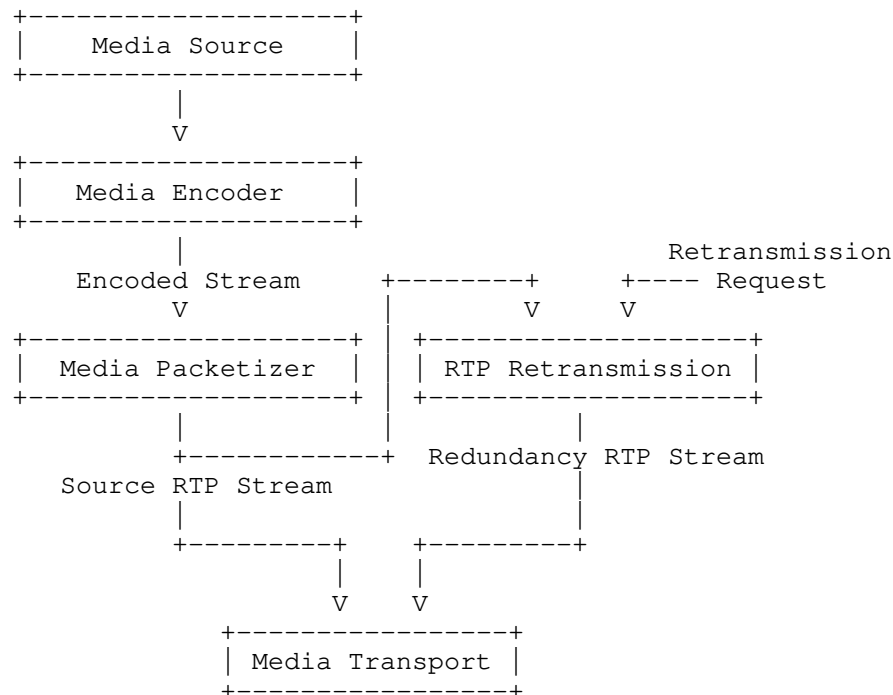


Figure 12: Example of Media Source Retransmission Flows

The RTP Retransmission example (Figure 12) illustrates that this mechanism works purely on the Source RTP Stream. The RTP Retransmission transform buffers the sent Source RTP Stream and, upon request, emits a retransmitted packet with an extra payload header as a Redundancy RTP Stream. The RTP Retransmission mechanism [RFC4588] is specified such that there is a one to one relation between the Source RTP Stream and the Redundancy RTP Stream. Therefore, a Redundancy RTP Stream needs to be associated with its Source RTP Stream. This is done based on CNAME selectors and heuristics to match requested packets for a given Source RTP Stream with the original sequence number in the payload of any new Redundancy RTP Stream using the RTX payload format. In cases where the Redundancy RTP Stream is sent in a different RTP Session than the Source RTP Stream, the RTP Session relation is signaled by using the SDP Media Grouping's [RFC5888] Flow Identification (FID identification-tag) semantics.

3.11. Forward Error Correction

Figure 13 shows an example where two Media Sources' Source RTP Streams are protected by Forward Error Correction (FEC). Source RTP Stream A has a RTP-based Redundancy transformation in FEC Encoder 1. This produces a Redundancy RTP Stream 1, that is only related to Source RTP Stream A. The FEC Encoder 2, however, takes two Source RTP Streams (A and B) and produces a Redundancy RTP Stream 2 that protects them jointly, i.e. Redundancy RTP Stream 2 relates to two Source RTP Streams (a FEC group). FEC decoding, when needed due to packet loss or packet corruption at the receiver, requires knowledge about which Source RTP Streams that the FEC encoding was based on.

In Figure 13 all RTP Streams are sent on the same Media Transport. This is however not the only possible choice. Numerous combinations exist for spreading these RTP Streams over different Media Transports to achieve the communication application's goal.

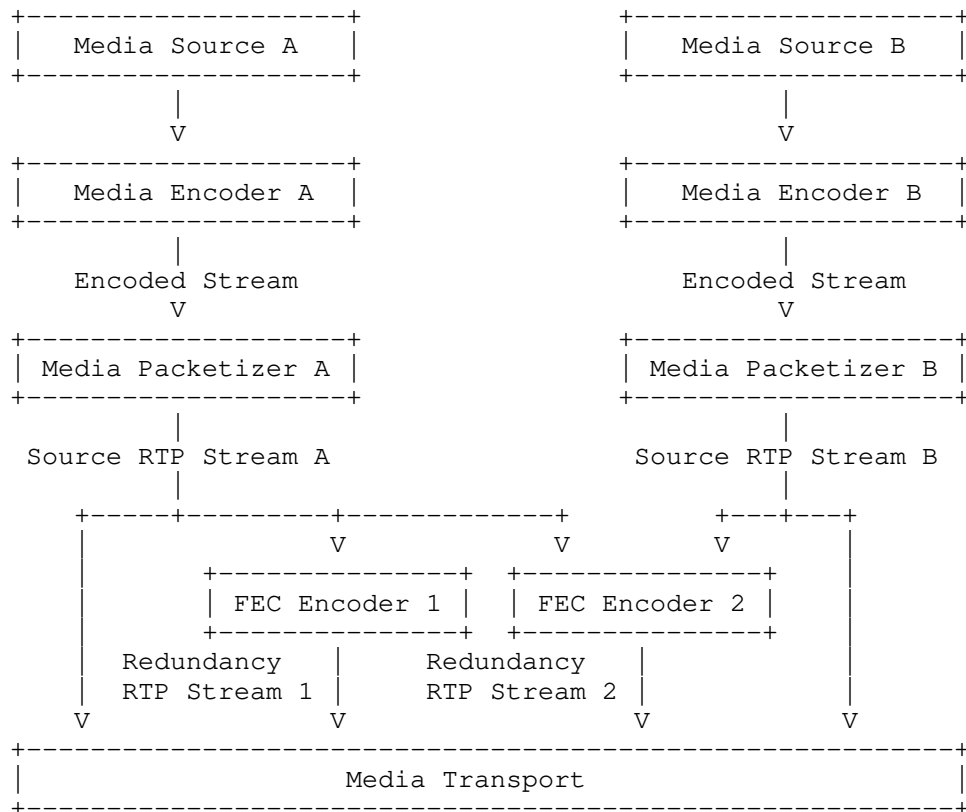


Figure 13: Example of FEC Redundancy RTP Streams

As FEC Encoding exists in various forms, the methods for relating FEC Redundancy RTP Streams with its source information in Source RTP Streams are many. The XOR based RTP FEC Payload format [RFC5109] is defined in such a way that a Redundancy RTP Stream has a one to one relation with a Source RTP Stream. In fact, the RFC requires the Redundancy RTP Stream to use the same SSRC as the Source RTP Stream. This requires the use of either a separate RTP Session, or the Redundancy RTP Payload format [RFC2198]. The underlying relation requirement for this FEC format and a particular Redundancy RTP Stream is to know the related Source RTP Stream, including its SSRC.

3.12. RTP Stream Separation

RTP Streams can be separated exclusively based on their SSRCS, at the RTP Session level, or at the Multi-Media Session level.

When the RTP Streams that have a relationship are all sent in the same RTP Session and are uniquely identified based on their SSRC only, it is termed an SSRC-Only Based Separation. Such streams can be related via RTCP CNAME to identify that the streams belong to the same Endpoint. SSRC-based approaches [RFC5576], when used, can explicitly relate various such RTP Streams.

On the other hand, when RTP Streams that are related are sent in the context of different RTP Sessions to achieve separation, it is known as RTP Session-based separation. This is commonly used when the different RTP Streams are intended for different Media Transports.

Several mechanisms that use RTP Session-based separation rely on it to enable an implicit grouping mechanism expressing the relationship. The solutions have been based on using the same SSRC value in the different RTP Sessions to implicitly indicate their relation. That way, no explicit RTP level mechanism has been needed, only signaling level relations have been established using semantics from Grouping of Media lines framework [RFC5888]. Examples of this are RTP Retransmission [RFC4588], SVC Multi-Session Transmission [RFC6190] and XOR Based FEC [RFC5109]. RTCP CNAME explicitly relates RTP Streams across different RTP Sessions, as explained in the previous section. Such a relationship can be used to perform inter-media synchronization.

RTP Streams that are related and need to be associated can be part of different Multimedia Sessions, rather than just different RTP Sessions within the same Multimedia Session context. This puts further demand on the scope of the mechanism(s) and its handling of identifiers used for expressing the relationships.

3.13. Multiple RTP Sessions over one Media Transport

[I-D.westerlund-avtcore-transport-multiplexing] describes a mechanism that allows several RTP Sessions to be carried over a single underlying Media Transport. The main reasons for doing this are related to the impact of using one or more Media Transports (using a common network path or potentially have different ones). The fewer Media Transports used, the less need for NAT/FW traversal resources and smaller number of flow based Quality of Service (QoS).

However, Multiple RTP Sessions over one Media Transport imply that a single Media Transport 5-tuple is not sufficient to express in which RTP Session context a particular RTP Stream exists. Complexities in the relationship between Media Transports and RTP Session already exist as one RTP Session contains multiple Media Transports, e.g. even a Peer-to-Peer RTP Session with RTP/RTCP Multiplexing requires two Media Transports, one in each direction. The relationship between Media Transports and RTP Sessions as well as additional levels of identifiers need to be considered in both signaling design and when defining terminology.

4. Mapping from Existing Terms

This section describes a selected set of terms from some relevant IETF RFC and Internet Drafts (at the time of writing), using the concepts from previous sections.

4.1. Telepresence Terms

The terms in this sub-section are used in the context of CLUE [I-D.ietf-clue-framework]. Note that some terms listed in this sub-section use the same names as terms defined elsewhere in this document. Unless explicitly stated (as "RTP Taxonomy") and in this sub-section, they are to be read as references to the CLUE-specific term within this sub-section.

4.1.1. Audio Capture

Defined in CLUE as a Media Capture (Section 4.1.7) for audio.
Describes an audio Media Source (Section 2.1.4).

4.1.2. Capture Device

Defined in CLUE as a device that converts physical input into an electrical signal. Identifies a physical entity performing an RTP Taxonomy Media Capture (Section 2.1.2) transformation.

4.1.3. Capture Encoding

Defined in CLUE as a specific encoding (Section 4.1.6) of a Media Capture (Section 4.1.7). Describes an Encoded Stream (Section 2.1.7) related to CLUE specific semantic information.

4.1.4. Capture Scene

Defined in CLUE as a structure representing a spatial region captured by one or more Capture Devices (Section 4.1.2), each capturing media representing a portion of the region. Describes a set of spatially related Media Sources (Section 2.1.4).

4.1.5. Endpoint

Defined in CLUE as a CLUE-capable device which is the logical point of final termination through receiving, decoding and rendering and/or initiation through capturing, encoding, and sending of media streams (Section 4.1.10). CLUE further defines it to consist of one or more physical devices with source and sink media streams, and exactly one [RFC4353] Participant. Describes exactly one Participant (Section 2.2.3) and one or more RTP Taxonomy Endpoints (Section 2.2.1).

4.1.6. Individual Encoding

Defined in CLUE as a set of parameters representing a way to encode a Media Capture (Section 4.1.7) to become a Capture Encoding (Section 4.1.3). Describes the configuration information needed to perform a Media Encoder (Section 2.1.6) transformation.

4.1.7. Media Capture

Defined in CLUE as a source of media, such as from one or more Capture Devices (Section 4.1.2) or constructed from other media streams (Section 4.1.10). Describes either an RTP Taxonomy Media Capture (Section 2.1.2) or a Media Source (Section 2.1.4), depending on in which context the term is used.

4.1.8. Media Consumer

Defined in CLUE as a CLUE-capable device that intends to receive Capture Encodings (Section 4.1.3). Describes the media receiving part of an RTP Taxonomy Endpoint (Section 2.2.1).

4.1.9. Media Provider

Defined in CLUE as a CLUE-capable device that intends to send Capture Encodings (Section 4.1.3). Describes the media sending part of an RTP Taxonomy Endpoint (Section 2.2.1).

4.1.10. Stream

Defined in CLUE as a Capture Encoding (Section 4.1.3) sent from a Media Provider (Section 4.1.9) to a Media Consumer (Section 4.1.8) via RTP. Describes an RTP Stream (Section 2.1.10).

4.1.11. Video Capture

Defined in CLUE as a Media Capture (Section 4.1.7) for video. Describes a video Media Source (Section 2.1.4).

4.2. Media Description

A single Session Description Protocol (SDP) [RFC4566] media description (or media block; an m-line and all subsequent lines until the next m-line or the end of the SDP) describes part of the necessary configuration and identification information needed for a Media Encoder transformation, as well as the necessary configuration and identification information for the Media Decoder to be able to correctly interpret a received RTP Stream.

A Media Description typically relates to a single Media Source. This is for example an explicit restriction in WebRTC. However, nothing prevents that the same Media Description (and same RTP Session) is re-used for multiple Media Sources [I-D.ietf-avtcore-rtp-multi-stream]. It can thus describe properties of one or more RTP Streams, and can also describe properties valid for an entire RTP Session (via [RFC5576] mechanisms, for example).

4.3. Media Stream

RTP [RFC3550] uses media stream, audio stream, video stream, and stream of (RTP) packets interchangeably, which are all RTP Streams.

4.4. Multimedia Conference

A Multimedia Conference is a Communication Session (Section 2.2.5) between two or more Participants (Section 2.2.3), along with the software they are using to communicate.

4.5. Multimedia Session

SDP [RFC4566] defines a Multimedia Session as a set of multimedia senders and receivers and the data streams flowing from senders to receivers, which would correspond to a set of Endpoints and the RTP Streams that flow between them. In this document, Multimedia Session (Section 2.2.4) also assumes those Endpoints belong to a set of Participants that are engaged in communication via a set of related RTP Streams.

RTP [RFC3550] defines a Multimedia Session as a set of concurrent RTP Sessions among a common group of Participants. For example, a video conference may contain an audio RTP Session and a video RTP Session. This would correspond to a group of Participants (each using one or more Endpoints) sharing a set of concurrent RTP Sessions. In this document, Multimedia Session also defines those RTP Sessions to have some relation and be part of a communication among the Participants.

4.6. Multipoint Control Unit (MCU)

This term is commonly used to describe the central node in any type of star topology [I-D.ietf-avtcore-rtp-topologies-update] conference. It describes a device that includes one Participant (Section 2.2.3) (usually corresponding to a so-called conference focus) and one or more related Endpoints (Section 2.2.1) (sometimes one or more per conference Participant).

4.7. Multi-Session Transmission (MST)

One of two transmission modes defined in H.264 based SVC [RFC6190], the other mode being SST (Section 4.13). In Multi-Session Transmission (MST), the SVC Media Encoder sends Encoded Streams and Dependent Streams distributed across two or more RTP Streams in one or more RTP Sessions. The term "MST" is ambiguous in RFC 6190, especially since the name indicates the use of multiple "sessions", while MST type packetization is in fact required whenever two or more RTP Streams are used for the Encoded and Dependent Streams, regardless if those are sent in one or more RTP Sessions. Corresponds either to MRST or MRMT (Section 3.7) stream relations defined in this document. The SVC RTP Payload RFC [RFC6190] is not particularly explicit about how the common Media Encoder (Section 2.1.6) relation between Encoded Streams (Section 2.1.7) and Dependent Streams (Section 2.1.8) is to be implemented.

4.8. Recording Device

WebRTC specifications use this term to refer to locally available entities performing a Media Capture (Section 2.1.2) transformation.

4.9. RtcMediaStream

A WebRTC RtcMediaStream is a set of Media Sources (Section 2.1.4) sharing the same Synchronization Context (Section 3.1).

4.10. RtcMediaStreamTrack

A WebRTC RtcMediaStreamTrack is a Media Source (Section 2.1.4).

4.11. RTP Sender

RTP [RFC3550] uses this term, which can be seen as the RTP protocol part of a Media Packetizer (Section 2.1.9).

4.12. RTP Session

Within the context of SDP, a single m= line can map to a single RTP Session (Section 2.2.2) or multiple m= lines can map to a single RTP Session. The latter is enabled via multiplexing schemes such as BUNDLE [I-D.ietf-mmusic-sdp-bundle-negotiation], for example, which allows mapping of multiple m= lines to a single RTP Session.

4.13. Single Session Transmission (SST)

One of two transmission modes defined in H.264 based SVC [RFC6190], the other mode being MST (Section 4.7). In Single Session Transmission (SST), the SVC Media Encoder sends Encoded Streams (Section 2.1.7) and Dependent Streams (Section 2.1.8) combined into a single RTP Stream (Section 2.1.10) in a single RTP Session (Section 2.2.2), using the SVC RTP Payload format. The term "SST" is ambiguous in RFC 6190, in that it sometimes refers to the use of a single RTP Stream, like in sections relating to packetization, and sometimes appears to refer to use of a single RTP Session, like in the context of discussing SDP. Closely corresponds to SRST (Section 3.7) defined in this document.

4.14. SSRC

RTP [RFC3550] defines this as "the source of a stream of RTP packets", which indicates that an SSRC is not only a unique identifier for the Encoded Stream (Section 2.1.7) carried in those packets, but is also effectively used as a term to denote a Media Packetizer (Section 2.1.9). In [RFC3550], it is stated that "a

synchronization source may change its data format, e.g., audio encoding, over time". The related Encoded Stream data format in an RTP Stream (Section 2.1.10) is identified by the RTP Payload Type. Changing data format for an Encoded Stream effectively also changes what Media Encoder (Section 2.1.6) that is used for the Encoded Stream. No ambiguity is introduced to SSRC as Encoded Stream identifier by allowing RTP Payload Type changes, as long as only a single RTP Payload Type is valid for any given RTP Time Stamp. This is aligned with and further described by Section 5.2 of [RFC3550].

5. Security Considerations

The purpose of this document is to make clarifications and reduce the confusion prevalent in RTP taxonomy because of inconsistent usage by multiple technologies and protocols making use of the RTP protocol. It does not introduce any new security considerations beyond those already well documented in the RTP protocol [RFC3550] and each of the many respective specifications of the various protocols making use of it.

Having a well-defined common terminology and understanding of the complexities of the RTP architecture will help lead us to better standards, avoiding security problems.

6. Acknowledgement

This document has many concepts borrowed from several documents such as WebRTC [I-D.ietf-rtcweb-overview], CLUE [I-D.ietf-clue-framework], and Multiplexing Architecture [I-D.westerlund-avtcore-transport-multiplexing]. The authors would like to thank all the authors of each of those documents.

The authors would also like to acknowledge the insights, guidance and contributions of Magnus Westerlund, Roni Even, Paul Kyzivat, Colin Perkins, Keith Drage, Harald Alvestrand, Alex Eleftheriadis, Mo Zanaty, Stephan Wenger, and Bernard Aboba.

7. Contributors

Magnus Westerlund has contributed the concept model for the media chain using transformations and streams model, including rewriting pre-existing concepts into this model and adding missing concepts. The first proposal for updating the relationships and the topologies based on this concept was also performed by Magnus.

8. IANA Considerations

This document makes no request of IANA.

9. Informative References

- [I-D.ietf-avtcore-rtp-multi-stream]
Lennox, J., Westerlund, M., Wu, W., and C. Perkins,
"Sending Multiple Media Streams in a Single RTP Session",
draft-ietf-avtcore-rtp-multi-stream-08 (work in progress),
July 2015.
- [I-D.ietf-avtcore-rtp-topologies-update]
Westerlund, M. and S. Wenger, "RTP Topologies", draft-
ietf-avtcore-rtp-topologies-update-10 (work in progress),
July 2015.
- [I-D.ietf-clue-framework]
Duckworth, M., Pepperell, A., and S. Wenger, "Framework
for Telepresence Multi-Streams", draft-ietf-clue-
framework-22 (work in progress), April 2015.
- [I-D.ietf-mmusic-sdp-bundle-negotiation]
Holmberg, C., Alvestrand, H., and C. Jennings,
"Negotiating Media Multiplexing Using the Session
Description Protocol (SDP)", draft-ietf-mmusic-sdp-bundle-
negotiation-23 (work in progress), July 2015.
- [I-D.ietf-mmusic-sdp-simulcast]
Burman, B., Westerlund, M., Nandakumar, S., and M. Zanaty,
"Using Simulcast in SDP and RTP Sessions", draft-ietf-
mmusic-sdp-simulcast-00 (work in progress), January 2015.
- [I-D.ietf-rtcweb-overview]
Alvestrand, H., "Overview: Real Time Protocols for
Browser-based Applications", draft-ietf-rtcweb-overview-14
(work in progress), June 2015.
- [I-D.westerlund-avtcore-transport-multiplexing]
Westerlund, M. and C. Perkins, "Multiplexing Multiple RTP
Sessions onto a Single Lower-Layer Transport", draft-
westerlund-avtcore-transport-multiplexing-07 (work in
progress), October 2013.

- [RFC2198] Perkins, C., Kouvelas, I., Hodson, O., Hardman, V., Handley, M., Bolot, J., Vega-Garcia, A., and S. Fosse-Parisis, "RTP Payload for Redundant Audio Data", RFC 2198, DOI 10.17487/RFC2198, September 1997, <<http://www.rfc-editor.org/info/rfc2198>>.
- [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>>.
- [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>>.
- [RFC4353] Rosenberg, J., "A Framework for Conferencing with the Session Initiation Protocol (SIP)", RFC 4353, DOI 10.17487/RFC4353, February 2006, <<http://www.rfc-editor.org/info/rfc4353>>.
- [RFC4566] Handley, M., Jacobson, V., and C. Perkins, "SDP: Session Description Protocol", RFC 4566, DOI 10.17487/RFC4566, July 2006, <<http://www.rfc-editor.org/info/rfc4566>>.
- [RFC4588] Rey, J., Leon, D., Miyazaki, A., Varsa, V., and R. Hakenberg, "RTP Retransmission Payload Format", RFC 4588, DOI 10.17487/RFC4588, July 2006, <<http://www.rfc-editor.org/info/rfc4588>>.
- [RFC4867] Sjöberg, J., Westerlund, M., Lankaniemi, A., and Q. Xie, "RTP Payload Format and File Storage Format for the Adaptive Multi-Rate (AMR) and Adaptive Multi-Rate Wideband (AMR-WB) Audio Codecs", RFC 4867, DOI 10.17487/RFC4867, April 2007, <<http://www.rfc-editor.org/info/rfc4867>>.
- [RFC5109] Li, A., Ed., "RTP Payload Format for Generic Forward Error Correction", RFC 5109, DOI 10.17487/RFC5109, December 2007, <<http://www.rfc-editor.org/info/rfc5109>>.
- [RFC5404] Westerlund, M. and I. Johansson, "RTP Payload Format for G.719", RFC 5404, DOI 10.17487/RFC5404, January 2009, <<http://www.rfc-editor.org/info/rfc5404>>.

- [RFC5481] Morton, A. and B. Claise, "Packet Delay Variation Applicability Statement", RFC 5481, DOI 10.17487/RFC5481, March 2009, <<http://www.rfc-editor.org/info/rfc5481>>.
- [RFC5576] Lennox, J., Ott, J., and T. Schierl, "Source-Specific Media Attributes in the Session Description Protocol (SDP)", RFC 5576, DOI 10.17487/RFC5576, June 2009, <<http://www.rfc-editor.org/info/rfc5576>>.
- [RFC5888] Camarillo, G. and H. Schulzrinne, "The Session Description Protocol (SDP) Grouping Framework", RFC 5888, DOI 10.17487/RFC5888, June 2010, <<http://www.rfc-editor.org/info/rfc5888>>.
- [RFC5905] Mills, D., Martin, J., Ed., Burbank, J., and W. Kasch, "Network Time Protocol Version 4: Protocol and Algorithms Specification", RFC 5905, DOI 10.17487/RFC5905, June 2010, <<http://www.rfc-editor.org/info/rfc5905>>.
- [RFC6190] Wenger, S., Wang, Y., Schierl, T., and A. Eleftheriadis, "RTP Payload Format for Scalable Video Coding", RFC 6190, DOI 10.17487/RFC6190, May 2011, <<http://www.rfc-editor.org/info/rfc6190>>.
- [RFC7160] Petit-Huguenin, M. and G. Zorn, Ed., "Support for Multiple Clock Rates in an RTP Session", RFC 7160, DOI 10.17487/RFC7160, April 2014, <<http://www.rfc-editor.org/info/rfc7160>>.
- [RFC7197] Begen, A., Cai, Y., and H. Ou, "Duplication Delay Attribute in the Session Description Protocol", RFC 7197, DOI 10.17487/RFC7197, April 2014, <<http://www.rfc-editor.org/info/rfc7197>>.
- [RFC7198] Begen, A. and C. Perkins, "Duplicating RTP Streams", RFC 7198, DOI 10.17487/RFC7198, April 2014, <<http://www.rfc-editor.org/info/rfc7198>>.
- [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>>.
- [RFC7273] Williams, A., Gross, K., van Brandenburg, R., and H. Stokking, "RTP Clock Source Signalling", RFC 7273, DOI 10.17487/RFC7273, June 2014, <<http://www.rfc-editor.org/info/rfc7273>>.

Appendix A. Changes From Earlier Versions

NOTE TO RFC EDITOR: Please remove this section prior to publication.

A.1. Modifications Between WG Version -07 and -08

Addresses comments from IESG evaluation.

- o Made text more firm around what improvements this document introduces.
- o Clarified the distinction between analog and digital in sections 2.1.1 and 2.1.2.
- o Removed the explicit requirement that a Source RTP Stream must send at least some data from an Encoded Stream, replacing it with a statement that it is directly related to the Encoded Stream.
- o Moved the clarification that RTP-based Redundancy excludes Media Encoder redundancy data in an Encoded Stream from Section 2.1.10 (RTP Stream) to 2.1.11 (RTP-based Redundancy), since that statement applies to RTP-based Redundancy rather than to RTP Stream.
- o Added clarification that a Media Transport Sender can intentionally pace packet transmission.
- o Aligned text around delay variation to use this term throughout, and added a reference to RFC 5481.
- o Added that RTP Session is a group communications channel that can potentially carry a number of RTP Streams, as an additional clarification below Figure 7.
- o Added a clarification in Section 4.1 around Telepresence Terms on which references are to CLUE terms and which are to other sections of this document, for terms that have the same name in CLUE as in this document.
- o Clarified in Section 4.14 what SSRC data format changes means, since the RFC 3550 SSRC definition mentions this possibility.
- o Editorial improvements.

A.2. Modifications Between WG Version -06 and -07

Addresses comments from AD review and GenArt review.

- o Added RTP-based Security and RTP-based Validation transform sections, as well as Secured RTP Stream and Received Secured RTP Stream sections.
- o Improved wording in Abstract and Introduction sections.
- o Clarified what is considered "media" in section 2.1.2 Media Capture.
- o Changed a number of "Characteristics" lists to more suitable prose text.
- o Re-worded text around use of Encoded and Dependent RTP Streams in section 2.1.9 Media Packetizer.
- o Clarified description of Source RTP Stream in section 2.1.10.
- o Clarified motivation to use separate Media Transports for Simulcast in section 3.6.
- o Added local descriptions of terms imported from CLUE framework.
- o Editorial improvements.

A.3. Modifications Between WG Version -05 and -06

- o Clarified that a Redundancy RTP Stream can be used standalone to generate Repaired RTP Streams.
- o Clarified that (in accordance with above) RTP-based Repair takes zero or more Received RTP Streams and one or more Received Redundancy RTP Streams as input.
- o Changed Figure 6 to more clearly show that Media Transport is terminated in the Endpoint, not in the Participant.
- o Added a sentence to Endpoint section that clarifies there may be contexts where a single "host" can serve multiple Participants, making those Endpoints share some properties.
- o Merged previous section 3.5 on SST/MST with previous section 3.8 on Layered Multi-Stream into a common section discussing the scalable/layered stream relation, and moved improved, descriptive

text on SST and MST to new sub-sections 4.7 and 4.13, describing them as existing terms.

- o Editorial improvements.

