

Network Working Group
Internet-Draft
Updates: 3550 (if approved)
Intended status: Standards Track
Expires: May 27, 2014

M. Petit-Huguenin
Impedance Mismatch
G. Zorn, Ed.
Network Zen
November 23, 2013

Support for Multiple Clock Rates in an RTP Session
draft-ietf-avtext-multiple-clock-rates-11

Abstract

This document clarifies the RTP specification when different clock rates are used in an RTP session. It also provides guidance on how to interoperate with legacy RTP implementations that use multiple clock rates. It updates RFC 3550.

Status of This Memo

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

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

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

This Internet-Draft will expire on May 27, 2014.

Copyright Notice

Copyright (c) 2013 IETF Trust and the persons identified as the document authors. All rights reserved.

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

Table of Contents

1. Introduction	2
2. Terminology	4
3. Legacy RTP	4
3.1. Different SSRC	4
3.2. Same SSRC	4
3.2.1. Monotonic timestamps	5
3.2.2. Non-monotonic timestamps	5
4. Recommendations	6
4.1. RTP Sender (with RTCP)	6
4.2. RTP Sender (without RTCP)	6
4.3. RTP Receiver	7
5. Security Considerations	7
6. IANA Considerations	7
7. Acknowledgements	7
8. References	8
8.1. Normative References	8
8.2. Informative References	8
Appendix A. Example Values	9
Appendix B. Using a Fixed Clock Rate	11
Appendix C. Behavior of Legacy Implementations	11
C.1. libccrtp 2.0.2	11
C.2. libmediastreamer0 2.6.0	11
C.3. libpjmedia 1.0	12
C.4. Android RTP stack 4.0.3	12
Authors' Addresses	12

1. Introduction

The clock rate is a parameter of the payload format as identified in RTP and RTCP by the payload type value. It is often defined as being the same as the sampling rate but that is not always the case (see, for example, the G722 and MPA audio codecs [RFC3551]).

An RTP sender can switch between different payloads during the lifetime of an RTP session and because clock rates are defined by payload format, it is possible that the clock rate will also vary during an RTP session. Schulzrinne, et al. [RFC3550] lists using multiple clock rates as one of the reasons to not use different payloads on the same Synchronization Source (SSRC). Unfortunately this advice has not always been followed and some RTP implementations change the payload in the same SSRC even if the different payloads use different clock rates.

This creates three problems:

- o The method used to calculate the RTP timestamp field in an RTP packet is underspecified.
- o When the same SSRC is used for different clock rates, it is difficult to know what clock rate was used for the RTP timestamp field in an RTCP Sender Report (SR) packet.
- o When the same SSRC is used for different clock rates, it is difficult to know what clock rate was used for the interarrival jitter field in an RTCP Receiver Report (RR) packet.

Table 1 contains a non-exhaustive list of fields in RTCP packets that uses a clock rate as unit:

Field name	RTCP packet type	Reference
RTP timestamp	SR	[RFC3550]
Interarrival jitter	RR	[RFC3550]
min_jitter	XR Summary Block	[RFC3611]
max_jitter	XR Summary Block	[RFC3611]
mean_jitter	XR Summary Block	[RFC3611]
dev_jitter	XR Summary Block	[RFC3611]
Interarrival jitter	IJ	[RFC5450]
RTP timestamp	SMPTE TC	[RFC5484]
Jitter	RSI Jitter Block	[RFC5760]
Median jitter	RSI Stats Block	[RFC5760]

Table 1

This document first tries to list in Section 3 and subsections all of the algorithms known to be used in existing RTP implementations at the time of writing. These sections are not normative.

Section 4 and subsections then recommend a unique algorithm that modifies RFC 3550. These sections are normative.

2. Terminology

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

Clock rate	The multiplier used to convert from a wallclock value in seconds to an equivalent RTP timestamp value (without the fixed random offset). Note that RFC 3550 uses various terms like "clock frequency", "media clock rate", "timestamp unit", "timestamp frequency", and "RTP timestamp clock rate" as synonymous to clock rate.
RTP Sender	A logical network element that sends RTP packets, sends RTCP SR packets, and receives RTCP reception report blocks.
RTP Receiver	A logical network element that receives RTP packets, receives RTCP SR packets, and sends RTCP reception report blocks.

3. Legacy RTP

The following sections describe the various ways legacy RTP implementations behave when multiple clock rates are used. Legacy RTP refers to RFC 3550 without the modifications introduced by this document.

3.1. Different SSRC

One way of managing multiple clock rates is to use a different SSRC for each different clock rate, as in this case there is no ambiguity on the clock rate used by fields in the RTCP packets. This method also seems to be the original intent of RTP as can be deduced from points 2 and 3 of section 5.2 of RFC 3550.

On the other hand, changing the SSRC can be a problem for some implementations designed to work only with unicast IP addresses, where having multiple SSRCs is considered a corner case. Lip synchronization can also be a problem in the interval between the beginning of the new stream and the first RTCP SR packet.

3.2. Same SSRC

The simplest way of managing multiple clock rates is to use the same SSRC for all the payload types regardless of the clock rates.

Unfortunately there is no clear definition on how the RTP timestamp should be calculated in this case. The following subsections present the algorithms used in the field.

3.2.1. Monotonic timestamps

This method of calculating the RTP timestamp ensures that the value increases monotonically. The formula used by this method is as follows:

```
timestamp = previous_timestamp
           + (current_capture_time - previous_capture_time)
           * current_clock_rate
```

The problem with this method is that the jitter calculation on the receiving side gives an invalid result during the transition between two clock rates, as shown in Table 2 (Appendix A). The capture and arrival time are in seconds, starting at the beginning of the capture of the first packet; clock rate is in Hz; the RTP timestamp does not include the random offset; the transit, jitter, and average jitter use the clock rate as unit.

Calculating the correct transit time on the receiving side can be done by using the following formulas:

1. $\text{current_capture_time} = (\text{current_timestamp} - \text{previous_timestamp}) / \text{current_clock_rate} + \text{previous_capture_time}$
2. $\text{transit} = \text{current_clock_rate} * (\text{arrival_time} - \text{current_capture_time})$
3. $\text{previous_capture_time} = \text{current_capture_time}$

The main problem with this method, in addition to the fact that the jitter calculation described in RFC 3550 cannot be used, is that it is dependent on the previous RTP packets, packets that can be reordered or lost in the network.

3.2.2. Non-monotonic timestamps

An alternate way of generating the RTP timestamps is to use the following formula:

```
timestamp = capture_time * clock_rate
```

With this formula, the jitter calculation is correct but the RTP timestamp values are no longer increasing monotonically as shown in Table 3 (Appendix A). RFC 3550 states that "[t]he sampling instant MUST be derived from a clock that increments monotonically[...]" but nowhere says that the RTP timestamp must increment monotonically.

The advantage with this method is that it works with the jitter calculation described in RFC 3550, as long as the correct clock rates are used. It seems that this is what most implementations are using (based on a survey done at Sipit26 and on a survey of open source implementations, see Appendix C).

4. Recommendations

The following subsections describe behavioral recommendations for RTP senders (with and without RTCP) and RTP receivers.

4.1. RTP Sender (with RTCP)

An RTP Sender with RTCP turned on MUST use a different SSRC for each different clock rate. An RTCP BYE MUST be sent and a new SSRC MUST be used if the clock rate switches back to a value already seen in the RTP stream.

To accelerate lip synchronization, the next compound RTCP packet sent by the RTP sender MUST contain multiple SR packets, the first one containing the mapping for the current clock rate and the subsequent SR packet(s) containing the mapping for the other clock rates seen during the last period.

The RTP extension defined in Perkins & Schierl [RFC6051] MAY be used to accelerate the synchronization.

4.2. RTP Sender (without RTCP)

An RTP Sender with RTCP turned off (i.e. having set the RS and RR bandwidth modifiers [RFC3556] to 0) SHOULD use a different SSRC for each different clock rate but MAY use different clock rates on the same SSRC as long as the RTP timestamp is calculated as explained below:

Each time the clock rate changes, the `start_offset` and `capture_start` values are calculated with the following formulas:

```
start_offset += (capture_time - capture_start) * previous_clock_rate
capture_start = capture_time
```

For the first RTP packet, the values are initialized with the following values:

```
start_offset = random_initial_offset
capture_start = capture_time
```

After eventually updating these values, the RTP timestamp is calculated with the following formula:

```
timestamp = (capture_time - capture_start) * clock_rate
            + start_offset
```

Note that in all the formulas, capture_start is the first instant that the new timestamp rate is used. The output of the above method is exemplified in Table 4 (Appendix A).

4.3. RTP Receiver

An RTP Receiver MUST calculate the jitter using the following formula:

```
D(i,j) = (arrival_time_j * clock_rate_i - timestamp_j)
          - (arrival_time_i * clock_rate_i - timestamp_i)
```

An RTP Receiver MUST be able to handle a compound RTCP packet with multiple SR packets.

5. Security Considerations

When the algorithm described in Section 4.1 is used the security considerations described in RFC 3550 apply.

The algorithm described in Section 4.2 is new and so its security properties were not considered in RFC 3550. Although the RTP timestamp is initialized with a random value like before, the timestamp value depends on the current and previous clock rates and this may or may not introduce a security vulnerability in the protocol.

6. IANA Considerations

This document requires no IANA actions.

7. Acknowledgements

Thanks to Colin Perkins, Ali C. Begen, Harald Alvestrand, Qin Wu, Jonathan Lennox, Barry Leiba, David Harrington, Stephen Farrell, Spencer Dawkins, Wassim Haddad and Magnus Westerlund for comments, suggestions and questions that helped to improve this document.

Thanks to Bo Burman (who provided the values in Table 4 of Appendix A).

Thanks to Robert Sparks and the attendees of SIPit 26 for the survey on multiple clock rates interoperability.

This document was written with the xml2rfc tool described in Rose [RFC2629].

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.

8.2. Informative References

- [I-D.ietf-avt-variable-rate-audio] Wenger, S. and C. Perkins, "RTP Timestamp Frequency for Variable Rate Audio Codecs", draft-ietf-avt-variable-rate-audio-00 (work in progress), October 2004.
- [RFC2629] Rose, M.T., "Writing I-Ds and RFCs using XML", RFC 2629, June 1999.
- [RFC3551] Schulzrinne, H. and S. Casner, "RTP Profile for Audio and Video Conferences with Minimal Control", STD 65, RFC 3551, July 2003.
- [RFC3556] Casner, S., "Session Description Protocol (SDP) Bandwidth Modifiers for RTP Control Protocol (RTCP) Bandwidth", RFC 3556, July 2003.
- [RFC3611] Friedman, T., Caceres, R., and A. Clark, "RTP Control Protocol Extended Reports (RTCP XR)", RFC 3611, November 2003.

- [RFC5450] Singer, D. and H. Desineni, "Transmission Time Offsets in RTP Streams", RFC 5450, March 2009.
- [RFC5484] Singer, D., "Associating Time-Codes with RTP Streams", RFC 5484, March 2009.
- [RFC5760] Ott, J., Chesterfield, J., and E. Schooler, "RTP Control Protocol (RTCP) Extensions for Single-Source Multicast Sessions with Unicast Feedback", RFC 5760, February 2010.
- [RFC6051] Perkins, C. and T. Schierl, "Rapid Synchronisation of RTP Flows", RFC 6051, November 2010.

Appendix A. Example Values

The following tables illustrate the timestamp and jitter values produced when the various methods discussed in the text are used.

The values shown are purely exemplary, illustrative and non-normative.

Capt. time	Clock rate	RTP timestamp	Arrival time	Transit	Jitter	Average jitter
0	8000	0	0.1	800		
0.02	8000	160	0.12	800	0	0
0.04	8000	320	0.14	800	0	0
0.06	8000	480	0.16	800	0	0
0.08	16000	800	0.18	2080	480	30
0.1	16000	1120	0.2	2080	0	28
0.12	16000	1440	0.22	2080	0	26
0.14	8000	1600	0.24	320	720	70
0.16	8000	1760	0.26	320	0	65

Table 2: Monotonic Timestamps

Capt. time	Clock rate	RTP timestamp	Arrival time	Transit	Jitter	Average jitter
0	8000	0	0.1	800		
0.02	8000	160	0.12	800	0	0
0.04	8000	320	0.14	800	0	0
0.06	8000	480	0.16	800	0	0
0.08	16000	1280	0.18	1600	0	0
0.1	16000	1600	0.2	1600	0	0
0.12	16000	1920	0.22	1600	0	0
0.14	8000	1120	0.24	800	0	0
0.16	8000	1280	0.26	800	0	0

Table 3: Non-monotonic Timestamps

Capt. time	Clock rate	RTP timestamp	Arrival time	Transit	Jitter	Average jitter
0	8000	0	0.1	800		
0.02	8000	160	0.12	800	0	0
0.04	8000	320	0.14	800	0	0
0.06	8000	480	0.16	800	0	0
0.08	16000	640	0.18	1600	0	0
0.1	16000	960	0.2	1600	0	0
0.12	16000	1280	0.22	1600	0	0
0.14	8000	1600	0.24	320	0	0
0.16	8000	1760	0.26	320	0	0

Table 4: Recommended Method for RTP Sender (without RTCP)

Appendix B. Using a Fixed Clock Rate

An alternate way of fixing the multiple clock rates issue was proposed by Wenger & Perkins [I-D.ietf-avt-variable-rate-audio]. This document proposed to define a unified clock rate, but the proposal was rejected at IETF 61.

Appendix C. Behavior of Legacy Implementations

C.1. libccrtp 2.0.2

This library uses the formula described in Section 3.2.2.

Note that this library uses `gettimeofday(2)` which is not guaranteed to increment monotonically, like when the clock is adjusted by NTP.

C.2. libmediastreamer0 2.6.0

This library (which uses the `oRTP` library) uses the formula described in Section 3.2.2.

Note that in some environments this library uses `gettimeofday(2)` which is not guaranteed to increment monotonically.

C.3. libpjmedia 1.0

This library uses the formula described in Section 3.2.2.

C.4. Android RTP stack 4.0.3

This library changes the SSRC each time the format changes, as described in Section 3.1.

Authors' Addresses

Marc Petit-Huguenin
Impedance Mismatch

Email: petithug@acm.org

Glen Zorn (editor)
Network Zen
227/358 Thanon Sanphawut
Bang Na, Bangkok 10260
Thailand

Phone: +66 (0) 8-1000-4155
Email: glenzorn@gmail.com

Network Working Group
Internet-Draft
Intended status: Informational
Expires: August 18, 2014

J. Lennox
Vidyo
K. Gross
AVA
S. Nandakumar
G. Salgueiro
Cisco Systems
B. Burman
Ericsson
February 14, 2014

A Taxonomy of Grouping Semantics and Mechanisms for Real-Time Transport
Protocol (RTP) Sources
draft-ietf-avtext-rtp-grouping-taxonomy-01

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 relationships among RTP sources, and attempts to define 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 August 18, 2014.

Copyright Notice

Copyright (c) 2014 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. Concepts	4
2.1. Media Chain	4
2.1.1. Physical Stimulus	8
2.1.2. Media Capture	8
2.1.3. Raw Stream	8
2.1.4. Media Source	9
2.1.5. Source Stream	10
2.1.6. Media Encoder	10
2.1.7. Encoded Stream	11
2.1.8. Dependent Stream	11
2.1.9. Media Packetizer	12
2.1.10. Packet Stream	12
2.1.11. Media Redundancy	13
2.1.12. Redundancy Packet Stream	14
2.1.13. Media Transport	14
2.1.14. Received Packet Stream	16
2.1.15. Received Redundancy Packet Stream	16
2.1.16. Media Repair	16
2.1.17. Repaired Packet Stream	17
2.1.18. Media Depacketizer	17
2.1.19. Received Encoded Stream	17
2.1.20. Media Decoder	17
2.1.21. Received Source Stream	18
2.1.22. Media Sink	18
2.1.23. Received Raw Stream	18
2.1.24. Media Render	18
2.2. Communication Entities	19
2.2.1. End Point	19
2.2.2. RTP Session	19
2.2.3. Participant	20
2.2.4. Multimedia Session	20
2.2.5. Communication Session	21
3. Relations at Different Levels	22
3.1. Media Source Relations	22
3.1.1. Synchronization Context	22
3.1.2. End Point	23
3.1.3. Participant	24

3.1.4. WebRTC MediaStream	24
3.2. Packetization Time Relations	24
3.2.1. Single and Multi-Session Transmission of SVC	24
3.2.2. Multi-Channel Audio	25
3.2.3. Redundancy Format	25
3.3. Packet Stream Relations	26
3.3.1. Simulcast	27
3.3.2. Layered Multi-Stream	28
3.3.3. Robustness and Repair	29
3.3.4. Packet Stream Separation	32
3.4. Multiple RTP Sessions over one Media Transport	33
4. Topologies and Communication Entities	33
4.1. Point-to-Point Communication	33
4.2. Centralized Conferencing	34
4.3. Full Mesh Conferencing	37
4.4. Source-Specific Multicast	39
5. Security Considerations	41
6. Acknowledgement	41
7. Contributors	41
8. IANA Considerations	41
9. References	41
9.1. Normative References	42
9.2. Informative References	42
Appendix A. Changes From Earlier Versions	44
A.1. Modifications Between WG Version -00 and -03	44
A.2. Modifications Between Version -02 and -03	44
A.3. Modifications Between Version -01 and -02	44
A.4. Modifications Between Version -00 and -01	44
Authors' Addresses	44

1. Introduction

The existing taxonomy of sources in RTP is often 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 attempts to provide some 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 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 an attempt is made to list any alternate definitions and usages that co-exist today 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 memo, Media is a sequence of synthetic or Physical Stimulus (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 stimulus. 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.
- o A Stream has one source transformation and one or more sink transformation (with the exception of Physical Stimulus (Section 2.1.1) that can have no 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, this may include loops if required by a particular transformation.

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 below first introduce the concepts for the sender side (Figure 1). This covers physical stimulus until media packets are emitted onto the network.

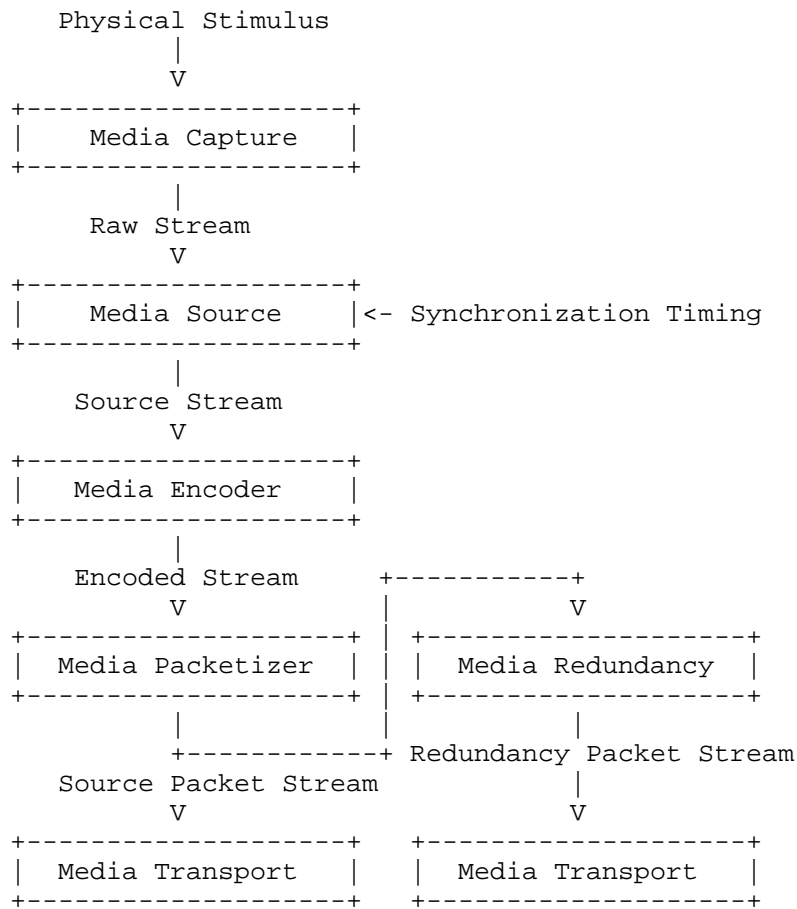


Figure 1: Sender Side Concepts in the Media Chain

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 below in Section 2.1.13.

Below we review a receiver media chain (Figure 2) matching the sender side to look at the inverse transformations and their attempts to recover possibly identical streams as in the sender chain. 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 Media Repair, if used, fails to repair. It should be noted that some transformations are not always present, like Media Repair that cannot operate without Redundancy Packet Streams.

