

Network Working Group  
Internet-Draft  
Updates: 3550 (if approved)  
Intended status: Standards Track  
Expires: April 25, 2013

M. Petit-Huguenin  
Unaffiliated  
G. Zorn, Ed.  
Network Zen  
October 22, 2012

Support for Multiple Clock Rates in an RTP Session  
draft-ietf-avtext-multiple-clock-rates-06

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.

Status of this Memo

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

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

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

This Internet-Draft will expire on April 25, 2013.

Copyright Notice

Copyright (c) 2012 IETF Trust and the persons identified as the document authors. All rights reserved.

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

## Table of Contents

1. Introduction . . . . .	3
2. Terminology . . . . .	4
3. Legacy RTP . . . . .	4
3.1. Different SSRC . . . . .	4
3.2. Same SSRC . . . . .	5
3.2.1. Monotonic timestamps . . . . .	5
3.2.2. Non-monotonic timestamps . . . . .	6
4. Recommendations . . . . .	7
4.1. RTP Sender (with RTCP) . . . . .	7
4.2. RTP Sender (without RTCP) . . . . .	7
4.3. RTP Receiver . . . . .	8
5. Security Considerations . . . . .	8
6. IANA Considerations . . . . .	8
7. Acknowledgements . . . . .	8
8. References . . . . .	8
8.1. Normative References . . . . .	8
8.2. Informative References . . . . .	9
Appendix A. Using a Fixed Clock Rate . . . . .	9
Appendix B. Behavior of Legacy Implementations . . . . .	10
B.1. libccrtp 2.0.2 . . . . .	10
B.2. libmediastreamer0 2.6.0 . . . . .	10
B.3. libpjmedia 1.0 . . . . .	10
B.4. Android RTP stack 4.0.3 . . . . .	10
Authors' Addresses . . . . .	10

## 1. Introduction

The clock rate is a parameter of the payload format. It is often defined as been the same as the sampling rate but it is not always the case (see e.g. 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 types, it is possible that the clock rate also varies 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 SSRC but unfortunately this advice was not always 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 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 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	[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 RR packets.
RTP Receiver	A logical network element that receives RTP packets, receives RTCP SR packets, and sends RTCP RR packets.

## 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. This is not different than what happen at the beginning of the RTP session but it can be more annoying for the end-user.

### 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. 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.

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

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:

$$\text{timestamp} = \text{capture\_time} * \text{clock\_rate}$$

With this formula, the jitter calculation is correct but the RTP timestamp values are no longer increasing monotonically as shown in Table 3. 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.

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

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.

#### 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 next SR packets 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. by setting 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\_time is the first instant the new timestamp rate is used.

#### 4.3. RTP Receiver

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

$$D(i,j) = (\text{arrival\_time\_j} * \text{clock\_rate\_i} - \text{timestamp\_j}) \\ - (\text{arrival\_time\_i} * \text{clock\_rate\_i} - \text{timestamp\_i})$$

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

#### 5. Security Considerations

This document is not believed to effect the security of the RTP sessions described here in any way.

#### 6. IANA Considerations

This document requires no IANA actions.

#### 7. Acknowledgements

Thanks to Colin Perkins, Ali C. Begen, Harald Alvestrand, Qin Wu and Magnus Westerlund for their comments, suggestions and questions that helped to improve this document.

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., "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. Using a Fixed Clock Rate

An alternate way of fixing the multiple clock rates issue was proposed in [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 B. Behavior of Legacy Implementations

## B.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.

## B.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.

## B.3. libpjmedia 1.0

This library uses the formula described in Section 3.2.2.

## B.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  
Unaffiliated

Email: [petithug@acm.org](mailto:petithug@acm.org)

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

Phone: +66 (0) 909-201060  
Email: [glenzorn@gmail.com](mailto:glenzorn@gmail.com)



AVTEXT  
Internet-Draft  
Intended status: Standards Track  
Expires: January 3, 2013

A. Begen  
Cisco  
C. Perkins  
University of Glasgow  
July 2, 2012

Duplicating RTP Streams  
draft-ietf-avtext-rtp-duplication-00

Abstract

Packet loss is undesirable for real-time multimedia sessions, but can occur due to congestion, or other unplanned network outages. This is especially true for IP multicast networks, where packet loss patterns can vary greatly between receivers. One technique that can be used to recover from packet loss without incurring unbounded delay for all the receivers is to duplicate the packets and send them in separate redundant streams. This document explains how Real-time Transport Protocol (RTP) streams can be duplicated without breaking RTP media streams, or RTP Control Protocol (RTCP) rules.