A.4. Modifications Between WG Version -04 and -05

- o Editorial improvements.

A.5. Modifications Between WG Version -03 and -04

- o Changed "Media Redundancy" and "Media Repair" to "RTP-based Redundancy" and "RTP-based Repair", since those terms are more specific and correct.
- o Changed "End Point" to "Endpoint" and removed Editor's Note on this.
- o Clarified that a Media Capture may impose constraints on clock handling.
- o Clarified that mixing multiple Raw Streams into a Source Stream is not possible, since that requires mixed streams to have a timing relation, requiring them to be Source Streams, and added an example.
- o Clarified that RTP-based Redundancy excludes the type of encoding redundancy found within the encoded media format in an Encoded Stream.
- o Clarified that a Media Transport contains only a single RTP Session, but a single RTP Session can span multiple Media Transports.
- o Clarified that packets with seemingly correct checksum that are received by a Media Transport Receiver may still be corrupt.
- o Clarified that a corrupt packet in a Media Transport Receiver is typically either discarded or somehow marked and passed on in the Received RTP Stream.
- o Added Synchronization Context to Figure 6.
- o Editorial improvements and clarifications.

A.6. Modifications Between WG Version -02 and -03

- o Changed section 3.5, removing SST-SS/MS and MST-SS/MS, replacing them with SRST, MRST, and MRMT.
- o Updated section 3.8 to align with terminology changes in section 3.5.
- o Added a new section 4.12, describing the term Multimedia Conference.
- o Changed reference from I-D to now published RFC 7273.
- o Editorial improvements and clarifications.

A.7. Modifications Between WG Version -01 and -02

- o Major re-structure
- o Moved media chain Media Transport detailing up one section level
- o Collapsed level 2 sub-sections of section 3 and thus moved level 3 sub-sections up one level, gathering some introductory text into the beginning of section 3
- o Added that not only SSRC collision, but also a clock rate change [RFC7160] is a valid reason to change SSRC value for an RTP stream
- o Added a sub-section on clock source signaling
- o Added a sub-section on RTP stream duplication
- o Elaborated a bit in section 2.2.1 on the relation between End Points, Participants and CNAMEs
- o Elaborated a bit in section 2.2.4 on Multimedia Session and synchronization contexts
- o Removed the section on CLUE scenes defining an implicit synchronization context, since it was incorrect
- o Clarified text on SVC SST and MST according to list discussions
- o Removed the entire topology section to avoid possible inconsistencies or duplications with draft-ietf-avtcore-rtp-topologies-update, but saved one example overview figure of Communication Entities into that section

- o Added a section 4 on mapping from existing terms with one sub-section per term, mainly by moving text from sections 2 and 3
- o Changed all occurrences of Packet Stream to RTP Stream
- o Moved all normative references to informative, since this is an informative document
- o Added references to RFC 7160, RFC 7197 and RFC 7198, and removed unused references

A.8. Modifications Between WG Version -00 and -01

- o WG version -00 text is identical to individual draft -03
- o Amended description of SVC SST and MST encodings with respect to concepts defined in this text
- o Removed UML as normative reference, since the text no longer uses any UML notation
- o Removed a number of level 4 sections and moved out text to the level above

A.9. Modifications Between Version -02 and -03

- o Section 4 rewritten (and new communication topologies added) to reflect the major updates to Sections 1-3
- o Section 8 removed (carryover from initial -00 draft)
- o General clean up of text, grammar and nits

A.10. Modifications Between Version -01 and -02

- o Section 2 rewritten to add both streams and transformations in the media chain.
- o Section 3 rewritten to focus on exposing relationships.

A.11. Modifications Between Version -00 and -01

- o Too many to list
- o Added new authors
- o Updated content organization and presentation

Authors' Addresses

Jonathan Lennox
Vidyo, Inc.
433 Hackensack Avenue
Seventh Floor
Hackensack, NJ 07601
US

Email: jonathan@vidyo.com

Kevin Gross
AVA Networks, LLC
Boulder, CO
US

Email: kevin.gross@avanw.com

Suhas Nandakumar
Cisco Systems
170 West Tasman Drive
San Jose, CA 95134
US

Email: snandaku@cisco.com

Gonzalo Salgueiro
Cisco Systems
7200-12 Kit Creek Road
Research Triangle Park, NC 27709
US

Email: gsalguei@cisco.com

Bo Burman (editor)
Ericsson
Kistavagen 25
SE-16480 Stockholm
Sweden

Email: bo.burman@ericsson.com

Network Working Group
Internet-Draft
Updates: 5104 (if approved)
Intended status: Standards Track
Expires: March 14, 2016

B. Burman
A. Akram
Ericsson
R. Even
Huawei Technologies
M. Westerlund
Ericsson
September 11, 2015

RTP Stream Pause and Resume
draft-ietf-avtext-rtp-stream-pause-10

Abstract

With the increased popularity of real-time multimedia applications, it is desirable to provide good control of resource usage, and users also demand more control over communication sessions. This document describes how a receiver in a multimedia conversation can pause and resume incoming data from a sender by sending real-time feedback messages when using Real-time Transport Protocol (RTP) for real time data transport. This document extends the Codec Control Messages (CCM) RTP Control Protocol (RTCP) feedback package by explicitly allowing and describing specific use of existing CCM messages and adding a group of new real-time feedback messages used to pause and resume RTP data streams. This document updates RFC 5104.

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 14, 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.

This document may contain material from IETF Documents or IETF Contributions published or made publicly available before November 10, 2008. The person(s) controlling the copyright in some of this material may not have granted the IETF Trust the right to allow modifications of such material outside the IETF Standards Process. Without obtaining an adequate license from the person(s) controlling the copyright in such materials, this document may not be modified outside the IETF Standards Process, and derivative works of it may not be created outside the IETF Standards Process, except to format it for publication as an RFC or to translate it into languages other than English.

Table of Contents

1. Introduction	4
2. Definitions	5
2.1. Abbreviations	5
2.2. Terminology	6
2.3. Requirements Language	7
3. Use Cases	7
3.1. Point to Point	8
3.2. RTP Mixer to Media Sender	8
3.3. RTP Mixer to Media Sender in Point-to-Multipoint	9
3.4. Media Receiver to RTP Mixer	10
3.5. Media Receiver to Media Sender Across RTP Mixer	10
4. Design Considerations	11
4.1. Real-time Nature	11
4.2. Message Direction	11
4.3. Apply to Individual Sources	12
4.4. Consensus	12
4.5. Message Acknowledgments	12
4.6. Request Retransmission	13
4.7. Sequence Numbering	13

4.8. Relation to Other Solutions	13
5. Solution Overview	14
5.1. Expressing Capability	15
5.2. PauseID	15
5.3. Requesting to Pause	16
5.4. Media Sender Pausing	17
5.5. Requesting to Resume	18
5.6. TMMBR/TMMBN Considerations	19
6. Participant States	20
6.1. Playing State	21
6.2. Pausing State	21
6.3. Paused State	22
6.3.1. RTCP BYE Message	22
6.3.2. SSRC Time-out	23
6.4. Local Paused State	23
7. Message Format	24
8. Message Details	27
8.1. PAUSE	28
8.2. PAUSED	29
8.3. RESUME	29
8.4. REFUSED	30
8.5. Transmission Rules	31
9. Signaling	32
9.1. Offer-Answer Use	36
9.2. Declarative Use	37
10. Examples	38
10.1. Offer-Answer	38
10.2. Point-to-Point Session	40
10.3. Point-to-Multipoint using Mixer	44
10.4. Point-to-Multipoint using Translator	46
11. IANA Considerations	49
12. Security Considerations	50
13. Contributors	51
14. Acknowledgements	52
15. References	52
15.1. Normative References	52
15.2. Informative References	53
Appendix A. Changes From Earlier Versions	54
A.1. Modifications Between Version -09 and -10	54
A.2. Modifications Between Version -08 and -09	54
A.3. Modifications Between Version -07 and -08	55
A.4. Modifications Between Version -06 and -07	56
A.5. Modifications Between Version -05 and -06	57
A.6. Modifications Between Version -04 and -05	58
A.7. Modifications Between Version -03 and -04	58
A.8. Modifications Between Version -02 and -03	58
A.9. Modifications Between Version -01 and -02	59
A.10. Modifications Between Version -00 and -01	59

Authors' Addresses	59
------------------------------	----

1. Introduction

As real-time communication attracts more people, more applications are created; multimedia conversation applications being one example. Multimedia conversation further exists in many forms, for example, peer-to-peer chat application and multiparty video conferencing controlled by central media nodes, such as RTP Mixers.

Multimedia conferencing may involve many participants; each has its own preferences for the communication session, not only at the start but also during the session. This document describes several scenarios in multimedia communication where a conferencing node or participant chooses to temporarily pause an incoming RTP [RFC3550] stream and later resume it when needed. The receiver does not need to terminate or inactivate the RTP session and start all over again by negotiating the session parameters, for example using SIP [RFC3261] with SDP [RFC4566] Offer/Answer [RFC3264].

Centralized nodes, like RTP Mixers or Multipoint Control Units (MCU), which either use logic based on voice activity, other measurements, or user input could reduce the resources consumed in both the sender and the network by temporarily pausing the RTP streams that aren't required by the RTP Mixer. If the number of conference participants are greater than what the conference logic has chosen to present simultaneously to receiving participants, some participant RTP streams sent to the RTP Mixer may not need to be forwarded to any other participant. Those RTP streams could then be temporarily paused. This becomes especially useful when the media sources are provided in multiple encoding versions (Simulcast) [I-D.ietf-mmusic-sdp-simulcast] or with Multi-Session Transmission (MST) of scalable encoding such as SVC [RFC6190]. There may be some of the defined encodings or combination of scalable layers that are not used or cannot be used all of the time. As an example, a centralized node may choose to pause such unused RTP streams without being explicitly requested to do so, maybe due to temporarily limited network or processing resources. It may then also send an explicit indication that the streams are paused.

As the set of RTP streams required at any given point in time is highly dynamic in such scenarios, using the out-of-band signaling channel for pausing, and even more importantly resuming, an RTP stream is difficult due to the performance requirements. Instead, the pause and resume signaling should be in the media plane and go directly between the affected nodes. When using RTP [RFC3550] for media transport, using the Extended RTP Profile for Real-time Transport Control Protocol (RTCP)-Based Feedback (RTP/AVPF) [RFC4585]

appears appropriate. No currently existing RTCP feedback message explicitly supports pausing and resuming an incoming RTP stream. As this affects the generation of packets and may even allow the encoding process to be paused, the functionality appears to match Codec Control Messages in the RTP Audio-Visual Profile with Feedback (AVPF) [RFC5104]. This document defines the solution as a Codec Control Message (CCM) extension.

The Temporary Maximum Media Bitrate Request (TMMBR) message of CCM is used by video conferencing systems for flow control. It is desirable to be able to use that method with a bitrate value of zero for pause, whenever possible. This specification updates RFC 5104 to add the new Pause and Resume semantics to the TMMBR/TMMBN messages.

2. Definitions

2.1. Abbreviations

AVPF: Audio-Visual Profile with Feedback (RFC 4585)

CCM: Codec Control Messages (RFC 5104)

CNAME: Canonical Name (RTCP SDES)

CSRC: Contributing Source (RTP)

FCI: Feedback Control Information (AVPF)

FIR: Full Intra Refresh (CCM)

FMT: Feedback Message Type (AVPF)

MCU: Multipoint Control Unit

MTU: Maximum Transfer Unit

PT: Payload Type (RTP)

RTP: Real-time Transport Protocol (RFC 3550)

RTCP: RTP Control Protocol (RFC 3550)

RTCP RR: RTCP Receiver Report

RTCP SR: RTCP Sender Report

SDP: Session Description Protocol (RFC 4566)

SIP: Session Initiation Protocol (RFC 3261)

SSRC: Synchronization Source (RTP)

SVC: Scalable Video Coding

TMMBR: Temporary Maximum Media Bitrate Request (CCM)

TMMBN: Temporary Maximum Media Bitrate Notification (CCM)

UA: User Agent (SIP)

UDP: User Datagram Protocol (RFC 768)

2.2. Terminology

In addition to the following, the definitions from RTP [RFC3550], AVPF [RFC4585], CCM [RFC5104], and RTP Taxonomy [I-D.ietf-avtext-rtp-grouping-taxonomy] also apply in this document.

Feedback Messages: CCM [RFC5104] categorized different RTCP feedback messages into four types, Request, Command, Indication and Notification. This document places the PAUSE and RESUME messages into Request category, PAUSED as Indication and REFUSED as Notification.

PAUSE: Request from an RTP stream receiver to pause a stream

RESUME: Request from an RTP stream receiver to resume a paused stream

PAUSED: Indication from an RTP stream sender that a stream is paused

REFUSED: Notification from an RTP stream sender that a PAUSE or RESUME request will not be honored

Mixer: The intermediate RTP node which receives an RTP stream from different endpoints, combines them to make one RTP stream and forwards to destinations, in the sense described in Topo-Mixer of RTP Topologies [I-D.ietf-avtcore-rtp-topologies-update].

Participant: A member which is part of an RTP session, acting as receiver, sender or both.

Paused sender: An RTP stream sender that has stopped its transmission, i.e. no other participant receives its RTP

transmission, either based on having received a PAUSE request, defined in this specification, or based on a local decision.

Pausing receiver: An RTP stream receiver which sends a PAUSE request, defined in this specification, to other participant(s).

Stream: Used as a short term for RTP stream, unless otherwise noted.

Stream receiver: Short for RTP stream receiver; the RTP entity responsible for receiving an RTP stream, usually a Media Depacketizer.

Stream sender: Short for RTP stream sender; the RTP entity responsible for creating an RTP stream, usually a Media Packetizer.

2.3. Requirements Language

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 RFC 2119 [RFC2119].

3. Use Cases

This section discusses the main use cases for RTP stream pause and resume.

RTCWEB WG's use case and requirements document [RFC7478] defines the following API requirements in Appendix A, used also by W3C WebRTC WG:

A8 The Web API must provide means for the web application to mute/unmute a stream or stream component(s). When a stream is sent to a peer mute, status must be preserved in the stream received by the peer.

A9 The Web API must provide means for the web application to cease the sending of a stream to a peer.

This document provides means to optimize transport usage by stop sending muted streams and start sending again when unmuting. It is here assumed that "mute" above can be taken to apply also to other media than audio. At the time of publication for this specification, RTCWEB did not specify any pause / resume functionality.

3.1. Point to Point

This is the most basic use case with an RTP session containing two Endpoints. Each Endpoint sends one or more streams.

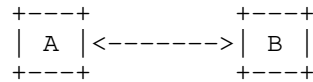


Figure 1: Point to Point

The usage of RTP stream pause in this use case is to temporarily halt delivery of streams that the sender provides but the receiver does not currently use. This can for example be due to minimized applications where the video stream is not actually shown on any display, and neither is it used in any other way, such as being recorded.

In this case, since there is only a single receiver of the stream, pausing or resuming a stream does not impact anyone else than the sender and the single receiver of that stream.

3.2. RTP Mixer to Media Sender

One of the most commonly used topologies in centralized conferencing is based on the RTP Mixer [I-D.ietf-avtcore-rtp-topologies-update]. The main reason for this is that it provides a very consistent view of the RTP session towards each participant. That is accomplished through the Mixer originating its own streams, identified by distinct SSRC values, and any RTP streams sent to the participants will be sent using those SSRC values. If the Mixer wants to identify the underlying media sources for its conceptual streams, it can identify them using CSRC. The stream the Mixer provides can be an actual mix of multiple media sources, but it might also be switching received streams as described in Sections 3.6-3.8 of [I-D.ietf-avtcore-rtp-topologies-update].



Figure 2: RTP Mixer in Unicast-only

Which streams from clients B, C and D that are delivered to a given receiver, A, can depend on several things. It can either be the RTP Mixer's own logic and measurements such as voice activity on the incoming audio streams. It can be that the number of sent media sources exceed what is reasonable to present simultaneously at any given receiver. It can also be a human controlling the conference that determines how the media should be mixed; this would be more common in lecture or similar applications where regular listeners may be prevented from breaking into the session unless approved by the moderator. The streams may also be part of a Simulcast [I-D.ietf-mmusic-sdp-simulcast] or scalable encoded (for Multi-Session Transmission) [RFC6190], thus providing multiple versions that can be delivered by the RTP stream sender. These examples indicate that there are numerous reasons why a particular stream would not currently be in use, but must be available for use at very short notice if any dynamic event occurs that causes a different stream selection to be done in the Mixer.

Because of this, it would be highly beneficial if the Mixer could request the RTP stream sender to pause a particular stream. The Mixer also needs to be able to request the RTP stream sender to resume delivery with minimal delay.

In some cases, especially when the Mixer sends multiple RTP streams per receiving client, there may be situations that makes it desirable for the Mixer to pause some of its sent RTP streams, even without being explicitly asked to do so by the receiving client. Such situations can for example be caused by a temporary lack of available Mixer network or processing resources. An RTP stream receiver that no longer receives an RTP stream could interpret this as an error condition and try to take action to re-establish the RTP stream. Such action would likely be undesirable if the RTP stream was in fact deliberately paused by the Mixer. Undesirable RTP stream receiver actions could be avoided if the Mixer is able to explicitly indicate that an RTP stream is deliberately paused.

Just as for point-to-point (Section 3.1), there is only a single receiver of the stream, the RTP Mixer, and pausing or resuming a stream does not affect anyone else than the sender and single receiver of that stream.

3.3. RTP Mixer to Media Sender in Point-to-Multipoint

This use case is similar to the previous section, however the RTP Mixer is involved in three domains that need to be separated; the Multicast Network (including participants A and C), participant B, and participant D. The difference from above is that A and C share a multicast domain, which is depicted below.

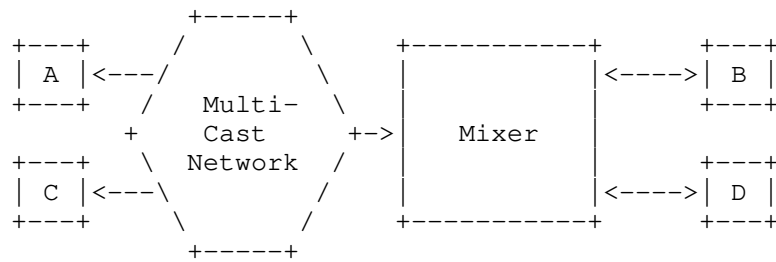


Figure 3: RTP Mixer in Point-to-Multipoint

If the RTP Mixer pauses a stream from A, it will not only pause the stream towards itself, but will also stop the stream from arriving to C, which C is heavily impacted by, might not approve of, and should thus have a say on.

If the Mixer resumes a paused stream from A, it will be resumed also towards C. In this case, if C is not interested it can simply ignore the stream and is not impacted as much as above.

In this use case there are several receivers of a stream and the Mixer must take special care so as not to pause a stream that is still wanted by some receivers.

3.4. Media Receiver to RTP Mixer

In this use case, the direction of the request to pause is the opposite compared to the two previous use cases. An Endpoint in Figure 2 could potentially request to pause the delivery of a given stream. Possible reasons include the ones in the point to point case (Section 3.1) above.

When the RTP Mixer is only connected to individual unicast paths, the use case and any considerations are identical to the point to point use case.

However, when the Endpoint requesting stream pause is connected to the RTP Mixer through a multicast network, such as A or C in Figure 3, the use case instead becomes identical to the one in Section 3.3, only with reverse direction of the streams and pause/resume requests.

3.5. Media Receiver to Media Sender Across RTP Mixer

An Endpoint, like A in Figure 2, could potentially request to pause the delivery of a given stream, like one of B's, over any of the SSRCs used by the Mixer by sending a pause request for the CSRC

identifying the stream. However, the authors are of the opinion that this is not a suitable solution, for several reasons:

1. The Mixer might not include CSRC in its stream indications.
2. An Endpoint cannot rely on the CSRC to correctly identify the stream to be paused when the delivered media is some type of mix. A more elaborate stream identification solution is needed to support this in the general case.
3. The Endpoint cannot determine if a given stream is still needed by the RTP Mixer to deliver to another session participant.

Due to the above reasons, we exclude this use case from further consideration.

4. Design Considerations

This section describes the requirements that this specification needs to meet.

4.1. Real-time Nature

The first section (Section 1) of this specification describes some possible reasons why a receiver may pause an RTP sender. Pausing and resuming is time-dependent, i.e. a receiver may choose to pause an RTP stream for a certain duration, after which the receiver may want the sender to resume. This time dependency means that the messages related to pause and resume must be transmitted to the sender in a timely fashion in order for them to be purposeful. The pause operation is arguably not as time critical as the resume operation, since it mainly provides a reduction of resource usage. Timely handling of the resume operation is however likely to directly impact the end-user's perceived quality experience, since it affects the availability of media that the user expects to receive more or less instantly. It may also be highly desirable for a receiver to quickly learn that an RTP stream is intentionally paused on the RTP sender's own behalf.

4.2. Message Direction

It is the responsibility of an RTP stream receiver that wants to pause or resume a stream from the sender(s) to transmit PAUSE and RESUME messages. An RTP stream sender that wants to pause itself can often simply do it, but sometimes this will adversely affect the receiver and an explicit indication that the RTP stream is paused may then help. Any indication that an RTP stream is paused is the

responsibility of the RTP stream sender and may in some cases not even be needed by the stream receiver.

4.3. Apply to Individual Sources

The PAUSE and RESUME messages apply to single RTP streams identified by their SSRC, which means the receiver targets the sender's SSRC in the PAUSE and RESUME requests. If a paused sender starts sending with a new SSRC, the receivers will need to send a new PAUSE request in order to pause it. PAUSED indications refer to a single one of the sender's own, paused SSRC.

4.4. Consensus

An RTP stream sender should not pause an SSRC that some receiver still wishes to receive.

The reason is that in RTP topologies where the stream is shared between multiple receivers, a single receiver on that shared network must not single-handedly cause the stream to be paused without letting all other receivers to voice their opinions on whether or not the stream should be paused. Such shared networks can for example be multicast, a mesh with a joint RTP session, or a transport Translator based network. A consequence of this is that a newly joining receiver firstly needs to learn the existence of paused streams, and secondly should be able to resume any paused stream. A newly joining receiver can for example be detected through an RTCP Receiver Report containing both a new SSRC and a CNAME that does not already occur in the session. Any single receiver wanting to resume a stream should also cause it to be resumed. An important exception to this is when the RTP stream sender is aware of conditions that makes it desirable or even necessitates to pause the RTP stream on its own behalf, without being explicitly asked to do so. Such local consideration in the RTP sender takes precedence over RTP receiver wishes to receive the stream.

4.5. Message Acknowledgments

RTP and RTCP does not guarantee reliable data transmission. It uses whatever assurance the lower layer transport protocol can provide. However, this is commonly UDP that provides no reliability guarantees. Thus it is possible that a PAUSE and/or RESUME message transmitted from an RTP Endpoint does not reach its destination, i.e. the targeted RTP stream sender. When PAUSE or RESUME reaches the RTP stream sender and are effective, i.e., an active RTP stream sender pauses, or a resuming RTP stream sender have media data to transmit, it is immediately seen from the arrival or non-arrival of RTP packets

for that RTP stream. Thus, no explicit acknowledgments are required in this case.

In some cases when a PAUSE or RESUME message reaches the RTP stream sender, it will not be able to pause or resume the stream due to some local consideration, for example lack of data to transmit. In this error condition, a negative acknowledgment may be needed to avoid unnecessary retransmission of requests (Section 4.6).

4.6. Request Retransmission

When the stream is not affected as expected by a PAUSE or RESUME request, the request may have been lost and the sender of the request will need to retransmit it. The retransmission should take the round trip time into account, and will also need to take the normal RTCP bandwidth and timing rules applicable to the RTP session into account, when scheduling retransmission of feedback.

When it comes to resume requests or unsolicited paused indications that are more time critical, the best performance may be achieved by repeating the message as often as possible until a sufficient number have been sent to reach a high probability of message delivery, or at an explicit indication that the message was delivered. For resume requests, such explicit indication can be delivery of the RTP stream being requested to be resumed.

4.7. Sequence Numbering

A PAUSE request message will need to have a sequence number to separate retransmissions from new requests. A retransmission keeps the sequence number unchanged, while it is incremented every time a new PAUSE request is transmitted that is not a retransmission of a previous request.

Since RESUME always takes precedence over PAUSE and are even allowed to avoid pausing a stream, there is a need to keep strict ordering of PAUSE and RESUME. Thus, RESUME needs to share sequence number space with PAUSE and implicitly references which PAUSE it refers to. For the same reasons, the explicit PAUSED indication also needs to share sequence number space with PAUSE and RESUME.

4.8. Relation to Other Solutions

A performance comparison between SIP/SDP and RTCP signaling technologies was made and included in draft versions of this specification. Using SIP and SDP to carry pause and resume information means that it will need to traverse the entire signaling path to reach the signaling destination (either the remote Endpoint

or the entity controlling the RTP Mixer), across any signaling proxies that potentially also has to process the SDP content to determine if they are expected to act on it. The amount of bandwidth required for a SIP/SDP-based signaling solution is in the order of at least 10 times more than an RTCP-based solution. Especially for UA sitting on mobile wireless access, this will risk introducing delays that are too long (Section 4.1) to provide a good user experience, and the bandwidth cost may also be considered infeasible compared to an RTCP-based solution. RTCP data is sent through the media path, which is likely shorter (contains fewer intermediate nodes) than the signaling path, may anyway have to traverse a few intermediate nodes. The amount of processing and buffering required in intermediate nodes to forward those RTCP messages is however believed to be significantly less than for intermediate nodes in the signaling path. Based on those considerations, RTCP is chosen as signaling protocol for the pause and resume functionality.

5. Solution Overview

The proposed solution implements PAUSE and RESUME functionality based on sending AVPF RTCP feedback messages from any RTP session participant that wants to pause or resume a stream targeted at the stream sender, as identified by the sender SSRC.

This solution re-uses CCM TMMBR and TMMBN [RFC5104] to the extent possible, and defines a small set of new RTCP feedback messages where new semantics is needed.

A single Feedback message specification is used to implement the new messages. The message consists of a number of Feedback Control Information (FCI) blocks, where each block can be a PAUSE request, a RESUME request, PAUSED indication, a REFUSED notification, or an extension to this specification. This structure allows a single feedback message to handle pause functionality on a number of streams.

The PAUSED functionality is also defined in such a way that it can be used standalone by the RTP stream sender to indicate a local decision to pause, and inform any receiver of the fact that halting media delivery is deliberate and which RTP packet was the last transmitted.

Special considerations that apply when using TMMBR/TMMBN for pause and resume purposes are described in Section 5.6. This specification applies to both the new messages defined in herein as well as their TMMBR/TMMBN counterparts, except when explicitly stated otherwise. An obvious exception are any reference to the message parameters that are only available in the messages defined here. For example, any reference to PAUSE in the text below is equally applicable to TMMBR

0, and any reference to PAUSED is equally applicable to TMMBN 0. Therefore and for brevity, TMMBR/TMMBN will not be mentioned in the text, unless there is specific reason to do so.

This section is intended to be explanatory and therefore intentionally contains no mandatory statements. Such statements can instead be found in other parts of this specification.

5.1. Expressing Capability

An Endpoint can use an extension to CCM SDP signaling to declare capability to understand the messages defined in this specification. Capability to understand only a subset of messages is possible, to support partial implementation, which is specifically believed to be feasible for the RTP Mixer to Media Sender use case (Section 3.2). In that use case, only the RTP Mixer has capability to request the Media Sender to pause or resume. Consequently, in that same use case only the Media Sender has capability to pause and resume its sent streams based on requests from the RTP Mixer. Allowing for partial implementation of this specification is not believed to hamper interoperability, as long as the subsets are well defined and describe a consistent functionality, including a description of how a more capable implementation must perform fallback.

For the case when TMMBR/TMMBN are used for pause and resume purposes, it is possible to explicitly express joint support for TMMBR and TMMBN, but not for TMMBN only.