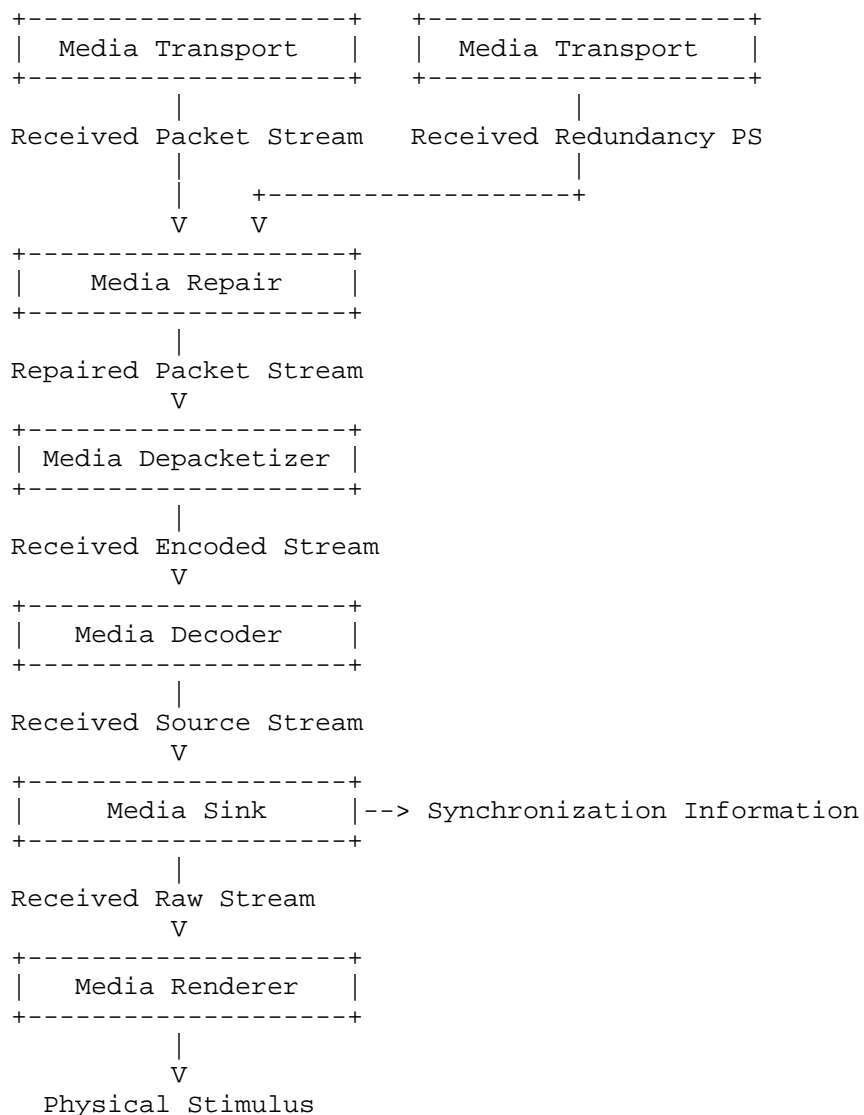


Figure 2: Receiver Side Concepts of the Media Chain

2.1.1. Physical Stimulus

The physical stimulus is a physical event that can be measured and converted to digital form by an appropriate sensor or transducer. This include sound waves making up audio, photons in a light field that is visible, 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 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 due to its periodical sampling, or at least being timed asynchronous events, some form of a stream of media data. 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.

Alternate usages:

- o The CLUE WG uses the term "Capture Device" to identify a physical capture device.
- o WebRTC WG uses the term "Recording Device" to refer to the locally available capture devices in an end-system.

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 support such a configuration.

2.1.3. Raw Stream

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 reference clock synchronized, time progressing, digital media stream, 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. This output has been synchronized with some reference clock, even if just a system local wall 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 Raw Streams (Figure 3), a mixed selection of the three loudest inputs regarding speech activity, a selection of a particular video based on the current speaker, i.e. typically based on other Media Sources.

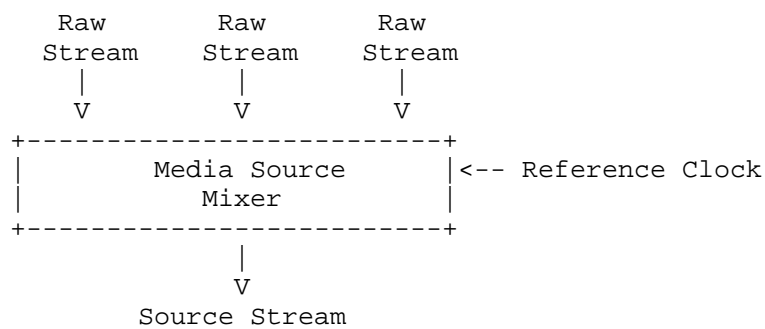


Figure 3: Conceptual Media Source in form of Audio Mixer

The CLUE WG uses the term "Media Capture" for this purpose. A CLUE Media Capture is identified via indexed notation. The terms Audio Capture and Video Capture are used to identify Audio Sources and Video Sources respectively. Concepts such as "Capture Scene", "Capture Scene Entry" and "Capture" provide a flexible framework to represent media captured spanning spatial regions.

The WebRTC WG defines the term "RtcMediaStreamTrack" to refer to a Media Source. An "RtcMediaStreamTrack" is identified by the ID attribute.

Typically a Media Source is mapped to a single m=line via the Session Description Protocol (SDP) [RFC4566] unless mechanisms such as Source-Specific attributes are in place [RFC5576]. In the latter cases, an m=line can represent either multiple Media Sources, multiple Packet Streams (Section 2.1.10), or both.

Characteristics:

- o At any point, it can represent a physical captured source or conceptual source.

2.1.5. Source Stream

A time progressing 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 bit-rate, start points for decoding, resolution, bandwidth or other fidelity affecting properties. The actually used codec is also an important factor in many communication systems, not only its parameters.

Scalable Media Encoders need special mentioning as they produce multiple outputs that are potentially of different types. 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) that requires at least one Encoded Stream and zero or more Dependent Streams to be possible to decode. A Dependent Stream's dependency is one of the grouping relations this document discusses further in Section 3.3.2.

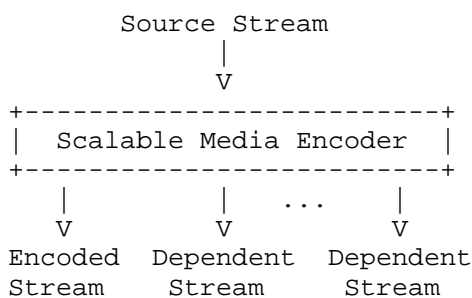


Figure 4: Scalable Media Encoder Input and Outputs

There are also other variants of encoders, like so-called Multiple Description Coding (MDC). Such Media Encoder produce multiple independent and thus individually decodable Encoded Streams that are possible to combine into a Received Source Stream that is somehow a better representation of the original Source Stream than using only a single Encoded Stream.

Alternate usages:

- o Within the SDP usage, an SDP media description (m=line) describes part of the necessary configuration required for encoding purposes.
- o CLUE's "Capture Encoding" provides specific encoding configuration for this purpose.

Characteristics:

- o A Media Source can be multiply encoded by different Media Encoders to provide various encoded representations.

2.1.7. Encoded Stream

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

Characteristics:

- o 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.

Characteristics:

- o Each Dependent Stream has a set of dependencies. These dependencies must be understood by the parties in a multi-media 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 Stream (Section 2.1.8) and put their content into one or more sequences of packets, normally RTP packets, and output Source Packet Streams (Section 2.1.10). This step includes both generating RTP payloads as well as RTP packets.

The Media Packetizer can use multiple inputs when producing a single Packet Stream. One such example is SST packetization when using SVC (Section 3.2.1).

The Media Packetizer can also produce multiple Packet Streams, for example when Encoded and/or Dependent Streams are distributed over multiple Packet Streams. One example of this is MST packetization when using SVC (Section 3.2.1).

Alternate usages:

- o An RTP sender is part of the Media Packetizer.

Characteristics:

- o The Media Packetizer will select which Synchronization source(s) (SSRC) [RFC3550] in which RTP sessions that are used.
- o Media Packetizer can combine multiple Encoded or Dependent Streams into one or more Packet Streams.

2.1.10. Packet Stream

A stream of RTP packets containing media data, source or redundant. The Packet 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 Packet Stream is a packet stream containing at least some content from an Encoded Stream. Source material is any media material that is produced for transport over RTP without any additional redundancy applied to cope with network transport losses. Compare this with the Redundancy Packet Stream (Section 2.1.12).

Alternate usages:

- o The term "Stream" is used by the CLUE WG to define an encoded Media Source sent via RTP. "Capture Encoding", "Encoding Groups" are defined to capture specific details of the encoding scheme.

- o RFC3550 [RFC3550] uses the terms media stream, audio stream, video stream and streams of (RTP) packets interchangeably. It defines the SSRC as the "The source of a stream of RTP packets, ...".
- o The equivalent mapping of a Packet Stream in SDP [RFC4566] is defined per usage. For example, each Media Description (m=line) and associated attributes can describe one Packet Stream OR properties for multiple Packet Streams OR for an RTP session (via [RFC5576] mechanisms for example).

Characteristics:

- o Each Packet Stream is identified by a unique Synchronization source (SSRC) [RFC3550] that is carried in every RTP and RTP Control Protocol (RTCP) packet header in a specific RTP session context.
- o At any given point in time, a Packet Stream can have one and only one SSRC. SSRC collision is a valid reason to change SSRC for a Packet Stream, since the Packet Stream itself is not changed in any way, only the identifying SSRC number.
- o Each Packet Stream defines a unique RTP sequence numbering and timing space.
- o Several Packet Streams may map to a single Media Source via the source transformations.
- o Several Packet Streams can be carried over a single RTP Session.

2.1.11. Media Redundancy

Media redundancy is a transformation that generates redundant or repair packets sent out as a Redundancy Packet Stream to mitigate network transport impairments, like packet loss and delay.

The Media Redundancy exists in many flavors; they may be generating independent Repair Streams that are used in addition to the Source Stream (RTP Retransmission [RFC4588] and some FEC [RFC5109]), they may generate a new Source Stream by combining redundancy information with source information (Using XOR FEC [RFC5109] as a redundancy payload [RFC2198]), or completely replace the source information with only redundancy packets.

2.1.12. Redundancy Packet Stream

A Packet Stream (Section 2.1.10) that contains no original source data, only redundant data that may be combined with one or more Received Packet Stream (Section 2.1.14) to produce Repaired Packet Streams (Section 2.1.17).

2.1.13. Media Transport

A Media Transport defines the transformation that the Packet 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 may contain multiple RTP receivers per sender). Each Media Transport is defined by a transport association that is identified by a 5-tuple (source address, source port, destination address, destination port, transport protocol). Each transport association normally contains only a single RTP session, although a proposal exists for sending multiple RTP sessions over one transport association [I-D.westerlund-avtcore-transport-multiplexing].

Characteristics:

- o Media Transport transmits Packet Streams of RTP Packets from a source transport address to a destination transport address.

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 5).

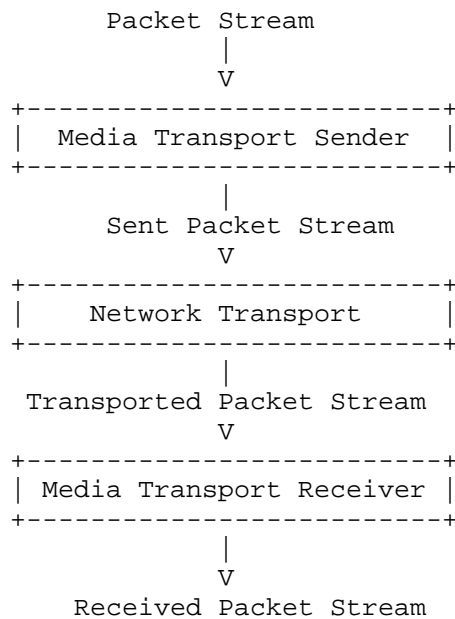


Figure 5: Decomposition of Media Transport

2.1.13.1. Media Transport Sender

The first transformation within the Media Transport (Section 2.1.13) is the Media Transport Sender, where the sending End-Point (Section 2.2.1) takes a Packet Stream and emits the packets onto the network using the transport association established for this Media Transport thus creating a Sent Packet Stream (Section 2.1.13.2). In this process it transforms the Packet Stream in several ways. First, it gains 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, pace or otherwise affect how the packets are emitted onto the network. Thus adding delay, jitter and inter packet spacings that characterize the Sent Packet Stream.

2.1.13.2. Sent Packet Stream

The Sent Packet Stream is the Packet Stream as entering the first hop of the network path to its destination. The Sent Packet 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)).

2.1.13.3. Network Transport

Network Transport is the transformation that the Sent Packet Stream (Section 2.1.13.2) is subjected to by traveling from the source to the destination through the network. These transformations include, loss of some packets, varying delay on a per packet basis, packet duplication, and packet header or data corruption. These transformations produces a Transported Packet Stream (Section 2.1.13.4) at the exit of the network path.

2.1.13.4. Transported Packet Stream

The Packet Stream that is emitted out of the network path at the destination, subjected to the Network Transport's transformation (Section 2.1.13.3).

2.1.13.5. Media Transport Receiver

The receiver End-Point's (Section 2.2.1) transformation of the Transported Packet Stream (Section 2.1.13.4) by its reception process that result in the Received Packet Stream (Section 2.1.14). This transformation includes transport checksums being verified and if non-matching, causing discarding of the corrupted packet. Other transformations can include delay variations in receiving a packet on the network interface and providing it to the application.

2.1.14. Received Packet Stream

The Packet Stream (Section 2.1.10) resulting from the Media Transport's transformation, i.e. subjected to packet loss, packet corruption, packet duplication and varying transmission delay from sender to receiver.

2.1.15. Received Redundancy Packet Stream

The Redundancy Packet Stream (Section 2.1.12) resulting from the Media Transport's transformation, i.e. subjected to packet loss, packet corruption, and varying transmission delay from sender to receiver.

2.1.16. Media Repair

A Transformation that takes as input one or more Source Packet Streams (Section 2.1.10) as well as Redundancy Packet Streams (Section 2.1.12) and attempts to combine them to counter the transformations introduced by the Media Transport (Section 2.1.13) to minimize the difference between the Source Stream (Section 2.1.5) and the Received Source Stream (Section 2.1.21) after Media Decoder

(Section 2.1.20). The output is a Repaired Packet Stream (Section 2.1.17).

2.1.17. Repaired Packet Stream

A Received Packet Stream (Section 2.1.14) for which Received Redundancy Packet Stream (Section 2.1.15) information has been used to try to re-create the Packet Stream (Section 2.1.10) as it was before Media Transport (Section 2.1.13).

2.1.18. Media Depacketizer

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

It should be noted that in practical implementations, the Media Depacketizer and the Media Decoder 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.

2.1.19. Received Encoded Stream

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

2.1.20. 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).

It should be noted that 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.

Alternate usages:

- o Within the context of SDP, an m=line describes the necessary configuration and identification (RTP Payload Types) required to decode either one or more incoming Media Streams.

Characteristics:

- o A Media Decoder is the entity that will have to deal with any errors in the encoded streams that resulted from corruptions or failures to repair packet losses. This as a media decoder generally is forced to produce some output periodically. It thus commonly includes concealment methods.

2.1.21. Received Source Stream

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

2.1.22. 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 sent in synchronization with the output from other Media Sinks to a Media Render (Section 2.1.24). 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.

Characteristics:

- o 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.

2.1.23. Received Raw Stream

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

2.1.24. 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, D/A converters connected to amplifiers and loudspeakers.

Characteristics:

- o An End Point can potentially have multiple Media Renders for each media type.

2.2. Communication Entities

This section contains concept for entities involved in the communication.

2.2.1. End Point

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 "End Point".

Alternate usages:

- o The CLUE Working Group (WG) uses the terms "Media Provider" and "Media Consumer" to describes aspects of End Point pertaining to sending and receiving functionalities.

Characteristics:

- o End Points 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 3.1.1). Thus one End Point can have multiple CNAMEs. Therefore, mechanisms outside the scope of RTP, such as application defined mechanisms, must be used to ensure End Point identification when outside this Synchronization Context.

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 Packet Streams. Within an RTP session, every participant can find meta-data and control information (over RTCP) about all the Packet Streams in the RTP session. The bandwidth of the RTCP control channel is shared between all participants within an RTP Session.

Alternate usages:

- o Within the context of SDP, a single m=line can map to a single RTP Session 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.

Characteristics:

- o Typically, an RTP Session can carry one or more Packet Streams.
- o An RTP Session shares a single SSRC space as defined in RFC3550 [RFC3550]. That is, the End Points participating in an RTP Session can see an SSRC identifier transmitted by any of the other End Points. An End Point 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 2.1.13), one for sending and one for receiving. Commonly, the receiving one is the reverse direction of the same one as used for sending. An RTP Session may use many Media Transports and these define the session's network interconnection topology. A single Media Transport can normally not transport more than one RTP Session, unless a solution for multiplexing multiple RTP sessions over a single Media Transport is used. One example of such a scheme is Multiple RTP Sessions on a Single Lower-Layer Transport [I-D.westerlund-avtcore-transport-multiplexing].
- o Multiple RTP Sessions can be related.

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 have several Multimedia Sessions (Section 2.2.4).
- o A participant can have several associated transport flows, including several separate local transport addresses for those transport flows.

2.2.4. Multimedia Session

A multimedia session is an association among a group of participants 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.

Alternate usages:

- o RFC4566 [RFC4566] defines a multimedia session as a set of multimedia senders and receivers and the data streams flowing from senders to receivers.
- o RFC3550 [RFC3550] defines it as set of concurrent RTP sessions among a common group of participants. For example, a video conference (which is a multimedia session) may contain an audio RTP session and a video RTP session.

Characteristics:

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

2.2.5. Communication Session

A Communication Session is an association among group of participants communicating with each other via a set of Multimedia Sessions.

Alternate usages:

- o The Session Description Protocol (SDP) [RFC4566] defines a multimedia session as a set of multimedia senders and receivers and the data streams flowing from senders to receivers. In that definition it is however not clear if a multimedia session includes both the sender's and the receiver's view of the same RTP Packet Stream.

Characteristics:

- o Each participant in a Communication Session is identified via an application-specific signaling address.
- o A Communication Session is composed of at least one Multimedia Session per participant, involving one or more parallel RTP Sessions with potentially multiple Packet 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. Relations at Different Levels

This section uses the concepts from previous section and look at different types of relationships among them. These relationships occur at different levels and for different purposes. The section is organized such as to look at the level where a relation is required. The reason for the relationship may exist at another step in the media handling chain. For example, using Simulcast (discussed in Section 3.3.1) needs to determine relations at Packet Stream level, however the reason to relate Packet Streams is that multiple Media Encoders use the same Media Source, i.e. to be able to identify a common Media Source.

3.1. Media Source Relations

Media Sources (Section 2.1.4) are commonly grouped and related to an End Point (Section 2.2.1) or a Participant (Section 2.2.3). This occurs for several reasons; both application logic as well as media handling purposes. These cases are further discussed below.

3.1.1. Synchronization Context

A Synchronization Context defines a requirement on a strong timing relationship between the Media Sources, typically requiring alignment of clock sources. Such 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.

A single Multimedia Session can contain media from one or more Synchronization Contexts. 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.

3.1.1.1. 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 [I-D.ietf-avtcore-clksrc], despite using NTP format timestamps, it is not required that the clock be synchronized to an NTP source.

3.1.1.2. Clock Source Signaling

[I-D.ietf-avtcore-clksrc] provides a mechanism to signal the clock source in SDP 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.

3.1.1.3. CLUE Scenes

In CLUE "Capture Scene", "Capture Scene Entry" and "Captures" define an implied Synchronization Context.

3.1.1.4. Implicitly via RtcMediaStream

The WebRTC WG defines "RtcMediaStream" with one or more "RtcMediaStreamTracks". All tracks in a "RtcMediaStream" are intended to be possible to synchronize when rendered.

3.1.1.5. Explicitly via SDP Mechanisms

RFC5888 [RFC5888] defines m=line grouping mechanism called "Lip Synchronization (LS)" for establishing the synchronization requirement across m=lines when they map to individual sources.

RFC5576 [RFC5576] extends the above mechanism when multiple media sources are described by a single m=line.

3.1.2. End Point

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

In RTP, this identification has been overloaded with the Synchronization Context through the usage of the source description CNAME item. This works for some usages, but sometimes it breaks down. For example, if an End Point 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. Thus, an End Point may have multiple CNAMEs. The CNAMEs or the Media Sources themselves can be related to the End Point.

3.1.3. Participant

In communication scenarios, it is commonly needed to know which Media Sources that originate from which Participant (Section 2.2.3). Thus enabling the application to for example display Participant Identity information correctly associated with the Media Sources. This association is currently 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 End Point.