Status of this Memo

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

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

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

This Internet-Draft will expire on January 3, 2013.

Copyright Notice

Copyright (c) 2012 IETF Trust and the persons identified as the document authors. All rights reserved.

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

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

## Table of Contents

1. Introduction . . . . .	3
2. Terminology and Requirements Notation . . . . .	3
3. Dual Streaming Use Cases . . . . .	3
3.1. Temporal Redundancy . . . . .	4
3.2. Spatial Redundancy . . . . .	4
3.3. Dual Streaming over a Single Path or Multiple Paths . . . . .	5
4. Use of RTP and RTCP with Temporal Redundancy . . . . .	6
4.1. RTCP Considerations . . . . .	6
4.2. Signaling Considerations . . . . .	6
5. Use of RTP and RTCP with Spatial Redundancy . . . . .	7
5.1. RTCP Considerations . . . . .	8
5.2. Signaling Considerations . . . . .	8
6. Use of RTP and RTCP with Temporal and Spatial Redundancy . . . . .	9
7. Security Considerations . . . . .	9
8. IANA Considerations . . . . .	9
9. Acknowledgments . . . . .	10
10. References . . . . .	10
10.1. Normative References . . . . .	10
10.2. Informative References . . . . .	10
Authors' Addresses . . . . .	11

## 1. Introduction

The Real-time Transport Protocol (RTP) [RFC3550] is widely used today for delivering IPTV traffic, and other real-time multimedia sessions. Many of these applications support very large numbers of receivers, and rely on intra-domain UDP/IP multicast for efficient distribution of traffic within the network.

While this combination has proved successful, there does exist a weakness. As [RFC2354] noted, packet loss is not avoidable, even in a carefully managed network. This loss might be due to congestion, it might also be a result of an unplanned outage caused by a flapping link, link or interface failure, a software bug, or a maintenance person accidentally cutting the wrong fiber. Since UDP/IP flows do not provide any means for detecting loss and retransmitting packets, it leaves up to the RTP layer and the applications to detect, and recover from, packet loss.

One technique to recover from packet loss without incurring unbounded delay for all the receivers is to duplicate the packets and send them in separate redundant streams. Variations on this idea have been implemented and deployed today [IC2011]. However, duplication of RTP streams without breaking the RTP and RTCP functionality has not been documented properly. This document explains how duplication can be achieved for RTP streams.

Stream duplication offers a simple way to protect media flows from packet loss. It has a comparatively high bandwidth overhead, since everything is sent twice, but with a low processor overhead. It is also very predictable in its overheads. Alternative approaches may be suitable in some cases, for example retransmission-based recovery [RFC4588] or forward error correction [RFC5109].

## 2. Terminology and Requirements Notation

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

## 3. Dual Streaming Use Cases

Dual streaming refers to a technique that involves transmitting two redundant RTP streams of the same content, with each stream capable of supporting the playback when there is no packet loss. Therefore, adding an additional RTP stream provides a protection against packet

loss. The level of protection depends on how the packets are sent and transmitted inside the network.

It is important to note that dual streaming can easily be extended to support cases when more than two streams are desired. However, using three or more streams is rare in practice, due to the high overhead that it incurs.

### 3.1. Temporal Redundancy

From a routing perspective, two streams are considered identical if the following two IP header fields are the same, since they will be both routed over the same path:

- o IP Source Address
- o IP Destination Address

Two routing-plane identical RTP streams might carry the same payload, but can use different Synchronization Sources (SSRC) to differentiate the RTP packets belonging to each stream. In the context of dual RTP streaming, we assume that the source duplicates the RTP packets and sends them in separate RTP streams, each with a unique SSRC. All the redundant streams are transmitted in the same RTP session.

For example, one main and one redundant RTP stream can be sent to the same IP destination address and UDP destination port with a certain delay between them [I-D.begen-mmusic-temporal-interleaving]. The streams carry the same payload in their respective RTP packets with identical sequence numbers. This allows receivers (or other nodes responsible for gap filling and duplicate suppression) to identify and suppress the duplicate packets, and subsequently produce a hopefully loss-free and duplication-free output stream. This process is called stream merging.