5.2. PauseID

All messages defined in this specification (Section 8) contain a PauseID, satisfying the design consideration on sequence numbering (Section 4.7). This PauseID is scoped by and thus a property of the targeted RTP stream (SSRC), not only a sequence number for individual messages. Instead, it numbers an entire "pause and resume operation" for the RTP stream, typically keeping PauseID constant for multiple, related messages. The PauseID value used during such operation is called the current PauseID. A new "pause and resume operation" is defined to start when the RTP stream sender resumes the RTP stream after it was being paused. The current PauseID is then incremented by one, in modulo arithmetic. In the subsequent descriptions below, it is sometimes necessary to refer to PauseID values that were already used as current PauseID, which is denoted as past PauseID. It should be noted that since PauseID uses modulo arithmetic, a past PauseID may have a larger value than the current PauseID. Since PauseID uses modulo arithmetic, it is also useful to define what PauseID values that are considered "past", to clearly separate it from what could be considered "future" PauseID values. Half of the

entire PauseID value range is chosen to represent past PauseID, while a quarter of the PauseID value range is chosen to represent future values. The remaining quarter of the PauseID value range is intentionally left undefined in that respect.

5.3. Requesting to Pause

An RTP stream receiver can choose to send a PAUSE request at any time, subject to AVPF timing rules.

The PAUSE request contains the current PauseID (Section 5.2).

When a non-paused RTP stream sender receives the PAUSE request, it continues to send the RTP stream while waiting for some time to allow other RTP stream receivers in the same RTP session that saw this PAUSE request to disapprove by sending a RESUME (Section 5.5) for the same stream and with the same current PauseID as in the PAUSE being disapproved. If such disapproving RESUME arrives at the RTP stream sender during the hold-off period before the stream is paused, the pause is not performed. In point-to-point configurations, the hold-off period may be set to zero. Using a hold-off period of zero is also appropriate when using TMMBR 0 and in line with the semantics for that message.

If the RTP stream sender receives further PAUSE requests with the current PauseID while waiting as described above, those additional requests are ignored.

If the PAUSE request is lost before it reaches the RTP stream sender, it will be discovered by the RTP stream receiver because it continues to receive the RTP stream. It will also not see any PAUSED indication (Section 5.4) for the stream. The same condition can be caused by the RTP stream sender having received a disapproving RESUME from a stream receiver A for a PAUSE request sent by a stream sender B, but that the PAUSE sender (B) did not receive the RESUME (from A) and may instead think that the PAUSE was lost. In both cases, a PAUSE request can be re-transmitted using the same current PauseID. If using TMMBR 0, the request MAY be re-transmitted when the requester fails to receive a TMMBN 0 confirmation.

If the pending stream pause is aborted due to a disapproving RESUME, the pause and resume operation for that PauseID is concluded, the current PauseID is updated, and any new PAUSE must therefore use the new current PauseID to be effective.

An RTP stream sender receiving a PAUSE not using the current PauseID informs the RTP stream receiver sending the ineffective PAUSE of this

condition by sending a REFUSED notification that contains the current PauseID value.

A situation where an ineffective PauseID is chosen can appear when a new RTP stream receiver joins a session and wants to PAUSE a stream, but does not yet know the current PauseID to use. The REFUSED notification will then provide sufficient information to create a valid PAUSE. The required extra signaling round-trip is not considered harmful, since it is assumed that pausing a stream is not time-critical (Section 4.1).

There may be local considerations making it impossible or infeasible to pause the stream, and the RTP stream sender can then respond with a REFUSED. In this case, if the used current PauseID would otherwise have been effective, REFUSED contains the same current PauseID as in the PAUSE request. Note that when using TMMBR 0 as PAUSE, that request cannot be refused ($TMMBN > 0$) due to the existing restriction in section 4.2.2.2 of [RFC5104] that TMMBN shall contain the current bounding set, and the fact that a TMMBR 0 will always be the most restrictive point in any bounding set, regardless of bounding set overhead value.

If the RTP stream sender receives several identical PAUSE for an RTP stream that was already at least once responded with REFUSED and the condition causing REFUSED remains, those additional REFUSED should be sent with regular RTCP timing. A single REFUSED can respond to several identical PAUSE requests.

5.4. Media Sender Pausing

An RTP stream sender can choose to pause the stream at any time. This can either be as a result of receiving a PAUSE, or be based on some local sender consideration. When it does, it sends a PAUSED indication, containing the current PauseID. Note that current PauseID in an unsolicited PAUSED (without having received a PAUSE), is incremented compared to previously sent PAUSED. It also sends the PAUSED indication in the next two regular RTCP reports, given that the pause condition is then still effective.

There is no reply to a PAUSED indication; it is simply an explicit indication of the fact that an RTP stream is paused. This can be helpful for the RTP stream receiver, for example to quickly understand that transmission is deliberately and temporarily suspended and no specific corrective action is needed.

The RTP stream sender may want to apply some local consideration to exactly when the RTP stream is paused, for example completing some

media unit or a forward error correction block, before pausing the stream.

The PAUSED indication also contains information about the RTP extended highest sequence number when the pause became effective. This provides RTP stream receivers with firsthand information allowing them to know whether they lost any packets just before the stream paused or when the stream is resumed again. This allows RTP stream receivers to quickly and safely take into account that the stream is paused, in for example retransmission or congestion control algorithms.

If the RTP stream sender receives PAUSE requests with the current PauseID while the stream is already paused, those requests are ignored.

As long as the stream is being paused, the PAUSED indication MAY be sent together with any regular RTCP Sender Report (SR) or Receiver Report (RR). Including PAUSED in this way allows RTP stream receivers joining while the stream is paused to quickly know that there is a paused stream, what the last sent extended RTP sequence number was, and what the current PauseID is to be able to construct valid PAUSE and RESUME requests at a later stage.

When the RTP stream sender learns that a new Endpoint has joined the RTP session, for example by a new SSRC and a CNAME that was not previously seen in the RTP session, it should send PAUSED indications for all its paused streams at its earliest opportunity. It should in addition continue to include PAUSED indications in at least two regular RTCP reports.

5.5. Requesting to Resume

An RTP stream receiver can request the RTP stream sender to resume a stream with a RESUME request at any time, subject to AVPF timing rules. The RTP stream receiver must include the current PauseID in the RESUME request for it to be effective.

A pausing RTP stream sender that receives a RESUME including the current PauseID resumes the stream at the earliest opportunity. Receiving RESUME requests for a stream that is not paused does not require any action and can be ignored.

There may be local considerations at the RTP stream sender, for example that the media device is not ready, making it temporarily impossible to resume the stream at that point in time, and the RTP stream sender can then respond with a REFUSED containing the current PauseID. When receiving such REFUSED with a current PauseID

identical to the one in the sent RESUME, RTP stream receivers should avoid sending further RESUME requests for some reasonable amount of time, to allow the condition to clear. An RTP stream sender having sent a REFUSED SHOULD resume the stream through local considerations (see below) when the condition that caused the REFUSED is no longer true.

If the RTP stream sender receives several identical RESUME for an RTP stream that was already at least once responded with REFUSED and the condition causing REFUSED remains, those additional REFUSED should be sent with regular RTCP timing. A single REFUSED can respond to several identical RESUME requests.

A pausing RTP stream sender can apply local considerations and can resume a paused RTP stream at any time. If TMMBR 0 was used to pause the RTP stream, resumption is prevented by protocol, even if the RTP sender would like to resume due to local considerations. If TMMBR/TMMBN signaling is used, if the RTP stream is paused due to local considerations (Section 5.4), and the RTP stream sender thus owns the TMMBN bounding set, the RTP stream can be resumed due to local considerations.

When resuming a paused stream, especially for media that makes use of temporal redundancy between samples such as video, it may not be appropriate to use such temporal dependency in the encoding between samples taken before the pause and at the time instant the stream is resumed. Should such temporal dependency between media samples before and after the media was paused be used by the RTP stream sender, it requires the RTP stream receiver to have saved the samples from before the pause for successful continued decoding when resuming. The use of this temporal dependency of media samples from before the pause is left up to the RTP stream sender. If temporal dependency on samples from before the pause is not used when the RTP stream is resumed, the first encoded sample after the pause will not contain any temporal dependency on samples before the pause (for video it may be a so-called intra picture). If temporal dependency on samples from before the pause is used by the RTP stream sender when resuming, and if the RTP stream receiver did not save any sample from before the pause, the RTP stream receiver can use a FIR request [RFC5104] to explicitly ask for a sample without temporal dependency (for video a so-called intra picture), even at the same time as sending the RESUME.

5.6. TMMBR/TMMBN Considerations

As stated above, TMMBR/TMMBN may be used to provide pause and resume functionality for the point-to-point case. If the topology is not point-to-point, TMMBR/TMMBN cannot safely be used for pause or

resume. This use is expected to be mainly for interworking with implementations that don't support the messages defined in this specification (Section 8), but make use of TMMBR/TMMBN to achieve a similar effect.

This is a brief summary of what functionality is provided when using TMMBR/TMMBN:

TMMBR 0: Corresponds to PAUSE, without the requirement for any hold-off period to wait for RESUME before pausing the RTP stream.

TMMBR >0: Corresponds to RESUME when the RTP stream was previously paused with TMMBR 0. Since there is only a single RTP stream receiver, there is no need for the RTP stream sender to delay resuming the stream until after sending TMMBN >0, or to apply the hold-off period specified in [RFC5104] before increasing the bitrate from zero. The bitrate value used when resuming after pausing with TMMBR 0 is either according to known limitations, or based on starting a stream with the configured maximum for the stream or session, for example given by b-parameter in SDP.

TMMBN 0: Corresponds to PAUSED when the RTP stream was paused with TMMBR 0, but may, just as PAUSED, also be used unsolicited. An unsolicited RTP stream pause based on local sender considerations uses the RTP stream's own SSRC as TMMBR restriction owner in the TMMBN message bounding set. Also corresponds to a REFUSED notification when a stream is requested to be resumed with TMMBR >0, thus resulting in the stream sender becoming the owner of the bounding set in the TMMBN message.

TMMBN >0: Cannot be used as REFUSED notification when a stream is requested to be paused with TMMBR 0, for reasons stated in Section 5.3.

6. Participant States

This document introduces three new states for a stream in an RTP sender, according to the figure and sub-sections below. Any references to PAUSE, PAUSED, RESUME and REFUSED in this section SHALL be taken to apply to the extent possible also when TMMBR/TMMBN are used (Section 5.6) for this functionality.

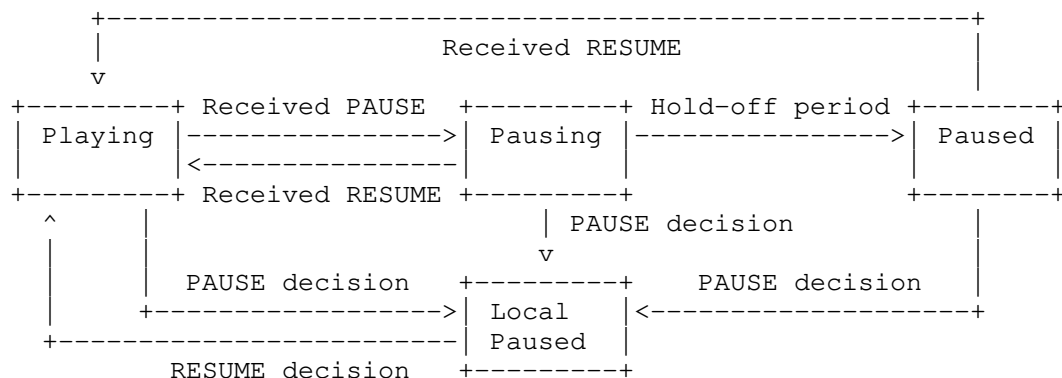


Figure 4: RTP Pause States in Sender

6.1. Playing State

This state is not new, but is the normal media sending state from [RFC3550]. When entering the state, the current PauseID MUST be incremented by one in modulo arithmetic. The RTP sequence number for the first packet sent after a pause SHALL be incremented by one compared to the highest RTP sequence number sent before the pause. The first RTP Time Stamp for the first packet sent after a pause SHOULD be set according to capture times at the source, meaning the RTP Time Stamp difference compared to before the pause reflects the time the RTP stream was paused.

6.2. Pausing State

In this state, the RTP stream sender has received at least one PAUSE message for the stream in question. The RTP stream sender SHALL wait during a hold-off period for the possible reception of RESUME messages for the RTP stream being paused before actually pausing RTP stream transmission. The hold-off period to wait SHALL be long enough to allow another RTP stream receiver to respond to the PAUSE with a RESUME, if it determines that it would not like to see the stream paused. This hold-off period is determined by the formula:

$$2 * RTT + T_dither_max,$$

where RTT is the longest round trip known to the RTP stream sender and T_dither_max is defined in section 3.4 of [RFC4585]. The hold-off period MAY be set to 0 by some signaling (Section 9) means when it can be determined that there is only a single receiver, for example in point-to-point or some unicast situations.

If the RTP stream sender has set the hold-off period to 0 and receives information that it was an incorrect decision and that there are in fact several receivers of the stream, it MUST change the hold-off to instead be based on the above formula.

An RTP stream sender SHOULD use the following criteria to determine if there is only a single receiver, unless it has explicit and more reliable information:

- o Observing only a single CNAME across all received SSRCS (CNAMEs for received CSRCs are insignificant), or
- o If RTCP reporting groups [I-D.ietf-avtcore-rtp-multi-stream-optimisation] is used, observing only a single, endpoint external RTCP reporting group.

6.3. Paused State

An RTP stream is in paused state when the sender pauses its transmission after receiving at least one PAUSE message and the hold-off period has passed without receiving any RESUME message for that stream. Pausing transmission SHOULD only be done when reaching an appropriate place to pause in the stream, like a media boundary that avoids a media receiver to trigger repair or concealment actions.

When entering the state, the RTP stream sender SHALL send a PAUSED indication to all known RTP stream receivers, and SHALL also repeat PAUSED in the next two regular RTCP reports, as long as it is then still in paused state.

Pausing an RTP stream MUST NOT affect the sending of RTP keepalive [RFC6263][RFC5245] applicable to that RTP stream.

Following sub-sections discusses some potential issues when an RTP sender goes into paused state. These conditions are also valid if an RTP Translator is used in the communication. When an RTP Mixer implementing this specification is involved between the participants (which forwards the stream by marking the RTP data with its own SSRC), it SHALL be a responsibility of the Mixer to control sending PAUSE and RESUME requests to the sender. The below conditions also apply to the sender and receiver parts of the RTP Mixer, respectively.

6.3.1. RTCP BYE Message

When a participant leaves the RTP session, it sends an RTCP BYE message. In addition to the semantics described in section 6.3.4 and

6.3.7 of RTP [RFC3550], following two conditions MUST also be considered when an RTP participant sends an RTCP BYE message,

- o If a paused sender sends an RTCP BYE message, receivers observing this SHALL NOT send further PAUSE or RESUME requests to it.
- o Since a sender pauses its transmission on receiving the PAUSE requests from any receiver in a session, the sender MUST keep record of which receiver that caused the RTP stream to pause. If that receiver sends an RTCP BYE message observed by the sender, the sender SHALL resume the RTP stream. No receivers that were in the RTP session when the stream was paused objected that the stream was paused, but if there were so far undetected receivers added to the session during pause, those may not have learned about the existence of the paused stream, either because there was no PAUSED sent for the paused RTP stream or those receivers did not support PAUSED. Resuming the stream when the pausing party leaves the RTP session allows those potentially undetected receivers to learn that the stream exists.

6.3.2. SSRC Time-out

Section 6.3.5 in RTP [RFC3550] describes the SSRC time-out of an RTP participant. Every RTP participant maintains a sender and receiver list in a session. If a participant does not get any RTP or RTCP packets from some other participant for the last five RTCP reporting intervals it removes that participant from the receiver list. Any streams that were paused by that removed participant SSRC SHALL be resumed.

6.4. Local Paused State

This state can be entered at any time, based on local decision from the RTP stream sender. Pausing transmission SHOULD only be done when reaching an appropriate place to pause in the stream, like a media boundary that avoids a media receiver to trigger repair or concealment actions.

As with Paused State (Section 6.3), the RTP stream sender SHALL send a PAUSED indication to all known RTP stream receivers, when entering the state, unless the stream was already in paused state (Section 6.3). Such PAUSED indication SHALL be repeated a sufficient number of times to reach a high probability that the message is correctly delivered, stopping such repetition whenever leaving the state.

When using TMMBN 0 as PAUSED indication and when already in paused state, the actions when entering local paused state depends on the

bounding set overhead value in the received TMMBR 0 that caused the paused state, and the bounding set overhead value used in (the RTP stream sender's own) TMMBN 0:

TMMBN 0 overhead \leq TMMBR 0 overhead: The RTP stream sender SHALL NOT send any new TMMBN 0 replacing that active (more restrictive) bounding set, even if entering local paused state.

TMMBN 0 overhead $>$ TMMBR 0 overhead: The RTP stream sender SHALL send TMMBN 0 with itself in the TMMBN bounding set when entering local paused state.

The case above when using TMMBN 0 as PAUSED indication, being in local paused state, and having received a TMMBR 0 with a bounding set overhead value greater than the value the RTP stream sender would itself use in a TMMBN 0 requires further consideration and is for clarity henceforth referred to as "restricted local paused state".

As indicated in Figure 4, local paused state has higher precedence than paused state (Section 6.3) and RESUME messages alone cannot resume a paused RTP stream as long as the local decision still applies. An RTP stream sender in local paused state is responsible for leaving the state whenever the conditions that caused the decision to enter the state no longer apply.

If the RTP stream sender is in restricted local paused state, it cannot leave that state until the TMMBR 0 limit causing the state is removed by a TMMBR >0 (RESUME). If the RTP stream sender then needs to stay in local paused state due to local considerations, it MAY continue pausing the RTP stream by entering local paused state and MUST then act accordingly, including sending a TMMBN 0 with itself in the bounding set.

Pausing an RTP stream MUST NOT affect the sending of RTP keepalive [RFC6263][RFC5245] applicable to that RTP stream.

When leaving the local paused state, the stream state SHALL become Playing, regardless whether or not there were any RTP stream receivers that sent PAUSE for that stream during the local paused state, effectively clearing the RTP stream sender's memory for that stream.

7. Message Format

Section 6 of AVPF [RFC4585] defines three types of low-delay RTCP feedback messages, i.e. Transport layer, Payload-specific, and Application layer feedback messages. This document defines a new Transport layer feedback message, which is further sub-typed into

either a PAUSE request, a RESUME request, a PAUSED indication, or a REFUSED notification.

The Transport layer feedback messages are identified by having the RTCP payload type be RTPFB (205) as defined by AVPF [RFC4585]. This Transport layer feedback message, containing one or more of the sub-typed messages, is henceforth referred to as the PAUSE-RESUME message. The specific FCI format is identified by a Feedback Message Type (FMT) value in common packet header for feedback message defined in section 6.1 of AVPF [RFC4585]. The PAUSE-RESUME transport feedback message FCI is identified by FMT value = TBA1.

The Common Packet Format for Feedback Messages defined by AVPF [RFC4585] is:

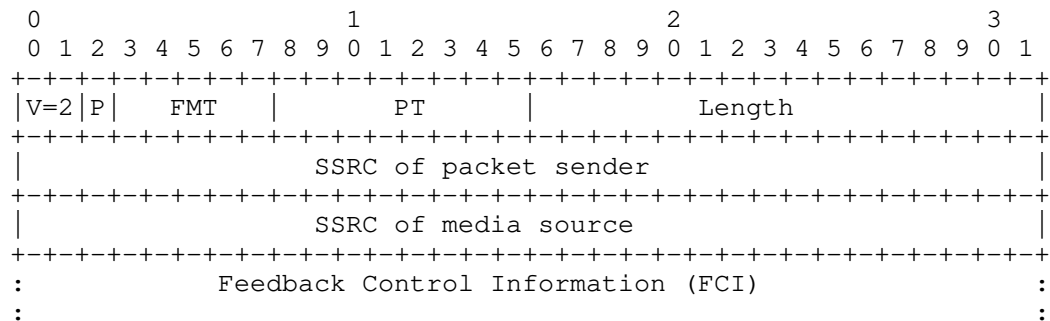


Figure 5: AVPF Common Feedback Message Packet Format

For the PAUSE-RESUME message defined in this memo, the following interpretations of the packet fields apply:

FMT: The FMT value identifying the PAUSE-RESUME FCI: TBA1

PT: Payload Type = 205 (RTPFB)

Length: As defined by AVPF, i.e. the length of this packet in 32-bit words minus one, including the header and any padding.

SSRC of packet sender: The SSRC of the RTP session participant sending the messages in the FCI. Note, for Endpoints that have multiple SSRCs in an RTP session, any of its SSRCs MAY be used to send any of the pause message types.

SSRC of media source: Not used, SHALL be set to 0. The FCI identifies the SSRC the message is targeted for.

The Feedback Control Information (FCI) field consists of one or more PAUSE, RESUME, PAUSED, REFUSED, or any future extension. These messages have the following FCI format:

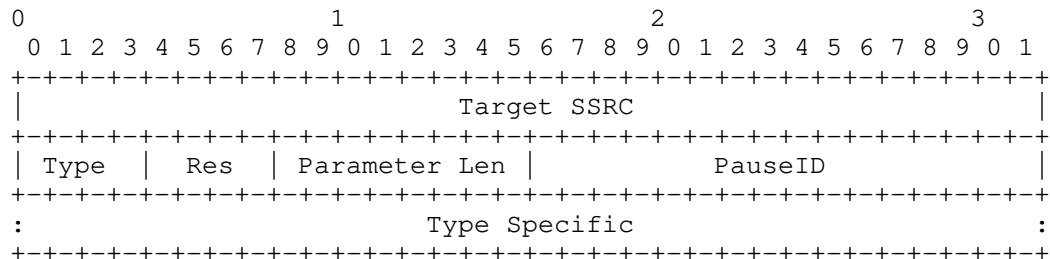


Figure 6: Syntax of FCI Entry in the PAUSE and RESUME message

The FCI fields have the following definitions:

Target SSRC (32 bits): For a PAUSE-RESUME message, this value is the SSRC that the request is intended for. For PAUSED, it MUST be the SSRC being paused. If pausing is the result of a PAUSE request, the value in PAUSED is effectively the same as Target SSRC in a related PAUSE request. For REFUSED, it MUST be the Target SSRC of the PAUSE or RESUME request that cannot change state. A CSRC MUST NOT be used as a target as the interpretation of such a request is unclear.

Type (4 bits): The pause feedback type. The values defined in this specification are as follows,

0: PAUSE request message.

1: RESUME request message.

2: PAUSED indication message.

3: REFUSED notification message.

4-15: Reserved for future use. FCI fields with these Type values SHALL be ignored on reception by receivers and MUST NOT be used by senders implementing this specification.

Res: (4 bits): Type specific reserved. SHALL be ignored by receivers implementing this specification and MUST be set to 0 by senders implementing this specification.

Parameter Len: (8 bits): Length of the Type Specific field in 32-bit words. MAY be 0.

PauseID (16 bits): Message sequence identification, as described in Section 5.2. SHALL be incremented by one modulo 2^{16} for each new PAUSE message, unless the message is re-transmitted. The initial value SHOULD be 0. The PauseID is scoped by the Target SSRC, meaning that PAUSE, RESUME, and PAUSED messages therefore share the same PauseID space for a specific Target SSRC.

Type Specific: (variable): Defined per pause feedback Type. MAY be empty. A receiver implementing this specification MUST be able to skip and ignore any unknown Type Specific data, even for Type values defined in this specification.

8. Message Details

This section contains detailed explanations of each message defined in this specification. All transmissions of requests and indications are governed by the transmission rules as defined by Section 8.5.

Any references to PAUSE, PAUSED, RESUME and REFUSED in this section SHALL be taken to apply to the extent possible also when TMMBR/TMMBN are used (Section 5.6) for this functionality. TMMBR/TMMBN MAY be used instead of the messages defined in this specification when the effective topology is point-to-point. This use is expected to be mainly for interworking with implementations that don't support the messages defined in this specification, but make use of TMMBR/TMMBN to achieve a similar effect. If either sender or receiver learns that the topology is not point-to-point, TMMBR/TMMBN MUST NOT be used for pause/resume functionality. If the messages defined in this specification are supported in addition to TMMBR/TMMBN by all involved parties, pause/resume signaling MUST use messages from this specification. If the topology is not point-to-point and the messages defined in this specification are not supported, pause/resume functionality with TMMBR/TMMBN MUST NOT be used.

For the scope of this specification, a past PauseID (Section 5.2) is defined as having a value between and including $(\text{PauseID} - 2^{15}) \bmod 2^{16}$ and $(\text{PauseID} - 1) \bmod 2^{16}$, where "MOD" is the modulo operator. Similarly, a future PauseID is defined as having a value between and including $(\text{PauseID} + 1) \bmod 2^{16}$ and $(\text{PauseID} + 2^{14}) \bmod 2^{16}$. Future PauseID is intentionally not defined as the entire range that was not already defined as past PauseID. The remaining range of PauseID is simply "not current".

8.1. PAUSE

An RTP stream receiver MAY schedule PAUSE for transmission at any time.

PAUSE has no defined Type Specific parameters.

PauseID SHOULD be the current PauseID, as indicated by PAUSED (Section 8.2), REFUSED (Section 8.4), or implicitly determined by previously received PAUSE or RESUME (Section 8.3) requests. A randomly chosen PauseID MAY be used if it was not possible to retrieve current PauseID information, in which case the PAUSE will either succeed, or the current PauseID can be found in the returned REFUSED (Section 8.4).

It can be noted that as a result of what is described in Section 6.1, PauseID is incremented by one, in modulo arithmetic, for each PAUSE request that is not a retransmission, compared to what was used in the last PAUSED indication sent by the media sender. PauseID in the message is supposed to match current PauseID at the RTP stream sender.

If an RTP stream receiver that sent a PAUSE with a certain PauseID for a target SSRC receives a RESUME or a REFUSED with the same PauseID for the same target SSRC, it is RECOMMENDED that it refrains from scheduling further PAUSE requests for some appropriate time. This is because the RESUME indicates that there are other receivers that still wishes to receive the stream, and the REFUSED indicates that the RTP stream sender is currently not able to pause the stream. What is an appropriate time can vary from application to application and will also depend on the importance of achieving the bandwidth saving, but 2-5 regular RTCP intervals is expected to be appropriate.

If the targeted RTP stream does not pause, if no PAUSED indication with a future PauseID compared to the one used in PAUSE is received, and if no REFUSED with the current or a future PauseID is received within $2 * RTT + T_dither_max$, the PAUSE MAY be scheduled for retransmission, using the same current PauseID. RTT is the observed round-trip to the RTP stream sender and T_dither_max is defined in section 3.4 of [RFC4585].

When an RTP stream sender in Playing State (Section 6.1) receives a PAUSE with the current PauseID, and unless local considerations currently makes it impossible to pause the stream, it SHALL enter Pausing State (Section 6.2) and act accordingly.

If an RTP stream sender receives a PAUSE with the current PauseID while in Pausing, Paused (Section 6.3) or Local Paused (Section 6.4) States, the received PAUSE SHALL be ignored.

8.2. PAUSED

The PAUSED indication, if supported, MUST be sent whenever entering Paused State (Section 6.3) or Local Paused State (Section 6.4).

PauseID in the PAUSED message MUST contain the current PauseID that can be included in a subsequent RESUME (Section 8.3). For Local Paused State, this means that PauseID in the message is the current PauseID, just as if the RTP stream sender had sent a PAUSE to itself.

PAUSED SHALL contain a fixed-length 32-bit parameter at the start of the Type Specific field with the RTP extended highest sequence number (Section 6.4.1 of [RFC3550]) valid when the RTP stream was paused.

After having entered Paused or Local Paused State and thus having sent PAUSED once, PAUSED MUST also be included in (at least) the next two regular RTCP reports, given that the pause condition is then still effective.