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 Contributing Sources (CSRC) that carries such information about the previous step origin of the included media content on RTP level.

3.1.4. WebRTC MediaStream

An RtcMediaStream, in addition to requiring a single Synchronization Context as discussed above, is also an explicit grouping of a set of Media Sources, as identified by RtcMediaStreamTracks, within the RtcMediaStream.

3.2. Packetization Time Relations

At RTP Packetization time, there exists a possibility for a number of different types of relationships between Encoded Streams (Section 2.1.7), Dependent Streams (Section 2.1.8) and Packet Streams (Section 2.1.10). These are caused by grouping together or distributing these different types of streams into Packet Streams. This section will look at such relationships.

3.2.1. Single and Multi-Session Transmission of SVC

Scalable Video Coding [RFC6190] has a mode of operation called Single Session Transmission (SST), where Encoded Streams and Dependent Streams from the SVC Media Encoder are sent in a single RTP Session (Section 2.2.2) using the SVC RTP Payload format. There is another mode of operation where Encoded Streams and Dependent Streams are distributed across multiple RTP Sessions, called Multi-Session Transmission (MST). Regardless if used with SST or MST, as they are

defined, each of those RTP Sessions may contain one or more Packet Streams (SSRC) per Media Source.

To elaborate, what could be called SST-SingleStream (SST-SS) uses a single Packet Stream in a single RTP Session to send all Encoded and Dependent Streams. Similarly, SST-MultiStream (SST-MS) uses multiple Packet Streams in a single RTP Session to send the Encoded and Dependent Streams. MST-SS uses a single Packet Stream in each of multiple RTP Sessions and MST-MS uses multiple Packet Streams in each of the multiple RTP Sessions:

	Single RTP Session	Multiple RTP Sessions
Single Packet Stream	SST-SS	MST-SS
Multiple Packet Streams	SST-MS	MST-MS

3.2.2. Multi-Channel Audio

There exist a number of RTP payload formats that can carry multi-channel audio, despite the codec being a 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 then packetized together in a time synchronized way into a single Source Packet Stream using the used codec's RTP Payload format. Example of such codecs are, PCMA and PCMU [RFC3551], AMR [RFC4867], and G.719 [RFC5404].

3.2.3. Redundancy Format

The RTP Payload for Redundant Audio Data [RFC2198] defines how one can transport 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 below in Figure 6.

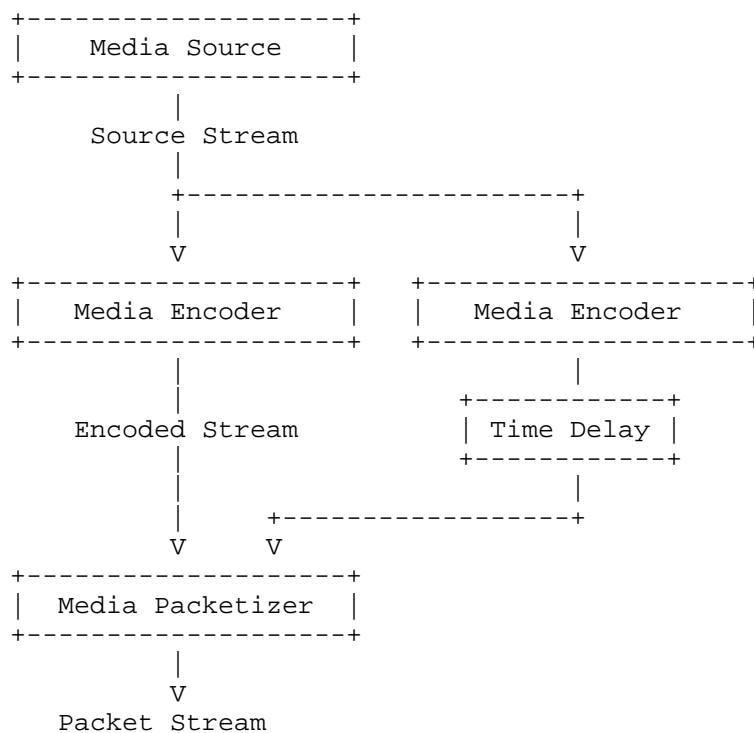


Figure 6: Concept for usage of Audio Redundancy with different Media Encoders

The Redundancy format is thus providing the necessary meta information to correctly relate different parts of the same Encoded Stream, or in the case depicted above (Figure 6) relate the Received Source Stream fragments coming out of different Media Decoders to be able to combine them together into a less erroneous Source Stream.

3.3. Packet Stream Relations

This section discusses various cases of relationships among Packet Streams. This is a common relation to handle in RTP due to that Packet Streams are separate and have their own SSRC, implying independent sequence numbers and timestamp spaces. The underlying reasons for the Packet Stream relationships are different, as can be seen in the cases below. The different Packet Streams can be handled within the same RTP Session or different RTP Sessions to accomplish different transport goals. This separation of Packet Streams is further discussed in Section 3.3.4.

3.3.1. Simulcast

A Media Source represented as multiple independent Encoded Streams constitutes a simulcast of that Media Source. Figure 7 below represents an example of a Media Source that is encoded into three separate and different Simulcast streams, that are in turn sent on the same Media Transport flow. When using Simulcast, the Packet Streams may be sharing RTP Session and Media Transport, or be separated on different RTP Sessions and Media Transports, or be any combination of these two. It is other considerations that affect which usage is desirable, as discussed in Section 3.3.4.

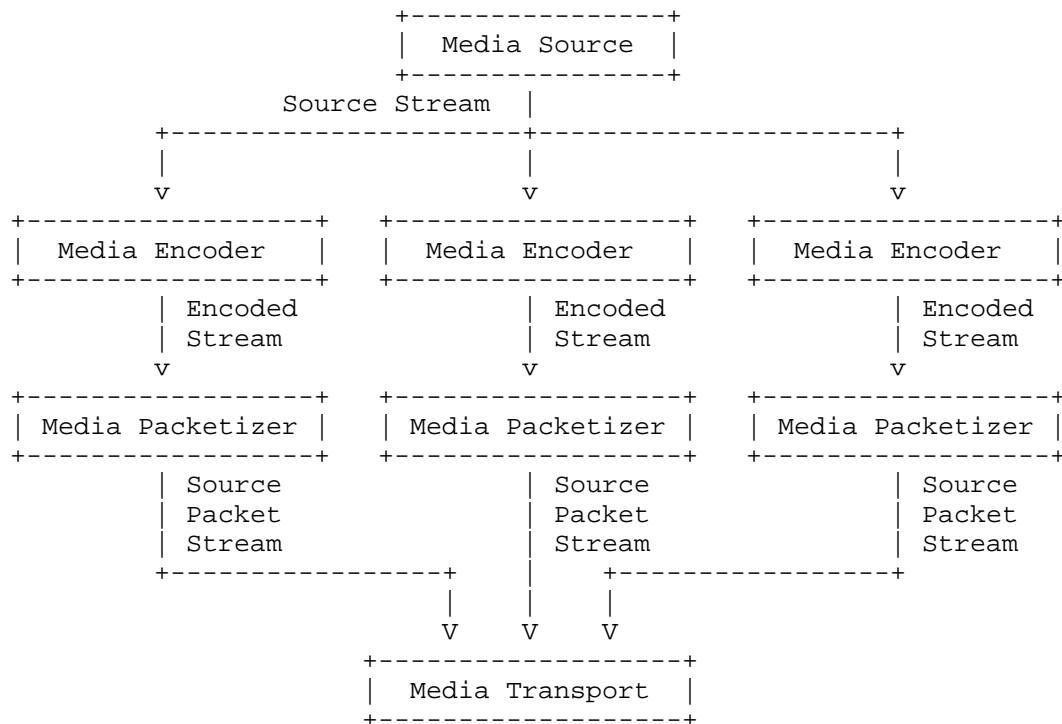


Figure 7: Example of Media Source Simulcast

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

3.3.2. Layered Multi-Stream

Layered Multi-Stream (LMS) is a mechanism by which different portions of a layered encoding of a Source Stream are sent using separate Packet 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. The figure below 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 Packet Streams, i.e. SSRCs, and the third layer is sent on a separate Media Transport, i.e. a different RTP Session.

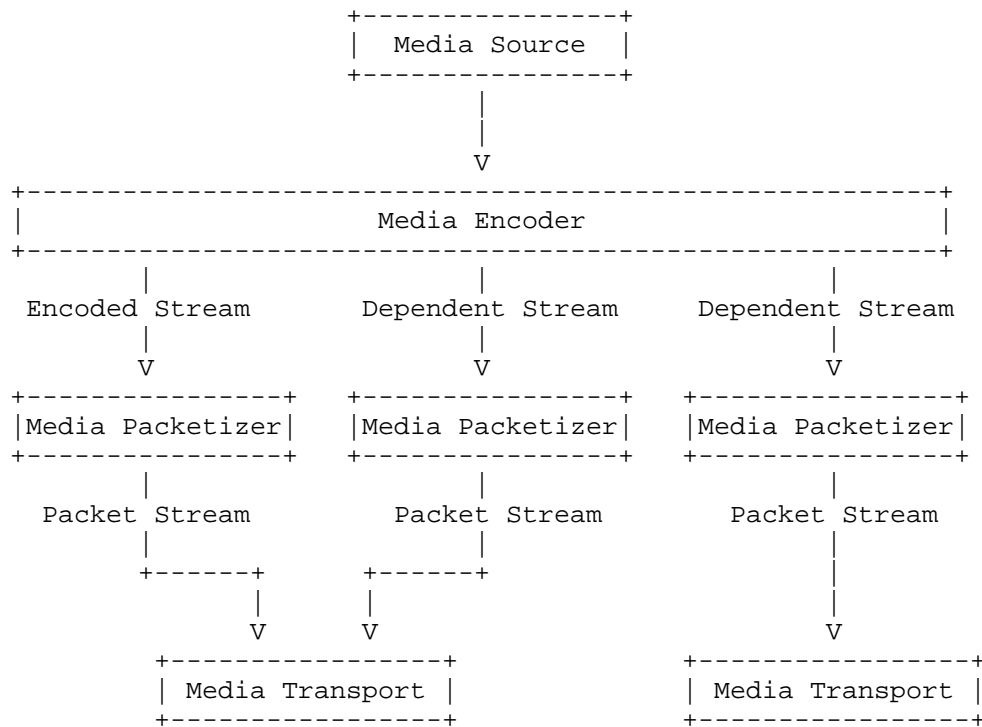


Figure 8: Example of Media Source Layered Dependency

As an example, the SVC MST (Section 3.2.1) relation needs to identify the common Media Encoder origin for the Encoded and Dependent Streams. The SVC RTP Payload RFC is not particularly explicit about how this relation is to be implemented. When using different RTP Sessions, thus different Media Transports, and as long as there is

only one Packet Stream per Media Encoder and a single Media Source in each RTP Session (MST-SS (Section 3.2.1)), common SSRC and CNAMEs can be used to identify the common Media Source. When multiple Packet Streams are sent from one Media Encoder in the same RTP Session (SST-MS), 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 need to be applied to create the MST relationship between the Packet Streams.

3.3.3. Robustness and Repair

Packet Streams may be protected by Redundancy Packet Streams during transport. Several approaches listed below can achieve the same result;

- o Duplication of the original Packet Stream
- o Duplication of the original Packet Stream with a time offset,
- o Forward Error Correction (FEC) techniques, and
- o Retransmission of lost packets (either globally or selectively).

3.3.3.1. RTP Retransmission

The figure below (Figure 9) represents an example where a Media Source's Source Packet Stream is protected by a retransmission (RTX) flow [RFC4588]. In this example the Source Packet Stream and the Redundancy Packet Stream share the same Media Transport.

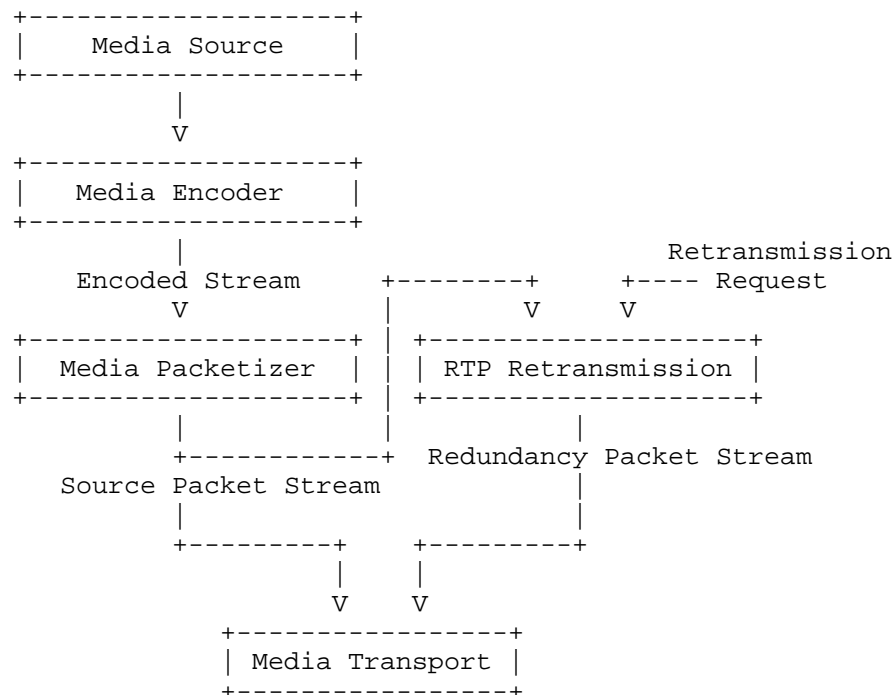


Figure 9: Example of Media Source Retransmission Flows

The RTP Retransmission example (Figure 9) helps illustrate that this mechanism works purely on the Source Packet Stream. The RTP Retransmission transform buffers the sent Source Packet Stream and upon requests emits a retransmitted packet with some extra payload header as a Redundancy Packet Stream. The RTP Retransmission mechanism [RFC4588] is specified so that there is a one to one relation between the Source Packet Stream and the Redundancy Packet Stream. Thus a Redundancy Packet Stream needs to be associated with its Source Packet Stream upon being received. This is done based on CNAME selectors and heuristics to match requested packets for a given Source Packet Stream with the original sequence number in the payload of any new Redundancy Packet Stream using the RTX payload format. In cases where the Redundancy Packet Stream is sent in a separate RTP Session from the Source Packet Stream, these sessions are related, e.g. using the SDP Media Grouping's [RFC5888] FID semantics.

3.3.3.2. Forward Error Correction

The figure below (Figure 10) represents an example where two Media Sources' Source Packet Streams are protected by FEC. Source Packet Stream A has a Media Redundancy transformation in FEC Encoder 1.

This produces a Redundancy Packet Stream 1, that is only related to Source Packet Stream A. The FEC Encoder 2, however takes two Source Packet Streams (A and B) and produces a Redundancy Packet Stream 2 that protects them together, i.e. Redundancy Packet Stream 2 relate to two Source Packet Streams (a FEC group). FEC decoding, when needed due to packet loss or packet corruption at the receiver, requires knowledge about which Source Packet Streams that the FEC encoding was based on.

In Figure 10 all Packet Streams are sent on the same Media Transport. This is however not the only possible choice. Numerous combinations exist for spreading these Packet Streams over different Media Transports to achieve the communication application's goal.

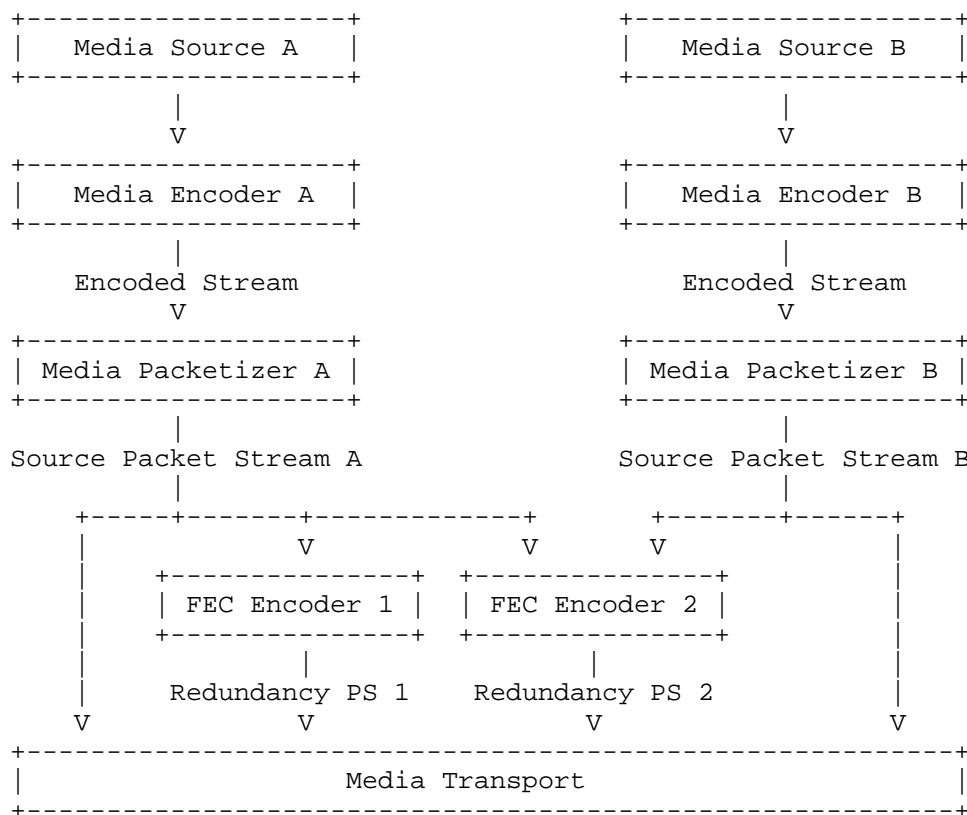


Figure 10: Example of FEC Flows

As FEC Encoding exists in various forms, the methods for relating FEC Redundancy Packet Streams with its source information in Source Packet Streams are many. The XOR based RTP FEC Payload format

[RFC5109] is defined in such a way that a Redundancy Packet Stream has a one to one relation with a Source Packet Stream. In fact, the RFC requires the Redundancy Packet Stream to use the same SSRC as the Source Packet Stream. This requires to either use a separate RTP session or to use the Redundancy RTP Payload format [RFC2198]. The underlying relation requirement for this FEC format and a particular Redundancy Packet Stream is to know the related Source Packet Stream, including its SSRC.

3.3.4. Packet Stream Separation

Packet Streams can be separated exclusively based on their SSRCs or at the RTP Session level or at the Multi-Media Session level as explained below.

When the Packet 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 End Point. [RFC5576]-based approaches, when used, can explicitly relate various such Packet Streams.

On the other hand, when Packet Streams that are related but 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 Packet 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 Packet Streams across different RTP Sessions, as explained in the previous section. Such a relationship can be used to perform inter-media synchronization.

Packet 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.

3.4. Multiple RTP Sessions over one Media Transport

[I-D.westerlund-avtcore-transport-multiplexing] describes a mechanism that allow 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. Thus using a common network path or potentially have different ones. There is reduced need for NAT/FW traversal resources and no need for flow based QoS.

However, Multiple RTP Sessions over one Media Transport makes it clear that a single Media Transport 5-tuple is not sufficient to express which RTP Session context a particular Packet Stream exists in. 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. Topologies and Communication Entities

This section reviews some communication topologies and looks at the relationship among the communication entities that are defined in Section 2.2. It does not deal with discussions about the streams and their relation to the transport. Instead, it covers the aspects that enable the transport of those streams. For example, the Media Transports (Section 2.1.13) that exists between the End Points (Section 2.2.1) that are part of an RTP session (Section 2.2.2) and their relationship to the Multi-Media Session (Section 2.2.4) between Participants (Section 2.2.3) and the established Communication session (Section 2.2.5) are explained.

The text provided below is neither any exhaustive listing of possible topologies, nor does it cover all topologies described in [I-D.ietf-avtcore-rtp-topologies-update].

4.1. Point-to-Point Communication

Figure 11 shows a very basic point-to-point communication session between A and B. It uses two different audio and video RTP sessions between A's and B's end points. Assume that the Multi-media session shared by the participants is established using SIP (i.e., there is a SIP Dialog between A and B). The high level representation of this communication scenario can be demonstrated using Figure 11.

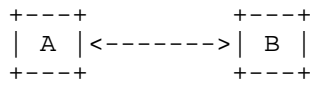


Figure 11: Point to Point Communication

However, this picture gets slightly more complex when redrawn using the communication entities concepts defined earlier in this document.