### 3.2. Spatial Redundancy

An RTP source might be associated with multiple network interfaces, allowing it to send two redundant streams from two separate source addresses. Such streams can be routed over diverse or identical paths depending on the routing algorithm used inside the network. At the receiving end, the node responsible for duplicate suppression can look into various RTP header fields, for example SSRC and sequence number, to identify and suppress the duplicate packets.

If source-specific multicast (SSM) transport is used to carry such redundant streams, there will be a separate SSM session for each redundant stream since the streams are sourced from different

interfaces (i.e., IP addresses). Thus, the receiving host has to join each SSM session separately.

Alternatively, an RTP source might send the redundant streams to separate IP destination addresses.

### 3.3. Dual Streaming over a Single Path or Multiple Paths

Having described the characteristics of the streams, one can reach the following conclusions:

1. When two routing-plane identical streams are used, the two streams will have identical IP headers. This makes it impractical to forward the packets onto different paths. In order to minimize packet loss, the packets belonging to one stream are often interleaved with packets belonging to the other, and with a delay, so that if there is a packet loss, such a delay would allow the same packet from the other stream to reach the receiver because the chances that the same packet is lost in transit again is often small. This is what is also known as Time-shifted Redundancy, Temporal Redundancy or simply Delayed Duplication [I-D.begen-mmusic-temporal-interleaving] [IC2011]. This approach can be used with both types of dual streaming, described in Section 3.1 and Section 3.2.
2. If the two streams have different IP headers, an additional opportunity arises in that one is able to build a network, with physically diverse paths, to deliver the two streams concurrently to the intended receivers. This reduces the delay when packet loss occurs and needs to be recovered. Additionally, it also further reduces chances for packet loss. An unrecoverable loss happens only when two network failures happen in such a way that the same packet is affected on both paths. This is referred to as Spatial Diversity or Spatial Redundancy [IC2011]. The techniques used to build diverse paths are beyond the scope of this document.

Note that spatial redundancy often offers less delay in recovering from packet loss provided that the forwarding delay of the network paths are more or less the same. For both temporal and spatial redundancy approaches, packet misordering might still happen and needs to be handled using the sequence numbers of some sort (e.g., RTP sequence numbers).

To summarize, dual streaming allows an application and a network to work together to provide a near zero-loss transport with a bounded or minimum delay. The additional advantage includes a predictable bandwidth overhead that is proportional to the minimum bandwidth



needed for the multimedia session, but independent of the number of receivers experiencing a packet loss and requesting a retransmission. For a survey and comparison of similar approaches, refer to [IC2011].

#### 4. Use of RTP and RTCP with Temporal Redundancy

To achieve temporal redundancy, the main and redundant RTP streams MUST be sent using the same 5-tuple of transport protocol, source and destination IP addresses, and source and destination transport ports. This is perhaps overly restrictive, but with the possible presence of network address and port translation (NAPT) devices, using anything other than an identical 5-tuple can also cause spatial redundancy.

Since main and redundant RTP streams follow an identical path, they are part of the same RTP session. Accordingly, the sender MUST choose a different SSRC for the redundant RTP stream than it chose for the main RTP stream, following the rules in [RFC3550] Section 8.

##### 4.1. RTCP Considerations

If RTCP is being sent for the main RTP stream, then the sender MUST also generate RTCP for the redundant RTP stream. The RTCP for the redundant RTP stream is generated exactly as-if the redundant RTP stream were a regular media stream. The sender MUST NOT duplicate the RTCP packets sent for the main RTP stream when sending the duplicate stream, instead it MUST generate new RTCP reports for the duplicate stream. The sender MUST use the same RTCP CNAME in the RTCP reports it sends for the main and redundant streams, so that the receiver can synchronize them.

Both the main and redundant RTP streams, and their corresponding RTCP reports, will be received. If RTCP is used, receivers MUST generate RTCP reports for both main and redundant streams in the usual way, treating them as entirely separate media streams.

##### 4.2. Signaling Considerations

Signaling is needed to allow the receiver to determine that an RTP stream is a redundant copy of another, rather than a separate stream that needs to be rendered in parallel. There are two parts to this: an SDP extension is needed in the offer/answer exchange to negotiate support for temporal redundancy; and signalling is needed to indicate which stream is the duplicate (the latter can be done in-band using an RTCP extension, or out-of-band by signalling the SSRCs used by the duplicate streams in SDP).