PAUSED indications MAY be retransmitted, subject to transmission rules (Section 8.5), to increase the probability that the message reaches the receiver in a timely fashion. This can be especially important when entering Local Paused State. The number of repetitions to use could be tuned to observed loss rate and desired loss probability, for example based on RTCP reports received from the intended message target.

While remaining in Paused or Local Paused States, PAUSED MAY be included in all compound RTCP reports, as long as the negotiated RTCP bandwidth is not exceeded.

When in Paused or Local Paused States, whenever the RTP stream sender learns that there are Endpoints that did not previously receive the stream, for example by RTCP reports with an SSRC and a CNAME that was not previously seen in the RTP session, it is RECOMMENDED to send PAUSED at the earliest opportunity and also to include it in (at least) the next two regular RTCP reports, given that the pause condition is then still effective.

8.3. RESUME

An RTP stream receiver MAY schedule RESUME for transmission whenever it wishes to resume a paused stream, or to disapprove a stream from being paused.

PauseID SHOULD be the current PauseID, as indicated by PAUSED (Section 8.2) or implicitly determined by previously received PAUSE (Section 8.1) or RESUME requests. A randomly chosen PauseID MAY be used if it was not possible to retrieve current PauseID information, in which case the RESUME will either succeed, or the current PauseID can be found in a returned REFUSED (Section 8.4).

If an RTP stream receiver that sent a RESUME with a certain PauseID receives a REFUSED with the same PauseID, it is RECOMMENDED that it refrains from scheduling further RESUME requests for some appropriate time since the REFUSE indicates that it is currently not possible to resume the stream. What is an appropriate time can vary from application to application and will also depend on the importance of resuming the stream, but 1-2 regular RTCP intervals is expected to be appropriate.

RESUME requests MAY be retransmitted, subject to transmission rules (Section 8.5), to increase the probability that the message reaches the receiver in a timely fashion. The number of repetitions to use could be tuned to observed loss rate and desired loss probability, for example based on RTCP reports received from the intended message target. Such retransmission SHOULD stop as soon as RTP packets from the targeted stream are received, or a REFUSED with the current PauseID for the targeted RTP stream is received.

RESUME has no defined Type Specific parameters.

When an RTP stream sender in Pausing (Section 6.2), Paused (Section 6.3) or Local Paused State (Section 6.4) receives a RESUME with the current PauseID, and unless local considerations currently makes it impossible to resume the stream, it SHALL enter Playing State (Section 6.1) and act accordingly. If the RTP stream sender is incapable of honoring a RESUME request with the current PauseID, or if it receives a RESUME request with a PauseID that is not the current PauseID while in Paused or Pausing state, the RTP stream sender SHALL schedule a REFUSED message for transmission as specified below.

If an RTP stream sender in Playing State receives a RESUME containing either the current PauseID or a past PauseID, the received RESUME SHALL be ignored.

8.4. REFUSED

If an RTP stream sender receives a PAUSE (Section 8.1) or RESUME (Section 8.3) request containing the current PauseID, where the requested action cannot be fulfilled by the RTP stream sender due to some local consideration, it SHALL schedule transmission of a REFUSED

notification containing the current PauseID from the rejected request.

REFUSED has no defined Type Specific parameters.

If an RTP stream sender receives a PAUSE or RESUME request with a PauseID that is not the current PauseID, it SHALL schedule a REFUSED notification containing the current PauseID, except if the RTP stream sender is in Playing State and receives a RESUME with a past PauseID, in which case the RESUME SHALL be ignored.

If several PAUSE or RESUME that would render identical REFUSED notifications are received before the scheduled REFUSED is sent, duplicate REFUSED MUST NOT be scheduled for transmission. This effectively lets a single REFUSED respond to several ineffective PAUSE or RESUME requests.

An RTP stream receiver that sent a PAUSE or RESUME request and receives a REFUSED containing the same PauseID as in the request SHOULD refrain from sending an identical request for some appropriate time to allow the condition that caused REFUSED to clear. For PAUSE, an appropriate time is suggested in Section 8.1. For RESUME, an appropriate time is suggested in Section 8.3.

An RTP stream receiver that sent a PAUSE or RESUME request and receives a REFUSED containing a PauseID different from the request MAY schedule another request using the PauseID from the REFUSED notification.

8.5. Transmission Rules

The transmission of any RTCP feedback messages defined in this specification MUST follow the normal AVPF defined timing rules and depends on the session's mode of operation.

All messages defined in this specification, as well as TMMBR/TMMBN used for pause/resume purposes (Section 5.6), can use either Regular, Early or Immediate timings, but should make a trade-off between timely transmission (Section 4.1) and RTCP bandwidth consumption. This can be achieved by taking the following into consideration:

- o It is recommended that PAUSE use Early or Immediate timing, except for retransmissions where RTCP bandwidth can motivate the use of Regular timing.
- o The first transmission of PAUSED for each (non-wrapped) PauseID is recommended to be sent with Immediate or Early timing, to stop unnecessary repetitions of PAUSE. It is recommended that

subsequent transmissions of PAUSED for that PauseID use Regular timing, to avoid that multiple PAUSE requests cause excessive PAUSED RTCP bandwidth.

- o It is recommended that unsolicited PAUSED (sent when entering Local Paused State (Section 6.4)) always use Immediate or Early timing, until PAUSED for that PauseID is considered delivered at least once to all receivers of the paused RTP stream, to avoid that RTP stream receivers take unnecessary corrective action when the RTP stream is no longer received, after which it is recommended that PAUSE uses Regular timing (as for PAUSED triggered by PAUSE above).
- o RESUME is often time-critical and it is recommended that it always uses Immediate or Early timing.
- o The first transmission of REFUSED for each (non-wrapped) PauseID is recommended to be sent with Immediate or Early timing, to stop unnecessary repetitions of PAUSE or RESUME. It is recommended that subsequent REFUSED for that PauseID use Regular timing, to avoid that multiple unreasonable requests cause excessive REFUSED RTCP bandwidth.

9. Signaling

The capability of handling messages defined in this specification MAY be exchanged at a higher layer such as SDP. This document extends the rtcp-fb attribute defined in section 4 of AVPF [RFC4585] to include the request for pause and resume. This specification follows all the rules defined in AVPF [RFC4585] and CCM [RFC5104] for an rtcp-fb attribute relating to payload type in a session description.

This specification defines a new parameter "pause" to the "ccm" feedback value defined in CCM [RFC5104], representing the capability to understand the RTCP feedback message and all of the defined FCIs of PAUSE, RESUME, PAUSED and REFUSED.

Note: When TMMBR 0 / TMMBN 0 are used to implement pause and resume functionality (with the restrictions described in this specification), signaling rtcp-fb attribute with ccm tmmbr parameter is sufficient and no further signaling is necessary. There is however no guarantee that TMMBR/TMMBN implementations pre-dating this specification work exactly as described here when used with a bitrate value of 0.

The "pause" parameter has two optional attributes, "nowait" and "config":

- o "nowait" indicates that the hold-off period defined in Section 6.2 can be set to 0, reducing the latency before the stream can be paused after receiving a PAUSE request. This condition occurs when there will only be a single receiver per direction in the RTP session, for example in point-to-point sessions. It is also possible to use in scenarios using unidirectional media. The conditions that allow "nowait" to be set (Section 6.2) also indicate that it would be possible to use CCM TMMBR/TMMBN as pause/resume signaling.
- o "config" allows for partial implementation of this specification according to the different roles in the use cases section (Section 3), and takes a value that describes what sub-set is implemented:
 - 1 Full implementation of this specification. This is the default configuration. A missing config attribute MUST be treated equivalent to providing a config value of 1.
 - 2 The implementation intends to send PAUSE and RESUME requests for received RTP streams and is thus also capable of receiving PAUSED and REFUSED. It does not support receiving PAUSE and RESUME requests, but may pause sent RTP streams due to local considerations and then intends to send PAUSED for them.
 - 3 The implementation supports receiving PAUSE and RESUME requests targeted for RTP streams it sends. It will send PAUSED and REFUSED as needed. The node will not send any PAUSE and RESUME requests, but supports and desires receiving PAUSED if received RTP streams are paused.
 - 4 The implementation intends to send PAUSE and RESUME requests for received RTP streams and is thus also capable of receiving PAUSED and REFUSED. It cannot pause any RTP streams it sends, and thus does not support receiving PAUSE and RESUME requests, and also does not support sending PAUSED indications.
 - 5 The implementation supports receiving PAUSE and RESUME requests targeted for RTP streams it sends. It will send PAUSED and REFUSED as needed. It does not support sending PAUSE and RESUME requests to pause received RTP streams, and also does not support receiving PAUSED indications.
 - 6 The implementation supports sent and received RTP streams being paused due to local considerations, and thus supports sending and receiving PAUSED indications.

- 7 The implementation supports and desires to receive PAUSED indications for received RTP streams, but does not pause or send PAUSED indications for sent RTP streams. It does not support any other messages defined in this specification.
- 8 The implementation supports pausing sent RTP streams and sending PAUSED indications for them, but does not support receiving PAUSED indications for received RTP streams. It does not support any other messages defined in this specification.

All implementers of this specification are encouraged to include full support for all messages (config=1), but it is recognized that this is sometimes not meaningful for implementations operating in an environment where only parts of the functionality provided by this specification are needed. The above defined "config" functionality sub-sets provide a trade-off between completeness and the need for implementation interoperability, achieving at least a level of functionality corresponding to what is desired by the least capable party when used as specified here. Implementation of any other functionality sub-sets of this specification than the above defined is NOT RECOMMENDED.

When signaling a config value other than 1, an implementation MUST ignore non-supported messages on reception, and SHOULD omit sending messages not supported by the remote peer. One example where is can be motivated to send messages that some receivers do not support, is when there are multiple message receivers with different message support (different config values). That approach avoids that the least capable receiver limits the functionality provided to others. The below table summarizes per-message send and receive support for the different config attribute values ("X" indicating support and "-" indicating non-support):

#	Send				Receive			
	PAUSE	RESUME	PAUSED	REFUSED	PAUSE	RESUME	PAUSED	REFUSED
1	X	X	X	X	X	X	X	X
2	X	X	X	-	-	-	X	X
3	-	-	X	X	X	X	X	-
4	X	X	-	-	-	-	X	X
5	-	-	X	X	X	X	-	-
6	-	-	X	-	-	-	X	-
7	-	-	-	-	-	-	X	-
8	-	-	X	-	-	-	-	-

Figure 7: Supported messages for different config values

In the above description of partial implementations, config=2 and 4 correspond to the RTP Mixer in the RTP Mixer to Media Sender use case (Section 3.2), and config=3 and 5 correspond to the Media Sender in that same use case. For that use case, it should be clear that an RTP Mixer implementing only config=3 or 5 will not provide a working solution. Similarly, for that use case, a Media Sender implementing only config=2 or 4 will not provide a working solution. Both the RTP Mixer and the Media Sender will of course work when implementing the full set of messages, corresponding to config=1.

A partial implementation is not suitable for pause / resume support between cascaded RTP Mixers, but would require support corresponding to config=1 between such RTP Mixers. This is because an RTP Mixer is then also Media Sender towards the other RTP Mixer, requiring support for the union of config=2 and 3 or config=4 and 5, which effectively becomes config=1.

As can be seen from Figure 7 above, config=2 and 3 differ from config=4 and 5 only in that in the latter, the PAUSE / RESUME message sender (e.g. the RTP Mixer side) does not support local pause (Section 6.4) for any of its own streams and therefore also does not support sending PAUSED.

Partial implementations that only support local pause functionality can declare this capability through config=6-8.

Viable fallback rules between different config are described in Section 9.1 and Figure 9.

This is the resulting ABNF [RFC5234], extending existing ABNF in section 7.1 of CCM [RFC5104]:

```
rtcp-fb-ccm-param  =/ SP "pause" *(SP pause-attr)
pause-attr         = pause-config ; partial message support
                   / "nowait"      ; no hold-off
                   / byte-string   ; for future extensions
pause-config       = "config=" pause-config-value
pause-config-value = 1*2DIGIT
; byte-string as defined in RFC 4566
```

Figure 8: ABNF

An endpoint implementing this specification and using SDP to signal capability SHOULD indicate the new "pause" parameter with ccm signaling, but MAY instead use existing ccm tmmbr signaling [RFC5104] if the limitations in functionality when using TMMBR/TMMBN as described in this specification (Section 5.6) are considered

acceptable. In that case, no partial message support is possible. The messages from this specification (Section 8) SHOULD NOT be used towards receivers that did not declare capability to receive those messages.

The pause functionality can normally be expected to work independently of the payload type. However, there might exist situations where an endpoint needs to restrict or at least configure the capabilities differently depending on the payload type carrying the media stream. Reasons for this might relate to capabilities to correctly handle media boundaries and avoid any pause or resume operation to occur where it would leave a receiver or decoder with no choice than to attempt to repair or discard the media received just prior to or at the point of resuming.

There MUST NOT be more than one "a=rtcp-fb" line with "pause" applicable to a single payload type in the SDP, unless the additional line uses "*" as payload type, in which case "*" SHALL be interpreted as applicable to all listed payload types that do not have an explicit "pause" specification. The "config" pause attribute MUST NOT appear more than once for each "pause" CCM parameter. The "nowait" pause attribute MUST NOT appear more than once for each "pause" CCM parameter.

9.1. Offer-Answer Use

An offerer implementing this specification needs to include "pause" CCM parameter with suitable configuration attribute ("config") in the SDP, according to what messages it intends to send and desires to receive in the session.

In SDP offer/answer, the "config" attribute and its message directions are interpreted based on the agent providing the SDP. The offerer is described in an offer, and the answerer is described in an answer.

An answerer receiving an offer with a "pause" CCM line and a config attribute with a certain value, describing a certain capability to send and receive messages, MAY change the config attribute value in the answer to another configuration. The permitted answers are listed in the below table.

SDP Offer config value	Permitted SDP Answer config values
1	1, 2, 3, 4, 5, 6, 7, 8
2	3, 4, 5, 6, 7, 8
3	2, 4, 5, 6, 7, 8
4	5, 6, 7, 8
5	4, 6, 7, 8
6	6, 7, 8
7	8
8	7

Figure 9: Config values in Offer/Answer

An offer or answer omitting the config attribute, MUST be interpreted as equivalent to config=1. Implementations of this specification MUST NOT use any other config values than the ones defined above in an offer or answer, and MUST remove the "pause" CCM line in the answer when receiving an offer with a config value it does not understand. In all cases the answerer MAY also completely remove any "pause" CCM line to indicate that it does not understand or desire to use any pause functionality for the affected payload types.

If the offerer believes that itself and the intended answerer are likely the only Endpoints in the RTP session, it MAY include the "nowait" attribute on the "pause" line in the offer. If an answerer receives the "nowait" attribute on the "pause" line in the SDP, and if it has information that the offerer and itself are not the only Endpoints in the RTP session, it MUST NOT include any "nowait" attribute on its "pause" line in the SDP answer. The answerer MUST NOT add "nowait" on the "pause" line in the answer unless it is present on the "pause" line in the offer. If both offer and answer contained a "nowait" parameter, then the hold-off period is configured to 0 at both offerer and answerer.

Unknown pause attributes MUST be ignored in the offer and MUST then be omitted from the answer.

If both "pause" and "tmmbr" are present in the offer, both MAY be included also in the answer, in which case TMMBR/TMMBN MUST NOT be used for pause/resume purposes (with a bitrate value of 0), to avoid signaling ambiguity.

9.2. Declarative Use

In declarative use, the SDP is used to configure the node receiving the SDP. This has implications on the interpretation of the SDP signaling extensions defined in this specification.

First, the "config" attribute and its message directions are interpreted based on the node receiving the SDP, and describes the RECOMMENDED level of operation. If the joining client does not support the indicated config value, some RTP session stream optimizations may not be possible in that some RTP streams will not be paused by the joining client, and/or the joining client may not be able to resume and receive wanted streams because they are paused.

Second, the "nowait" parameter, if included, is followed as specified. It is the responsibility of the declarative SDP sender to determine if a configured node will participate in a session that will be point to point, based on the usage. For example, a conference client being configured for an any source multicast session using SAP [RFC2974] will not be in a point to point session, thus "nowait" cannot be included. An RTSP [RFC2326] client receiving a declarative SDP may very well be in a point to point session, although it is highly doubtful that an RTSP client would need to support this specification, considering the inherent PAUSE support in RTSP.

Unknown pause attributes MUST be ignored.

If both "pause" and "tmmbrr" are present in the SDP, TMMBR/TMMBN MUST NOT be used for pause/resume purposes (with a bitrate value of 0), to avoid signaling ambiguity.

10. Examples

The following examples shows use of PAUSE and RESUME messages, including use of offer-answer:

1. Offer-Answer
2. Point-to-Point session
3. Point-to-Multipoint using Mixer
4. Point-to-Multipoint using Relay

10.1. Offer-Answer

The below figures contains an example how to show support for pausing and resuming the streams, as well as indicating whether or not the hold-off period can be set to 0.

```
v=0
o=alice 3203093520 3203093520 IN IP4 alice.example.com
s=Pausing Media
t=0 0
c=IN IP4 alice.example.com
m=audio 49170 RTP/AVPF 98 99
a=rtpmap:98 G719/48000
a=rtpmap:99 PCMA/8000
a=rtcp-fb:* ccm pause nowait
```

Figure 10: SDP Offer With Pause and Resume Capability

The offerer supports all of the messages defined in this specification, leaving out the optional config attribute. The offerer also believes that it will be the sole receiver of the answerer's stream as well as that the answerer will be the sole receiver of the offerer's stream and thus includes the "nowait" sub-parameter for the "pause" parameter.

This is the SDP answer:

```
v=0
o=bob 293847192 293847192 IN IP4 bob.example.com
s=-
t=0 0
c=IN IP4 bob.example.com
m=audio 49202 RTP/AVPF 98
a=rtpmap:98 G719/48000
a=rtcp-fb:98 ccm pause config=2
```

Figure 11: SDP Answer With Pause and Resume Capability

The answerer will not allow its sent streams to be paused or resumed and thus restricts the answer to indicate config=2. It also supports pausing its own RTP streams due to local considerations, which is why config=2 is chosen rather than config=4. The answerer somehow knows that it will not be a point-to-point RTP session and has therefore removed "nowait" from the "pause" line, meaning that the offerer must use a non-zero hold-off period when being requested to pause the stream.

When using TMMBR 0 / TMMBN 0 to achieve pause and resume functionality, there are no differences in SDP compared to CCM [RFC5104] and therefore no such examples are included here.

10.2. Point-to-Point Session

This is the most basic scenario, which involves two participants, each acting as a sender and/or receiver. Any RTP data receiver sends PAUSE or RESUME messages to the sender, which pauses or resumes transmission accordingly. The hold-off period before pausing a stream is 0.

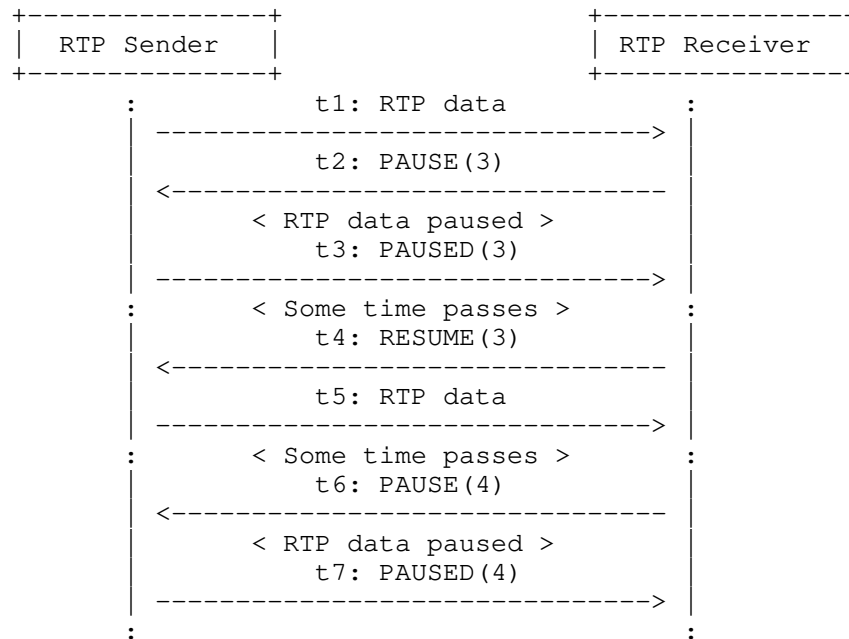


Figure 12: Pause and Resume Operation in Point-to-Point

Figure 12 shows the basic pause and resume operation in Point-to-Point scenario. At time t1, an RTP sender sends data to a receiver. At time t2, the RTP receiver requests the sender to pause the stream, using PauseID 3 (which it knew since before in this example). The sender pauses the data and replies with a PAUSED containing the same PauseID. Some time later (at time t4) the receiver requests the sender to resume, which resumes its transmission. The next PAUSE, sent at time t6, contains an updated PauseID (4), with a corresponding PAUSED being sent at time t7.

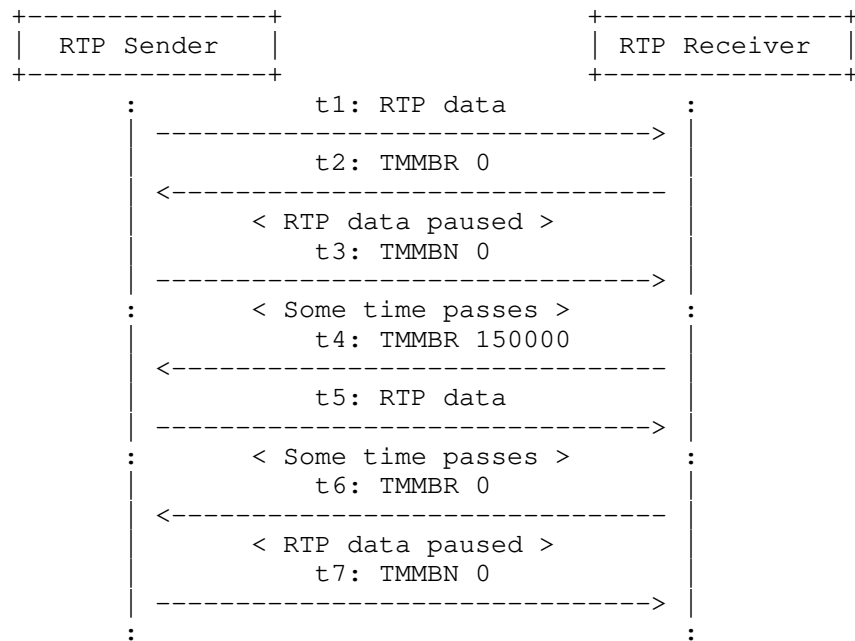


Figure 13: TMMBR Pause and Resume in Point-to-Point

Figure 13 describes the same point-to-point scenario as above, but using TMMBR/TMMBN signaling.

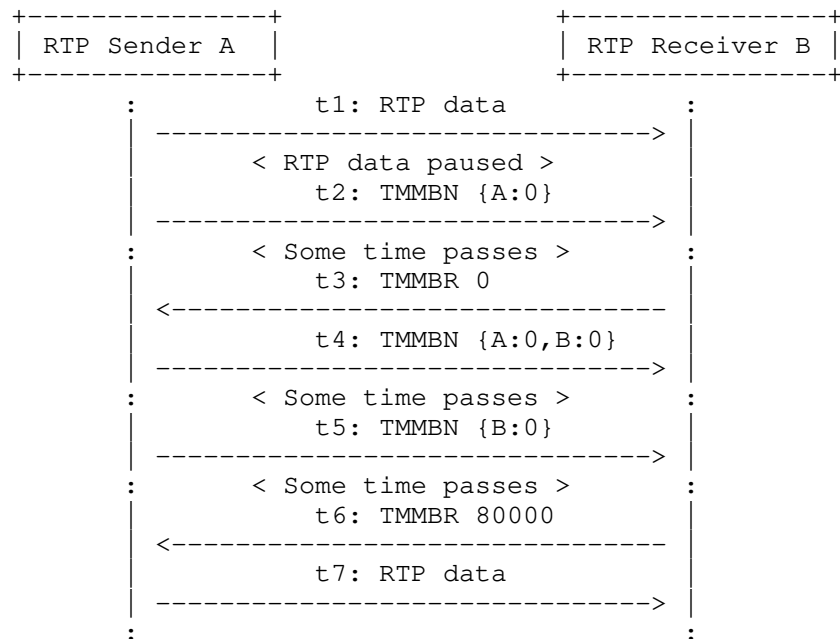


Figure 14: Unsolicited PAUSED using TMMBN

Figure 14 describes the case when an RTP stream sender (A) chooses to pause an RTP stream due to local considerations. Both the RTP stream sender (A) and the RTP stream receiver (B) use TMMBR/TMMBN signaling for pause/resume purposes. A decides to pause the RTP stream at time t2 and uses TMMBN 0 to signal PAUSED, including itself in the TMMBN bounding set. At time t3, despite the fact that the RTP stream is still paused, B decides that it is no longer interested to receive the RTP stream and signals PAUSE by sending a TMMBR 0. As a result of that, the bounding set now contains both A and B, and A sends out a new TMMBN reflecting that. After a while, at time t5, the local considerations that caused A to pause the RTP stream no longer apply, causing it to remove itself from the bounding set and to send a new TMMBN indicating this. At time t6, B decides that it is now interested to receive the RTP stream again and signals RESUME by sending a TMMBR containing a bitrate value greater than 0, causing A to resume sending RTP data.

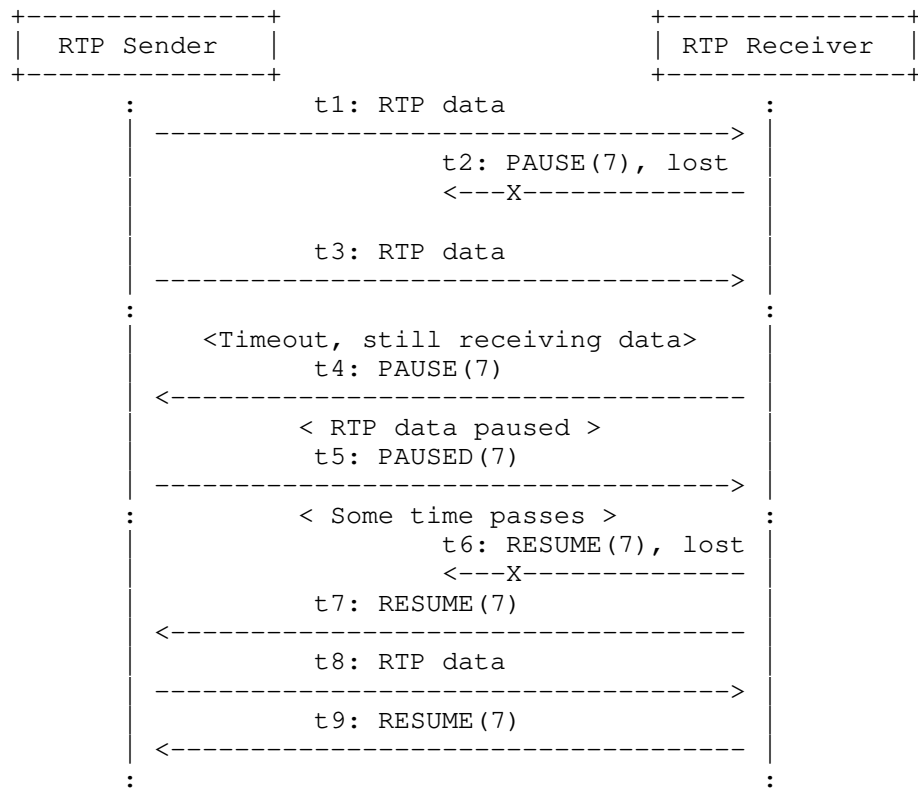


Figure 15: Pause and Resume Operation With Messages Lost

Figure 15 describes what happens if a PAUSE message from an RTP stream receiver does not reach the RTP stream sender. After sending a PAUSE message, the RTP stream receiver waits for a time-out to detect if the RTP stream sender has paused the data transmission or has sent PAUSED indication according to the rules discussed in Section 6.3. As the PAUSE message is lost on the way (at time t2), RTP data continues to reach to the RTP stream receiver. When the timer expires, the RTP stream receiver schedules a retransmission of the PAUSE message, which is sent at time t4. If the PAUSE message now reaches the RTP stream sender, it pauses the RTP stream and replies with PAUSED.