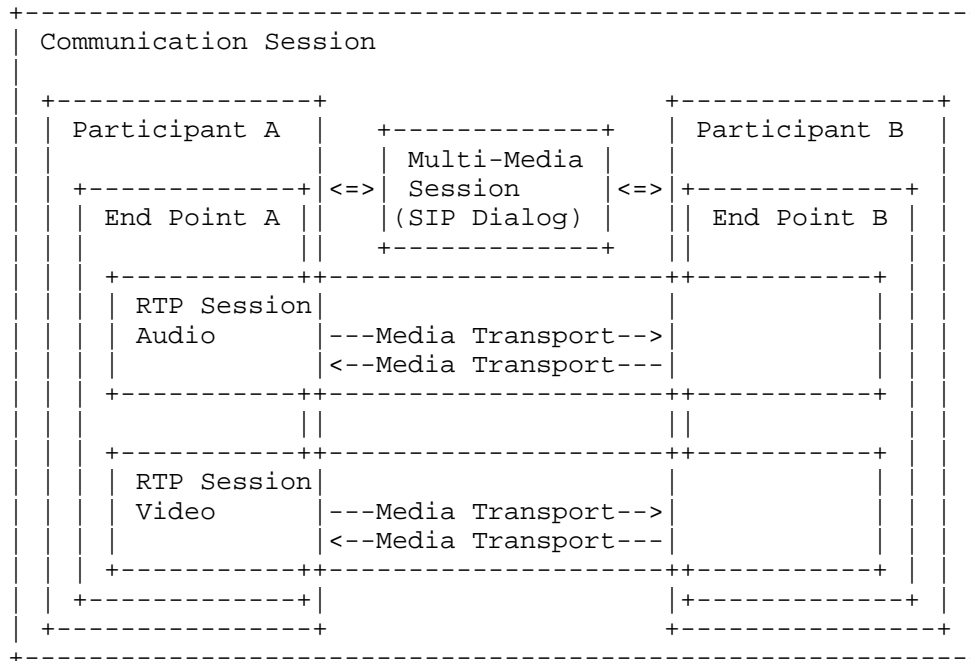


Figure 12: Point to Point Communication Session with two RTP Sessions

Figure 12 shows the two RTP Sessions only exist between the two End Points A and B and over their respective Media Transports. The Multi-Media Session establishes the association between the two Participants and configures these RTP sessions and the Media Transports that are used.

4.2. Centralized Conferencing

This section looks at the centralized conferencing communication topology, where a number of participants, like A, B, C, and D in Figure 13, communicate using an RTP mixer.



Figure 13: Centralized Conferencing using an RTP Mixer

In this case each of the Participants establish their Multi-media session with the Conference Bridge. Thus, negotiation for the establishment of the used RTP sessions and their configuration happens between these entities. The participants have their End Points (A, B, C, D) and the Conference Bridge has the host running the RTP mixer, referred to as End Point M in Figure 14. However, despite the individual establishment of four Multi-Media Sessions and the corresponding Media Transports for each of the RTP sessions between the respective End Points and the Conference Bridge, there is actually only two RTP sessions. One for audio and one for Video, as these RTP sessions are, in this topology, shared between all the Participants.

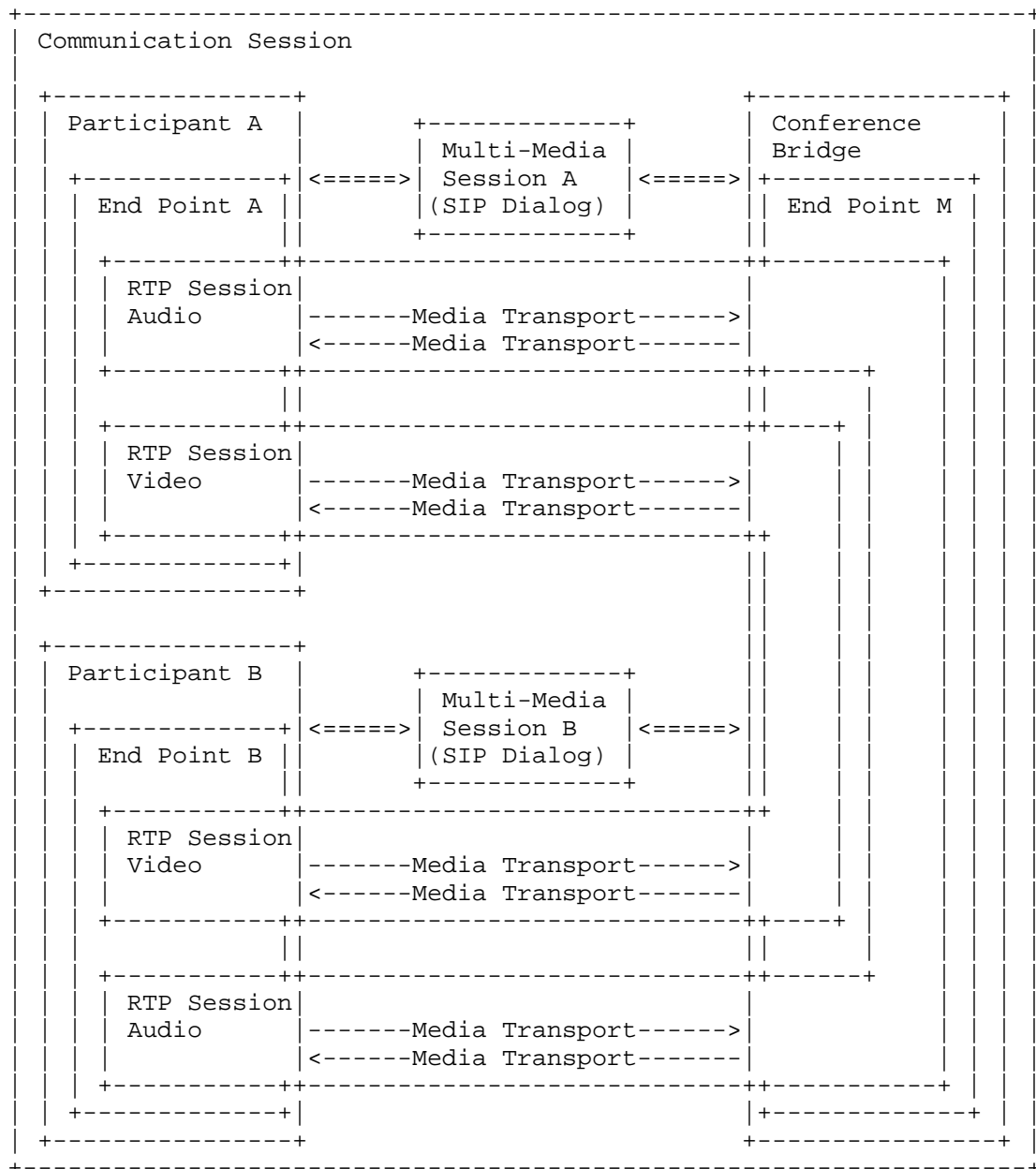


Figure 14: Centralized Conferencing with Two Participants A and B communicating over a Conference Bridge

It is important to stress that in the case of Figure 14, it might appear that the Multi-Media Sessions context is scoped between A and B over M. This might not be always true and they can have contexts that extend further. In this case the RTP session, its common SSRC space goes beyond what occurs between A and M and B and M respectively.

4.3. Full Mesh Conferencing

This section looks at the case where the three Participants (A, B and C) wish to communicate. They establish individual Multi-Media Sessions and RTP sessions between themselves and the other two peers. Thus, each providing two copies of their media to every other participant. Figure 15 shows a high level representation of such a topology.

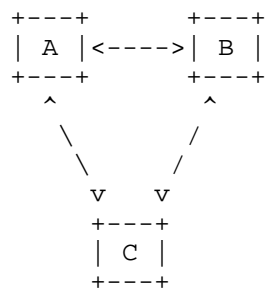


Figure 15: Full Mesh Conferencing with three Participants A, B and C

In this particular case there are two aspects worth noting. The first is there will be multiple Multi-Media Sessions per Communication Session between the participants. This, however, hasn't been true in the earlier examples; the Centralized Conferencing in Section 4.2 being the exception. The second aspect is consideration of whether one needs to maintain relationships between entities and concepts, for example Media Sources, between these different Multi-Media Sessions and between Packet Streams in the independent RTP sessions configured by those Multi-Media Sessions.

It is noteworthy that the full mesh conferencing topologies described here have the potential for creating loops. For example, if one compares the above full mesh with a mixing three party communication session as depicted in (Figure 17). In this example A's Media Source A1 is sent to B over a Multi-Media Session (A-B). In B the Media Source A1 is mixed with Media Source B1 and the resulting Media Source (MS AB) is sent to C over a Multi-Media Session (B-C). If C and A would establish a Multi-Media Session (A-C) and C would act in the same role as B, then A would receive a Media Source from C that contains a mix of A, B and C's individual Media Sources. This would result in A playing out a time delay version of its own signal (i.e., the system has created an echo path).

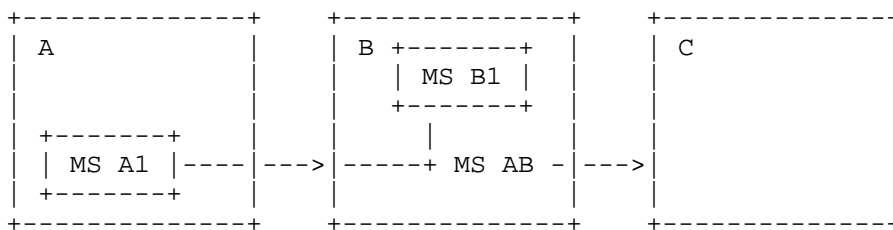
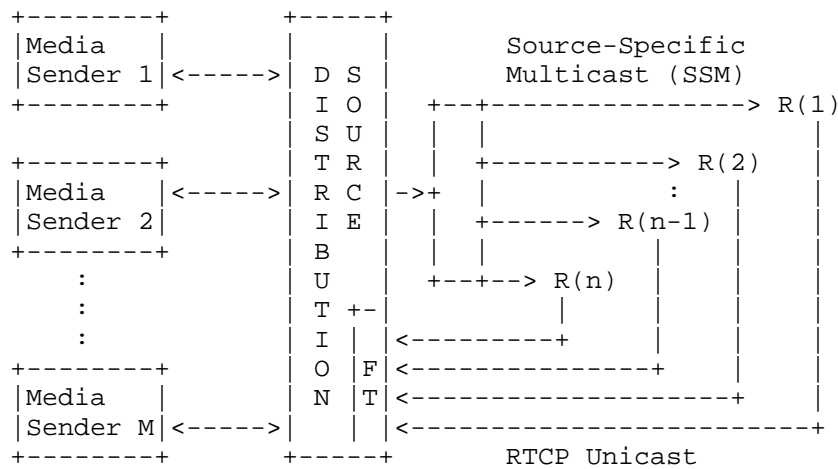


Figure 17: Mixing Three Party Communication Session

The looping issue can be avoided, detected or prevented using two general methods. The first method is to use great care when setting up and establishing the communication session if participants have any mixing or forwarding capacity, so that one doesn't end up getting back a partial or full representation of one's own media believing it is someone else's. The other method is to maintain some unique identifiers at the communication session level for all Media Sources and ensure that any Packet Streams received identify those Media Sources that contributed to the content of the Packet Stream.

4.4. Source-Specific Multicast

In one-to-many media distribution cases (e.g., IPTV), where one Media Sender or a set of Media Senders is allowed to send Packet Streams on a particular Source-Specific Multicast (SSM) group to many receivers (R), there are some different aspects to consider. Figure 18 presents a high level SSM system for RTP/RTCP defined in [RFC5760]. In this case, several Media Senders send their Packet Streams to the Distribution Source, which is the only one allowed to send to the SSM group. The Receivers joining the SSM group can provide RTCP feedback on its reception by sending unicast feedback to a Feedback Target (FT).



FT = Feedback Target

Figure 18: Source-Specific Multicast Communication Topology

Here the Media Transport from the Distribution Source to all the SSM receivers (R) have the same 5-tuple, but in reality have different paths. Also, the Multi-Media Sessions between the Distribution Source and the individual receivers are normally identical. This is due to one-way communication from the Distribution Source to the receiver of configuration information. This is information typically embedded in Electronic Program Guides (EPGs), distributed by the Session Announcement Protocol (SAP) [RFC2974] or other one-way protocols. In some cases load balancing occurs, for example, by providing the receiver with a set of Feedback Targets and then it randomly selects one out of the set.

This scenario varies significantly from previously described communication topologies due to the asymmetric nature of the RTP Session context across the Distribution Source. The Distribution Source forms a focal point in collecting the unicasted RTCP feedback from the receivers and then re-distributing it to the Media Senders. Each Media Sender and the Distribution Source establish their own Multi-Media Session Context for the underlying RTP Sessions but with shared RTCP context across all the receivers.

To improve the readability, Figure 18 intentionally hides the details of the various entities. Expanding on this, one can think of Media Senders being part of one or more Multi-Media Sessions grouped under a Communication Session. The Media Sender in this scenario refers to the Media Packetizer transformation Section 2.1.9. The Packet Stream generated by such a Media Sender can be part of its own RTP Session

or can be multiplexed with other Packet Streams within an End Point. The latter case requires careful consideration since the re-distributed RTCP packets now correspond to a single RTP Session Context across all the Media Senders.

5. Security Considerations

This document simply tries to clarify 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.

Hopefully 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], 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, and Alex Eleftheriadis.

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. References

9.1. Normative References

- [RFC3550] Schulzrinne, H., Casner, S., Frederick, R., and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications", STD 64, RFC 3550, July 2003.

9.2. Informative References

- [I-D.ietf-avtcore-clksrc]
Williams, A., Gross, K., Brandenburg, R., and H. Stokking, "RTP Clock Source Signalling", draft-ietf-avtcore-clksrc-09 (work in progress), December 2013.
- [I-D.ietf-avtcore-rtp-topologies-update]
Westerlund, M. and S. Wenger, "RTP Topologies", draft-ietf-avtcore-rtp-topologies-update-01 (work in progress), October 2013.
- [I-D.ietf-clue-framework]
Duckworth, M., Pepperell, A., and S. Wenger, "Framework for Telepresence Multi-Streams", draft-ietf-clue-framework-14 (work in progress), February 2014.
- [I-D.ietf-mmusic-sdp-bundle-negotiation]
Holmberg, C., Alvestrand, H., and C. Jennings, "Multiplexing Negotiation Using Session Description Protocol (SDP) Port Numbers", draft-ietf-mmusic-sdp-bundle-negotiation-05 (work in progress), October 2013.
- [I-D.ietf-rtcweb-overview]
Alvestrand, H., "Overview: Real Time Protocols for Browser-based Applications", draft-ietf-rtcweb-overview-08 (work in progress), September 2013.
- [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, September 1997.
- [RFC2974] Handley, M., Perkins, C., and E. Whelan, "Session Announcement Protocol", RFC 2974, October 2000.

- [RFC3264] Rosenberg, J. and H. Schulzrinne, "An Offer/Answer Model with Session Description Protocol (SDP)", RFC 3264, June 2002.
- [RFC3551] Schulzrinne, H. and S. Casner, "RTP Profile for Audio and Video Conferences with Minimal Control", STD 65, RFC 3551, July 2003.
- [RFC4566] Handley, M., Jacobson, V., and C. Perkins, "SDP: Session Description Protocol", RFC 4566, July 2006.
- [RFC4588] Rey, J., Leon, D., Miyazaki, A., Varsa, V., and R. Hakenberg, "RTP Retransmission Payload Format", RFC 4588, July 2006.
- [RFC4867] Sjoberg, J., Westerlund, M., Lakanienmi, 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, April 2007.
- [RFC5109] Li, A., "RTP Payload Format for Generic Forward Error Correction", RFC 5109, December 2007.
- [RFC5404] Westerlund, M. and I. Johansson, "RTP Payload Format for G.719", RFC 5404, January 2009.
- [RFC5576] Lennox, J., Ott, J., and T. Schierl, "Source-Specific Media Attributes in the Session Description Protocol (SDP)", RFC 5576, June 2009.
- [RFC5760] Ott, J., Chesterfield, J., and E. Schooler, "RTP Control Protocol (RTCP) Extensions for Single-Source Multicast Sessions with Unicast Feedback", RFC 5760, February 2010.
- [RFC5888] Camarillo, G. and H. Schulzrinne, "The Session Description Protocol (SDP) Grouping Framework", RFC 5888, June 2010.
- [RFC5905] Mills, D., Martin, J., Burbank, J., and W. Kasch, "Network Time Protocol Version 4: Protocol and Algorithms Specification", RFC 5905, June 2010.
- [RFC6190] Wenger, S., Wang, Y., Schierl, T., and A. Eleftheriadis, "RTP Payload Format for Scalable Video Coding", RFC 6190, May 2011.
- [RFC6222] Begen, A., Perkins, C., and D. Wing, "Guidelines for Choosing RTP Control Protocol (RTCP) Canonical Names (CNAMEs)", RFC 6222, April 2011.

Appendix A. Changes From Earlier Versions

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

A.1. Modifications Between WG Version -00 and -03

- 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.2. 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.3. 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.4. 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
Ericsson
Farogatan 6
SE-164 80 Kista
Sweden

Phone: +46 10 714 13 11
Email: bo.burman@ericsson.com

Network Working Group
Internet-Draft
Updates: 5104 (if approved)
Intended status: Standards Track
Expires: August 18, 2014

A. Akram
B. Burman
Ericsson
R. Even
Huawei Technologies
M. Westerlund
Ericsson
February 14, 2014

RTP Media Stream Pause and Resume
draft-westerlund-avtext-rtp-stream-pause-05

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) 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 August 18, 2014.

Copyright Notice

Copyright (c) 2014 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. Definitions	4
2.1. Abbreviations	4
2.2. Terminology	6
2.3. Requirements Language	7
3. Use Cases	7
3.1. Point to Point	7
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	9
3.5. Media Receiver to Media Sender Across RTP Mixer	10
4. Design Considerations	10
4.1. Real-time Nature	10
4.2. Message Direction	11
4.3. Apply to Individual Sources	11
4.4. Consensus	11
4.5. Acknowledgements	11
4.6. Retransmitting Requests	12
4.7. Sequence Numbering	12
5. Relation to Other Solutions	12
5.1. Signaling Technology Performance Comparison	12
5.2. CCM TMMBR / TMMBN	20
5.3. SDP "inactive" Attribute	21
5.4. Media Source Selection in SDP	21
5.5. Conclusion	22
6. Solution Overview	22
6.1. Expressing Capability	23
6.2. Requesting to Pause	23
6.3. Media Sender Pausing	25
6.4. Requesting to Resume	26
6.5. TMMBR/TMMBN Considerations	27

7.	Participant States	27
7.1.	Playing State	28
7.2.	Pausing State	28
7.3.	Paused State	29
7.3.1.	RTCP BYE Message	29
7.3.2.	SSRC Time-out	29
7.4.	Local Paused State	30
8.	Message Format	30
9.	Message Details	32
9.1.	PAUSE	33
9.2.	PAUSED	34
9.3.	RESUME	34
9.4.	REFUSE	35
9.5.	Transmission Rules	36
10.	Signalling	36
10.1.	Offer-Answer Use	39
10.2.	Declarative Use	40
11.	Examples	40
11.1.	Offer-Answer	41
11.2.	Point-to-Point Session	42
11.3.	Point-to-multipoint using Mixer	45
11.4.	Point-to-multipoint using Translator	47
12.	IANA Considerations	50
13.	Security Considerations	51
14.	Contributors	51
15.	Acknowledgements	51
16.	References	51
16.1.	Normative References	51
16.2.	Informative References	52
	Authors' Addresses	54

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] media stream from a specific source 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 Offer/Answer [RFC3264].

Centralized nodes, like RTP Mixers or MCUs, which either uses logic based on voice activity, other measurements, or user input could reduce the resources consumed in both the media sender and the network by temporarily pausing the media 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 media streams sent to the RTP Mixer may not need to be forwarded to any other participant. Those media streams could then be temporarily paused. This becomes especially useful when the media sources are provided in multiple encoding versions (Simulcast) [I-D.westerlund-avtcore-rtp-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 all of the time.

As the media streams required at any given point in time is highly dynamic in such scenarios, using the out-of-band signalling channel for pausing, and even more importantly resuming, a media stream is difficult due to the performance requirements. Instead, the pause and resume signalling should be in the media plane and go directly between the affected nodes. When using RTP [RFC3550] for media transport, using 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 media 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] and it is proposed to define 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 and resume, whenever possible.