We require out-of-band signalling for both features. The required

SDP attribute to signal duplication in the SDP offer/answer exchange ('duplication-delay') is defined in [I-D.begen-mmusic-temporal-interleaving]. The required SDP grouping semantics are defined in [I-D.begen-mmusic-redundancy-grouping].

In the following SDP example, a video stream is duplicated, and the main and redundant streams are transmitted in two separate SSRCs (1000 and 1010):

```
v=0
o=ali 1122334455 1122334466 IN IP4 dup.example.com
s=Delayed Duplication
t=0 0
m=video 30000 RTP/AVP 100
c=IN IP4 233.252.0.1/127
a=source-filter:incl IN IP4 233.252.0.1 198.51.100.1
a=rtpmap:100 MP2T/90000
a=ssrc:1000 cname:chl@example.com
a=ssrc:1010 cname:chl@example.com
a=ssrc-group:DUP 1000 1010
a=duplication-delay:100
a=mid:Group1
```

It is RECOMMENDED that the SSRC listed first in the "a=ssrc-group:" line is sent first, with the other RTP SSRC being the time-delayed duplicate. This is not critical, however, and receivers should size their playout buffers based on the "a=duplication-delay:" attribute, and play the stream that arrives first in preference, with the other stream acting as a repair stream, irrespective of the order in which they are signalled.

## 5. Use of RTP and RTCP with Spatial Redundancy

When using spatial redundancy, the redundant RTP stream is sent on using a different source and/or destination address/port pair. This will be a separate RTP session to the session conveying the main RTP stream.

The SSRCs used for the main and redundant streams MUST be chosen randomly, following the rules in Section 8 of [RFC3550]. Accordingly, they will almost certainly not match each other. The sender MUST, however, use the same RTCP CNAME for both the main and redundant streams, and MUST include an "a=ssrc:... srcname:..." attribute to correlate the flows. An "a=group:DUP" attribute is used to indicate duplication.

### 5.1. RTCP Considerations

If RTCP is being sent for the main RTP stream, then the sender MUST also generate RTCP for the redundant RTP stream. The RTCP for the redundant RTP stream is generated exactly as-if the redundant RTP stream were a regular media stream; the sender MUST NOT duplicate the RTCP packets sent for the main RTP stream. The sender MUST use the same RTCP CNAME in the RTCP reports it sends for the main and redundant streams, so that the receiver can synchronize them.

The main and redundant streams are conceptually synchronised using the standard RTCP SR-based mechanism, deriving a mapping between their timelines. The RTP timestamps and sequence numbers SHOULD be identical in the main and redundant streams, however, making the mapping trivial in most cases.

Both main and redundant streams, and their corresponding RTCP, will be received. If RTCP is used, receivers MUST generate RTCP reports for both main and redundant streams in the usual way, treating them as entirely separate media streams.

### 5.2. Signaling Considerations

The required SDP grouping semantics have been defined in [I-D.begen-mmusic-redundancy-grouping]. In the following example, the redundant streams have different IP destination addresses. The example shows the same UDP port number and IP source addresses, but either or both could have been different for the two streams.

```
v=0
o=ali 1122334455 1122334466 IN IP4 dup.example.com
s=DUP Grouping Semantics
t=0 0
a=group:DUP S1a S1b
m=video 30000 RTP/AVP 100
c=IN IP4 233.252.0.1/127
a=source-filter:incl IN IP4 233.252.0.1 198.51.100.1
a=rtpmap:100 MP2T/90000
a=ssrc:1000 cname:chl@example.com
a=ssrc:1000 srcname:45:a8:f4:19:b4:c3
a=mid:S1a
m=video 30000 RTP/AVP 101
c=IN IP4 233.252.0.2/127
a=source-filter:incl IN IP4 233.252.0.2 198.51.100.1
a=rtpmap:101 MP2T/90000
a=ssrc:1010 cname:chl@example.com
a=ssrc:1010 srcname:45:a8:f4:19:b4:c3
a=mid:S1b
```

## 6. Use of RTP and RTCP with Temporal and Spatial Redundancy

This uses the same RTP/RTCP mechanisms, plus a combination of both sets of signaling.

## 7. Security Considerations

The security considerations of [RFC3550], [I-D.begen-mmusic-temporal-interleaving], and [I-D.begen-mmusic-redundancy-grouping] apply.