At time t6, the RTP stream receiver wishes to resume the stream again and sends a RESUME, which is lost. This does not cause any severe effect, since there is no requirement to wait until further RESUME are sent and another RESUME are sent already at time t7, which now reaches the RTP stream sender that consequently resumes the stream at

time t8. The time interval between t6 and t7 can vary, but may for example be one RTCP feedback transmission interval as determined by the AVPF rules.

The RTP stream receiver did not realize that the RTP stream was resumed in time to stop yet another scheduled RESUME from being sent at time t9. This is however harmless since RESUME contains a past PauseID and will be ignored by the RTP stream sender. It will also not cause any unwanted resume even if the stream was paused again based on a PAUSE from some other receiver before receiving the RESUME, since the current PauseID was updated compared to the one in the stray RESUME, which contains a past PauseID and will be ignored by the RTP stream sender.

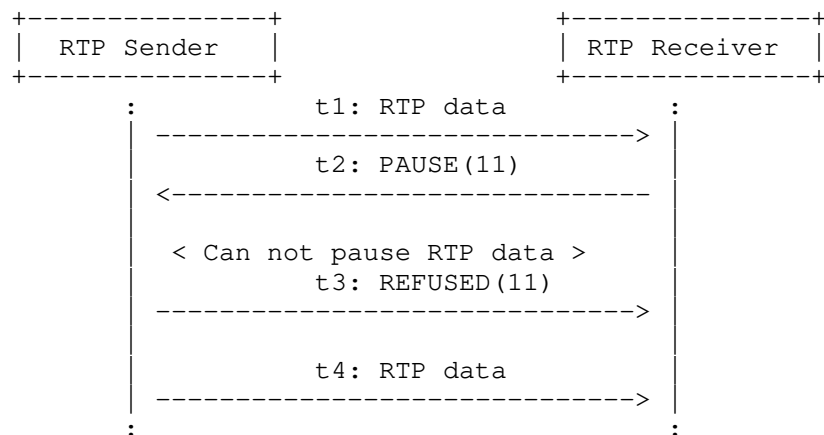


Figure 16: Pause Request is Refused in Point-to-Point

In Figure 16, the receiver requests to pause the sender, which refuses to pause due to some consideration local to the sender and responds with a REFUSED message.

10.3. Point-to-Multipoint using Mixer

An RTP Mixer is an intermediate node connecting different transport-level clouds. The Mixer receives streams from different RTP sources, selects or combines them based on the application's needs and forwards the generated stream(s) to the destination. The Mixer typically puts its' own SSRC(s) in RTP data packets instead of the original source(s).

The Mixer keeps track of all the streams delivered to the Mixer and how they are currently used. In this example, it selects the video

stream to deliver to the receiver R based on the voice activity of the RTP stream senders. The video stream will be delivered to R using M's SSRC and with an CSRC indicating the original source.

Note that PauseID is not of any significance for the example and is therefore omitted in the description.

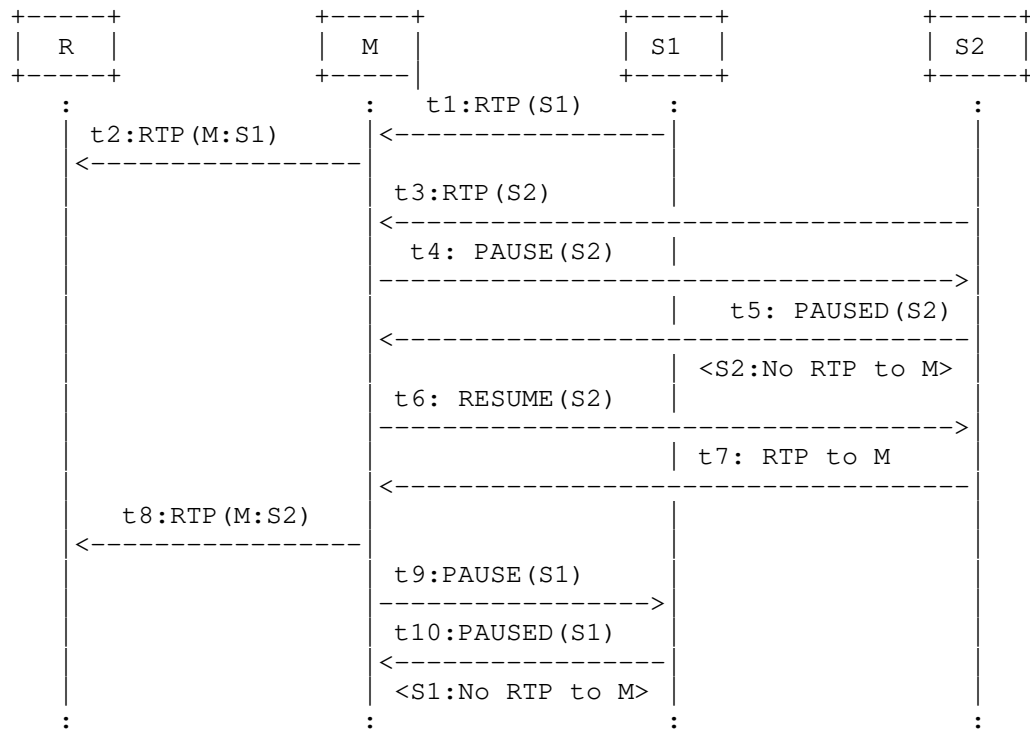


Figure 17: Pause and Resume Operation for a Voice Activated Mixer

The session starts at t1 with S1 being the most active speaker and thus being selected as the single video stream to be delivered to R (t2) using the Mixer SSRC but with S1 as CSRC (indicated after the colon in the figure). Then S2 joins the session at t3 and starts delivering an RTP stream to the Mixer. As S2 has less voice activity than S1, the Mixer decides to pause S2 at t4 by sending S2 a PAUSE request. At t5, S2 acknowledges with a PAUSED and at the same instant stops delivering RTP to the Mixer. At t6, the user at S2 starts speaking and becomes the most active speaker and the Mixer decides to switch the video stream to S2, and therefore quickly sends a RESUME request to S2. At t7, S2 has received the RESUME request and acts on it by resuming RTP stream delivery to M. When the RTP

stream from t7 arrives at the Mixer, it switches this RTP stream into its SSRC (M) at t8 and changes the CSRC to S2. As S1 now becomes unused, the Mixer issues a PAUSE request to S1 at t9, which is acknowledged at t10 with a PAUSED and the RTP stream from S1 stops being delivered.

10.4. Point-to-Multipoint using Translator

A transport Relay in an RTP session forwards the message from one peer to all the others. Unlike Mixer, the Relay does not mix the streams or change the SSRC of the messages or RTP media. These examples are to show that the messages defined in this specification can be safely used also in a transport Relay case. The parentheses in the figures contains (Target SSRC, PauseID) information for the messages defined in this specification.

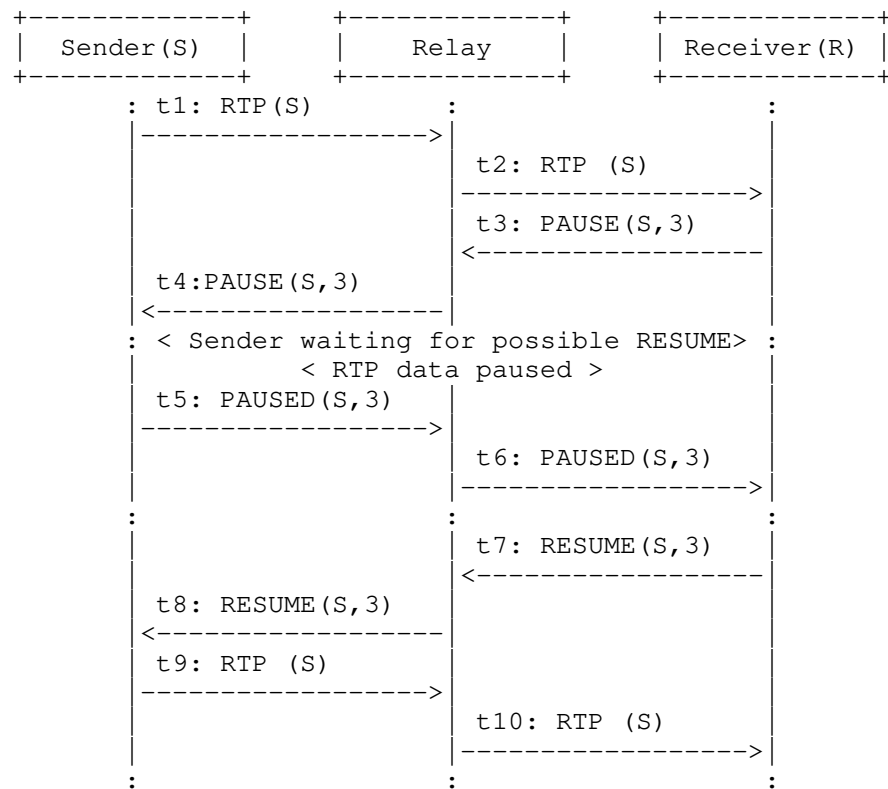


Figure 18: Pause and Resume Operation Between Two Participants Using a Relay

Figure 18 describes how a Relay can help the receiver in pausing and resuming the sender. The sender S sends RTP data to the receiver R through Relay, which just forwards the data without modifying the SSRCs. The receiver sends a PAUSE request to the sender, which in this example knows that there may be more receivers of the stream and waits a non-zero hold-off period to see if there is any other receiver that wants to receive the data, does not receive any disapproving RESUME, hence pauses itself and replies with PAUSED. Similarly the receiver resumes the sender by sending RESUME request through Relay. Since this describes only a single pause and resume operation for a single RTP stream sender, all messages uses a single PauseID, in this example 3.

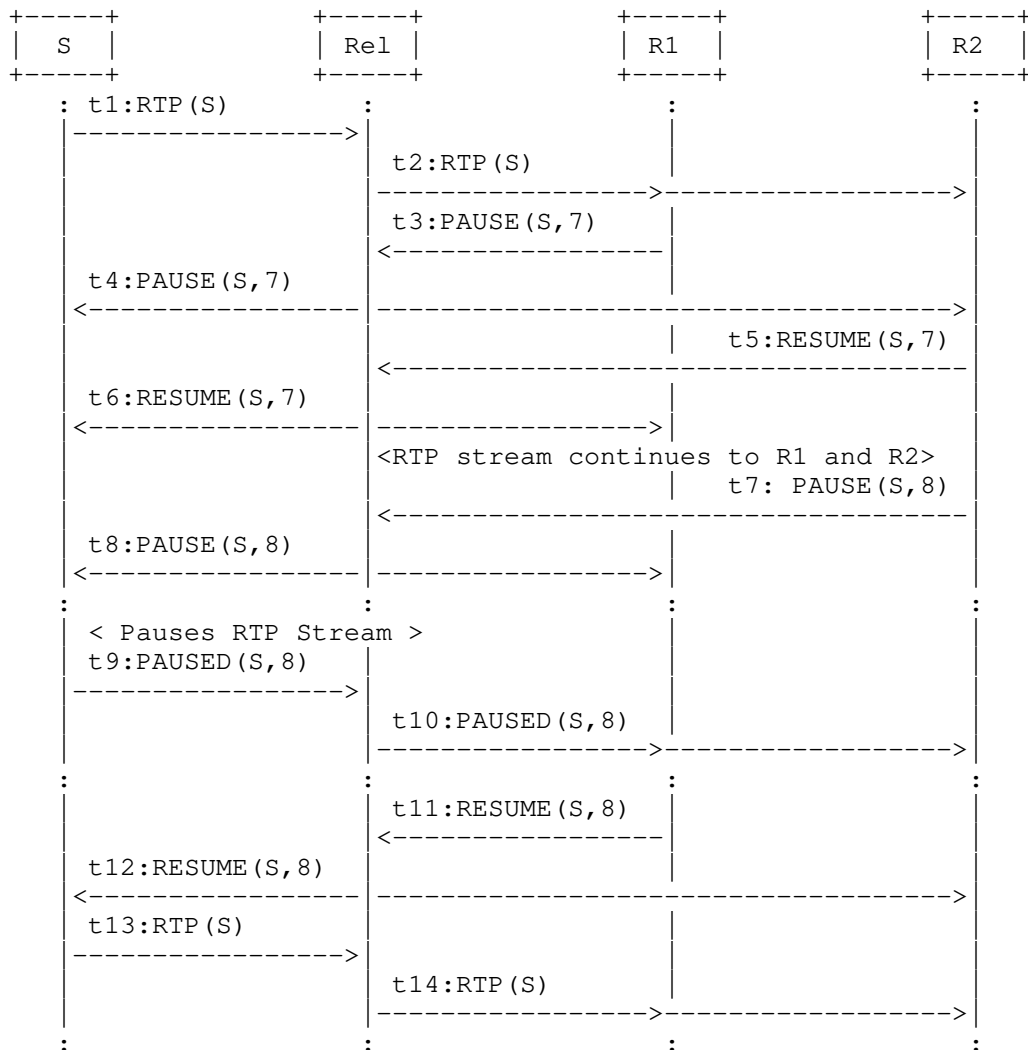


Figure 19: Pause and Resume Operation Between One Sender and Two Receivers Through Relay

Figure 19 explains the pause and resume operations when a transport Relay is involved between a sender and two receivers in an RTP session. Each message exchange is represented by the time it happens. At time t1, Sender (S) starts sending an RTP stream to the Relay, which is forwarded to R1 and R2 through the Relay, Rel. R1 and R2 receives RTP data from Relay at t2. At this point, both R1 and R2

will send RTCP Receiver Reports to S informing that they receive S's stream.

After some time (at t3), R1 chooses to pause the stream. On receiving the PAUSE request from R1 at t4, S knows that there are at least one receiver that may still want to receive the data and uses a non-zero hold-off period to wait for possible RESUME messages. R2 did also receive the PAUSE request at time t4 and since it still wants to receive the stream, it sends a RESUME for it at time t5, which is forwarded to the sender S by the Relay. The sender S sees the RESUME at time t6 and continues to send data to Relay which forwards to both R1 and R2. At t7, the receiver R2 chooses to pause the stream by sending a PAUSE request with an updated PauseID. The sender S still knows that there are more than one receiver (R1 and R2) that may want the stream and again waits a non-zero hold-off period, after which and not having received any disapproving RESUME, it concludes that the stream must be paused. S now stops sending the stream and replies with PAUSED to R1 and R2. When any of the receivers (R1 or R2) chooses to resume the stream from S, in this example R1, it sends a RESUME request to the sender (also seen by R2). The RTP sender immediately resumes the stream.

Consider also an RTP session which includes one or more receivers, paused sender(s), and a Relay. Further assume that a new participant joins the session, which is not aware of the paused sender(s). On receiving knowledge about the newly joined participant, e.g. any RTP traffic or RTCP report (i.e. either SR or RR) from the newly joined participant, the paused sender(s) immediately sends PAUSED indications for the paused streams since there is now a receiver in the session that did not pause the sender(s) and may want to receive the streams. Having this information, the newly joined participant has the same possibility as any other participant to resume the paused streams.

11. IANA Considerations

This specification requests the following registrations from IANA:

1. A new value for media stream pause / resume to be registered with IANA in the "FMT Values for RTPFB Payload Types" registry located at the time of publication at: <http://www.iana.org/assignments/rtp-parameters/rtp-parameters.xhtml#rtp-parameters-8>

Value: TBA1

Name: PAUSE-RESUME

Long Name: Media Pause / Resume

Reference: This RFC

2. A new value "pause" to be registered with IANA in the "Codec Control Messages" registry located at the time of publication at: <http://www.iana.org/assignments/sdp-parameters/sdp-parameters.xhtml#sdp-parameters-19>

Value Name: pause

Long Name: Media Pause / Resume

Usable with: ccm

Reference: This RFC

12. Security Considerations

This document extends the CCM [RFC5104] and defines new messages, i.e. PAUSE, RESUME, PAUSED, and REFUSED. The exchange of these new messages have some security implications, which need to be addressed by the user.

The messages defined in this specification can have substantial impact on the perceived media quality if used in a malicious way. First of all, there is the risk for Denial of Service (DoS) on any RTP session that uses the PAUSE-RESUME functionality. By injecting one or more PAUSE requests into the RTP session, an attacker can potentially prevent any media from flowing, especially when the hold-off period is zero. The injection of PAUSE messages is quite simple, requiring knowledge of the SSRC and the PauseID. This information is visible to an on-path attacker unless RTCP messages are encrypted. Even off-path attacks are possible as signalling messages often carry the SSRC value, while the 16-bit PauseID have to be guessed or tried. The way of protecting the RTP session from these injections is to perform source authentication combined with message integrity, to prevent other than intended session participants from sending these messages. The security solution should provide replay protection. Otherwise, if a session is long-lived enough for the PauseID value to wrap, an attacker could replay old messages at the appropriate time to influence the media sender state. There exist several different choices for securing RTP sessions to prevent this type of attack. SRTP is the most common, but also other methods exist as discussed in "Options for Securing RTP Sessions" [RFC7201].

Most of the methods for securing RTP however do not provide source authentication of each individual participant in a multi-party use case. In case one of the session participants is malicious, it can wreck significant havoc within the RTP session and similarly cause a

DoS on the RTP session from within. That damage can also be attempted to be obfuscated by having the attacker impersonate other endpoints within the session. These attacks can be mitigated by using a solution that provides true source authentication of all participants' RTCP packets. However, that has other implications. For multi-party sessions including a middlebox, that middlebox is RECOMMENDED to perform checks on all forwarded RTCP packets so that each participant only uses its set of SSRCs, to prevent the attacker utilizing another participant's SSRCs. An attacker that can send a PAUSE request that does not reach any other participants than the media sender can cause a stream to be paused without providing opportunity for opposition. This is mitigated in multi-party topologies that ensure that requests are seen by all or most of the RTP session participants, enabling these participants to send RESUME. In topologies with middleboxes that consume and process PAUSE requests, the middlebox can also mitigate such behavior as it will commonly not generate or forward a PAUSE message if it knows of another participant having use for the media stream.

The above text has been focused on using the PAUSE message as the tool for malicious impact on the RTP session. That is because of the greater impact from denying users access to RTP media streams. In contrast, if an attacker attempts to use RESUME in a malicious purpose, it will result in that the media streams are delivered. However, such an attack basically prevents the use the Pause and Resume functionality. Thus, potentially forcing a reduction of the media quality due to limitation in available resources, like bandwidth that must be shared.

The session establishment signalling is also a potential venue of attack, as that can be used to prevent the enabling of Pause and Resume functionality by modifying the signalling messages. The above mitigation of attacks based on source authentication also requires the signalling system to securely handle identities, and assert that only the intended identities are allowed into the RTP session and provided the relevant security contexts.

13. Contributors

Daniel Grondal contributed in the creation and writing of early versions of this specification. Christian Groves contributed significantly to the SDP config attribute and its use in Offer/Answer.

14. Acknowledgements

Daniel Grondal made valuable contributions during the initial versions of this draft. The authors would also like to thank Emil Ivov, Christian Groves, David Mandelberg, Meral Shirazipour, Spencer Dawkins, Bernard Aboba, and Ben Campbell, who provided valuable review comments.

15. References

15.1. 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>>.
- [RFC3264] Rosenberg, J. and H. Schulzrinne, "An Offer/Answer Model with Session Description Protocol (SDP)", RFC 3264, DOI 10.17487/RFC3264, June 2002, <<http://www.rfc-editor.org/info/rfc3264>>.
- [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>>.
- [RFC4566] Handley, M., Jacobson, V., and C. Perkins, "SDP: Session Description Protocol", RFC 4566, DOI 10.17487/RFC4566, July 2006, <<http://www.rfc-editor.org/info/rfc4566>>.
- [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>>.
- [RFC5234] Crocker, D., Ed. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", STD 68, RFC 5234, DOI 10.17487/RFC5234, January 2008, <<http://www.rfc-editor.org/info/rfc5234>>.

- [RFC5245] Rosenberg, J., "Interactive Connectivity Establishment (ICE): A Protocol for Network Address Translator (NAT) Traversal for Offer/Answer Protocols", RFC 5245, DOI 10.17487/RFC5245, April 2010, <<http://www.rfc-editor.org/info/rfc5245>>.
- [RFC6263] Marjou, X. and A. Sollaud, "Application Mechanism for Keeping Alive the NAT Mappings Associated with RTP / RTP Control Protocol (RTCP) Flows", RFC 6263, DOI 10.17487/RFC6263, June 2011, <<http://www.rfc-editor.org/info/rfc6263>>.

15.2. Informative References

- [I-D.ietf-avtc core-rtp-multi-stream-optimisation]
Lennox, J., Westerlund, M., Wu, W., and C. Perkins,
"Sending Multiple Media Streams in a Single RTP Session:
Grouping RTCP Reception Statistics and Other Feedback",
draft-ietf-avtc core-rtp-multi-stream-optimisation-06 (work
in progress), July 2015.
- [I-D.ietf-avtc core-rtp-topologies-update]
Westerlund, M. and S. Wenger, "RTP Topologies", draft-
ietf-avtc core-rtp-topologies-update-10 (work in progress),
July 2015.
- [I-D.ietf-avtext-rtp-grouping-taxonomy]
Lennox, J., Gross, K., Nandakumar, S., Salgueiro, G., and
B. Burman, "A Taxonomy of Semantics and Mechanisms for
Real-Time Transport Protocol (RTP) Sources", draft-ietf-
avtext-rtp-grouping-taxonomy-08 (work in progress), July
2015.
- [I-D.ietf-mmusic-sdp-simulcast]
Burman, B., Westerlund, M., Nandakumar, S., and M. Zanaty,
"Using Simulcast in SDP and RTP Sessions", draft-ietf-
mmusic-sdp-simulcast-01 (work in progress), July 2015.
- [RFC2326] Schulzrinne, H., Rao, A., and R. Lanphier, "Real Time
Streaming Protocol (RTSP)", RFC 2326,
DOI 10.17487/RFC2326, April 1998,
<<http://www.rfc-editor.org/info/rfc2326>>.
- [RFC2974] Handley, M., Perkins, C., and E. Whelan, "Session
Announcement Protocol", RFC 2974, DOI 10.17487/RFC2974,
October 2000, <<http://www.rfc-editor.org/info/rfc2974>>.

- [RFC3261] Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston, A., Peterson, J., Sparks, R., Handley, M., and E. Schooler, "SIP: Session Initiation Protocol", RFC 3261, DOI 10.17487/RFC3261, June 2002, <<http://www.rfc-editor.org/info/rfc3261>>.
- [RFC6190] Wenger, S., Wang, Y., Schierl, T., and A. Eleftheriadis, "RTP Payload Format for Scalable Video Coding", RFC 6190, DOI 10.17487/RFC6190, May 2011, <<http://www.rfc-editor.org/info/rfc6190>>.
- [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>>.
- [RFC7478] Holmberg, C., Hakansson, S., and G. Eriksson, "Web Real-Time Communication Use Cases and Requirements", RFC 7478, DOI 10.17487/RFC7478, March 2015, <<http://www.rfc-editor.org/info/rfc7478>>.

Appendix A. Changes From Earlier Versions

NOTE TO RFC EDITOR: Please remove this section prior to publication.

A.1. Modifications Between Version -09 and -10

Changes based on AD review of changes:

- o Editorial rewordings in Security Consideration

A.2. Modifications Between Version -08 and -09

Changes based on IETF last call and IESG comments.

- o Expanded some acronyms on first usage.
- o Clarified why this document updated RFC 5104.
- o Removed some unused abbreviations.
- o Clarified when TMMBR/TMMBN is expected to be used in Section 5.6.
- o Mandate using SHALL repetition of PAUSED indications in Section 6.4.
- o Removed paragraph in Section 8.4 on PauseID and REFUSED as being redundant.

- o Transmission rules in Section 8.5 was reworded and stoped using RFC 2119 terms, instead uses recommendations and lists motivations.
- o In Section 9 (Signalling) it was clarified that one SHOULD only send messages supported by receivers, but provide example of cases when one may still send messages not supported by every receiver of that message. Also clarified that TMMBR/TMMBN for pause is monolithic in capability.
- o Security consideration was updated to consider replay attacks.
- o Security consideration was updated to describe the mitigation against malicious RTP session participants in multi-party cases that seeing all messages provide.

A.3. Modifications Between Version -07 and -08

Changes based on IESG AD Evaluation.

- o Moved the mentioning of RTCWEB RFC7478 API requirements out from 3.1 to section 3, adding a couple of clarifying sentences.
- o Highlighted that the use case in section 3.4 deals with a different direction of the pause request than the previous use cases.
- o Added text on partial capability and interoperability to section 5.1.
- o Added an overview explanation of PauseID as a new section 5.2, and moved a few sentences on PauseID from other 5.x sections in there.
- o Changed all occurrences of "available" and "valid" PauseID to the more clear "current" PauseID, and re-phrased sentences involving that to become more clear.
- o Changed all occurrences of "smaller" and "larger" PauseID to "past" and "future", respectively, to better align with "current".
- o Removed an incorrect sentence in 5.2 about when it is not feasible to send repeated PAUSE.
- o Changed a few capitalized words that could be taken as normative text from section 5, which is intended to be a non-normative description.

- o Added some explanatory text on why RTP stream is resumed when the stream receiver that paused the stream leaves the RTP session to last bullet in 6.3.1.
- o Added caption to Figure 5.
- o Moved the detailed description on what PauseID ranges are defined as "past" and "future" before section 8.1, instead of having it in section 8.1, and added a comment on the "not current" part of the value range.
- o Added text in section 8.1 on appropriate time to wait between sending PAUSE, when the first PAUSE was rejected by a RESUME or a REFUSED.
- o Added text in section 8.3 on appropriate time to wait between sending RESUME, when the first RESUME was rejected by a REFUSED.
- o Added text in section 8.4 on time to wait before sending the REFUSED request again, referencing sections 8.1 and 8.3.
- o Added a couple of paragraphs in section 9 on partial capability and interoperability, including a description on when different config values are expected to be useful, and when they are not.
- o Added arrows in Figure 19 to highlight that the Relay sends out all received messages to all receivers, not only the first PAUSE message.
- o Changed references to RFC3264 and RFC4566 to be normative.
- o Updated ietf-rtcweb-use-cases-and-requirements reference to be RFC7478.
- o Editorial improvements and clarifications.

A.4. Modifications Between Version -06 and -07

- o Completely rewrote the Security Consideration section.
- o Aligned text such that REFUSED is always referred to as a notification, not indication.
- o Added and changed text in several places, clarifying the case when TMMBR/TMMBN bounding set overhead value matters, related to whether local RTP stream sender or remote RTP stream receiver owns the TMMBR 0 restriction, and the consequences this has on pause/resume logic.

- o Moved text on when to stop media stream transmission from when receiving PAUSE and entering pausing state, to when entering paused or local paused states.
- o Added text on how to determine if there is a single receiver or not, aligned with what is specified in draft-ietf-avtcore-multi-stream, adding a reference to draft-ietf-avtcore-multi-stream-optimisation to be able to use a single RTCP reporting group as one criteria.
- o Added clarifying text on repeating PAUSED and RESUME messages only as long as remaining in the relevant state.
- o Clarified that it is the RTP stream sender's responsibility to leave local paused state when the condition causing that state is no longer true.
- o Added text to better allow for extensions to this specification, since there is already some text on extensions.
- o Corrected and amended ABNF to make CCM pause parameters order-independent, allow for a larger config pause attribute value range, and added corresponding text to handle that additional flexibility.
- o Added SDP rules on how to handle unknown pause attribute values.
- o Clarified how to handle an SDP with both "ccm pause" and "ccm tmmbr".
- o Changed from "Translator" to "Relay" in examples, to make it clearer in relation to the updated topologies draft.
- o Editorial improvements.