2. Definitions

2.1. Abbreviations

3GPP: 3rd Generation Partnership Project

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

BGW: Border Gateway

CCM: Codec Control Messages (RFC 5104)

CNAME: Canonical Name (RTCP SDES)
CSRC: Contributing Source (RTP)
FB: Feedback (AVPF)
FCI: Feedback Control Information (AVPF)
FIR: Full Intra Refresh (CCM)
FMT: Feedback Message Type (AVPF)
LTE: Long-Term Evolution (3GPP)
MCU: Multipoint Control Unit
MTU: Maximum Transfer Unit
PT: Payload Type (RTP)
RTP: Real-time Transport Protocol (RFC 3550)
RTCP: Real-time Transport Control Protocol (RFC 3550)
RTCP RR: RTCP Receiver Report
SDP: Session Description Protocol (RFC 4566)
SGW: Signaling Gateway
SIP: Session Initiation Protocol (RFC 3261)
SSRC: Synchronization Source (RTP)
SVC: Scalable Video Coding
TCP: Transmission Control Protocol (RFC 793)
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 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 REFUSE as Notification.

PAUSE Request from a media receiver to pause a stream

RESUME Request from a media receiver to resume a paused stream

PAUSED Indication from a media sender that a stream is paused

REFUSE Notification from a media sender that a PAUSE or RESUME request will not be honored

Acknowledgement: The confirmation from receiver to sender that the message has been received.

Sender: The RTP entity that sends an RTP Packet Stream.

Receiver: The RTP entity that receives an RTP Packet Stream.

Mixer: The intermediate RTP node which receives a Packet Stream from different nodes, combines them to make one 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 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 receiver which sends a PAUSE request, defined in this specification, to other participant(s).

Stream: Used as a short term for Source Packet Stream, unless otherwise noted.

2.3. 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].

3. Use Cases

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

3.1. Point to Point

This is the most basic use case with an RTP session containing two end-points. Each end-point sends one or more streams.

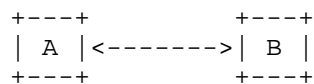


Figure 1: Point to Point

The usage of media stream pause in this use case is to temporarily halt media 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.

RTCWEB WG's use case and requirements document [I-D.ietf-rtcweb-use-cases-and-requirements] 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 memo provides means to optimize transport usage by stop sending muted streams and start sending again when unmuting.

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 SSRC, and any media sent to the participants will be sent using those SSRCs. 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 media 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].

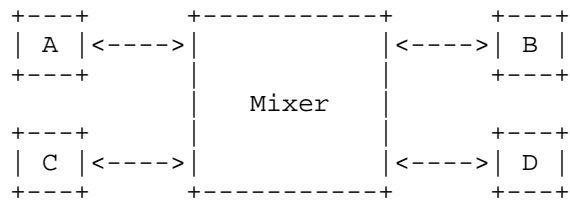


Figure 2: RTP Mixer in Unicast-only

Which streams 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.westerlund-avtcore-rtp-simulcast] or scalable encoded (for Multi-Stream Transmission) [RFC6190], thus providing multiple versions that can be delivered by the media 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 to pause a particular stream from being delivered to it. It also needs to be able to resume delivery with minimal delay.

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.

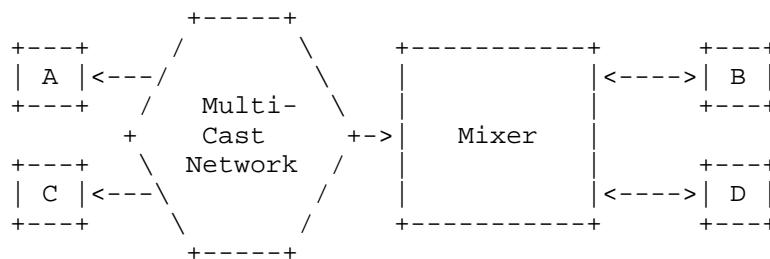


Figure 3: RTP Mixer in Point-to-Multipoint

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 special care must be taken as not to pause a stream that is still wanted by some receivers.

3.4. Media Receiver to RTP Mixer

An end-point in Figure 2 could potentially request to pause the delivery of a given media 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 end-point requesting media 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 end-point, like A in Figure 2, could potentially request to pause the delivery of a given media 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 media 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 end-point cannot rely on the CSRC to correctly identify the media stream to be paused when the delivered media is some type of mix. A more elaborate media stream identification solution is needed to support this in the general case.
3. The end-point cannot determine if a given media 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 real-time in order for them to be purposeful. The pause operation is arguably not very time critical 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.

4.2. Message Direction

It is the responsibility of a media receiver, who wants to pause or resume a media stream from the sender(s), to transmit PAUSE and RESUME messages. A media sender who likes to pause itself, can simply do it. Any indication that an RTP media stream is paused is the responsibility of the RTP media stream sender and may in some cases not even be needed by the media stream receiver.

4.3. Apply to Individual Sources

The PAUSE and RESUME messages apply to single RTP media 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 media stream sender should not pause an SSRC that some receiver still wishes to receive. The reason is that in RTP topologies where the media stream is shared between multiple receivers, a single receiver on that shared network, independent of it being multicast, a mesh with joint RTP session or a transport Translator based, must not single-handedly cause the media stream to be paused without letting all other receivers to voice their opinions on whether or not the stream should be paused. A consequence of this is that a newly joining receiver, for example indicated by an RTCP Receiver Report containing both a new SSRC and a CNAME that does not already occur in the session, firstly needs to learn the existence of paused streams, and secondly should be able to resume any paused stream. Any single receiver wanting to resume a stream should also cause it to be resumed.

4.5. Acknowledgements

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 end-point does not reach its destination, i.e. the targeted RTP media stream sender. When PAUSE or RESUME reaches the RTP media stream sender and are effective, i.e., an active media sender pauses, or a resuming have media data to transmit, it is immediately seen from the arrival or non-arrival of RTP packets for that RTP media stream. Thus, no explicit acknowledgements are required in this case.

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

4.6. Retransmitting Requests

When the media 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 that are more time critical, the best resume performance may be achieved by repeating the request as often as possible until a sufficient number have been sent to reach a high probability of request delivery, or the media stream gets delivered.

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.

5. Relation to Other Solutions

This section compares other possible solutions to achieve a similar functionality, along with motivations why the current solution is chosen.

5.1. Signaling Technology Performance Comparison

Editor's note: This section is related to the motivation for selecting RTCP as signaling technology rather than SIP/SDP and should be considered to be removed or at least significantly reduced if and when this draft is adopted as a working group

draft, since there now seems to be consensus that RTCP is the preferred technology.

This section contains what is thought to be a realistic estimate of one-way data transmission times for signaling implementing functionalities of this specification.

Two signaling protocols are compared. SIP is chosen to represent signaling in the control plane and RTCP is chosen to represent signaling in the media plane. For the sake of the comparison, each of these two protocols are listed with one favorable and one unfavorable condition to give the reader a hint of what range of delays that can be expected. The favorable condition is chosen as good as possible, while still realistic. The unfavorable condition is also chosen to be realistically occurring, and is not the worst possible or imaginable. Actual delays can in most cases be expected to lie somewhere between those two values.

It would also be possible to include a signaling protocol using a some dedicated signaling channel, separate from SIP and RTCP, into the comparison. Such signaling protocol can be expected to show performance somewhere in the range covered by the SIP and RTCP comparison below. The protocol can either use UDP as transport, like RTCP, or it can use TCP, like SIP, when the messages becomes too large for the MTU. The data sent on such channel can either be text based, in which case the amount of data can be similar to SIP, or it can be binary, in which case the amount of data can be similar to RTCP. Therefore, the dedicated signaling channel case is not described further in this specification.

Two different access technologies are compared:

- o Wired, fixed access is chosen as a representative low-delay alternative.
- o Mobile wireless access according to 3GPP LTE [TS36.201], also known as "4G", is chosen as a representative high-delay alternative.

NOTE: LTE is at the time of writing the most recent and best performing mobile wireless access. If an earlier mobile wireless access was to be used instead, the estimated transmission times would be considerably increased. For example, it is estimated that using 3GPP HSPA [TS25.308] (evolved 3G, just previous to LTE) would increase RTCP signaling times somewhat and significantly increase signaling times for SIP, although those estimates are too preliminary to provide any values here.

The target scenario includes two UA, residing in two different provider's (operator's) network. Those networks are assumed to be geographically close, that is no inter-continental transmission delays are included in the estimates.

Three signaling alternatives are compared:

- o Wireless UA to wireless UA, including two wireless links, uplink and downlink.
- o Wireless UA to media server (MCU), including a single wireless uplink.
- o Media server (MCU) to wireless UA, including a single wireless downlink.

The reason to include separate results for wireless uplink and downlink is that delay times can differ significantly.

The targeted topology is outlined in the following figure.

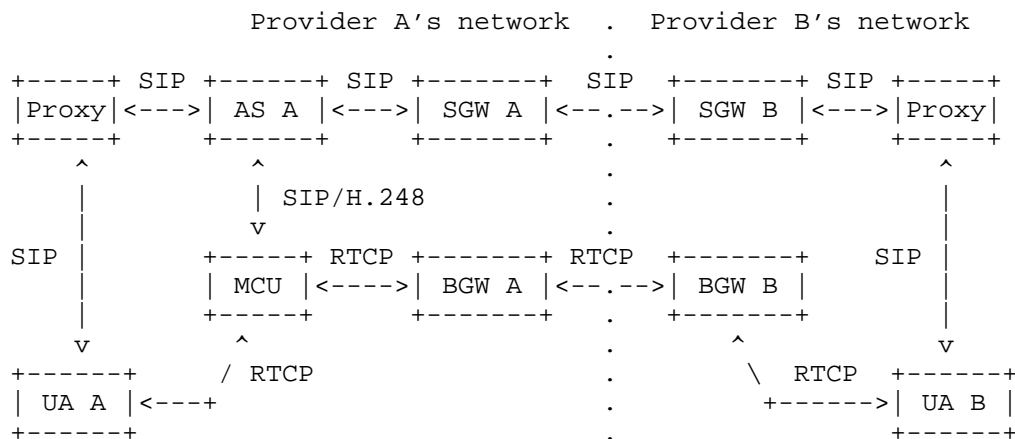


Figure 4: Comparison Signaling Topology

In the figure above, UA is a SIP User Agent, Proxy is a SIP Proxy, AS is an Application Server, MCU is a Multipoint Conference Unit, SGW a Signaling GateWay, and BGW a media Border GateWay.

It can be noted that when either one or both UAs use call forwarding or have roamed into yet another provider's network, several more signaling path nodes and a few more media path nodes could be included in the end-to-end signaling path.

The MCU is assumed to be located in one of the provider's network. Signaling delays between the MCU and a UA are presented as the average of MCU and UA being located in the same and different provider's networks.

These assumptions are used for SIP signaling:

- o A SIP UPDATE is used within an established session to dynamically impact individual streams to achieve the pause and resume functionality. The offer and answer SDP contains one audio and one video media, compliant with what is suggested in 3GPP MTSI [TS26.114], with the addition of SDP feedback message indication outlined in this specification (Section 10). A more complex media session with more streams would significantly add to the SDP size.
- o UDP is used as transport, except when risking to exceed MTU, in which case TCP is used instead. This is evaluated on a per-message basis.
- o Only SIP forward direction is included in the delay estimate, that is, delays needed to receive a response such as 200 OK are not included.
- o Favorable case:
 - * SIP SigComp [RFC5049] in dynamic mode is used for SIP and SDP signaling on the mobile link, reducing the SIP message size to approximately 1/3 of the original size.
- o Unfavorable case:
 - * SIP message is not compressed on the mobile link.
 - * SIP signaling on the mobile link uses a dedicated mobile wireless access radio channel that was idle for some time, has entered low power state and thus has to be re-established by radio layer signaling before any data can be sent.

These assumptions are used for RTCP signaling:

- o A minimal compound RTCP feedback packet is used, including one SR and one SDES with only the CNAME item present, with the addition of the feedback message outlined in Section 8.
- o RTCP bandwidth is chosen based on a 200 kbit/s session, which is considered to be a low bandwidth for media that would be worth pausing, and using the default 5% of this for RTCP traffic results

in 10 kbit/s. This low bandwidth makes RTCP scheduling delays be a significant factor in the unfavorable case.

- o Since there are random delay factors in RTCP transmission, the expected, most probable value is used in the estimates.
- o The mobile wireless access channel used for RTCP will always be active, that is there will be sufficient data to send at any time such that the radio channel will never have to be re-established. This is considered reasonable since it is assumed that the same channel is not only used for the messages defined in this specification, but also for other RTP and RTCP data.
- o Favorable case:
 - * It is assumed that AVPF Early or Immediate mode can always be used for the signaling described in this specification, since such signaling will be small in size and only occur occasionally in RTCP time scale.
 - * Early mode does not use dithering of send times ($T_{\text{dither_max}}$ is set to 0), that is, sender and receiver of the message are connected point-to-point. It can be noted that in case of a multiparty session where multiple end-points can see each others' messages, and unless the number of end-points is very large, it is very unlikely that more than a single end-point has the desire to send the same message (defined in this specification) as another end-point, and at almost exactly the same time. It is therefore arguably not very meaningful for messages in this specification to try to do feedback suppression by using a non-zero $T_{\text{dither_max}}$, even in multiparty sessions, but AVPF does not allow for any exemption from that rule.
 - * Reduced-size RTCP is used, which is considered appropriate for the type of messages defined in this specification.
 - * RTP/RTCP header compression [RFC5225] is not used, not even on the mobile link.
- o Unfavorable case:
 - * The expected, regular AVPF RTCP interval is used, including an expected value for timer re-consideration.
 - * A full, not reduced-size, minimal compound RTCP feedback packet without header compression is always used. No reduction of scheduling delays from the use of reduced-size RTCP is included

in the evaluation, since that would also require a reasonable estimate of the mix of compound and non-compound RTCP, which was considered too difficult for this study. The given unfavorable delays are thus an over-estimate compared to a more realistic case.

Common to both SIP and RTCP signaling estimates is that no UA processing delays are included. The reason for that decision is that processing delays are highly implementation and UA dependent. It is expected that wireless UA will be more limited than fixed UA by processing, but they are also constantly and quickly improving so any estimate will very quickly be outdated. More realistic estimates will however have to add such delays, which can be expected to be in the order of a few to a few tens of milliseconds. It is expected that SIP will be more penalized than RTCP by including processing delays, since it has larger and more complex messages. The processing may also include SigComp [RFC5049] compression and decompression in the favorable cases.

As a partial result, the message sizes can be compared, based on the messages defined in this specification (Section 8) and a SIP UPDATE with contents (Section 10) as discussed above. Favorable and unfavorable message sizes are presented as stacked bars in the figure below. Message sizes include IPv4 headers but no lower layer data, are rounded to the nearest 25 bytes, and the bars are to scale.

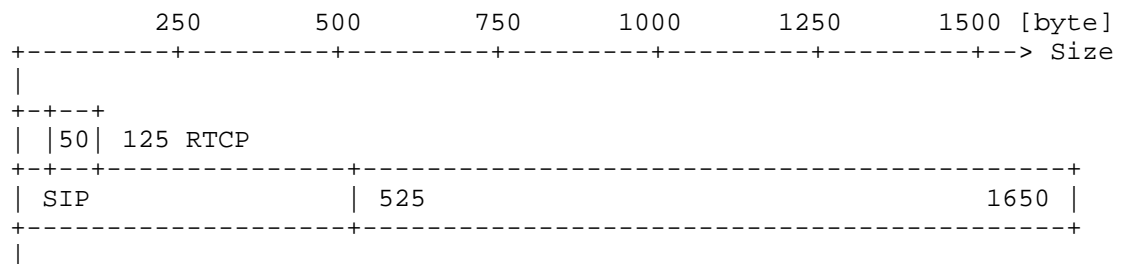


Figure 5: Message Size Comparison

The signaling delay results of the study are summarized in the following two figures. Favorable and unfavorable values are presented as stacked bars. Since there are many factors that impact the calculations, including some random processes, there are uncertainty in the calculations and delay values are thus rounded to nearest 5 ms. The bars are to scale.

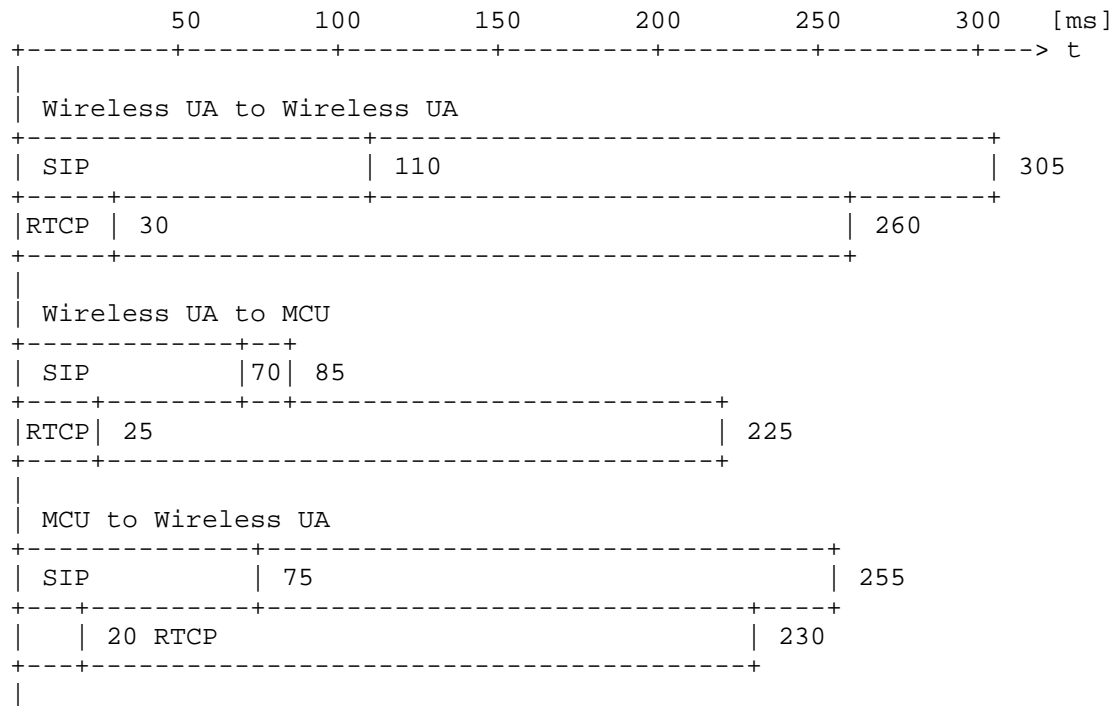


Figure 6: Mobile Access Transmission Delay Comparison

As can be seen, RTCP has a smaller signaling delay than SIP in a majority of cases for this mobile access. Non-favorable RTCP is however always worse than favorable SIP.

The UA to MCU signaling corresponds to the use case in Section 3.4. The reason that unfavorable SIP is more beneficial than unfavorable RTCP in this case comes from the fact that latency is fairly short to re-establish an uplink radio channel (as was assumed needed for unfavorable SIP), while unfavorable RTCP does not benefit from this since the delay is mainly due to RTCP Scheduling.

The MCU to UA signaling corresponds to the use case in Section 3.2. It has an unfavorable SIP signaling case with much longer delay than UA to MCU above, because the mixer cannot re-establish a downlink radio channel as quickly as the UA can establish an uplink. This case is applicable when an MCU wants to resume a paused stream, which is likely the most delay sensitive functionality, as discussed in Section 4.1.

Below are the same cases for fixed access depicted. Although delays are generally shorter, scales are kept the same for easy comparison with the previous figure.

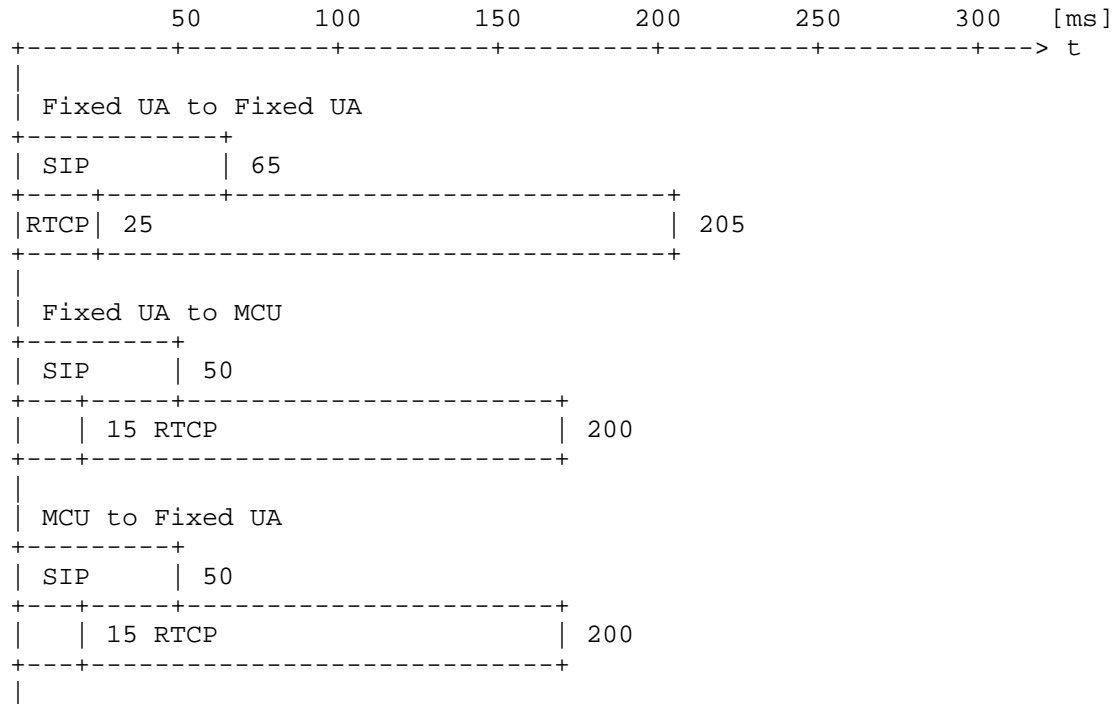


Figure 7: Fixed Access Transmission Delay Comparison

For fixed access, favorable RTCP is still significantly better than SIP, but unfavorable RTCP is significantly worse than SIP. There is no difference between favorable and unfavorable SIP, since in fixed access there is no channel that needs to be re-established.