If stream de-duplication is done by an in-network middlebox, rather than by an end system, that middlebox can work if Secure RTP (SRTP) encryption is used [RFC3711], since the RTP headers are in the clear. Doing so would break the authentication when the SSRC is rewritten, unless the de-duplication middlebox were trusted to re-authenticate the packets. This would require additional signalling which is not specified here, since de-duplication in the receiver end system is expected to be the more common use case.

## 8. IANA Considerations

No IANA actions are required.

## 9. Acknowledgments

Thanks to Magnus Westerlund for his suggestions.

## 10. References

### 10.1. 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.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [I-D.begen-mmusic-temporal-interleaving]  
Begen, A., Cai, Y., and H. Ou, "Delayed Duplication Attribute in the Session Description Protocol", draft-begen-mmusic-temporal-interleaving-04 (work in progress), March 2012.
- [I-D.begen-mmusic-redundancy-grouping]  
Begen, A., Cai, Y., and H. Ou, "Duplication Grouping Semantics in the Session Description Protocol", draft-begen-mmusic-redundancy-grouping-03 (work in progress), March 2012.
- [RFC3711] Baugher, M., McGrew, D., Naslund, M., Carrara, E., and K. Norrman, "The Secure Real-time Transport Protocol (SRTP)", RFC 3711, March 2004.

### 10.2. Informative References

- [RFC2354] Perkins, C. and O. Hodson, "Options for Repair of Streaming Media", RFC 2354, June 1998.
- [RFC4588] Rey, J., Leon, D., Miyazaki, A., Varsa, V., and R. Hakenberg, "RTP Retransmission Payload Format", RFC 4588, July 2006.
- [RFC5109] Li, A., "RTP Payload Format for Generic Forward Error Correction", RFC 5109, December 2007.
- [IC2011] Evans, J., Begen, A., Greengrass, J., and C. Filsfils, "Toward Lossless Video Transport (to appear in IEEE Internet Computing)", November 2011.

Authors' Addresses

Ali Begen  
Cisco  
181 Bay Street  
Toronto, ON M5J 2T3  
CANADA

Email: [abegen@cisco.com](mailto:abegen@cisco.com)

Colin Perkins  
University of Glasgow  
School of Computing Science  
Glasgow, G12 8QQ  
UK

Email: [csp@csperkins.org](mailto:csp@csperkins.org)



AVTEXT Working Group  
Internet-Draft  
Intended status: Informational  
Expires: April 25, 2013

J. Xia  
Huawei  
October 22, 2012

Content Splicing for RTP Sessions  
draft-ietf-avtext-splicing-for-rtp-11

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. Splicing is commonly used for local advertisement insertion by cable operators, replacing a national advertisement content with a local advertisement.

This memo describes some use cases for content splicing and a set of requirements for splicing content delivered by RTP. It provides concrete guidelines for how an RTP mixer can be used to handle content splicing.

Status of this Memo

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

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

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

This Internet-Draft will expire on April 25, 2013.

Copyright Notice

Copyright (c) 2012 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. System Model and Terminology . . . . .	3
3. Requirements for RTP Splicing . . . . .	6
4. Content Splicing for RTP sessions . . . . .	7
4.1. RTP Processing in RTP Mixer . . . . .	7
4.2. RTCP Processing in RTP Mixer . . . . .	8
4.3. Considerations for Handling Media Clipping at the RTP Layer . . . . .	10
4.4. Congestion Control Considerations . . . . .	11
4.5. Considerations for Implementing Undetectable Splicing . .	12
5. Implementation Considerations . . . . .	13
6. Security Considerations . . . . .	13
7. IANA Considerations . . . . .	14
8. Acknowledgments . . . . .	14
9. 10. Appendix- Why Mixer Is Chosen . . . . .	14
10. References . . . . .	15
10.1. Normative References . . . . .	15
10.2. Informative References . . . . .	15
Author's Address . . . . .	16

## 1. Introduction

This document outlines how content splicing can be used in RTP sessions. Splicing, in general, is a process where part of a multimedia content is replaced with other multimedia content, and delivered to the receivers for a period of time. The substitutive content can be provided for example via another stream or via local media file storage. One representative use case for splicing is local advertisement insertion, allowing content providers to replace the national advertising content with its own regional advertising content prior to delivering the regional advertising content to the receivers. Besides the advertisement insertion use case, there are other use cases in which splicing technology can be applied. For example, splicing a recorded video into a video conferencing session, or implementing a playlist server that stitches pieces of video together.