A.5. Modifications Between Version -05 and -06

- o Clarified in Message Details section for PAUSED that retransmission of the message can be used to increase the probability that the message reaches the receiver in a timely fashion, and also added text that says the number of repetitions can be tuned to observed loss rate and the desired loss probability. Also removed Editor's notes on potential ACK for unsolicited PAUSED, since the issue is solved by the above.
- o In the same section as above, added that PAUSED may be included in all compound RTCP reports, as long as the negotiated RTCP bandwidth is not exceeded.

- o In Message Details section for RESUME, added text on retransmission similar to the one mentioned for PAUSED above. Also included text that says such retransmission SHOULD stop as soon as RTP packets or a REFUSED with a valid PauseID from the targeted stream are received.
- o Changed simulcast reference, since that draft was moved from AVTCORE to MMUSIC and made WG draft.
- o Changed End Point to Endpoint to reflect change in RTP Grouping Taxonomy draft.
- o Editorial improvements.

A.6. Modifications Between Version -04 and -05

- o Added text in sections 4.1, 4.6, 6.4 and 8.5 on retransmission and timing of unsolicited PAUSED, to improve the message timeliness and probability of reception.

A.7. Modifications Between Version -03 and -04

- o Change of Copyright boilerplate

A.8. Modifications Between Version -02 and -03

- o Changed the section on SDP signaling to be more explicit and clear in what is supported, replacing the 'paused' parameter and the 'dir' attribute with a 'config' parameter that can take a value, and an explicit listing of what each value means.
- o Added a sentence in section on paused state (Section 6.3) that pause must not affect RTP keepalive.
- o Replaced REFUSE message name with REFUSED throughout, to better indicate that it is not a command but a notification.
- o Added text in a few places, clarifying that PAUSED message may be used unsolicited due to RTP sender local considerations, and also clarified the interaction between this usage and an RTP stream receiver pausing the stream. Also added an example describing this case.
- o Clarified that when TMMBN 0 is used as PAUSED message, and when sent unsolicited due to RTP sender local considerations, the TMMBN message includes the RTP stream sender itself as part of the bounding set.

- o Clarified that there is no reply to a PAUSED indication.
- o Improved the IANA section.
- o Editorial improvements.

A.9. Modifications Between Version -01 and -02

- o Replaced most text on relation with other signaling technologies in previous section 5 with a single, summarizing paragraph, as discussed at IETF 90 in Toronto, and placed it as the last subsection of section 4 (design considerations).
- o Removed unused references.

A.10. Modifications Between Version -00 and -01

- o Corrected text in section 6.5 and 6.2 to indicate that a PAUSE signaled via TMMBR 0 cannot be REFUSED using TMMBN > 0
- o Improved alignment with RTP Taxonomy draft, including the change of Packet Stream to RTP Stream
- o Editorial improvements

Authors' Addresses

Bo Burman
Ericsson
Kistavagen 25
SE - 164 80 Kista
Sweden

Email: bo.burman@ericsson.com

Azam Akram
Ericsson
Farogatan 6
SE - 164 80 Kista
Sweden

Phone: +46107142658
Email: muhammad.azam.akram@ericsson.com
URI: www.ericsson.com

Roni Even
Huawei Technologies
Tel Aviv
Israel

Email: roni.even@mail01.huawei.com

Magnus Westerlund
Ericsson
Farogatan 6
SE- 164 80 Kista
Sweden

Phone: +46107148287
Email: magnus.westerlund@ericsson.com

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: January 7, 2016

M. Westerlund
B. Burman
Ericsson
R. Even
Huawei Technologies
M. Zanaty
Cisco Systems
July 6, 2015

RTP Header Extension for RTCP Source Description Items
draft-ietf-avtext-sdes-hdr-ext-02

Abstract

Source Description (SDS) items are normally transported in RTP control protocol (RTCP). In some cases it can be beneficial to speed up the delivery of these items. Mainly when a new source (SSRC) joins an RTP session and the receivers needs this source's identity, relation to other sources, or its synchronization context, all of which may be fully or partially identified using SDS items. To enable this optimization, this document specifies a new RTP header extension that can carry SDS items.

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 7, 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. Definitions	3
2.1. Requirements Language	3
2.2. Terminology	3
3. Motivation	4
4. Specification	5
4.1. SDES Item Header Extension	5
4.1.1. One-Byte Format	5
4.1.2. Two-Byte Format	6
4.2. Usage of the SDES Item Header Extension	6
4.2.1. One or Two Byte Headers	6
4.2.2. MTU and Packet Expansion	7
4.2.3. Transmission Considerations	7
4.2.4. Different Usages	9
4.2.5. SDES Items in RTCP	9
4.2.6. Update Flaps	10
5. IANA Considerations	10
5.1. Reservation of the SDES URN sub-space	11
5.2. Registration of SDES Items	11
6. Security Considerations	12
7. Acknowledgements	12
8. References	12
8.1. Normative References	12
8.2. Informative References	13
Authors' Addresses	14

1. Introduction

This specification defines an RTP header extension [RFC3550][RFC5285] that can carry RTCP source description (SDES) items. By including selected SDES items in a header extension the determination of relationship and synchronization context for new RTP streams (SSRCs) in an RTP session can be optimized. Which relationship and what information depends on the SDES items carried. This becomes a complement to using only RTCP for SDES Item delivery.

It is important to note that not all SDES items are appropriate to transmit using RTP header extensions. Some SDES items performs

binding or identifies synchronization context with strict timeliness requirements, while many other SDES items do not have such requirements. In addition, security and privacy concerns for the SDES item information need to be considered. For example, the Name and Location SDES items are highly sensitive from a privacy perspective and should not be transported over the network without strong security. No use case has identified where this information is required at the same time as the first RTP packets arrive. A few seconds delay before such information is available to the receiver appears acceptable. Therefore only appropriate SDES items will be registered for use with this header extension, such as CNAME.

First, some requirements language and terminology are defined. The following section motivates why this header extension is sometimes required or at least provides a significant improvement compared to waiting for regular RTCP packet transmissions of the information. This is followed by a specification of the header extension and usage recommendations. Next, a sub-space of the header-extension URN is defined to be used for existing and future SDES items, and then the appropriate existing SDES items are registered.

2. Definitions

2.1. 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].

2.2. Terminology

This document uses terminology defined in "A Taxonomy of Grouping Semantics and Mechanisms for Real-Time Transport Protocol (RTP) Sources" [I-D.ietf-avtext-rtp-grouping-taxonomy]. In particular the following definitions:

Media Source

RTP Stream

Media Encoder

Encoded Stream

Participant

3. Motivation

Source Description (SDES) items are associated with a particular SSRC and thus RTP stream. The source description items provide various meta data associated with the SSRC. How important it is to have this data no later than when receiving the first RTP packets depends on the item itself. The CNAME item is one item that is commonly needed either at reception of the first RTP packet for this SSRC, or at least by the time the first media can be played out. If it is not available, the synchronization context cannot be determined and thus any related streams cannot be correctly synchronized. Thus, this is a valuable example for having this information early when a new RTP stream is received.

The main reason for new SSRCs in an RTP session is when media sources are added. This can be either because an end-point is adding a new actual media source, or additional participants in a multi-party session are added to the session. Another reason for a new SSRC can be an SSRC collision that forces both colliding parties to select new SSRCs.

For the case of rapid media synchronization, one may use the RTP header extension for Rapid Synchronization of RTP Flows [RFC6051]. This header extension carries the clock information present in the RTCP sender report (SR) packets. It however assumes that the CNAME binding is known, which can be provided via signaling [RFC5576] in some cases, but not all. Thus an RTP header extension for carrying SDES items like CNAME is a powerful combination to enable rapid synchronization in all cases.

The Rapid Synchronization of RTP Flows specification does provide an analysis of the initial synchronization delay for different sessions depending on number of receivers as well as on session bandwidth (Section 2.1 of [RFC6051]). These results are applicable also for other SDES items that have a similar time dependency until the information can be sent using RTCP. These figures can be used to determine the benefit of reducing the initial delay before information is available for some use cases.

That document also discusses the case of late joiners, and defines an RTCP Feedback format to request synchronization information, which is another potential use case for SDES items in RTP header extension. It would for example be natural to include CNAME SDES item with the header extension containing the NTP formatted reference clock to ensure synchronization.

There is an another SDES item that can benefit from timely delivery, and an RTP header extension SDES item was therefore defined for it:

MID: This is a media description identifier that matches the value of the Session Description Protocol (SDP) [RFC4566] a=mid attribute, to associate RTP streams multiplexed on the same transport with their respective SDP media description as described in [I-D.ietf-mmusic-sdp-bundle-negotiation].

4. Specification

This section first specifies the SDES item RTP header extension format, followed by some usage considerations.

4.1. SDES Item Header Extension

An RTP header extension scheme allowing for multiple extensions is defined in "A General Mechanism for RTP Header Extensions" [RFC5285]. That specification defines both short and long item headers. The short headers (One-byte) are restricted to 1 to 16 bytes of data, while the long format (Two-byte) supports a data length of 0 to 255 bytes. Thus the RTP header extension formats are capable of supporting any SDES item from a data length perspective.

The ID field, independent of short or long format, identifies both the type of RTP header extension and, in the case of the SDP item header extension, the type of SDP item. The mapping is done in signaling by identifying the header extension and SDP item type using a URN, which is defined in the IANA consideration (Section 5) for the known SDP items appropriate to use.

4.1.1. One-Byte Format

The one-byte header format for an SDES item extension element consists of the One-Byte header (defined in Section 4.2 of [RFC5285]), which consists of a 4-bit ID followed by a 4-bit length field (len) that identifies the number of data bytes (len value +1) following the header. The data part consists of len+1 bytes of UTF-8 text. The type of text and its mapping to the SDES item type is determined by the ID field value.

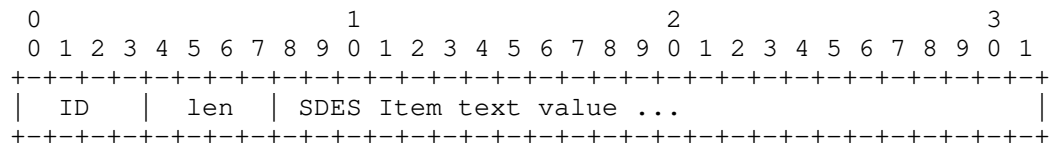


Figure 1

4.1.2. Two-Byte Format

The two-byte header format for an SDES item extension element consists of the two-byte header (defined in Section 4.3 of [RFC5285]), which consists of an 8-bit ID followed by an 8-bit length field (len) that identifies the number of data bytes following the header. The data part consists of len bytes of UTF-8 text. The type of text and its mapping to the SDES item type is determined by the ID field value.

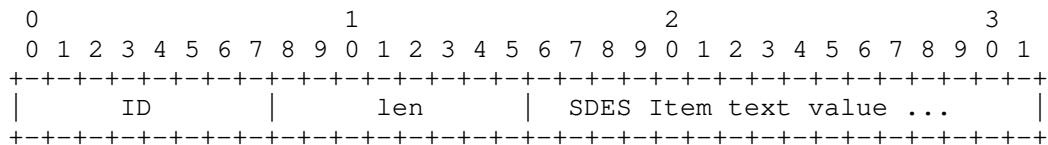


Figure 2

4.2. Usage of the SDES Item Header Extension

This section discusses various usage considerations; which form of header extension to use, the packet expansion, and when to send SDES items in header extension.

4.2.1. One or Two Byte Headers

The RTP header extensions for SDES items MAY use either the one-byte or two-byte header formats, depending on the text value size for the used SDES items and the requirement from any other header extensions used. The one-byte header SHOULD be used when all non SDES item header extensions supports the one-byte format and all SDES item text values contain at most 16 bytes. Note that the RTP header extension specification does not allow mixing one-byte and two-byte headers for the same RTP stream (SSRC), so if the value size of any of the SDES items value requires the two-byte header, then all other header extensions MUST also use the two-byte header format.

For example using CNAMEs that are generated according to "Guidelines for Choosing RTP Control Protocol (RTCP) Canonical Names (CNAMEs)" [RFC7022], using short term persistent values, and if 96-bit random values prior to base64 encoding are sufficient, then they will fit into the One-Byte header format.

An RTP middlebox needs to take care choosing between one byte headers to two-byte headers when creating the first packets for an outgoing stream (SSRC) with header extensions. First of all it needs to consider all the header extensions that may potentially be used. Secondly, it needs to know the size of the SDES items that are going

to be included, and use two bytes headers if any are longer than 16 bytes. An RTP middlebox that forwards a stream, i.e. not mixing it or combing it with other streams, may be able to base its choice on the header size in incoming streams. This is assuming that the middlebox does not modify the stream or add additional header extensions to the stream it sends, in which case it needs to make its own decision.

4.2.2. MTU and Packet Expansion

The RTP packet size will clearly increase when a header extension is included. How much depends on the type of header extensions and their data content. The SDES items can vary in size. There are also some use-cases that require transmitting multiple SDES items in the same packet to ensure that all relevant data reaches the receiver. An example of that is when both CNAME, a MID, and the rapid time synchronization extension from RFC 6051 are needed. Such a combination is quite likely to result in at least 16+3+8 bytes of data plus the headers, which will be another 7 bytes for one-byte headers, plus two bytes of header padding to make the complete header extension word aligned, thus in total 36 bytes.

If the packet expansion cannot be taken into account when producing the RTP payload it can cause an issue. An RTP payload that is created to meet a particular IP level Maximum Transmission Unit (MTU), taking the addition of IP/UDP/RTP headers but not RTP header extensions into account, could exceed the MTU when the header extensions are present, thus resulting in IP fragmentation. IP fragmentation is known to negatively impact the loss rate due to middleboxes unwilling or not capable of dealing with IP fragments, as well as increasing the target surface for other types of packet losses.

As this is a real issue, the media encoder and payload packetizer should be flexible and be capable of handling dynamically varying payload size restrictions to counter the packet expansion caused by header extensions. If that is not possible, some reasonable worst case packet expansion should be calculated and used to reduce the RTP payload size of all RTP packets the sender transmits.

4.2.3. Transmission Considerations

The general recommendation is to only send header extensions when needed. This is especially true for SDES items that can be sent in periodic repetitions of RTCP throughout the whole session. Thus, the different usages (Section 4.2.4) have different recommendations. First some general considerations for getting the header extensions delivered to the receiver:

1. The probability for packet loss and burst loss determine how many repetitions of the header extensions will be required to reach a targeted delivery probability, and if burst loss is likely, what distribution would be needed to avoid getting all repetitions of the header extensions lost in a single burst.
2. If a set of packets are all needed to enable decoding, there is commonly no reason for including the header extension in all of these packets, as they share fate. Instead, at most one instance of the header extension per independently decodable set of media data would be a more efficient use of the bandwidth.
3. How early the SDES item information is needed, from the first received RTP data or only after some set of packets are received, can guide if the header extension(s) should be in all of the first N packets or be included only once per set of packets, for example once per video frame.
4. The use of RTP level robustness mechanisms, such as RTP retransmission [RFC4588], or Forward Error Correction, e.g., [RFC5109] may treat packets differently from a robustness perspective, and SDES header extensions should be added to packets that get a treatment corresponding to the relative importance of receiving the information.

As a summary, the number of header extension transmissions should be tailored to a desired probability of delivery taking the receiver population size into account. For the very basic case, N repetitions of the header extensions should be sufficient, but may not be optimal. N is selected so that the header extension target delivery probability reaches $1-P^N$, where P is the probability of packet loss. For point to point or small receiver populations, it might also be possible to use feedback, such as RTCP, to determine when the information in the header extensions has reached all receivers and stop further repetitions. Feedback that can be used includes the RTCP XR Loss RLE report block [RFC3611], which will indicate successful delivery of particular packets. If the RTP/AVPF Transport Layer Feedback Messages for generic NACK [RFC4585] is used, it can indicate the failure to deliver an RTP packet with the header extension, thus indicating the need for further repetitions. The normal RTCP report blocks can also provide an indicator of successful delivery, if no losses are indicated for a reporting interval covering the RTP packets with the header extension. Note that loss of an RTCP packet reporting on an interval where RTP header extension packets were sent, does not necessarily mean that the RTP header extension packets themselves were lost.

4.2.4. Different Usages

4.2.4.1. New SSRC

A new SSRC joins an RTP session. As this SSRC is completely new for everyone, the goal is to ensure, with high probability, that all RTP session participants receives the information in the header extension. Thus, header extension transmission strategies that allow some margins in the delivery probability should be considered.

4.2.4.2. Late Joiner

In a multi-party RTP session where one or a small number of receivers join a session where the majority of receivers already have all necessary information, the use of header extensions to deliver relevant information should be tailored to reach the new receivers. The trigger to send header extensions can for example either be RTCP from new receiver(s) or an explicit request like the Rapid Resynchronization Request defined in [RFC6051]. In centralized topologies where an RTP middlebox is present, it can be responsible for transmitting the known information, possibly stored, to the new session participant only, and not repeat it to all the session participants.

4.2.4.3. Information Change

If the SDES information is tightly coupled with the RTP data, and the SDES information needs to be updated, then the use of the RTP header extension is superior to RTCP. Using the RTP header extension ensures that the information is updated on reception of the related RTP media, ensuring synchronization between the two. Continued use of the old SDES information can lead to undesired effects in the application. Thus, header extension transmission strategies with high probability of delivery should be chosen.

4.2.5. SDES Items in RTCP

The RTP header extensions information, i.e. SDES Items, can and will be sent also in RTCP. Therefore, it is worth making some reflections on this interaction. As an alternative to the header extension, it is possible to schedule a non-regular RTCP packet transmission containing important SDES items, if one uses an RTP/AVPF based RTP profile. Depending on which mode one's RTCP feedback transmitter is working on, extra RTCP packets may be sent as immediate or early packets, enabling more timely SDES information delivery.

There are however two aspects that differ between using RTP header extensions and any non-regular transmission of RTCP packets. First,

as the RTCP packet is a separate packet, there is no direct relation and also no fate sharing between the relevant media data and the SDES information. The order of arrival for the packets will matter. With a header-extension, the SDES items can be ensured to arrive if the media data to play out arrives. Secondly, it is difficult to determine if an RTCP packet is actually delivered. This, as the RTCP packets lack both sequence number or a mechanism providing feedback on the RTCP packets themselves.

4.2.6. Update Flaps

The SDES item may arrive both in RTCP and in RTP header extensions, potentially causing the value to flap back and forth at the time of updating. There are at least two reasons for these flaps. The first one is packet reordering, where a pre-update RTP or RTCP packet with an SDES item is delivered to the receiver after the first RTP/RTCP packet with the updated value. The second reason is the different code-paths for RTP and RTCP in implementations. An update to the sender's SDES item parameter can take a different time to propagate to the receiver than the corresponding media data. For example, an RTCP packet with the SDES item included that may have been generated prior to the update can still reside in a buffer and be sent unmodified. The update of the item's value can at the same time cause RTP packets to be sent including the header extension, prior to the RTCP packet being sent.

However, most of these issues can be avoided by performing some checks before updating the receiver's stored value. To handle flaps caused by reordering, only SDES items received in RTP packets with a higher extended sequence number than the last change shall be applied, i.e. discard items that can be determined to be older than the current one. For compound RTCP packets, which will contain an Sender Report (SR) packet (assuming an active RTP sender), the receiver can compare the RTCP Sender Report's Timestamp field, to determine at what approximate time it was transmitted. If the timestamp is earlier than the last received RTP packet extension carrying an SDES item, and especially if carrying a previously used value, the SDES item in the RTCP SDES packet can be ignored. Note that media processing and transmission pacing can easily cause the RTP header timestamp field as well as the RTCP SR timestamp field to not match with the actual transmission time.

5. IANA Considerations

This section makes the following requests to IANA:

- o Register and reserve for SDES items the URN sub-space "urn:ietf:params:rtp-hdext:sdes:" in the RTP Compact Header Extensions registry.
- o Register the SDES items appropriate for use with the RTP header extension defined in this document.

5.1. Reservation of the SDES URN sub-space

The reason to require registering a URN within an SDES sub-space is that the name represents an RTCP Source Description item, where a specification is strongly recommended. The formal policy is maintained from the main space, i.e. Expert Review. However, some additional considerations are provided here that needs to be considered when applying for a registration within this sub-space of the RTP Compact Header Extensions registry.

Any registration using an Extension URI that starts with "urn:ietf:params:rtp-hdext:sdes:" MUST also have a registered Source Description item in the "RTP SDES item types" registry. Secondly, a security and privacy consideration for the SDES item must be provided with the registration, preferably in a publicly available reference. Thirdly, information must be provided on why this SDES item requires timely delivery, motivating it to be transported in a header extension rather than as RTCP only.

IANA is requested to register the below in the RTP Compact Header Extensions:

Extension URI: urn:ietf:params:rtp-hdext:sdes
Description: Reserved as base URN for SDES items that are also defined as RTP Compact header extensions.
Contact: Authors of [RFCXXXX]
Reference: [RFCXXXX]

RFC-editor note: Please replace all occurrences of RFCXXXX with the RFC number this specification receives when published.

5.2. Registration of SDES Items

It is requested that the following SDES item is registered in the RTP Compact Header Extensions registry:

Extension URI: urn:ietf:params:rtp-hdext:sdes:cname
Description: Source Description: Canonical End-Point Identifier (SDES CNAME)
Contact: Authors of [RFCXXXX]
Reference: [RFCXXXX]

We also note that the MID SDES item is already registered in the registry by [I-D.ietf-mmusic-sdp-bundle-negotiation].

6. Security Considerations

Source Description items may contain data that are sensitive from a security perspective. There are SDES items that are or may be sensitive from a user privacy perspective, like CNAME, NAME, EMAIL, PHONE, LOC and H323-CADDR. Some may contain sensitive information, like NOTE and PRIV, while others may be sensitive from profiling implementations for vulnerability or other reasons, like TOOL. The CNAME sensitivity can vary depending on how it is generated and what persistence it has. A short term CNAME identifier generated using a random number generator [RFC7022] may have minimal security implications, while a CNAME of the form user@host has privacy concerns, and a CNAME generated from a MAC address has long term tracking potentials.

In RTP sessions where any type of confidentiality protection is enabled for RTCP, the SDES item header extensions MUST also be protected per default. This implies that to provide confidentiality, users of SRTP need to implement encrypted header extensions per [RFC6904]. Commonly, it is expected that the same security level is applied to RTCP packets carrying SDES items, and to an RTP header extension containing SDES items. If the security level is different, it is important to consider the security properties as the worst in each aspect for the different configurations.

As the SDES items are used by the RTP based application to establish relationships between RTP streams or between an RTP stream and information about the originating Participant, there SHOULD be strong requirements on integrity and source authentication of the header extensions. If not, an attacker can modify the SDES item value to create erroneous relationship bindings in the receiving application.

7. Acknowledgements

The authors likes to thanks the following individuals for feedback and suggestions; Colin Perkins.

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, July 2008.
- [RFC6904] Lennox, J., "Encryption of Header Extensions in the Secure Real-time Transport Protocol (SRTP)", RFC 6904, April 2013.

8.2. Informative References

- [I-D.ietf-avtext-rtp-grouping-taxonomy]
Lennox, J., Gross, K., Nandakumar, S., Salgueiro, G., and B. Burman, "A Taxonomy of Semantics and Mechanisms for Real-Time Transport Protocol (RTP) Sources", draft-ietf-avtext-rtp-grouping-taxonomy-07 (work in progress), June 2015.
- [I-D.ietf-mmusic-sdp-bundle-negotiation]
Holmberg, C., Alvestrand, H., and C. Jennings, "Negotiating Media Multiplexing Using the Session Description Protocol (SDP)", draft-ietf-mmusic-sdp-bundle-negotiation-22 (work in progress), June 2015.
- [RFC3611] Friedman, T., Caceres, R., and A. Clark, "RTP Control Protocol Extended Reports (RTCP XR)", RFC 3611, November 2003.
- [RFC4566] Handley, M., Jacobson, V., and C. Perkins, "SDP: Session Description Protocol", RFC 4566, July 2006.
- [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, July 2006.
- [RFC4588] Rey, J., Leon, D., Miyazaki, A., Varsa, V., and R. Hakenberg, "RTP Retransmission Payload Format", RFC 4588, July 2006.
- [RFC5109] Li, A., "RTP Payload Format for Generic Forward Error Correction", RFC 5109, December 2007.
- [RFC5576] Lennox, J., Ott, J., and T. Schierl, "Source-Specific Media Attributes in the Session Description Protocol (SDP)", RFC 5576, June 2009.

[RFC6051] Perkins, C. and T. Schierl, "Rapid Synchronisation of RTP Flows", RFC 6051, November 2010.

[RFC7022] Begen, A., Perkins, C., Wing, D., and E. Rescorla, "Guidelines for Choosing RTP Control Protocol (RTCP) Canonical Names (CNAMEs)", RFC 7022, September 2013.

Authors' Addresses

Magnus Westerlund
Ericsson
Farogatan 6
SE-164 80 Stockholm
Sweden

Phone: +46 10 714 82 87
Email: magnus.westerlund@ericsson.com

Bo Burman
Ericsson
Kistavagen 25
Stockholm 16480
Sweden

Email: bo.burman@ericsson.com

Roni Even
Huawei Technologies
Tel Aviv
Israel

Email: roni.even@mail01.huawei.com

Mo Zanaty
Cisco Systems
7100 Kit Creek
RTP, NC 27709
USA

Email: mzanaty@cisco.com

AVTEXT Working Group
INTERNET-DRAFT
Intended Status: Standards Track
Expires: October 31, 2015

J. Xia
R. Even
R. Huang
Huawei
L. Deng
China Mobile
April 29, 2015

RTP/RTCP extension for RTP Splicing Notification
draft-ietf-avtext-splicing-notification-02

Abstract

Content splicing is a process that replaces the content of a main multimedia stream with other multimedia content, and delivers the substitutive multimedia content to the receivers for a period of time. The splicer is designed to handle RTP splicing and needs to know when to start and end the splicing.

This memo defines two RTP/RTCP extensions to indicate the splicing related information to the splicer: an RTP header extension that conveys the information in-band and an RTCP packet that conveys the information out-of-band.

Status of this Memo

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

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

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

The list of current Internet-Drafts can be accessed at
<http://www.ietf.org/lid-abstracts.html>

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

Copyright and License 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	3
1.1	Terminology	3
2	Overview of RTP Splicing Notification	4
3	Conveying Splicing Interval in RTP/RTCP extensions	5
3.1	RTP Header Extension	5
3.2	RTCP Splicing Notification Message	6
4	Reducing Splicing Latency	7
5	Failure Cases	8
6	SDP Signaling	8
6.1	Declarative SDP	9
6.2	Offer/Answer without BUNDLE	9
6.3	Offer/Answer with BUNDLE: All Media are spliced	10
6.4	Offer/Answer with BUNDLE: a Subset of Media are Spliced	12
7	Security Considerations	13
8	IANA Considerations	14
8.1	RTCP Control Packet Types	14
8.2	RTCP Compact Header Extensions	14
8.3	SDP Grouping Semantic Extension	14
9	Acknowledges	15
10	References	15
10.1	Normative References	15
10.2	Informative References	15
	Authors' Addresses	16

1 Introduction

Splicing is a process that replaces some multimedia content with other multimedia content and delivers the substitutive multimedia content to the receivers for a period of time. In some predictable splicing cases, e.g., advertisement insertion, the splicing duration MUST be inside of the specific, pre-designated time slot. Certain timing information about when to start and end the splicing must be first acquired by the splicer in order to start the splicing. This document refers to this information as Splicing Interval.

[SCTE35] provides a method that encapsulates the Splicing Interval inside the MPEG2-TS layer in cable TV systems. But in the RTP splicing scenario described in [RFC6828], the RTP mixer designed as the splicer has to decode the RTP packets and search for the Splicing Interval inside the payloads. The need for such processing increases the workload of the mixer and limits the number of RTP sessions the mixer can support.