Regarding the unfavorable values above, it should be possible with reasonable effort to design UA and network nodes that show favorable delays in a majority of cases.

For SIP, the major delays in the unfavorable cases above comes from re-establishing a radio bearer that has entered low power state due to inactivity, and large size SIP messages. The inactivity problem can be removed by using for example SIP keep-alive [RFC5626], at the cost of reduced battery life to keep the signaling radio bearer active, and some very minimal amount of extra data transmission. The large SIP messages can to some extent be reduced by SIP SigComp

[RFC5049]. It may however prove harder to reduce delays that comes from forwarding the SDP many times between different signaling nodes.

For RTCP, the major delays comes from low RTCP bandwidth and not being able to use Immediate or Early mode, including use of timer re-consideration. UAs and network nodes can explicitly allocate an appropriate amount of RTCP bandwidth through use of the b=RS and b=RR RTCP bandwidth SDP attributes [RFC3556]. For RTP media streams of higher bandwidth than the 200 kbit/s used in this comparison, which will be even more interesting to pause, RTCP bandwidth will per default also be higher, significantly reducing the signaling delays. For example, using a 1000 kbit/s media stream instead of a 200 kbit/s stream will reduce the unfavorable RTCP delays from 260 ms to 115 ms for Wireless-Wireless, from 225 ms to 80 ms for Wireless-MCU, and from 230 ms to 80 ms for MCU-Wireless.

5.2. CCM TMMBR / TMMBN

The Codec Control Messages specification [RFC5104] contains two messages, Temporary Maximum Media Bitrate Request (TMMBR) and Temporary Maximum Media Bitrate Notification (TMMBN), which could provide some of the necessary functionality. TMMBR with a bitrate value of 0 could effectively constitute a PAUSE request and TMMBN 0 could effectively be a PAUSED indication, and there are already implementations making use of TMMBR 0 in this way. It is possible to signal per SSRC (Section 4.3) and using the media path for signaling (AVPF) [RFC4585] will in most cases provide the shortest achievable signaling delay (Section 4.1). However, in some cases the defined semantics for TMMBR differ from what is required for PAUSE.

When there is only a single receiver of a media stream, TMMBR 0 and PAUSE are effectively identical.

When there are several receivers of the same media stream, the stream must not be paused until there are no receiver that desires to receive it (Section 4.4), for example there is no disapproving RESUME for a PAUSE. In the presence of several simultaneous receivers, the TMMBR semantics is the opposite; the first media receiver that sends TMMBR 0 will pause the stream for all receivers.

When there is only a single receiver of a media stream that is paused, TMMBR with a bitrate greater than 0 can effectively function as a RESUME, resuming the media stream immediately as needed (Section 4.4).

For the case of multiple simultaneous receivers, TMMBR specifies to use a guard period when increasing the bandwidth. In this case, TMMBR/TMMBN semantics (Section 4.2.1.2 of [RFC5104]) requires a media

sender to wait $2*RTT+T_dither_max$ after having sent a TMMBN, indicating the intention to increase the bandwidth, before it actually increases its bandwidth usage. The RTT is specified to be the longest the media sender knows in the RTP session. So, there is both the delay between the media sender receiving the TMMBR until it can send a TMMBN, and the above delay for the guard period before the media sender are allowed to resume transmission. This delay before resuming transmission is the most time critical operation in this solution, making use of TMMBR as RESUME according to the defined semantics infeasible in practice when there are multiple simultaneous media stream receivers.

5.3. SDP "inactive" Attribute

In SDP [RFC4566], an "inactive" attribute is defined on media level and session level. The attribute is intended to be used to put media "on hold", either at the beginning of a session or as a result of session re-negotiation [RFC3264], for example using SIP re-INVITE [RFC3261], possibly in combination with ITU-T H.248 media gateway control.

This attribute is only possible to specify with media level resolution, is not possible to signal per individual media stream (SSRC) (Section 4.3), and is thus not usable for RTP sessions containing more than a single SSRC.

There is a per-ssrc attribute defined in [RFC5576], but that does currently not allow to set an individual stream (SSRC) inactive.

Using "inactive" does thus not provide sufficient functionality for the purpose of this specification.

5.4. Media Source Selection in SDP

There is a draft that selects sources based on SDP [I-D.lennox-mmusic-sdp-source-selection] information. It builds on the per-ssrc attribute [RFC5576] discussed above (Section 5.3).

The semantics differ between selecting a Media Source and pause / resume for a stream in topologies other than point-to-point. For example, in RTP Receiver to Mixer (Section 3.4), pausing a stream (SSRC) from the mixer should stop it being received altogether, while excluding a stream (CSRC) from the mix would just avoid that specific Media Source being included in the stream from the mixer. There is a similar difference between resuming a stream (SSRC) from the mixer and allowing a Media Source (CSRC) to be included in the mix again. This suffers from a lack of functionality for consensus (Section 4.4)

and would likely also suffer from lower real-time performance (Section 4.1).

5.5. Conclusion

As can be seen from Section 5.1, 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 end-point 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.

As seen in the same section, the RTCP data is sent through the media path, which is likely shorter (contains fewer intermediate nodes) than the signaling path but 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 reasons, RTCP is proposed as signaling protocol for the pause and resume functionality. Much of the wanted functionality can in a point-to-point case be achieved with the existing TMMBR/TMMBN CCM messages [RFC5104], but they cannot be used when the media stream is sent to multiple simultaneous receivers.

6. 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 media stream targeted at the media stream sender, as identified by the sender SSRC.

It is proposed to re-use CCM TMMBR and TMMBN [RFC5104] to the extent possible, and to define a small set of new RTCP feedback messages where new semantics is needed. Considerations that apply when using TMMBR/TMMBN for pause and resume purposes are also described.

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 REFUSE response, or an extension to this specification. This structure allows a single feedback message to handle pause functionality on a number of media streams.

The PAUSED functionality is also defined in such a way that it can be used standalone by the media 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.

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.

6.1. Expressing Capability

An end-point can use an extension to CCM SDP signaling to declare capability to understand the messages defined in this specification. Capability to understand PAUSED indication is defined separately from the others to support partial implementation, which is specifically believed to be feasible for the RTP Mixer to Media Sender use case (Section 3.2).

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.

6.2. Requesting to Pause

An RTP media stream receiver can choose to request PAUSE at any time, subject to AVPF timing rules. This also applies when using TMMBR 0 in the point-to-point case.

The PAUSE request contains a PauseID, which is incremented by one (in modulo arithmetic) with each PAUSE request that is not a re-transmission. The PauseID is scoped by and thus a property of the targeted RTP media stream (SSRC).

When a non-paused RTP media stream sender receives the PAUSE request, it continues to send media while waiting for some time to allow other RTP media stream receivers in the same RTP session that saw this PAUSE request to disapprove by sending a RESUME (Section 6.4) for the same stream and with the same PauseID as in the disapproved PAUSE. If such disapproving RESUME arrives at the RTP media stream sender during the wait period before the stream is paused, the pause is not performed. In point-to-point configurations, the wait period may be set to zero. Using a wait period of zero is also appropriate when using TMMBR 0 and in line with the semantics for that message.

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

If the PAUSE request or TMMBR 0 is lost before it reaches the RTP media stream sender, it will be discovered by the RTP media stream receiver because it continues to receive the RTP media stream. It will also not see any PAUSED indication (Section 6.3) or TMMBN 0 for the stream. The same condition can be caused by the RTP media stream sender having received a disapproving RESUME from a media stream receiver A for a PAUSE request sent by a media 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 PauseID. If using TMMBR 0 the request MAY be re-transmitted when the requestor fails to receive a TMMBN 0 confirmation.

If the pending stream pause is aborted due to a disapproving RESUME, the PauseID from the disapproved PAUSE is invalidated by the RESUME and any new PAUSE must use an incremented PauseID (in modulo arithmetic) to be effective.

An RTP media stream sender receiving a PAUSE not using the available PauseID informs the RTP media stream receiver sending the ineffective PAUSE of this condition by sending a REFUSE response that contains the next available PauseID value. This REFUSE also informs the RTP media stream receiver that it is probably not feasible to send another PAUSE for some time, not even with the available PauseID, since there are other RTP media stream receivers that wish to receive the stream.

A similar situation where an ineffective PauseID is chosen can appear when a new RTP media stream receiver joins a session and wants to PAUSE a stream, but does not yet know the available PauseID to use. The REFUSE response 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 media stream sender can then respond with a REFUSE. In this case, if the used PauseID would otherwise have been effective, the REFUSE contains the same PauseID as in the PAUSE request, and the PauseID is kept as available.

If the RTP media stream sender receives several identical PAUSE for an RTP media stream that was already at least once responded with REFUSE and the condition causing REFUSE remains, those additional

REFUSE should be sent with regular RTCP timing. A single REFUSE can respond to several identical PAUSE requests.

6.3. Media Sender Pausing

An RTP media 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 available PauseID. If the stream was paused by a TMMBR 0, TMMBN 0 is used as PAUSED indication. What is said on PAUSED in the rest of this paragraph apply also to the use of TMMBN 0, except for PAUSED message parameters. Note that PauseID is incremented when pausing locally (without having received a PAUSE). It also sends the PAUSED indication in the next two regular RTCP reports, given that the pause condition is then still effective.

The RTP media stream sender may want to apply some local consideration to exactly when the 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 media stream receivers with first hand 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 media 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 media stream sender receives PAUSE requests with the available 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 SR or RR. Including PAUSED in this way allows RTP media 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 next available PauseID is to be able to construct valid PAUSE and RESUME requests at a later stage.

When the RTP media stream sender learns that a new end-point 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.

6.4. Requesting to Resume

An RTP media stream receiver can request to resume a stream with a RESUME request at any time, subject to AVPF timing rules. If the stream was paused with TMMBR 0, resuming the stream is made with TMMBR containing a bitrate value larger than 0. The bitrate value used when resuming after a PAUSE with TMMBR 0 is either according to known limitations, or the configured maximum for the stream or session. What is said on RESUME in the rest of this paragraph apply also to the use of TMMBR with a bitrate value larger than 0, except for RESUME message parameters.

The RTP media stream receiver must include the available PauseID in the RESUME request for it to be effective.

A pausing RTP media stream sender that receives a RESUME including the correct available 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, 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 media stream sender MAY then respond with a REFUSE containing the same PauseID as in the RESUME. When receiving such REFUSE with a PauseID identical to the one in the sent RESUME, RTP media stream receivers SHOULD then avoid sending further RESUME requests for some reasonable amount of time, to allow the condition to clear.

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

When resuming a paused media stream, especially for media that makes use of temporal redundancy between samples such as video, the temporal dependency between samples taken before the pause and at the time instant the stream is resumed may not be appropriate to use in the encoding. Should such temporal dependency between before and after the media was paused be used by the media sender, it requires the media receiver to have saved the sample from before the pause for successful continued decoding when resuming. The use of this temporal dependency is left up to the media sender. If temporal dependency is not used when media is resumed, the first encoded sample after the pause will not contain any temporal dependency to samples before the pause (for video it may be a so-called intra picture). If temporal dependency to before the pause is used by the

media sender when resuming, and if the media receiver did not save any sample from before the pause, the media 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.

6.5. TMMBR/TMMBN Considerations

As stated, 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 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 media stream.

TMMBR >0: Corresponds to RESUME when the media stream was previously paused with TMMBR 0. Since there is only a single media receiver, there is no need for the media sender to delay resuming the media stream until after sending TMMBN >0, or to apply the hold-off period specified in [RFC5104] before increasing the bitrate from zero.

TMMBN 0: Corresponds to PAUSED. Also corresponds to a REFUSE indication when a media stream is requested to be resumed with TMMBR >0.

TMMBN >0: Corresponds to a REFUSE indication when a media stream is requested to be paused with TMMBR 0.

7. Participant States

This document introduces three new states for a media stream in an RTP sender, according to the figure and sub-sections below. Any references to PAUSE, PAUSED, RESUME and REFUSE in this section SHALL be taken to apply to the extent possible also when TMMBR/TMMBN are used (Section 6.5) for this functionality.

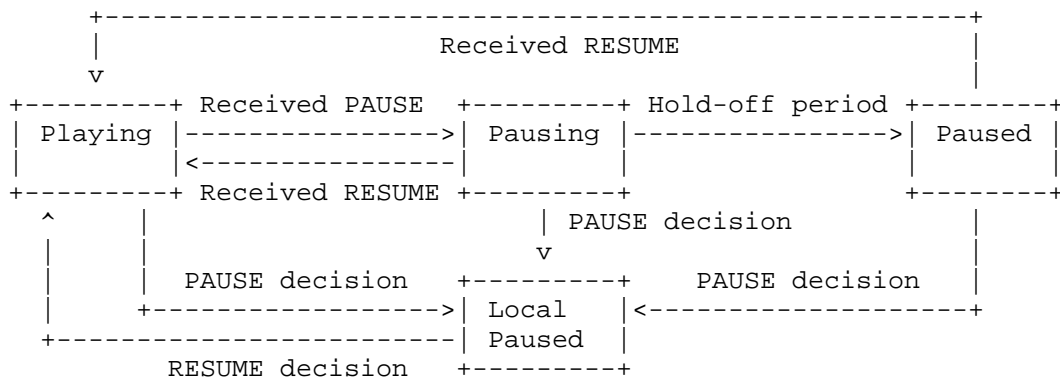


Figure 8: RTP Pause States

7.1. Playing State

This state is not new, but is the normal media sending state from [RFC3550]. When entering the state, the 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.

7.2. Pausing State

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

$$2 * RTT + T_dither_max,$$

where RTT is the longest round trip known to the media 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 10) 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 media 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, for example by RTCP RR, it MUST change the hold-off to instead be based on the above formula.

7.3. Paused State

An RTP media 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.

When entering the state, the media sender SHALL send a PAUSED indication to all known media receivers, and SHALL also repeat PAUSED in the next two regular RTCP reports.

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.

7.3.1. 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 media stream to pause. If that receiver sends an RTCP BYE message observed by the sender, the sender SHALL resume the RTP media stream.

7.3.2. 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 SHALL be resumed.

7.4. Local Paused State

This state can be entered at any time, based on local decision from the media sender. As for Paused State (Section 7.3), the media sender SHALL send a PAUSED indication to all known media receivers, when entering the state, and repeat it in the next two regular RTCP reports.

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

8. 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, this message is either a PAUSE request, a RESUME request, or one of four different types of acknowledgements in response to either PAUSE or RESUME requests.

The Transport layer feedback messages are identified by having the RTCP payload type be RTPFB (205) as defined by AVPF [RFC4585]. The PAUSE and RESUME messages are identified by Feedback Message Type (FMT) value in common packet header for feedback message defined in section 6.1 of AVPF [RFC4585]. The PAUSE and RESUME transport feedback message is identified by the FMT value = TBA1.

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

```

      0               1               2               3
      0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|V=2|P|   FMT   |         PT         |           Length           |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|                                     SSRC of packet sender         |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|                                     SSRC of media source          |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
:                               Feedback Control Information (FCI)   :
:                                                                       :

```

For the PAUSE and RESUME messages, the following interpretation of the packet fields will be:

FMT: The FMT value identifying the PAUSE and RESUME message: 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 end-points 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 consist of one or more PAUSE, RESUME, PAUSED, REFUSE, or any future extension. These messages have the following FCI format:

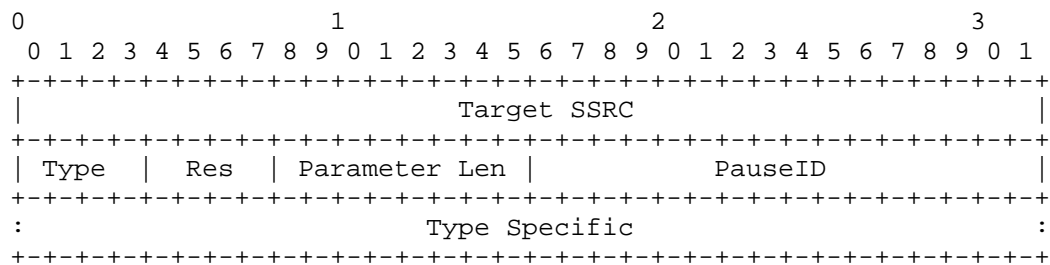


Figure 9: Syntax of FCI Entry in the PAUSE and RESUME message

The FCI fields have the following definitions:

Target SSRC (32 bits): For a PAUSE and RESUME messages, 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 REFUSE, 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: REFUSE indication message
- 4-15: Reserved for future use

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. 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.

9. Message Details

This section contains detailed explanations of each message defined in this specification. All transmissions of request and indications are governed by the transmission rules as defined by Section 9.5.

Any references to PAUSE, PAUSED, RESUME and REFUSE in this section SHALL be taken to apply to the extent possible also when TMMBR/TMMBN are used (Section 6.5) for this functionality. TMMBR/TMMBN MAY be used instead of the messages defined in this specification when the effective topology is point-to-point. 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, pause/resume signaling MUST revert to use those instead. 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.

9.1. PAUSE

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

PAUSE has no defined Type Specific parameters and Parameter Len MUST be set to 0.

PauseID SHOULD be the available PauseID, as indicated by PAUSED (Section 9.2) or implicitly determined by previously received PAUSE or RESUME (Section 9.3) requests. A randomly chosen PauseID MAY be used if it was not possible to retrieve PauseID information, in which case the PAUSE will either succeed, or the correct PauseID can be learnt from the returned REFUSE (Section 9.4). A PauseID that is matching the available PauseID is henceforth also called a valid PauseID.

PauseID needs to be 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. This is to ensure that the PauseID matches what is the current available PauseID at the media sender. The media sender increments what it considers to be the available PauseID when entering Playing State (Section 7.1).

For the scope of this specification, a PauseID larger than the current one is defined as having a value between and including $(\text{PauseID} + 1) \bmod 2^{16}$ and $(\text{PauseID} + 2^{14}) \bmod 2^{16}$, where "MOD" is the modulo operator. Similarly, a PauseID smaller than the current one is defined as having a value between and including $(\text{PauseID} - 2^{15}) \bmod 2^{16}$ and $(\text{PauseID} - 1) \bmod 2^{16}$.

If an RTP media stream receiver that sent a PAUSE with a certain PauseID receives a RESUME with the same PauseID, it is RECOMMENDED that it refrains from sending further PAUSE requests for some appropriate time since the RESUME indicates that there are other receivers that still wishes to receive the stream.

If the targeted RTP media stream does not pause, if no PAUSED indication with a larger PauseID than the one used in PAUSE, and if no REFUSE is received within $2 * \text{RTT} + \text{T_dither_max}$, the PAUSE MAY be scheduled for retransmission, using the same PauseID. RTT is the observed round-trip to the RTP media stream sender and T_dither_max is defined in section 3.4 of [RFC4585].

When an RTP media stream sender in Playing State (Section 7.1) receives a valid PAUSE, and unless local considerations currently makes it impossible to pause the stream, it SHALL enter Pausing State

(Section 7.2) when reaching an appropriate place to pause in the media stream, and act accordingly.

If an RTP media stream sender receives a valid PAUSE while in Pausing, Paused (Section 7.3) or Local Paused (Section 7.4) States, the received PAUSE SHALL be ignored.

9.2. PAUSED

The PAUSED indication MAY be sent either as a result of a valid PAUSE (Section 9.1) request, when entering Paused State (Section 7.3), or based on a RTP media stream sender local decision, when entering Local Paused State (Section 7.4).

PauseID MUST contain the available, valid value to be included in a subsequent RESUME (Section 9.3).

PAUSED SHALL contain a 32 bit parameter with the RTP extended highest sequence number valid when the RTP media stream was paused. Parameter Len MUST be set to 1.

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

While remaining in Paused or Local Paused States, PAUSED MAY be included in all regular RTCP reports.