Content splicing is a well-defined operation in MPEG-based cable TV systems. Indeed, the Society for Cable Telecommunications Engineers (SCTE) has created two standards, [SCTE30] and [SCTE35], to standardize MPEG2-TS splicing procedure. SCTE 30 creates a standardized method for communication between advertisements server and splicer, and SCTE 35 supports splicing of MPEG2 transport streams.

When using multimedia splicing into the internet, the media may be transported by RTP. In this case the original media content and substitutive media content will use the same time period, but may contain different numbers of RTP packets due to different media codecs and entropy coding. This mismatch may require some adjustments of the RTP header sequence number to maintain consistency. [RFC3550] provides the tools to enabled seamless content splicing in RTP session, but to date there has been no clear guidelines on how to use these tools.

This memo outlines the requirements for content splicing in RTP sessions and describes how an RTP mixer can be used to meet these requirements.

## 2. System Model and Terminology

In this document, an intermediary network element, the Splicer handles RTP splicing. The Splicer can receive main content and substitutive content simultaneously, but will send one of them at one point of time.

When RTP splicing begins, the splicer sends the substitutive content

to the RTP receiver instead of the main content for a period of time. When RTP splicing ends, the splicer switches back sending the main content to the RTP receiver.

A simplified RTP splicing diagram is depicted in Figure 1, in which only one main content flow and one substitutive content flow are given. Actually, the splicer can handle multiple splicing for multiple RTP sessions simultaneously. RTP splicing may happen more than once in multiple time slots during the lifetime of the main RTP stream. The methods how splicer learns when to start and end the splicing is out of scope for this document.

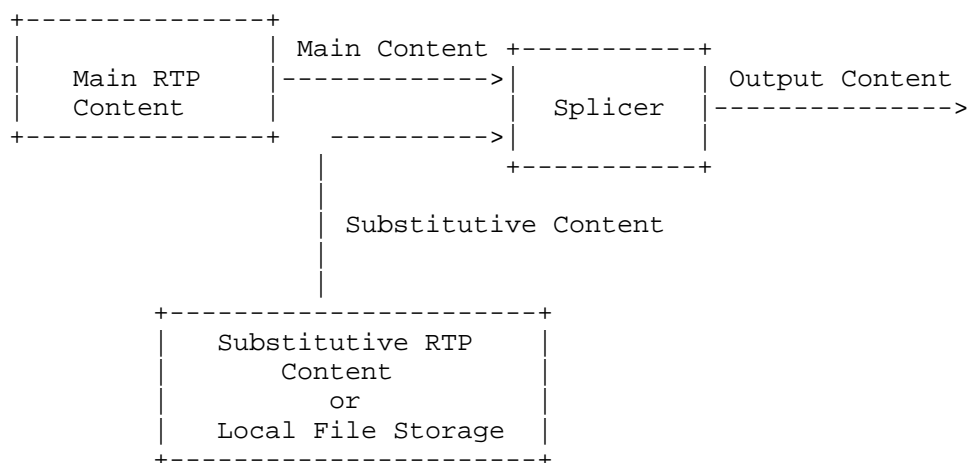


Figure 1: RTP Splicing Architecture

This document uses the following terminologies.

#### Output RTP Stream

The RTP stream that the RTP receiver is currently receiving. The content of output RTP stream can be either main content or substitutive content.

#### Main Content

The multimedia content that are conveyed in main RTP stream. Main content will be replaced by the substitutive content during splicing.

#### Main RTP Stream

The RTP stream that the splicer is receiving. The content of main RTP stream can be replaced by substitutive content for a period of time.

#### Main RTP Sender

The sender of RTP packets carrying the main RTP stream.

#### Substitutive Content

The multimedia content that replaces the main content during splicing. The substitutive content can for example be contained in an RTP stream from a media sender or fetched from local media file storage.

#### Substitutive RTP Stream

A RTP stream with new content that will replace the content in the main RTP stream. Substitutive RTP stream and main RTP stream are two separate streams. If the substitutive content is provided via substitutive RTP stream, the substitutive RTP Stream must pass through the splicer before the substitutive content is delivered to receiver.

#### Substitutive RTP Sender

The sender of RTP packets carrying the substitutive RTP stream.