The document defines an RTP header extension [RFC5285] used by the main RTP sender to provide the Splicing Interval by including it in the RTP packets.

Nevertheless, the Splicing Interval conveyed in the RTP header extension might not reach the mixer successfully, any splicing unaware middlebox on the path between the RTP sender and the mixer might strip this RTP header extension.

To increase robustness against such case, the document also defines a new RTCP packet type in a complementary fashion to carry the same Splicing Interval to the mixer.

1.1 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 RFC 2119 [RFC2119].

The terminology defined in "Content Splicing for RTP Sessions" [RFC6828] applies to this document and in addition, we define:

Splicing Interval:

The NTP timestamps for the Splicing-In point and Splicing-Out point per [RFC6828] allowing the mixer to know when to start and end the RTP splicing.

2 Overview of RTP Splicing Notification

According to [RFC6828], a mixer is designed to handle splicing on the RTP layer at the reserved time slots set by the main RTP sender. This implies that the mixer must first know the Splicing Interval from the main RTP sender before it can start splicing.

When a new splicing is forthcoming, the main RTP sender **MUST** send the Splicing Interval to the mixer. Usually, the Splicing Interval **SHOULD** be sent more than once to mitigate the possible packet loss. To enable the mixer to get the substitutive content before the splicing starts, the main RTP sender **MUST** send the Splicing Interval far ahead. For example, the main RTP sender can estimate when to send the Splicing Interval based on the round-trip time (RTT) following the mechanisms in section 6.4.1 of [RFC3550] when the mixer sends RTCP RR to the main sender.

The substitutive sender also needs to learn the Splicing Interval from the main RTP sender in advance, and thus estimates when to transfer the substitutive content to the mixer. The Splicing Interval could be transmitted from the main RTP sender to the substitutive content using some out-of-band mechanisms, the details how to achieve that are beyond the scope of this memo. To ensure the Splicing Interval is valid for both the main RTP sender and the substitutive RTP sender, the two senders **MUST** share a common reference clock, so the mixer can achieve accurate splicing.

In this document, the main RTP sender uses a pair of NTP-format timestamps, derived from the common reference clock, to indicate when to start and end the splicing to the mixer: the timestamp of the first substitutive RTP packet at the splicing in point, and the timestamp of the first main RTP packet at the splicing out point.

When the substitutive RTP sender gets the Splicing Interval, it must prepare the substitutive stream. The mixer **MUST** ensure that the RTP timestamp of the first substitutive RTP packet that would be presented to the receivers corresponds to the same time instant as the former NTP timestamp in the Splicing Interval. To enable the mixer to know the first substitutive RTP packet it needs to send, the substitutive RTP sender **MUST** send the substitutive RTP packet ahead of the Splicing In point, allowing the mixer to find out the timestamp of this first RTP packet in the substitutive RTP stream, e.g., using a prior RTCP SR message.

When the splicing will end, the mixer **MUST** ensure that the RTP timestamp of the first main RTP packet that would be presented on the receivers corresponds to the same time instant as the latter NTP timestamp in the Splicing Interval.

3 Conveying Splicing Interval in RTP/RTCP extensions

This memo defines two backwards compatible RTP extensions to convey the Splicing Interval to the mixer: an RTP header extension and an RTCP splicing notification message.

3.1 RTP Header Extension

The RTP header extension mechanism defined in [RFC5285] can be adapted to carry the Splicing Interval consisting of a pair of NTP-format timestamps.

One variant is defined for this header extension. It carries the 7 octets splicing-out NTP timestamp (lower 24-bit part of the Seconds of a NTP-format timestamp and the 32 bits of the Fraction of a NTP-format timestamp as defined in [RFC5905]), followed by the 8 octets splicing-in NTP timestamp (64-bit NTP-format timestamp as defined in [RFC5905]). The top 8 bits of the splicing-out NTP timestamp are referred from the top 8 bits of the splicing-in NTP timestamp. This is unambiguous, under the assumption that the splicing-out time is after the splicing-in time, and the splicing interval is less than 2^{25} seconds.

The format is shown in Figures 1.

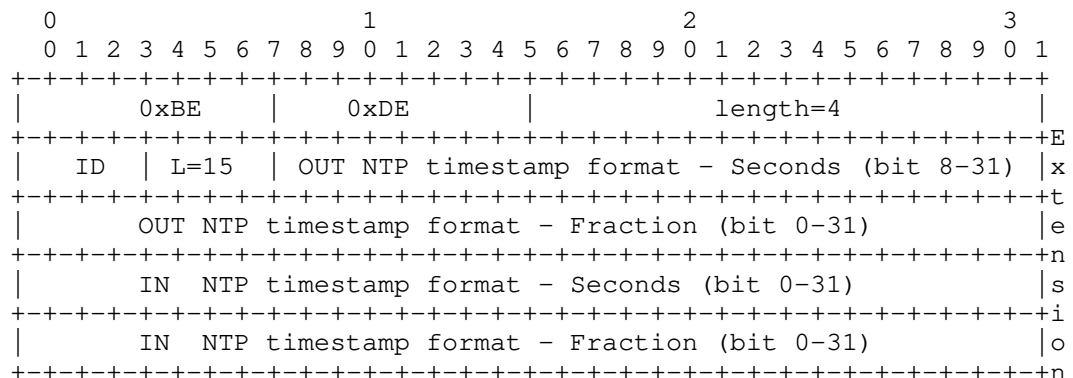


Figure 1: Sample hybrid NTP Encoding Using the One-Byte Header Format

Note that the inclusion of an RTP header extension will reduce the efficiency of RTP header compression. It is RECOMMENDED that the main sender begins to insert the RTP header extensions into a number of RTP packets prior to the splicing in, while leaving the remaining RTP packets unmarked.

After the mixer intercepts the RTP header extension and derives the Splicing Interval, it will generate its own stream and SHOULD NOT include the RTP header extension in outgoing packets to reduce header overhead.

Furthermore, whether the in-band NTP-format timestamps are included or not, RTCP splicing notification message, specified in the next section, MUST be sent to provide robustness in case of any splicing-unaware middlebox that might strip RTP header extensions.

3.2 RTCP Splicing Notification Message

In addition to the RTP header extension, the main RTP sender includes the Splicing Interval in an RTCP splicing notification message.

The RTCP splicing notification message is a new RTCP packet type. It has a fix header followed by a pair of NTP-format timestamps:

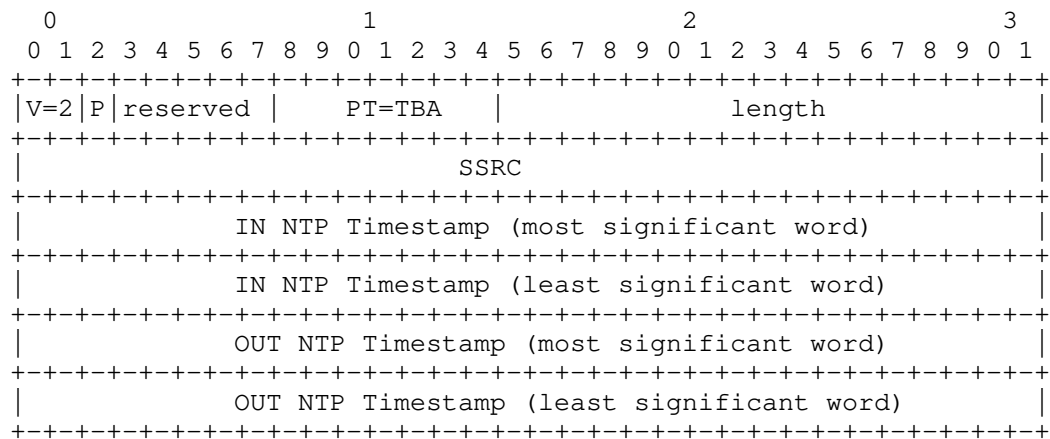


Figure 2: RTCP Splicing Notification Message

The RSI packet includes the following fields:

Length: 16 bits

As defined in [RFC3550], the length of the RTCP packet in 32-bit words minus one, including the header and any padding.

SSRC: 32 bits

The SSRC of the Main RTP Sender.

Timestamp: 64 bits

Indicates the wallclock time when this splicing starts and ends. The full-resolution NTP timestamp is used, which is a 64-bit, unsigned, fixed-point number with the integer part in the first 32 bits and the fractional part in the last 32 bits. This format is similar to RTCP Sender Report (Section 6.4.1 of [RFC3550]).

The RTCP splicing notification message can be appended to RTCP SR the main RTP sender generates in compound RTCP packets, and hence follows the compound RTCP rules defined in Section 6.1 in [RFC3550].

If the use of non-compound RTCP [RFC5506] was previously negotiated between the sender and the mixer, the RTCP splicing notification message may be sent as non-compound RTCP packets.

When the mixer intercepts the RTCP splicing notification message, it SHOULD NOT forward the message to the receivers in order to reduce RTCP bandwidth consumption. And it MUST NOT forward the message to the downstream receivers to avoid them from detecting splicing defined in Section 4.5 in [RFC6828].

4 Reducing Splicing Latency

When splicing starts or ends, the mixer outputs the multimedia content from another sender to the receivers. Given that the receivers must first acquire certain information ([RFC6285] refers to this information as Reference Information) to start processing the multimedia data, either the main RTP sender or the substitutive sender SHOULD provide the Reference Information align with its multimedia content to reduce the delay caused by acquiring the Reference Information. The methods by which the Reference Information is distributed to the receivers is out of scope of this memo.

Another latency element is synchronization caused delay. The receivers must receive enough synchronization metadata prior to synchronizing the separate components of the multimedia streams when splicing starts or ends. Either the main RTP sender or the substitutive sender SHOULD send the synchronization metadata early enough so that the receivers can play out the multimedia in a synchronized fashion. The mechanisms defined in [RFC6051] are RECOMMENDED to be adopted to reduce the possible synchronization delay.

5 Failure Cases

This section examines the implications of losing RTCP splicing notification message and other failure case, e.g., the RTP header extension is stripped on the path.

Given that there may be splicing un-aware middlebox on the path between the main RTP sender and the mixer, one heuristics will be used to verify whether or not the Splicing Interval reaches the mixers.

If the mixer does not get the Splicing Interval when the splicing starts, it will still output the main content to the downstream receivers and forward the RTCP RR packets sent from downstream receivers to the main RTP sender (see section 4.2 of [RFC6828]). In such case, the main RTP sender can learn that splicing failed.

In a similar manner, the substitutive sender can learn that splicing failed if it does not receive any RTCP RR packets from downstream receivers when the splicing starts.

Upon the detection of a failure, the main RTP sender or the substitutive sender SHOULD check the path to the failed mixer, or fallback to the payload specific mechanisms, e.g., MPEG-TS splicing solution defined in [SCTE35].

6 SDP Signaling

This document defines the URI for declaring this header extension in an extmap attribute to be "urn:ietf:params:rtp-hdext:splicing-interval".

This document extends the standard semantics defined in SDP Grouping Framework [RFC5888] with a new semantic: SPLICE to represent the relationship between the main RTP stream and the substitutive RTP stream. Only 2 m-lines are allowed in the SPLICE group. The main RTP stream is the one with the extended extmap attribute, and the other one is substitutive stream. A single m-line MUST NOT be included in different SPLICE groups at the same time. The main RTP sender provides the information about both main and substitutive sources.

The extended SDP attribute specified in this document is applicable for offer/answer content [RFC3264] and do not affect any rules when negotiating offer and answer. When used with multiple media, substitutive RTP MUST be applied only to the RTP packets whose SDP m-line is in the same group with the substitutive stream using SPLICE and has the extended splicing extmap attribute. This semantics is

also applicable for BUNDLE cases.

The following examples show how SDP signaling could be used for splicing in different cases.

6.1 Declarative SDP

```
v=0
o=xia 1122334455 1122334466 IN IP4 splicing.example.com
s=RTP Splicing Example
t=0 0
a=group:SPLICE 1 2
m=video 30000 RTP/AVP 100
i=Main RTP Stream
c=IN IP4 233.252.0.1/127
a=rtpmap:100 MP2T/90000
a=extmap:1 urn:ietf:params:rtp-hdrext:splicing-interval
a=mid:1
m=video 30002 RTP/AVP 100
i=Substitutive RTP Stream
c=IN IP4 233.252.0.2/127
a=sendonly
a=rtpmap:100 MP2T/90000
a=mid:2
```

Figure 3: Example SDP for a single-channel splicing scenario

The mixer receiving the SDP message above receives one MPEG2-TS stream (payload 100) from the main RTP sender (with multicast destination address of 233.252.0.1) on port 30000, and/or receives another MPEG2-TS stream from the substitutive RTP sender (with multicast destination address of 233.252.0.2) on port 30002. But at a particular point in time, the mixer only selects one stream and outputs the content from the chosen stream to the downstream receivers.

6.2 Offer/Answer without BUNDLE

SDP Offer - from main RTP sender

```
v=0
o=xia 1122334455 1122334466 IN IP4 splicing.example.com
s=RTP Splicing Example
t=0 0
a=group:SPLICE 1 2
m=video 30000 RTP/AVP 31 100
i=Main RTP Stream
c=IN IP4 splicing.example.com
```

```
a=rtpmap:31 H261/90000
a=rtpmap:100 MP2T/90000
a=extmap:1 urn:ietf:params:rtp-hdext:splicing-interval
a=sendonly
a=mid:1
m=video 40000 RTP/AVP 31 100
i=Substitutive RTP Stream
c=IN IP4 substitutive.example.com
a=rtpmap:31 H261/90000
a=rtpmap:100 MP2T/90000
a=sendonly
a=mid:2
```

SDP Answer - from splicer

```
v=0
o=xia 1122334455 1122334466 IN IP4 splicer.example.com
s=RTP Splicing Example
t=0 0
a=group:SPLICE 1 2
m=video 30000 RTP/AVP 100
i=Main RTP Stream
c=IN IP4 splicer.example.com
a=rtpmap:100 MP2T/90000
a=extmap:1 urn:ietf:params:rtp-hdext:splicing-interval
a=recvonly
a=mid:1
m=video 40000 RTP/AVP 100
i=Substitutive RTP Stream
c=IN IP4 splicer.example.com
a=rtpmap:100 MP2T/90000
a=recvonly
a=mid:2
```

Only codecs that are supported both by the main RTP stream and the substitutive RTP stream could be negotiated with SDP O/A. And the mixer MUST choose the same codec for both of these two streams.

6.3 Offer/Answer with BUNDLE: All Media are spliced

In this example, the bundled audio and video media have their own substitutive media for splicing:

1. An Offer, in which the offerer assigns a unique address and a substitutive media to each bundled "m="line for splicing within the BUNDLE group.

2. An answer, in which the answerer selects its own BUNDLE address, and leave the substitutive media untouched.

SDP Offer - from main RTP sender

```
v=0
o=alice 1122334455 1122334466 IN IP4 splicing.example.com
s=RTP Splicing Example
c=IN IP4 splicing.example.com
t=0 0
a=group:SPLICE foo 1
a=group:SPLICE bar 2
a=group:BUNDLE foo bar
m=audio 10000 RTP/AVP 0 8 97
a=mid:foo
b=AS:200
a=rtpmap:0 PCMU/8000
a=rtpmap:8 PCMA/8000
a=rtpmap:97 iLBC/8000
a=extmap:1 urn:ietf:params:rtp-hdrext:splicing-interval
a=sendonly
m=video 10002 RTP/AVP 31 32
a=mid:bar
b=AS:1000
a=rtpmap:31 H261/90000
a=rtpmap:32 MPV/90000
a=extmap:2 urn:ietf:params:rtp-hdrext:splicing-interval
a=sendonly
m=audio 20000 RTP/AVP 0 8 97
i=Substitutive audio RTP Stream
c=IN IP4 substitutive.example.com
a=rtpmap:0 PCMU/8000
a=rtpmap:8 PCMA/8000
a=rtpmap:97 iLBC/8000
a=sendonly
a=mid:1
m=video 20002 RTP/AVP 31 32
i=Substitutive video RTP Stream
c=IN IP4 substitutive.example.com
a=rtpmap:31 H261/90000
a=rtpmap:32 MPV/90000
a=mid:2
a=sendonly
```

SDP Answer - from the splicer

```
v=0
```

```
o=bob 2808844564 2808844564 IN IP4 splicer.example.com
s=RTP Splicing Example
c=IN IP4 splicer.example.com
t=0 0
a=group:SPLICE foo 1
a=group:SPLICE bar 2
a=group:BUNDLE foo bar
m=audio 30000 RTP/AVP 0
a=mid:foo
b=AS:200
a=rtpmap:0 PCMU/8000
a=extmap:1 urn:ietf:params:rtp-hdrext:splicing-interval
a=recvonly
m=video 30000 RTP/AVP 32
a=mid:bar
b=AS:1000
a=rtpmap:32 MPV/90000
a=extmap:2 urn:ietf:params:rtp-hdrext:splicing-interval
a=recvonly
m=audio 30002 RTP/AVP 0
i=Substitutive audio RTP Stream
c=IN IP4 splicer.example.com
a=rtpmap:0 PCMU/8000
a=recvonly
a=mid:1
m=video 30004 RTP/AVP 32
i=Substitutive video RTP Stream
c=IN IP4 splicer.example.com
a=rtpmap:32 MPV/90000
a=mid:2
a=recvonly
```

6.4 Offer/Answer with BUNDLE: a Subset of Media are Spliced

In this example, the substitutive media only applies for video when splicing:

1. An Offer, in which the offerer assigns a unique address to each bundled "m="line within the BUNDLE group, and assigns a substitutive media to the bundled video "m=" line for splicing.
2. An answer, in which the answerer selects its own BUNDLE address, and leave the substitutive media untouched.

SDP Offer - from the main RTP sender:

```
v=0
o=alice 1122334455 1122334466 IN IP4 splicing.example.com
```

```
s=RTP Splicing Example
c=IN IP4 splicing.example.com
t=0 0
a=group:SPLICE bar 2
a=group:BUNDLE foo bar
m=audio 10000 RTP/AVP 0 8 97
a=mid:foo
b=AS:200
a=rtpmap:0 PCMU/8000
a=rtpmap:8 PCMA/8000
a=rtpmap:97 iLBC/8000
a=sendonly
m=video 10002 RTP/AVP 31 32
a=mid:bar
b=AS:1000
a=rtpmap:31 H261/90000
a=rtpmap:32 MPV/90000
a=extmap:2 urn:ietf:params:rtp-hdext:splicing-interval
a=sendonly
m=video 20000 RTP/AVP 31 32
i=Substitutive video RTP Stream
c=IN IP4 substitutive.example.com
a=rtpmap:31 H261/90000
a=rtpmap:32 MPV/90000
a=mid:2
a=sendonly
```

SDP Answer - from the splicer:

```
v=0
o=bob 2808844564 2808844564 IN IP4 splicer.example.com
s=RTP Splicing Example
c=IN IP4 splicer.example.com
t=0 0
a=group:SPLICE bar 2
a=group:BUNDLE foo bar
m=audio 30000 RTP/AVP 0
a=mid:foo
b=AS:200
a=rtpmap:0 PCMU/8000
a=recvonly
m=video 30000 RTP/AVP 32
a=mid:bar
b=AS:1000
a=rtpmap:32 MPV/90000
a=extmap:2 urn:ietf:params:rtp-hdext:splicing-interval
a=recvonly
m=video 30004 RTP/AVP 32
```



```
i=Substitutive video RTP Stream
c=IN IP4 splicer.example.com
a=rtpmap:32 MPV/90000
a=mid:2
a=recvonly
```

7 Security Considerations

The security considerations of the RTP specification [RFC3550], the general mechanism for RTP header extensions [RFC5285] and the security considerations of the RTP splicing specification [RFC6828] apply.

The RTP header extension defined in Section 4.1 include two NTP-format timestamps. In the Secure Real-time Transport Protocol (SRTP) [RFC3711], RTP header extensions are authenticated but not encrypted. For a malicious endpoint without the key, it can observe the splicing time in the RTP header, and it can intercept the substitutive content and even replace it with a different one if the splicer does not use any security like SRTP and authenticate the main and substitutive content sources.

If there is a concern about the confidentiality of the splicing time information, header extension encryption [RFC6904] SHOULD be used. However, the malicious endpoint can get the splicing time information by other means, e.g., observing the RTP timestamp of the substitutive stream. To protect from different substitutive contents are inserted, the splicer MUST have some mechanisms to authenticate the substitutive stream source.

For cases that the splicing time information is changed by a malicious endpoint, the splicing may fail since it will not be available at the right time for the substitutive media to arrive, which may also break an undetectable splicing. To mitigate this effect, the splicer SHOULD NOT forward the splicing time information RTP header extension defined in Section 4.1 to the receivers. And it MUST NOT forward this header extension when considering an undetectable splicing.

8 IANA Considerations

8.1 RTCP Control Packet Types

Based on the guidelines suggested in [RFC5226], a new RTCP packet format has been registered with the RTCP Control Packet Type (PT) Registry:

Name: SNM

Long name: Splicing Notification Message

Value: TBA

Reference: This document

8.2 RTP Compact Header Extensions

The IANA has also registered a new RTP Compact Header Extension [RFC5285], according to the following:

Extension URI: urn:ietf:params:rtp-hdext:splicing-interval

Description: Splicing Interval

Contact: Jinwei Xia <xiajinwei@huawei.com>

Reference: This document

8.3 SDP Grouping Semantic Extension

This document request IANA to register the new SDP grouping semantic extension called "SPLICE".

Semantics: Splice

Token:SPLICE

Reference: This document

Contact: Jinwei Xia <xiajinwei@huawei.com>

9 Acknowledges

TBD

10 References

10.1 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.

- [RFC3264] Rosenberg, J., and H. Schulzrinne, "An Offer/Answer Model with the Session Description Protocol (SDP)", RFC 3264, June 2002.
- [RFC5285] Singer, D. and H. Desineni, "A General Mechanism for RTP Header Extensions", RFC 5285, July 2008.
- [RFC5888] Camarillo, G. and H. Schulzrinne, "The Session Description Protocol (SDP) Grouping Framework", RFC 5888, June 2010.
- [RFC5905] Mills, D., Martin, J., Ed., Burbank, J., and W. Kasch, "Network Time Protocol Version 4: Protocol and Algorithms Specification", RFC 5905, June 2010.
- [RFC6051] Perkins, C. and T. Schierl, "Rapid Synchronisation of RTP Flows", RFC 6051, November 2010.

10.2 Informative References

- [RFC3711] Baugher, M., McGrew, D., Naslund, M., Carrara, E., and K. Norrman, "The Secure Real-time Transport Protocol (SRTP)", RFC 3711, March 2004.
- [RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 5226, May 2008.
- [RFC5506] Johansson, I. and M. Westerlund, "Support for Reduced-Size Real-Time Transport Control Protocol (RTCP): Opportunities and Consequences", RFC 5506, April 2009.
- [RFC6285] Ver Steeg, B., Begen, A., Van Caenegem, T., and Z. Vax, "Unicast-Based Rapid Acquisition of Multicast RTP Sessions", RFC 6285, June 2011.
- [RFC6904] Lennox, J., "Encryption of Header Extensions in the Secure Real-Time Transport Protocol (SRTP)", April 2013.
- [SCTE35] Society of Cable Telecommunications Engineers (SCTE), "Digital Program Insertion Cueing Message for Cable", 2011.
- [RFC6828] Xia, J., "Content Splicing for RTP Sessions", RFC 6828, January 2013.

Authors' Addresses

Jinwei Xia
Huawei

Email: xiajinwei@huawei.com

Roni Even
Huawei

Email: ron.even.tlv@gmail.com

Rachel Huang
Huawei

Email: rachel.huang@huawei.com

Lingli Deng
China Mobile

Email: denglingli@chinamobile.com

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

P. Thatcher
Google
M. Zanaty
S. Nandakumar
Cisco Systems
B. Burman
Ericsson
A. Roach
B. Campen
Mozilla
October 19, 2015

RTP Payload Format Constraints
draft-pthatcher-mmusic-rid-02

Abstract

In this specification, we define a framework for identifying Source RTP Streams with the constraints on its payload format in the Session Description Protocol. This framework uses "rid" SDP attribute to: a) effectively identify the Source RTP Streams within a RTP Session, b) constrain their payload format parameters in a codec-agnostic way beyond what is provided with the regular Payload Types and c) enable unambiguous mapping between the Source RTP Streams to their media format specification in the SDP.

Note-1: The name 'rid' is not yet finalized. Please refer to Section 12 for more details on the naming.

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	3
2. Key Words for Requirements	4
3. Terminology	4
4. Motivation	4
5. SDP 'rid' Media Level Attribute	5
6. 'rid-level' constraints	6
7. SDP Offer/Answer Procedures	7
7.1. Generating the Initial SDP Offer	7
7.2. Answerer processing the SDP Offer	8
7.2.1. 'rid' unaware Answerer	8
7.2.2. 'rid' aware Answerer	8
7.3. Generating the SDP Answer	9
7.4. Offering Processing of the SDP Answer	10
7.5. Modifying the Session	10
8. Usage of 'rid' in RTP and RTCP	10
8.1. RTCP 'RID' SDES Extension	11
8.2. RTP 'rid' Header Extension	11
9. Interaction with Other Techniques	12
10. Formal Grammar	12
11. SDP Examples	14
11.1. Many Bundled Streams using Many Codecs	14
11.2. Scalable Layers	16
12. Open Issues	16
12.1. Name of the identifier	16
13. IANA Considerations	17
13.1. New RTP Header Extension URI	17
13.2. New SDES item	17
13.3. New SDP Media-Level attribute	18
13.4. Registry for RID-Level Parameters	18
14. Security Considerations	19
15. Acknowledgements	19

16. References	19
16.1. Normative References	20
16.2. Informative References	20
Authors' Addresses	21

1. Introduction

Payload Type (PT) in RTP provides mapping between the format of the RTP payload and the media format description specified in the signaling. For applications that use SDP for signaling, the constructs `rtptime` and/or `fmtp` describe the characteristics of the media that is carried in the RTP payload, mapped to a given PT.

Recent advances in standards such as RTCWEB and NETVC have given rise to rich multimedia applications requiring support for multiple RTP Streams within a RTP session

[I-D.ietf-mmusic-sdp-bundle-negotiation],
[I-D.ietf-mmusic-sdp-simulcast] or having to support multiple codecs, for example. These demands have unearthed challenges inherent with:

- o The restricted RTP PT space in specifying the various payload configurations,
- o The codec-specific constructs for the payload formats in SDP,
- o Missing or underspecified payload format parameters,
- o Ambiguity in mapping between the individual Source RTP Streams and their equivalent format specification in the SDP.

This specification defines a new SDP framework for constraining Source RTP Streams (Section 2.1.10

[I-D.ietf-avtext-rtp-grouping-taxonomy]), called "Restriction Identifier (rid)", along with the SDP attributes to constrain their payload formats in a codec-agnostic way. The "rid" framework can be thought of as complementary extension to the way the media format parameters are specified in SDP today, via the "a=fmtp" attribute. This specification also proposes a new RTCP SDES item to carry the "rid" value, to provide correlation between the RTP Packets and their format specification in the SDP. This SDES item also uses the header extension mechanism [I-D.ietf-avtext-sdes-hdr-ext] to provide correlation at stream startup, or stream changes where RTCP isn't sufficient.

Note that the "rid" parameters only serve to further constrain the parameters that are established on a PT format. They do not relax any existing constraints.

As described in Section 7.2.1, this mechanism achieves backwards compatibility via the normal SDP processing rules, which require unknown a= parameters to be ignored. This means that implementations need to be prepared to handle successful offers and answers from other implementations that neither indicate nor honor the constraints requested by this mechanism.

Further, as described in Section 7 and its subsections, this mechanism achieves extensibility by: (a) having offerers include all supported constraints in their offer, and (b) having answerers ignore a=rid lines that specify unknown constraints.