When in Paused or Local Paused States, It is RECOMMENDED to send PAUSED at the earliest opportunity and also to include it in the next two regular RTCP reports, whenever the RTP media sender learns that there are end-points 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.

9.3. RESUME

An RTP media 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 valid PauseID, as indicated by PAUSED (Section 9.2) or implicitly determined by previously received PAUSE (Section 9.1) or RESUME requests. A randomly chosen PauseID MAY be used if it was not possible to retrieve PauseID information, in which case the RESUME will either succeed, or the correct PauseID can be learnt from a returned REFUSE (Section 9.4).

RESUME has no defined Type Specific parameters and Parameter Len MUST be set to 0.

When an RTP media stream sender in Pausing (Section 7.2), Paused (Section 7.3) or Local Paused State (Section 7.4) receives a valid RESUME, and unless local considerations currently makes it impossible to resume the stream, it SHALL enter Playing State (Section 7.1) and act accordingly. If the RTP media stream sender is incapable of honoring the RESUME request with a valid PauseID, or receives a RESUME request with an invalid PauseID while in Paused or Pausing state, the RTP media stream sender sends a REFUSE message as specified below.

If an RTP media stream sender in Playing State receives a RESUME containing either a valid PauseID or a PauseID that is less than the valid PauseID, the received RESUME SHALL be ignored.

9.4. REFUSE

REFUSE has no defined Type Specific parameters and Parameter Len MUST be set to 0.

If an RTP media sender receives a valid PAUSE (Section 9.1) or RESUME (Section 9.3) request that cannot be fulfilled by the sender due to some local consideration, it SHALL schedule transmission of a REFUSE indication containing the valid PauseID from the rejected request.

If an RTP media stream sender receives PAUSE or RESUME requests with a non-valid PauseID it SHALL schedule a REFUSE response containing the available, valid PauseID, except if the RTP media stream sender is in Playing State and receives a RESUME with a PauseID less than the valid one, in which case the RESUME SHALL be ignored.

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

If REFUSE containing a certain PauseID was already sent and yet more PAUSE or RESUME messages are received that require additional REFUSE with that specific PauseID to be scheduled, and unless the PauseID number space has wrapped since REFUSE was last sent with that PauseID, further REFUSE messages with that PauseID SHOULD be sent in regular RTCP reports.

An RTP media stream receiver that sent a PAUSE or RESUME request and receives a REFUSE 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 REFUSE to clear.

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

9.5. 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 MAY use either Regular, Early or Immediate timings, taking the following into consideration:

- o PAUSE SHOULD use Early or Immediate timing, except for retransmissions that SHOULD use Regular timing.
- o The first transmission of PAUSED for each (non-wrapped) PauseID SHOULD be sent with Immediate or Early timing, while subsequent PAUSED for that PauseID SHOULD use Regular timing.
- o RESUME SHOULD always use Immediate or Early timing.
- o The first transmission of REFUSE for each (non-wrapped) PauseID SHOULD be sent with Immediate or Early timing, while subsequent REFUSE for that PauseID SHOULD use Regular timing.

10. Signalling

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. Like AVPF [RFC4585] and CCM [RFC5104], it is RECOMMENDED to use the rtcp-fb attribute at media level and it MUST NOT be used at session level. This specification follows all the rules defined in AVPF for rtcp-fb attribute relating to payload type in a session description.

Note: When TMMBR 0 / TMMBN 0 are used to implement pause and resume functionality (with the restrictions described in this memo), signaling rtcp-fb attribute with ccm tmmbr parameter is sufficient and no further signaling is necessary.

This specification defines two new parameters to the "ccm" feedback value defined in CCM [RFC5104], "pause" and "paused".

- o "pause" represents the capability to understand the RTCP feedback message and all of the defined FCIs of PAUSE, RESUME, PAUSED and REFUSE. A direction sub-parameter is used to determine if a given node desires to issue PAUSE or RESUME requests, can respond to PAUSE or RESUME requests, or both.
- o "paused" represents the functionality of supporting the playing and local paused states and generate PAUSED FCI when a media stream delivery is paused. A direction sub-parameter is used to determine if a given node desires to receive these indications, intends to send them, or both.

The reason for this separation is to make it possible for partial implementation of this specification, according to the different roles in the use cases section (Section 3).

A sub-parameter named "nowait", indicating that the hold-off time defined in Section 7.2 can be set to 0, reducing the latency before the media stream can be paused after receiving a PAUSE request. This condition occurs when there will be only 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 also indicate that it would be possible to use CCM TMMBR/TMMBN as pause/resume signaling.

A sub-parameter named "dir" is used to indicate in which directions a given node will use the pause or paused functionality. The node being configured or issuing an offer or an answer uses the directionality in the following way. Note that pause and paused have separate and different definitions.

Direction ("dir") values for "pause" is defined as follows:

sendonly: The node intends to send PAUSE and RESUME requests for other nodes' media streams and is thus also capable of receiving PAUSED and REFUSE. It will not support receiving PAUSE and RESUME requests.

recvonly: The node supports receiving PAUSE and RESUME requests targeted for media streams sent by the node. It will send PAUSED and REFUSE as needed. The node will not send any PAUSE and RESUME requests.

sendrecv: The node supports receiving PAUSE and RESUME requests targeted for media streams sent by the node. The node intends to send PAUSE and RESUME requests for other nodes' media streams. Thus the node is capable of sending and receiving all types of pause messages. This is the default value. If the "dir"

parameter is omitted, it MUST be interpreted to represent this value.

Direction values for "paused" is defined as follows:

sendonly: The node intends to send PAUSED indications whenever it pauses media delivery in any of its media streams. It has no need to receive PAUSED indications itself.

recvonly: The node desires to receive PAUSED indications whenever any media stream sent by another node is paused. It does not intend to send any PAUSED indications.

sendrecv: The nodes desires to receive PAUSED indications and intends to send PAUSED indications whenever any media stream is paused. This is the default value. If the "dir" parameter is omitted, it MUST be interpreted to represent this value.

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)
                  / SP "paused" *(SP paused-attr)
pause-attr        = direction
                  / "nowait"
                  / token ; for future extensions
paused-attr       = direction
                  / token ; for future extensions
direction         = "dir=" direction-alts
direction-alts    = "sendonly" / "recvonly" / "sendrecv"
```

Figure 10: ABNF

An endpoint implementing this specification and using SDP to signal capability SHOULD indicate both of the new "pause" and "paused" parameters with ccm signaling. When negotiating usage, it is possible select either of them, noting that "pause" contain the full "paused" functionality. A sender or receiver SHOULD NOT use the messages from this specification towards receivers that did not declare capability for it.

There MUST NOT be more than one "a=rtcp-fb" line with "pause" and one with "paused" 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 does not have an explicit "pause" or "paused" specification.

There MUST NOT be more than a single direction sub-parameter per "pause" and "paused" parameter. There MUST NOT be more than a single "nowait" sub-parameter per "pause" parameter.

10.1. Offer-Answer Use

An offerer implementing this specification needs to include "pause" and/or "paused" CCM parameters with suitable directionality parameter ("dir") in the SDP, according to what messages it intends to send and desires or is capable to receive in the session. It is RECOMMENDED to include both "pause" and "paused" if "pause" is supported, to enable at least the "paused" functionality if the answer only supports "paused" or different directionality for the two functionalities. The "pause" and "paused" functionalities are negotiated independently, although the "paused" functionality is part of the "pause" functionality. As a result, an answerer MAY remove "pause" or "paused" lines from the SDP depending on the agreed mode of functionality.

In offer/answer, the "dir" parameter is interpreted based on the agent providing the SDP. The node described in the offer is the offerer, and the answerer is described in an answer. In other words, an offer for "paused dir=sendonly" means that the offerer intends to send PAUSED indications whenever it pauses media delivery in any of its media streams.

An answerer receiving an offer with a "pause" parameter with dir=sendrecv MAY remove the pause line in its answer, respond with pause keeping sendrecv for full bi-directionality, or it may change dir value to either sendonly or recvonly based on its capabilities and desired functionality. An offer with a "pause" parameter with dir=sendonly or dir=recvonly is either completely removed or accepted with reverse directionality, i.e. sendonly becomes recvonly or recvonly becomes sendonly.

An answer receiving an offer with "paused" has the same choices as for "pause" above. It should be noted that the directionality of pause is the inverse of media direction, while the directionality of paused is the same as the media direction.

If the offerer believes that itself and the intended answerer are likely the only end-points in the RTP session, it MAY include the "nowait" sub-parameter on the "pause" line in the offer. If an answerer receives the "nowait" sub-parameter on the "pause" line in the SDP, and if it has information that the offerer and itself are not the only end-points in the RTP session, it MUST NOT include any "nowait" sub-parameter 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 "paused" line in the offer. If both offer and answer contained a "nowait" parameter, then the hold-off time is configured to 0 at both offerer and answerer.

10.2. 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 signalling extensions defined in this draft. First, it is normally only necessary to include either "pause" or "paused" parameter to indicate the level of functionality the node should use in this RTP session. Including both is only necessary if some implementations only understands "paused" and some other can understand both. Thus indicating both means use pause if you understand it, and if you only understand paused, use that.

The "dir" directionality parameter indicates how the configured node should behave. For example "pause" with sendonly:

sendonly: The node intends to send PAUSE and RESUME requests for other nodes' media streams and is thus also capable of receiving PAUSED and REFUSE. It will not support receiving PAUSE and RESUME requests.

In this example, the configured node should send PAUSE and RESUME requests if has reason for it. It does not need to respond to any PAUSE or RESUME requests as that is not supported.

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.

11. 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 Translator

11.1. 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
a=rtcp-fb:* ccm paused
```

Figure 11: SDP Offer With Pause and Resume Capability

The offerer supports all of the messages defined in this specification and offers a sendrecv stream. 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 both "pause" and "paused" parameters.