#### Splicing In Point

A virtual point in the RTP stream, suitable for substitutive content entry, typically in the boundary between two independently decodable frames.

#### Splicing Out Point

A virtual point in the RTP stream, suitable for substitutive content exist, typically in the boundary between two independently decodable frames.

#### Splicer

An intermediary node that inserts substitutive content into main RTP stream. The splicer sends substitutive content to RTP receiver instead of main content during splicing. It is also responsible for processing RTCP traffic between the RTP sender and

the RTP receiver.

### 3. Requirements for RTP Splicing

In order to allow seamless content splicing at the RTP layer, the following requirements must be met. Meeting these will also allow, but not require, seamless content splicing at layers above RTP.

#### REQ-1:

The splicer should be agnostic about the network and transport layer protocols used to deliver the RTP streams.

#### REQ-2:

The splicing operation at the RTP layer must allow splicing at any point required by the media content, and must not constrain when splicing in or splicing out operations can take place.

#### REQ-3:

Splicing of RTP content must be backward compatible with the RTP/RTCP protocol, associated profiles, payload formats, and extensions.

#### REQ-4:

The splicer will modify the content of RTP packets, and break the end-to-end security, e.g., breaking data integrity and source authentication. If the Splicer is designated to insert substitutive content, it must be trusted, i.e., be in the security context(s) as the main RTP sender, the substitutive RTP sender, and the receivers. If encryption is employed, the splicer must be able to decrypt the inbound RTP packets and re-encrypt the outbound RTP packets after splicing.

#### REQ-5:

The splicer should rewrite as necessary and forward RTCP messages (e.g., including packet loss, jitter, etc.) sent from downstream receiver to the main RTP sender or the substitutive RTP sender, and thus allow the main RTP sender or substitutive RTP sender to learn the performance of the downstream receiver when its content is being passed to RTP receiver. In addition, the splicer should rewrite RTCP messages from the main RTP sender or substitutive RTP sender to the receiver.

## REQ-6:

The splicer must not affect other RTP sessions running between the RTP sender and the RTP receiver, and must be transparent for the RTP sessions it does not splice.

## REQ-7:

The splicer should be able to modify the RTP stream such that the splicing point is not easy to be detected by the RTP receiver at the RTP layer. For the advertisement insertion use case, it is important to make it difficult for the RTP receiver to detect where an advertisement insertion is starting or ending from the RTP packets, and thus avoiding the RTP receiver from filtering out the advertisement content. This memo only focuses on making the splicing undetectable at the RTP layer. How (or if) the splicing is made undetectable in the media stream is outside the scope of this memo. The corresponding processing is depicted in section 4.5.

#### 4. Content Splicing for RTP sessions

The RTP specification [RFC3550] defines two types of middlebox: RTP translators and RTP mixers. Splicing is best viewed as a mixing operation. The splicer generates a new RTP stream that is a mix of the main RTP stream and the substitutive RTP stream. An RTP mixer is therefore an appropriate model for a content splicer. In next four subsections (from subsection 4.1 to subsection 4.4), the document analyzes how the mixer handles RTP splicing and how it satisfies the general requirements listed in section 3. In subsection 4.5, the document looks at REQ-7 in order to hide the fact that splicing take place.

##### 4.1. RTP Processing in RTP Mixer

A splicer could be implemented as a mixer that receives the main RTP stream and the substitutive content (possibly via a substitutive RTP stream), and sends a single output RTP stream to the receiver(s). That output RTP stream will contain either the main content or the substitutive content. The output RTP stream will come from the mixer, and will have the synchronization source (SSRC) of the mixer rather than the main RTP sender or the substitutive RTP sender.

The mixer uses its own SSRC, sequence number space and timing model when generating the output stream. Moreover, the mixer may insert the SSRC of main RTP stream into contributing source (CSRC) list in

the output media stream.

At the splicing in point, when the substitutive content becomes active, the mixer chooses the substitutive RTP stream as input stream at splicing in point, and extracts the payload data (i.e., substitutive content). If the substitutive content comes from local media file storage, the mixer directly fetches the substitutive content. After that, the mixer encapsulates substitutive content instead of main content as the payload of the output media stream, and then sends the output RTP media stream to receiver. The mixer may insert the SSRC of substitutive RTP stream into CSRC list in the output media stream. If the substitutive content comes from local media file storage, the mixer should leave the CSRC list blank.