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. Terminology

The terms Source RTP Stream, Endpoint, RTP Session, and RTP Stream are used as defined in [I-D.ietf-avtext-rtp-grouping-taxonomy].

[RFC4566] and [RFC3264] terminology is also used where appropriate.

4. Motivation

This section summarizes several motivations for proposing the "rid" framework.

1. RTP PT Space Exhaustion: [RFC3550] defines payload type (PT) that identifies the format of the RTP payload and determine its interpretation by the application. [RFC3550] assigns 7 bits for the PT in the RTP header. However, the assignment of static mapping of payload codes to payload formats and multiplexing of RTP with other protocols (such as RTCP) could result in limited number of payload type numbers available for the application usage. In scenarios where the number of possible RTP payload configurations exceed the available PT space within a RTP Session, there is need a way to represent the additional constraints on payload configurations and to effectively map a Source RTP Stream to its corresponding constraints.
1. Multi-source and Multi-stream Use Cases: Recently, there is a rising trend with real-time multimedia applications supporting multiple sources per endpoint with various temporal resolutions (Scalable Video Codec) and spatial resolutions (Simulcast) per source. These applications are being challenged by the limited

RTP PT space and/or by the underspecified SDP constructs for exercising granular control on configuring the individual Source RTP Streams.

5. SDP 'rid' Media Level Attribute

This section defines new SDP media-level attribute [RFC4566], "a=rid". Roughly speaking, this attribute takes the following form (see Section 10 for a formal definition).

```
a=rid:<rid-identifier> <direction> pt=<fmt-list>;<constraint>=<value>...
```

A given "a=rid" SDP media attribute specifies constraints defining a unique RTP payload configuration identified via the "rid-identifier". A set of codec-agnostic "rid-level" constraints are defined (Section 6) that describe the media format specification applicable to one or more Payload Types specified by the "a=rid" line.

The 'rid' framework MAY be used in combination with the 'a=fmtp' SDP attribute for describing the media format parameters for a given RTP Payload Type. However in such scenarios, the 'rid-level' constraints (Section 6) further constrains the equivalent 'fmtp' attributes.

The 'direction' identifies the either 'send', 'recv' directionality of the Source RTP Stream.

A given SDP media description MAY have zero or more "a=rid" lines describing various possible RTP payload configurations. A given 'rid-identifier' MUST NOT be repeated in a given media description.

The 'rid' media attribute MAY be used for any RTP-based media transport. It is not defined for other transports.

Though the 'rid-level' attributes specified by the 'rid' property follow the syntax similar to session-level and media-level attributes, they are defined independently. All 'rid-level' attributes MUST be registered with IANA, using the registry defined in Section 13

Section 10 gives a formal Augmented Backus-Naur Form (ABNF) [RFC5234] grammar for the "rid" attribute.

The "a=rid" media attribute is not dependent on charset.

6. 'rid-level' constraints

This section defines the 'rid-level' constraints that can be used to constrain the RTP payload encoding format in a codec-agnostic way.

The following constraints are intended to apply to video codecs in a codec-independent fashion.

- o max-width, for spatial resolution in pixels. In the case that stream orientation signaling is used to modify the intended display orientation, this attribute refers to the width of the stream when a rotation of zero degrees is encoded.
- o max-height, for spatial resolution in pixels. In the case that stream orientation signaling is used to modify the intended display orientation, this attribute refers to the width of the stream when a rotation of zero degrees is encoded.
- o max-fps, for frame rate in frames per second. For encoders that do not use a fixed framerate for encoding, this value should constrain the minimum amount of time between frames: the time between any two consecutive frames SHOULD NOT be less than 1/max-fps seconds.
- o max-fs, for frame size in pixels per frame. This is the product of frame width and frame height, in pixels, for rectangular frames.
- o max-br, for bit rate in bits per second. The restriction applies to the media payload only, and does not include overhead introduced by other layers (e.g., RTP, UDP, IP, or Ethernet). The exact means of keeping within this limit are left up to the implementation, and instantaneous excursions outside the limit are permissible. For any given one-second sliding window, however, the total number of bits in the payload portion of RTP SHOULD NOT exceed the value specified in "max-br."
- o max-pps, for pixel rate in pixels per second. This value SHOULD be handled identically to max-fps, after performing the following conversion: $\text{max-fps} = \text{max-pps} / (\text{width} * \text{height})$. If the stream resolution changes, this value is recalculated. Due to this recalculation, excursions outside the specified maximum are possible during near resolution change boundaries.

All the constraints are optional and are subjected to negotiation based on the SDP Offer/Answer rules described in Section 7

This list is intended to be an initial set of constraints; future documents may define additional constraints; see Section 13.4. While this document doesn't define constraints for audio codecs, there is no reason such constraints should be precluded from definition and registration by other documents.

Section 10 provides formal Augmented Backus-Naur Form (ABNF) [RFC5234] grammar for each of the "rid-level" attributes defined in this section.

7. SDP Offer/Answer Procedures

This section describes the SDP Offer/Answer [RFC3264] procedures when using the 'rid' framework.

Note that 'rid's are only required to be unique within a media section ("m-line"); they do not necessarily need to be unique within an entire RTP session. In traditional usage, each media section is sent on its own unique 5-tuple, which provides an unambiguous scope. Similarly, when using BUNDLE [I-D.ietf-mmusic-sdp-bundle-negotiation], MID values associate RTP streams uniquely to a single media description.

7.1. Generating the Initial SDP Offer

For each media description in the offer, the offerer MAY choose to include one or more "a=rid" lines to specify a configuration profile for the given set of RTP Payload Types.

In order to construct a given "a=rid" line, the offerer must follow the below steps:

1. It MUST generate a 'rid-identifier' that is unique within a media description
2. It MUST set the direction for the 'rid-identifier' to one of 'send' or 'recv'
3. It MAY include a listing of SDP format tokens (usually corresponding to RTP payload types) to which the constraints expressed by the 'rid-level' attributes apply. Any Payload Types chosen MUST be a valid payload type for the media section (that is, it must be listed on the "m=" line).
4. The Offerer then chooses the 'rid-level' constraints (Section 6) to be applied for the rid, and adds them to the "a=rid" line. If it wishes the answer to have the ability to specify a constraint, but does not wish to set a value itself, it MUST include the name

of the constraint in the "a=rid" line, but without any indicated value.

Note: If an 'a=fmtp' attribute is also used to provide media-format-specific parameters, then the 'rid-level' attributes will further constrain the equivalent 'fmtp' parameters for the given Payload Type for those streams associated with the 'rid'.

If a given codec would require "a=fmtp" line when used without "a=rid" then the offer MUST include a valid corresponding "a=fmtp" line even when using RID.

7.2. Answerer processing the SDP Offer

For each media description in the offer, and for each "a=rid" attribute in the media description, the receiver of the offer will perform the following steps:

7.2.1. 'rid' unaware Answerer

If the receiver doesn't support the 'rid' framework proposed in this specification, the entire "a=rid" line is ignored following the standard [RFC3264] Offer/Answer rules.

Section 7.1 requires the offer to include a valid "a=fmtp" line for any codecs that otherwise require it (in other words, the "a=rid" line cannot be used to replace "a=fmtp" configuration). As a result, ignoring the "a=rid" line is always guaranteed to result in a valid session description.

7.2.2. 'rid' aware Answerer

If the answerer supports 'rid' framework, the following steps are executed, in order, for each "a=rid" line in a given media description:

1. Extract the rid-identifier from the "a=rid" line and verify its uniqueness. In the case of a duplicate, the entire "a=rid" line, and all "a=rid" lines with rid-identifiers that duplicate this line, are rejected and MUST NOT be included in the SDP Answer.
2. If the "a=rid" line contains a "pt=" parameter, the list of payload types is verified against the list of valid payload types for the media section (that is, those listed on the "m=" line). If there is no match for the Payload Type listed in the "a=rid" line, then remove the "a=rid" line.

3. The answerer ensures that "rid-level" parameters listed are supported and syntactically well formed. In the case of a syntax error or an unsupported parameter, the "a=rid" line is removed.
4. If the 'depend' rid-level attribute is included, the answerer MUST make sure that the rid-identifiers listed unambiguously match the rid-identifiers in the SDP offer. Any lines that do not are removed.
5. if the "a=rid" line contains a "pt=" parameter, the answerer verifies that the attribute values provided in the "rid-level" attributes are consistent with the corresponding codecs and their other parameters. See Section 9 for more detail. If the rid-level parameters are incompatible with the other codec properties, then the "a=rid" line is removed.

7.3. Generating the SDP Answer

Having performed the verification of the SDP offer as described, the answerer shall perform the following steps to generate the SDP answer.

For each "a=rid" line:

1. The answerer MAY choose to modify specific 'rid-level' attribute value in the answer SDP. In such a case, the modified value MUST be more constrained than the ones specified in the offer. The answer MUST NOT include any constraints that were not present in the offer.
2. The answerer MUST NOT modify the 'rid-identifier' present in the offer.
3. The answerer is allowed to remove one or more media formats from a given 'a=rid' line. If the answerer chooses to remove all the media format tokens from an "a=rid" line, the answerer MUST remove the entire "a=rid" line.
4. In cases where the answerer is unable to support the payload configuration specified in a given "a=rid" line in the offer, the answerer MUST remove the corresponding "a=rid" line. This includes situations in which the answerer does not understand one or more of the constraints in the "a=rid" line that has an associated value.

Note: in the case that the answerer uses different PT values to represent a codec than the offerer did, the "a=rid" values in the answer use the PT values that were sent in the offer.

7.4. Offering Processing of the SDP Answer

The offerer shall follow the steps similar to answerer's offer processing with the following exceptions

1. The offerer MUST ensure that the 'rid-identifiers' aren't changed between the offer and the answer. If so, the offerer MUST consider the corresponding 'a=rid' line as rejected.
2. If there exist changes in the 'rid-level' attribute values, the offerer MUST ensure that the modifications can be supported or else consider the "a=rid" line as rejected.
3. If the SDP answer contains any "rid-identifier" that doesn't match with the offer, the offerer MUST ignore the corresponding "a=rid" line.
4. If the "a=rid" line contains a "pt=" parameter, the offerer verifies that the list of payload types is a subset of those sent in the corresponding "a=rid" line in the offer.
5. If the "a=rid" line contains a "pt=" parameter, the offerer verifies that the attribute values provided in the "rid-level" attributes are consistent with the corresponding codecs and their other parameters. See Section 9 for more detail. If the rid-level parameters are incompatible with the other codec properties, then the "a=rid" line is removed.

7.5. Modifying the Session

Offers and answers inside an existing session follow the rules for initial session negotiation. Such an offer MAY propose a change the number of RIDs in use. To avoid race conditions with media, any RIDs with proposed changes SHOULD use a new ID, rather than re-using one from the previous offer/answer exchange. RIDs without proposed changes SHOULD re-use the ID from the previous exchange.

8. Usage of 'rid' in RTP and RTCP

The RTP fixed header includes the payload type number and the SSRC values of the RTP stream. RTP defines how you de-multiplex streams within an RTP session, but in some use cases applications need further identifiers in order to effectively map the individual RTP Streams to their equivalent payload configurations in the SDP.

This specification defines a new RTCP SDPS item [RFC3550], 'RID', which is used to carry rids within RTCP SDPS packets. This makes it possible for a receiver to associate received RTP packets

(identifying the Source RTP Stream) with a media description having the format constraint specified.

This specification also uses the RTP header extension for RTCP SDES items [I-D.ietf-avtext-sdes-hdr-ext] to allow carrying RID information in RTP packets to provide correlation at stream startup, or after stream changes where the use of RTCP may not be sufficiently responsive.

8.1. RTCP 'RID' SDES Extension

```

  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
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|          RID=TBD          |      length      | rid                      ...
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+

```

The rid payload is UTF-8 encoded and is not null-terminated.

RFC EDITOR NOTE: Please replace TBD with the assigned SDES identifier value.

8.2. RTP 'rid' Header Extension

Because recipients of RTP packets will typically need to know which "a=rid" constraints they correspond to immediately upon receipt, this specification also defines a means of carrying RID identifiers in RTP extension headers, using the technique described in [I-D.ietf-avtext-sdes-hdr-ext].

As described in that document, the header extension element can be encoded using either the one-byte or two-byte header, and the identification-tag payload is UTF-8 encoded, as in SDP.

As the identification-tag is included in an RTP header extension, there should be some consideration about the packet expansion caused by the identification-tag. To avoid Maximum Transmission Unit (MTU) issues for the RTP packets, the header extension's size needs to be taken into account when the encoding media. Note that set of header extensions included in the packet needs to be padded to the next 32-bit boundary using zero bytes [RFC5285]

It is RECOMMENDED that the identification-tag is kept short. Due to the properties of the RTP header extension mechanism, when using the one-byte header, a tag that is 1-3 bytes will result in that a minimal number of 32-bit words are used for the RTP header extension, in case no other header extensions are included at the same time. In

many cases, a one-byte tag will be sufficient; it is RECOMMENDED that implementations use the shortest identifier that fits their purposes.

9. Interaction with Other Techniques

Historically, a number of other approaches have been defined that allow constraining media streams via SDP parameters. These include:

- o Codec-specific configuration set via format parameters ("a=fmtp"); for example, the H.264 "max-fs" format parameter
- o Size restrictions imposed by image attribute attributes ("a=imgattr") [RFC6236]

When the mechanism described in this document is used in conjunction with these other restricting mechanisms, it is intended to impose additional restrictions beyond those communicated in other techniques.

In an offer, this means that a=rid lines, when combined with other restrictions on the media stream, are expected to result in a non-empty union. For example, if image attributes are used to indicate that a PT has a minimum width of 640, then specification of "max-width=320" in an "a=rid" line that is then applied to that PT is nonsensical. According to the rules of Section 7.2.2, this will result in the corresponding "a=rid" line being ignored by the recipient.

Similarly, an answer the a=rid lines, when combined with the other restrictions on the media stream, are also expected to result in a non-empty union. If the implementation generating an answer wishes to restrict a property of the stream below that which would be allowed by other parameters (e.g., those specified in "a=fmtp" or "a=imgattr"), its only recourse is to remove the "a=rid" line altogether, as described in Section 7.3. If it instead attempts to constrain the stream beyond what is allowed by other mechanisms, then the offerer will ignore the corresponding "a=rid" line, as described in Section 7.4.

10. Formal Grammar

This section gives a formal Augmented Backus-Naur Form (ABNF) [RFC5234] grammar for each of the new media and rid-level attributes defined in this document.


```
rid-syntax      = "a=rid:" rid-identifier SP rid-dir  
                  [ rid-pt-param-list / rid-param-list ]  
  
rid-identifier  = 1*(alpha-numeric / "-" / "_")  
  
rid-dir         = "send" / "recv"  
  
rid-pt-param-list = SP rid-fmt-list *("; " rid-param)  
  
rid-param-list  = SP rid-param *("; " rid-param)  
  
rid-fmt-list    = "pt=" fmt *( " , " fmt )  
                  ; fmt defined in {{RFC4566}}  
  
rid-param       = rid-width-param  
                  / rid-height-param  
                  / rid-fps-param  
                  / rid-fs-param  
                  / rid-br-param  
                  / rid-pps-param  
                  / rid-depend-param  
                  / rid-param-other  
  
rid-width-param = "max-width" [ "=" int-param-val ]  
  
rid-height-param = "max-height" [ "=" int-param-val ]  
  
rid-fps-param    = "max-fps" [ "=" int-param-val ]  
  
rid-fs-param     = "max-fs" [ "=" int-param-val ]  
  
rid-br-param     = "max-br" [ "=" int-param-val ]  
  
rid-pps-param    = "max-pps" [ "=" int-param-val ]  
  
rid-depend-param = "depend=" rid-list  
  
rid-param-other  = 1*(alpha-numeric / "-") [ "=" param-val ]  
  
rid-list         = rid-identifier *( " , " rid-identifier )  
  
int-param-val    = 1*DIGIT  
  
param-val        = *( %x20-58 / %x60-7E )  
                  ; Any printable character except semicolon
```

11. SDP Examples

Note: see [I-D.ietf-mmusic-sdp-simulcast] for examples of RID used in simulcast scenarios.

11.1. Many Bundled Streams using Many Codecs

In this scenario, the offerer supports the Opus, G.722, G.711 and DTMF audio codecs, and VP8, VP9, H.264 (CBP/CHP, mode 0/1), H.264-SVC (SCBP/SCHP) and H.265 (MP/M10P) for video. An 8-way video call (to a mixer) is supported (send 1 and receive 7 video streams) by offering 7 video media sections (1 sendrecv at max resolution and 6 recvonly at smaller resolutions), all bundled on the same port, using 3 different resolutions. The resolutions include:

- o 1 receive stream of 720p resolution is offered for the active speaker.
- o 2 receive streams of 360p resolution are offered for the prior 2 active speakers.
- o 4 receive streams of 180p resolution are offered for others in the call.

Expressing all these codecs and resolutions using 32 dynamic PTs (2 audio + 10x3 video) would exhaust the primary dynamic space (96-127). RIDs are used to avoid PT exhaustion and express the resolution constraints.

NOTE: The SDP given below skips few lines to keep the example short and focused, as indicated by either the "..." or the comments inserted.

Example 1

Offer:

```
...
m=audio 10000 RTP/SAVPF 96 9 8 0 123
a=rtpmap:96 OPUS/48000
a=rtpmap:9 G722/8000
a=rtpmap:8 PCMA/8000
a=rtpmap:0 PCMU/8000
a=rtpmap:123 telephone-event/8000
a=mid:a1
...
m=video 10000 RTP/SAVPF 98 99 100 101 102 103 104 105 106 107
a=rtpmap:98 VP8/90000
```

```
a=fmtp:98 max-fs=3600; max-fr=30
a=rtpmap:99 VP9/90000
a=fmtp:99 max-fs=3600; max-fr=30
a=rtpmap:100 H264/90000
a=fmtp:100 profile-level-id=42401f; packetization-mode=0
a=rtpmap:101 H264/90000
a=fmtp:101 profile-level-id=42401f; packetization-mode=1
a=rtpmap:102 H264/90000
a=fmtp:102 profile-level-id=640c1f; packetization-mode=0
a=rtpmap:103 H264/90000
a=fmtp:103 profile-level-id=640c1f; packetization-mode=1
a=rtpmap:104 H264-SVC/90000
a=fmtp:104 profile-level-id=530c1f
a=rtpmap:105 H264-SVC/90000
a=fmtp:105 profile-level-id=560c1f
a=rtpmap:106 H265/90000
a=fmtp:106 profile-id=1; level-id=93
a=rtpmap:107 H265/90000
a=fmtp:107 profile-id=2; level-id=93
a=sendrecv
a=mid:v1 (max resolution)
a=rid:1 send max-width=1280;max-height=720;max-fps=30
a=rid:2 recv max-width=1280;max-height=720;max-fps=30
...
m=video 10000 RTP/SAVPF 98 99 100 101 102 103 104 105 106 107
...same rtpmap/fmtp as above...
a=recvonly
a=mid:v2 (medium resolution)
a=rid:3 recv max-width=640;max-height=360;max-fps=15
...
m=video 10000 RTP/SAVPF 98 99 100 101 102 103 104 105 106 107
...same rtpmap/fmtp as above...
a=recvonly
a=mid:v3 (medium resolution)
a=rid:3 recv max-width=640;max-height=360;max-fps=15
...
m=video 10000 RTP/SAVPF 98 99 100 101 102 103 104 105 106 107
...same rtpmap/fmtp as above...
a=recvonly
a=mid:v4 (small resolution)
a=rid:4 recv max-width=320;max-height=180;max-fps=15
...
m=video 10000 RTP/SAVPF 98 99 100 101 102 103 104 105 106 107
...same rtpmap/fmtp as above...
...same rid:4 as above for mid:v5,v6,v7 (small resolution)...
...
```

Answer:
...same as offer but swap send/recv...

11.2. Scalable Layers

Adding scalable layers to the above simulcast example gives the SFU further flexibility to selectively forward packets from a source that best match the bandwidth and capabilities of diverse receivers. Scalable encodings have dependencies between layers, unlike independent simulcast streams. RIDs can be used to express these dependencies using the "depend" parameter. In the example below, the highest resolution is offered to be sent as 2 scalable temporal layers (using MRST).

Example 3

Offer:
...
m=audio ...same as Example 1 ...
...
m=video ...same as Example 1 ...
...same rtpmap/fmt as Example 1...
a=sendrecv
a=mid:v1 (max resolution)
a=rid:0 send max-width=1280;max-height=720;max-fps=15
a=rid:1 send max-width=1280;max-height=720;max-fps=30;depend=0
a=rid:2 recv max-width=1280;max-height=720;max-fps=30
a=rid:5 send max-width=640;max-height=360;max-fps=15
a=rid:6 send max-width=320;max-height=180;max-fps=15
a=simulcast: send rid=0;1;5;6 recv rid=2
...
...same m=video sections as Example1 for mid:v2-v7...
...

Answer:
...same as offer but swap send/recv...

12. Open Issues

12.1. Name of the identifier

The name 'rid' is provisionally used and is open for further discussion.

Here are the few options that were considered while writing this draft

- o CID: Constraint ID, which is a rather precise description of what we are attempting to accomplish.
- o ESID: Encoded Stream ID, does not align well with taxonomy which defines Encoded Stream as before RTP packetization.
- o RSID or RID: RTP Stream ID, aligns better with taxonomy but very vague.
- o LID: Layer ID, aligns well for SVC with each layer in a separate stream, but not for other SVC layerings or independent simulcast which is awkward to view as layers.
- o EPT or XPT: EXTended Payload Type, conveys XPT.PT usage well, but may be confused with PT, for example people may mistakenly think they can use it in other places where PT would normally be used.

13. IANA Considerations

13.1. New RTP Header Extension URI

This document defines a new extension URI in the RTP Compact Header Extensions subregistry of the Real-Time Transport Protocol (RTP) Parameters registry, according to the following data:

Extension URI:	urn:ietf:params:rtp-hdext:sdes:rid
Description:	RTP Stream Restriction Identifier
Contact:	<mmusic@ietf.org>
Reference:	RFCXXXX

13.2. New SDDES item

RFC EDITOR NOTE: Please replace RFCXXXX with the RFC number of this document.

RFC EDITOR NOTE: Please replace TBD with the assigned SDDES identifier value.

This document adds the MID SDDES item to the IANA "RTCP SDDES item types" registry as follows:

Value:	TBD
Abbrev.:	RID
Name:	Restriction Identification
Reference:	RFCXXXX

13.3. New SDP Media-Level attribute

This document defines "rid" as SDP media-level attribute. This attribute must be registered by IANA under "Session Description Protocol (SDP) Parameters" under "att-field (media level only)".

The "rid" attribute is used to identify characteristics of RTP stream with in a RTP Session. Its format is defined in Section 10.

13.4. Registry for RID-Level Parameters

This specification creates a new IANA registry named "att-field (rid level)" within the SDP parameters registry. The rid-level parameters MUST be registered with IANA and documented under the same rules as for SDP session-level and media-level attributes as specified in [RFC4566].

Parameters for "a=rid" lines that modify the nature of encoded media MUST be of the form that the result of applying the modification to the stream results in a stream that still complies with the other parameters that affect the media. In other words, parameters always have to restrict the definition to be a subset of what is otherwise allowable, and never expand it.

New parameter registrations are accepted according to the "Specification Required" policy of [RFC5226], provided that the specification includes the following information:

- o contact name, email address, and telephone number
- o parameter name (as it will appear in SDP)
- o long-form parameter name in English
- o whether the parameter value is subject to the charset attribute
- o an explanation of the purpose of the parameter
- o a specification of appropriate attribute values for this parameter
- o an ABNF definition of the parameter

The initial set of rid-level parameter names, with definitions in Section 6 of this document, is given below:

Type	SDP Name	Reference
----	-----	-----
att-field	(rid level)	
	max-width	[RFCXXXX]
	max-height	[RFCXXXX]
	max-fps	[RFCXXXX]
	max-fs	[RFCXXXX]
	max-br	[RFCXXXX]
	max-pps	[RFCXXXX]
	depend	[RFCXXXX]

It is conceivable that a future document wants to define a RID-level parameter that contains string values. These extensions need to take care to conform to the ABNF defined for rid-param-other. In particular, this means that such extensions will need to define escaping mechanisms if they want to allow semicolons, unprintable characters, or byte values greater than 127 in the string.

OPEN ITEM: Do we need to do more than this regarding escaping?

14. Security Considerations

As with most SDP parameters, a failure to provide integrity protection over the a=rid attributes provides attackers a way to modify the session in potentially unwanted ways. This could result in an implementation sending greater amounts of data than a recipient wishes to receive. In general, however, since the "a=rid" attribute can only restrict a stream to be a subset of what is otherwise allowable, modification of the value cannot result in a stream that is of higher bandwidth than would be sent to an implementation that does not support this mechanism.

The actual identifiers used for RIDs are expected to be opaque. As such, they are not expected to contain information that would be sensitive, were it observed by third-parties.

15. Acknowledgements

Many thanks to review from Cullen Jennings, Magnus Westerlund, and Paul Kyzivat.

16. References

16.1. Normative References

- [I-D.ietf-avtext-sdes-hdr-ext]
Westerlund, M., Burman, B., Even, R., and M. Zanaty, "RTP Header Extension for RTCP Source Description Items", draft-ietf-avtext-sdes-hdr-ext-02 (work in progress), July 2015.
- [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>>.
- [RFC3264] Rosenberg, J. and H. Schulzrinne, "An Offer/Answer Model with Session Description Protocol (SDP)", RFC 3264, DOI 10.17487/RFC3264, June 2002, <<http://www.rfc-editor.org/info/rfc3264>>.
- [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>>.
- [RFC4566] Handley, M., Jacobson, V., and C. Perkins, "SDP: Session Description Protocol", RFC 4566, DOI 10.17487/RFC4566, July 2006, <<http://www.rfc-editor.org/info/rfc4566>>.
- [RFC5234] Crocker, D., Ed. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", STD 68, RFC 5234, DOI 10.17487/RFC5234, January 2008, <<http://www.rfc-editor.org/info/rfc5234>>.
- [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>>.

16.2. Informative References

- [I-D.ietf-avtext-rtp-grouping-taxonomy]
Lennox, J., Gross, K., Nandakumar, S., Salgueiro, G., and B. Burman, "A Taxonomy of Semantics and Mechanisms for Real-Time Transport Protocol (RTP) Sources", draft-ietf-avtext-rtp-grouping-taxonomy-08 (work in progress), July 2015.

- [I-D.ietf-mmusic-sdp-bundle-negotiation]
Holmberg, C., Alvestrand, H., and C. Jennings,
"Negotiating Media Multiplexing Using the Session
Description Protocol (SDP)", draft-ietf-mmusic-sdp-bundle-
negotiation-23 (work in progress), July 2015.
- [I-D.ietf-mmusic-sdp-simulcast]
Burman, B., Westerlund, M., Nandakumar, S., and M. Zanaty,
"Using Simulcast in SDP and RTP Sessions", draft-ietf-
mmusic-sdp-simulcast-02 (work in progress), October 2015.
- [RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an
IANA Considerations Section in RFCs", BCP 26, RFC 5226,
DOI 10.17487/RFC5226, May 2008,
<<http://www.rfc-editor.org/info/rfc5226>>.
- [RFC6236] Johansson, I. and K. Jung, "Negotiation of Generic Image
Attributes in the Session Description Protocol (SDP)", RFC
6236, DOI 10.17487/RFC6236, May 2011,
<<http://www.rfc-editor.org/info/rfc6236>>.

Authors' Addresses

Peter Thatcher
Google

Email: pthatcher@google.com

Mo Zanaty
Cisco Systems

Email: mzanaty@cisco.com

Suhas Nandakumar
Cisco Systems

Email: snandaku@cisco.com

Bo Burman
Ericsson

Email: bo.burman@ericsson.com

Adam Roach
Mozilla

Email: adam@nostrum.com

Byron Campen
Mozilla

Email: bcampen@mozilla.com