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 dir=sendonly
a=rtcp-fb:98 ccm paused
```

Figure 12: SDP Answer With Pause and Resume Capability

The answerer will not allow its sent streams to be paused or resumed and thus support pause only in sendonly mode. It does support paused

and intends to send it, and also desires to receive PAUSED indications. Thus paused in sendrecv mode is included in the answer. 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 time 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.

11.2. 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 time before pausing a stream is 0.

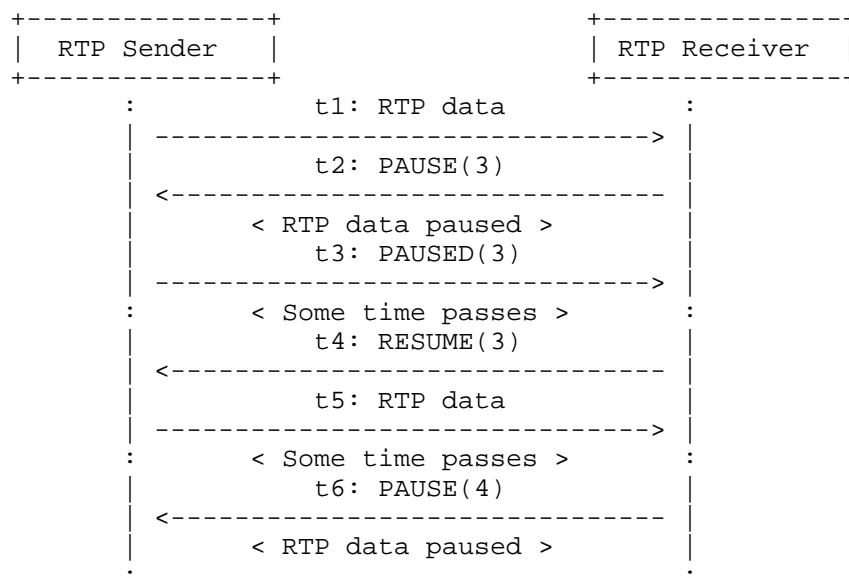


Figure 13: Pause and Resume Operation in Point-to-Point

Figure 13 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).

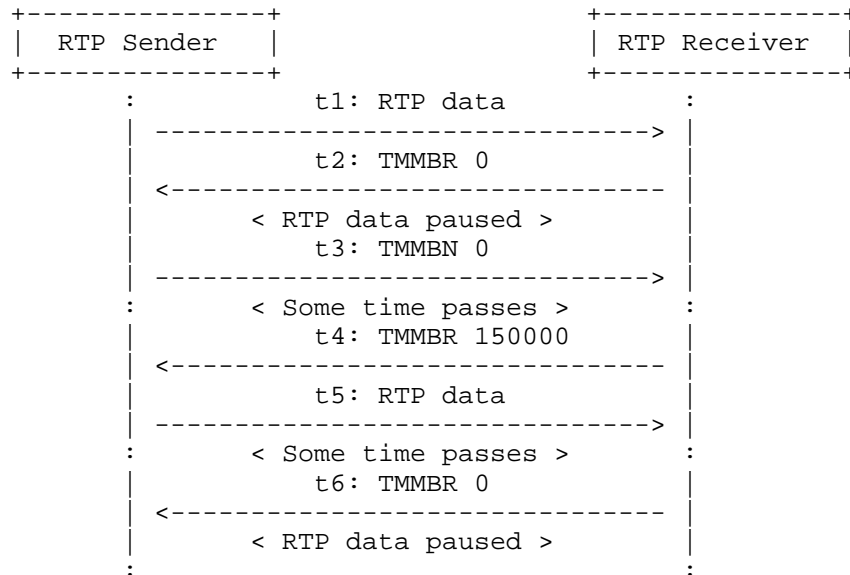


Figure 14: TMMBR Pause and Resume in Point-to-Point

Figure 14 describes the same point-to-point scenario as above, but using TMMBR/TMMBN signaling.

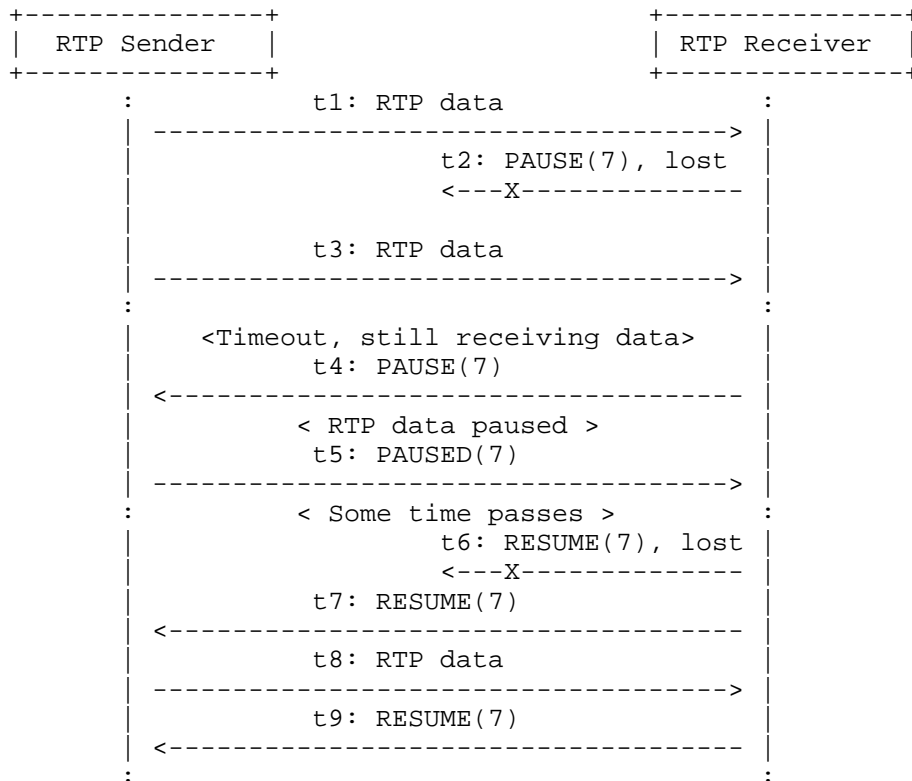


Figure 15: Pause and Resume Operation With Messages Lost

Figure 15 describes what happens if a PAUSE message from an RTP media stream receiver does not reach the RTP media stream sender. After sending a PAUSE message, the RTP media stream receiver waits for a time-out to detect if the RTP media stream sender has paused the data transmission or has sent PAUSED indication according to the rules discussed in Section 7.3. As the PAUSE message is lost on the way (at time t2), RTP data continues to reach to the RTP media stream receiver. When the timer expires, the RTP media stream receiver schedules a retransmission of the PAUSE message, which is sent at time t4. If the PAUSE message now reaches the RTP media stream sender, it pauses the RTP media stream and replies with PAUSED.

At time t6, the RTP media 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 media 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 media 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 the RESUME PauseID is less than the valid one and will be ignored by the RTP media stream sender. It will also not cause any unwanted resume even if the stream was paused based on a PAUSE from some other receiver before receiving the RESUME, since the valid PauseID is now larger than the one in the stray RESUME and will only cause a REFUSE containing the new valid PauseID from the RTP media stream sender.

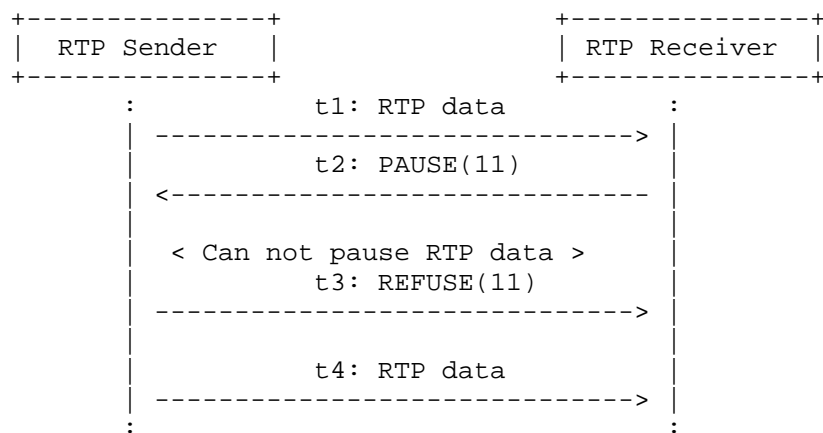


Figure 16: Pause Request is Refused in Point-to-Point

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 REFUSE message.

11.3. 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 media 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 media 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.

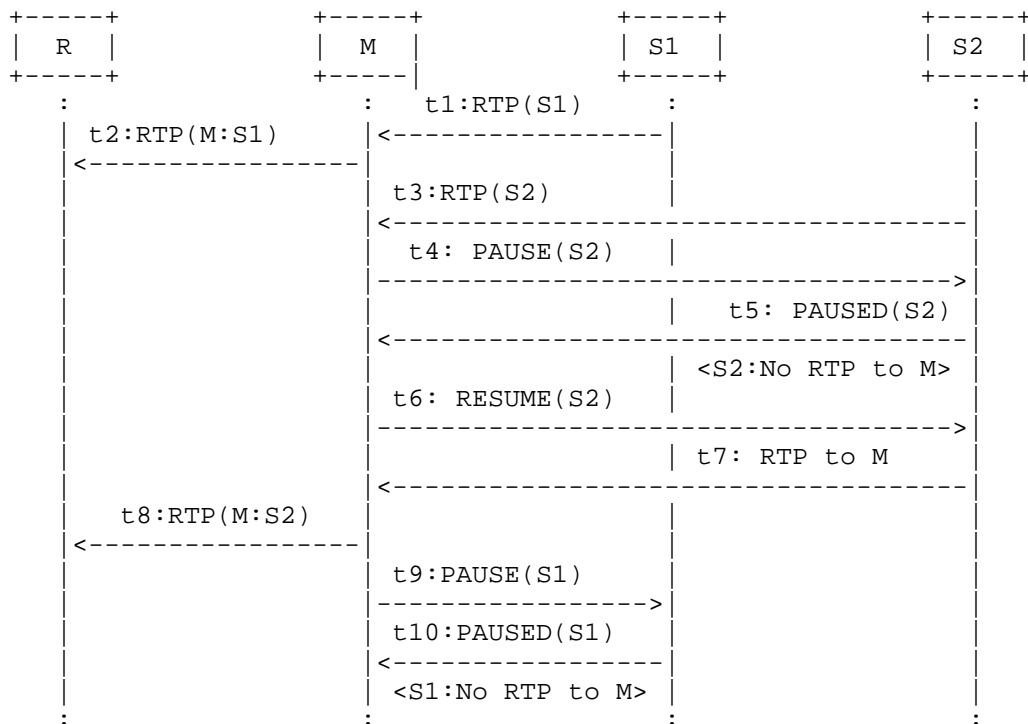


Figure 17: Pause and Resume Operation for a Voice Activated Mixer

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 media 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 media delivery to M. When the media

from t7 arrives at the Mixer, it switches this media 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 media stream from S1 stops being delivered.

11.4. Point-to-multipoint using Translator

A transport Translator in an RTP session forwards the message from one peer to all the others. Unlike Mixer, the Translator 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 Translator case. The parentheses in the figures contains (Target SSRC, PauseID) information for the messages defined in this specification.

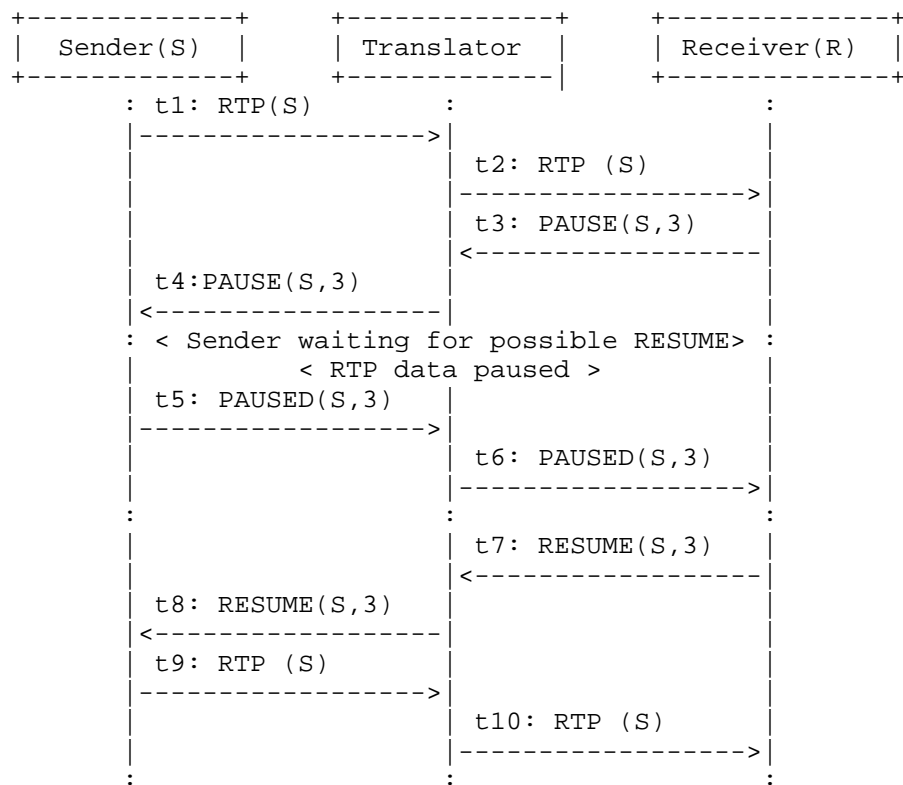


Figure 18: Pause and Resume Operation Between Two Participants Using a Translator

Figure 18 describes how a Translator can help the receiver in pausing and resuming the sender. The sender S sends RTP data to the receiver R through Translator, 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 time 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 Translator. Since this describes only a single pause operation for a single media sender, all messages uses a single PauseID, in this example 3.

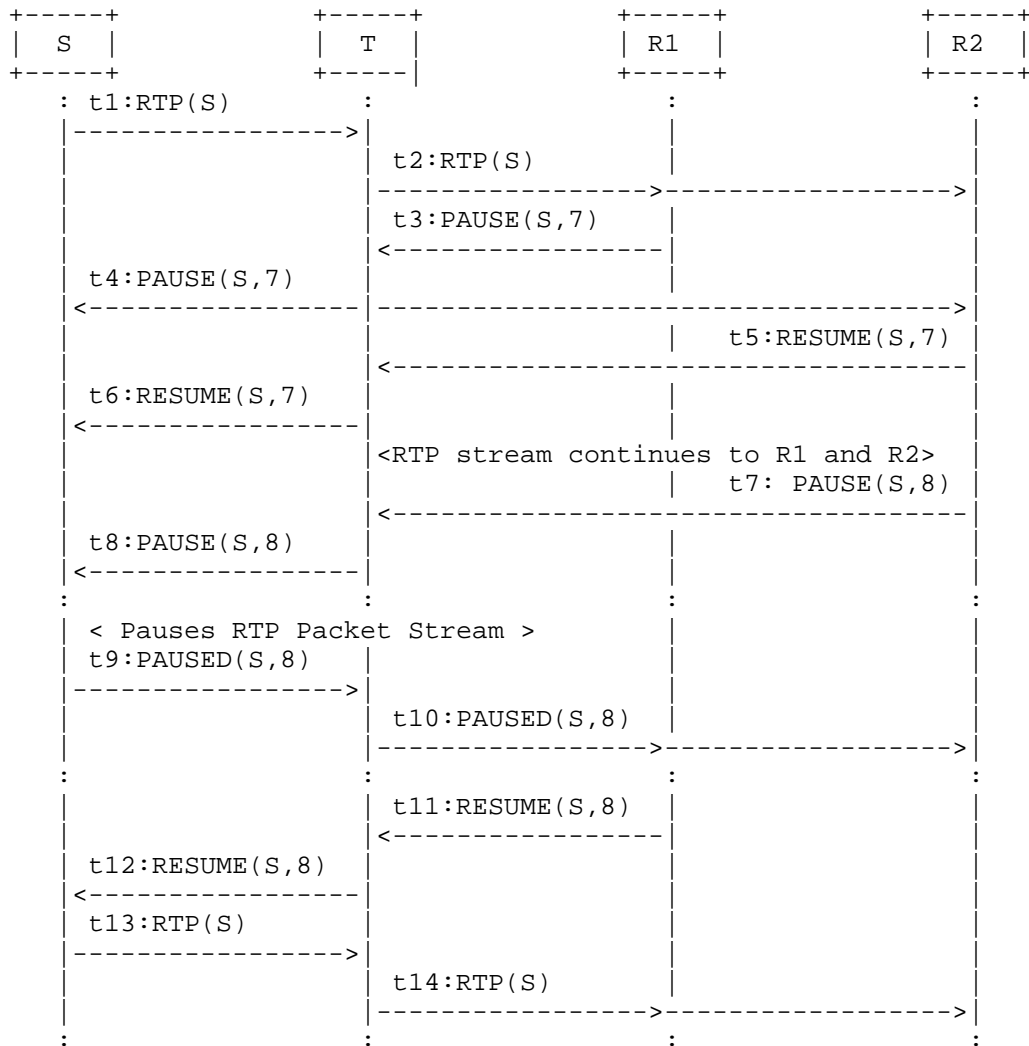


Figure 19: Pause and Resume Operation Between One Sender and Two Receivers Through Translator

Figure 19 explains the pause and resume operations when a transport Translator 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 media to the Translator, which is forwarded to R1 and R2 through the Translator, T. R1 and R2 receives RTP data from Translator at `t2`. At this point,

both R1 and R2 will send RTCP Receiver Reports to S informing that they receive S's media 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 translator T. The sender S sees the RESUME at time t6 and continues to send data to T 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 time, 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. The RTP sender immediately resumes the stream.

Consider also an RTP session which includes one or more receivers, paused sender(s), and a Translator. 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.

12. IANA Considerations

As outlined in Section 8, this specification requests IANA to allocate

1. The FMT number TBA1 to be allocated to the PAUSE and RESUME functionality from this specification.
2. The 'pause' and 'paused' tags to be used with ccm under rtcp-fb AVPF attribute in SDP.
3. The 'nowait' parameter to be used with the 'pause' and 'paused' tags in SDP.

4. A registry listing registered values for 'pause' Types.
5. PAUSE, RESUME, PAUSED, and REFUSE with the listed numbers in the pause Type registry.

13. Security Considerations

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

1. Identity spoofing - An attacker can spoof him/herself as an authenticated user and can falsely pause or resume any source transmission. In order to prevent this type of attack, a strong authentication and integrity protection mechanism is needed.
2. Denial of Service (DoS) - An attacker can falsely pause all source streams which MAY result in Denial of Service (DoS). An Authentication protocol may prevent this attack.
3. Man-in-Middle Attack (MiMT) - The pausing and resuming of an RTP source is prone to a Man-in-Middle attack. Public key authentication may be used to prevent MiMT.

14. Contributors

Daniel Groendal contributed in the creation and writing of earlier versions of this specification.

15. Acknowledgements

Daniel Grondal made valuable contributions during the initial versions of this draft.

16. References

16.1. 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.

- [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.
- [RFC5104] Wenger, S., Chandra, U., Westerlund, M., and B. Burman, "Codec Control Messages in the RTP Audio-Visual Profile with Feedback (AVPF)", RFC 5104, February 2008.
- [RFC5234] Crocker, D. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", STD 68, RFC 5234, January 2008.

16.2. Informative References

- [I-D.ietf-avtcore-rtp-topologies-update]
Westerlund, M. and S. Wenger, "RTP Topologies", draft-ietf-avtcore-rtp-topologies-update-01 (work in progress), October 2013.
- [I-D.ietf-avtext-rtp-grouping-taxonomy]
Lennox, J., Gross, K., Nandakumar, S., and G. Salgueiro, "A Taxonomy of Grouping Semantics and Mechanisms for Real-Time Transport Protocol (RTP) Sources", draft-ietf-avtext-rtp-grouping-taxonomy-00 (work in progress), November 2013.
- [I-D.ietf-rtcweb-use-cases-and-requirements]
Holmberg, C., Hakansson, S., and G. Eriksson, "Web Real-Time Communication Use-cases and Requirements", draft-ietf-rtcweb-use-cases-and-requirements-14 (work in progress), February 2014.
- [I-D.lennox-mmusic-sdp-source-selection]
Lennox, J. and H. Schulzrinne, "Mechanisms for Media Source Selection in the Session Description Protocol (SDP)", draft-lennox-mmusic-sdp-source-selection-05 (work in progress), October 2012.
- [I-D.westerlund-avtcore-rtp-simulcast]
Westerlund, M. and S. Nandakumar, "Using Simulcast in RTP Sessions", draft-westerlund-avtcore-rtp-simulcast-03 (work in progress), October 2013.
- [RFC2326] Schulzrinne, H., Rao, A., and R. Lanphier, "Real Time Streaming Protocol (RTSP)", RFC 2326, April 1998.
- [RFC2974] Handley, M., Perkins, C., and E. Whelan, "Session Announcement Protocol", RFC 2974, October 2000.

- [RFC3261] Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston, A., Peterson, J., Sparks, R., Handley, M., and E. Schooler, "SIP: Session Initiation Protocol", RFC 3261, June 2002.
- [RFC3264] Rosenberg, J. and H. Schulzrinne, "An Offer/Answer Model with Session Description Protocol (SDP)", RFC 3264, June 2002.
- [RFC3556] Casner, S., "Session Description Protocol (SDP) Bandwidth Modifiers for RTP Control Protocol (RTCP) Bandwidth", RFC 3556, July 2003.
- [RFC4566] Handley, M., Jacobson, V., and C. Perkins, "SDP: Session Description Protocol", RFC 4566, July 2006.
- [RFC5049] Bormann, C., Liu, Z., Price, R., and G. Camarillo, "Applying Signaling Compression (SigComp) to the Session Initiation Protocol (SIP)", RFC 5049, December 2007.
- [RFC5225] Pelletier, G. and K. Sandlund, "RObust Header Compression Version 2 (ROHCv2): Profiles for RTP, UDP, IP, ESP and UDP-Lite", RFC 5225, April 2008.
- [RFC5576] Lennox, J., Ott, J., and T. Schierl, "Source-Specific Media Attributes in the Session Description Protocol (SDP)", RFC 5576, June 2009.
- [RFC5626] Jennings, C., Mahy, R., and F. Audet, "Managing Client-Initiated Connections in the Session Initiation Protocol (SIP)", RFC 5626, October 2009.
- [RFC6190] Wenger, S., Wang, Y., Schierl, T., and A. Eleftheriadis, "RTP Payload Format for Scalable Video Coding", RFC 6190, May 2011.
- [TS25.308] 3GPP, "High Speed Downlink Packet Access (HSDPA); Overall description; Stage 2", 3GPP TS 25.308 10.6.0, December 2011, <<http://www.3gpp.org/ftp/Specs/html-info/25308.htm>>.
- [TS26.114] 3GPP, "IP Multimedia Subsystem (IMS); Multimedia telephony; Media handling and interaction", 3GPP TS 26.114 10.7.0, June 2013, <<http://www.3gpp.org/ftp/Specs/html-info/26114.htm>>.

[TS36.201]

3GPP, "Evolved Universal Terrestrial Radio Access
(E-UTRA); LTE physical layer; General description", 3GPP
TS 36.201 10.0.0, December 2010,
<<http://www.3gpp.org/ftp/Specs/html-info/36201.htm>>.

Authors' Addresses

Azam Akram
Ericsson
Farogatan 6
SE - 164 80 Kista
Sweden

Phone: +46107142658
Fax: +46107175550
Email: muhammad.azam.akram@ericsson.com
URI: www.ericsson.com

Bo Burman
Ericsson
Farogatan 6
SE - 164 80 Kista
Sweden

Phone: +46107141311
Fax: +46107175550
Email: bo.burman@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- Kista 164 80
Sweden

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

AVTEXT Working Group
INTERNET-DRAFT
Intended Status: Standards Track
Expires: August 17, 2014

J. Xia
R. Even
R. Huang
Huawei
L. Deng
China Mobile
February 13, 2014

RTP/RTCP extension for RTP Splicing Notification
draft-xia-avtext-splicing-notification-03

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 RTP mixer is designed to handle RTP splicing in [RFC6828], but how the RTP mixer knows when to start and end the splicing is still unspecified.

This memo defines two RTP/RTCP extensions to indicate the splicing related information to the RTP mixer: 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) 2013 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	7
6	SDP Signaling	8
7	Security Considerations	9
8	IANA Considerations	9
8.1	RTCP Control Packet Types	9
8.2	RTP Compact Header Extensions	10
9	Acknowledges	10
10	References	10
10.1	Normative References	10
10.2	Informative References	11
	Authors' Addresses	11

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 mixer 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 RTP splicing scenario described in [RFC6828], the mixer has to decode the RTP packets, search and solve the Splicing Interval inside the payloads. The need for such processing enhances the workload of the mixer and limits the size of RTP sessions the mixer can support.

The document defines an RTP header extension [RFC5285] through which the main RTP sender can 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 the RTP header extension.

To increase robustness against above case, the document also defines a new RTCP packet type in a complementary fashion to carry the Splicing Interval to the mixer even though RTCP is inherently unreliable too.

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].

Most terminology defined in "Content Splicing for RTP Sessions" [RFC6828] applies to this document except the following one.

Splicing Interval:

A set of certain metadata that allows the mixer to know when to start and end the RTP splicing. The information consists of a couple of NTP-format timestamps on the splicing in point and on the splicing out point.

2 Overview of RTP Splicing Notification

According to RTP Splicing draft [RFC6828], a mixer is designed to do splicing on the RTP layer, but it cannot insert the substitutive content randomly but only do that at the reserved time slots set by the main RTP sender. This implies the mixer must first know the Splicing Interval from the main RTP sender before splicing starts.

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 against 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 enough in advance. Alternatively, 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 to 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 couple 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 on the splicing in point, and the timestamp of the first main RTP packet on the splicing out point.

When the substitutive RTP sender gets the Splicing Interval, it must prepare the substitutive stream. The RTP timestamp of the first substitutive RTP packet that would be presented on the receivers MUST correspond to the same time instant as the former NTP timestamp in the Splicing Interval. To enable mixer to know the first substitutive RTP packet it begins to output, the substitutive RTP sender MUST enable the mixer to know above RTP timestamp in advance, e.g., from prior receipt of RTCP SR message.

When the splicing will end, the RTP timestamp of the first main RTP packet that would be presented on the receivers MUST correspond 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 couple 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, under the consumption that the splicing-out time is after the splicing-in time, and the splicing interval is less than 2^{25} seconds, this order allows full resolution for splicing-in NTP timestamp while keeping 4 octets alignment.

The format is shown in Figures 1.

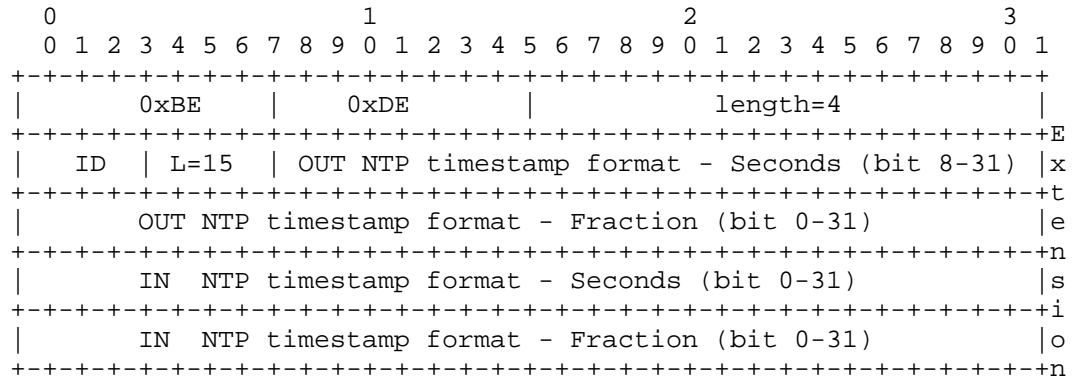


Figure 1: Sample hybrid NTP Encoding Using the One-Byte Header Format

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 in advance of the splicing starting, while leaving the

remain RTP packets unmarked.

After the mixer intercepts the RTP header extension and derives the Splicing Interval, it will generate its own stream and could 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 in next section MUST be sent to provide robustness in the case of any splicing-unaware middlebox that might strip RTP header extensions.

3.2 RTCP Splicing Notification Message

Besides 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 couple of NTP-format timestamps:

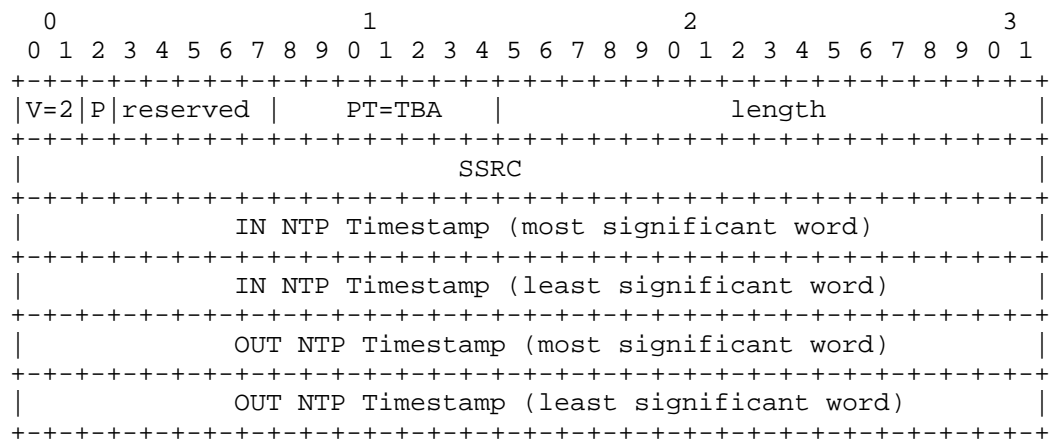


Figure 2: RTCP Splicing Notification Message

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 MAY NOT forward the message to the receivers in order to reduce RTCP bandwidth consumption or to avoid downstream receivers 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 means 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 there may be splicing un-aware middlebox on the path between the main RTP sender and the mixer, one heuristic 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. In such case, the main RTP sender can learn the splicing failed.

In a similar manner, the substitutive sender can learn the 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 also reuses the Flow Identification (FID) semantics defined in SDP Grouping Framework [RFC5888] to represent the relationship between the main RTP stream and the substitutive RTP stream.

The next example shows how the "group" attribute used with FID semantics can indicate RTP splicing support on RTP sender.

```
v=0
o=xia 1122334455 1122334466 IN IP4 splicing.example.com
s=RTP Splicing Example
t=0 0
a=group:FID 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-hdext:splicing-interval
```

```
a=mid: 1
m= video 30001 RTP/AVP 100
i=Substitutive RTP Stream
c=IN IP4 233.252.0.2/127
a=sendonly
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 30001. But at a particular point in time, the mixer only selects one stream and output the content from the chosen stream to the downstream receivers.

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. A malicious endpoint could choose to set the values in this header extension falsely, so as to falsely claim the splicing time.

In scenarios where this is a concern, additional mechanisms MUST be used to protect the confidentiality of the header extension. This mechanism could be header extension encryption [SRTP-ENCR-HDR], or a lower-level security and authentication mechanism such as IPsec [RFC4301].

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

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.
- [RFC4566] Handley, M., Jacobson, V., and C. Perkins, "SDP: Session Description Protocol", RFC 4566, July 2006.
- [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.

[RFC6828] Xia, J., "Content Splicing for RTP Sessions", RFC 6828, January 2013.

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.

[RFC4301] Kent, S. and K. Seo, "Security Architecture for the Internet Protocol", RFC 4301, December 2005.

[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.

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