At the splicing out point, when the substitutive content ends, the mixer retrieves the main RTP stream as input stream at splicing out point, and extracts the payload data (i.e., main content). After that, the mixer encapsulates main content instead of substitutive content as the payload of the output media stream, and then sends the output media stream to the receivers. Moreover, the mixer may insert the SSRC of main RTP stream into CSRC list in the output media stream as before.

Note that if the content is too large to fit into RTP packets sent to RTP receiver, the mixer needs to transcode or perform application-layer fragmentation. Usually the mixer is deployed as part of a managed system and MTU will be carefully managed by this system. This document does not raise any new MTU related issues compared to a standard mixer described in [RFC3550].

Splicing may occur more than once during the lifetime of main RTP stream, this means the mixer needs to send main content and substitutive content in turn with its own SSRC identifier. From receiver point of view, the only source of the output stream is the mixer regardless of where the content is coming from.

#### 4.2. RTCP Processing in RTP Mixer

By monitoring available bandwidth and buffer levels and by computing network metrics such as packet loss, network jitter, and delay, RTP receiver can learn the network performance and communicate this to the RTP sender via RTCP reception reports.

According to the description in section 7.3 of [RFC3550], the mixer splits the RTCP flow between sender and receiver into two separate RTCP loops, RTP sender has no idea about the situation on the receiver. But splicing is a processing that the mixer selects one media stream from multiple streams rather than mixing them, so the

mixer can leave the SSRC identifier in the RTCP report intact (i.e., the SSRC of downstream receiver), this enables the main RTP sender or the substitutive RTP sender to learn the situation on the receiver.

If the RTCP report corresponds to a time interval that is entirely main content or entirely substitutive content, the number of output RTP packets containing substitutive content is equal to the number of input substitutive RTP packets (from substitutive RTP stream) during splicing, in the same manner, the number of output RTP packets containing main content is equal to the number of input main RTP packets (from main RTP stream) during non-splicing unless the mixer fragment the input RTP packets. This means that the mixer does not need to modify the loss packet fields in reception report blocks in RTCP reports. But if the mixer fragments the input RTP packets, it may need to modify the loss packet fields to compensate for the fragmentation. Whether the input RTP packets are fragmented or not, the mixer still needs to change the SSRC field in report block to the SSRC identifier of the main RTP sender or the substitutive RTP sender, and rewrite the extended highest sequence number field to the corresponding original extended highest sequence number before forwarding the RTCP report to the main RTP sender or the substitutive RTP sender.

If the RTCP report spans the splicing in point or the splicing out point, it reflects the characteristics of the combination of main RTP packets and substitutive RTP packets. In this case, the mixer needs to divide the RTCP report into two separate RTCP reports and send them to their original RTP senders respectively. For each RTCP report, the mixer also needs to make the corresponding changes to the packet loss fields in report block besides the SSRC field and the extended highest sequence number field.

If the mixer receives an RTCP extended report (XR) block, it should rewrite the XR report block in a similar way to the reception report block in the RTCP report.

Besides forwarding the RTCP reports sent from RTP receiver, the mixer can also generate its own RTCP reports to inform the main RTP sender or the substitutive RTP sender of the reception quality of the content reaches the mixer when the content is not sent to the RTP receiver. These RTCP reports use the SSRC of the mixer. If the substitutive content comes from local media file storage, the mixer does not need to generate RTCP reports for the substitutive stream.

Based on above RTCP operating mechanism, the RTP sender whose content is being passed to receiver will see the reception quality of its stream as received by the mixer, and the reception quality of spliced stream as received by the receiver. The RTP sender whose content is



not being passed to receiver will only see the reception quality of its stream as received by the mixer.

The mixer must forward RTCP SDES and BYE packets from the receiver to the sender, and may forward them in inverse direction as defined in section 7.3 of [RFC3550].

Once the mixer receives an RTP/AVPF [RFC4585] transport layer feedback packet, it must handle it carefully as the feedback packet may contain the information of the content that come from different RTP senders. In this case the mixer needs to divide the feedback packet into two separate feedback packets and process the information in the feedback control information (FCI) in the two feedback packets, just as the RTCP report process described above.

If the substitutive content comes from local media file storage (i.e., the mixer can be regarded as the substitutive RTP sender), any RTCP packets received from downstream relate to the substitutive content must be terminated on the mixer without any further processing.

#### 4.3. Considerations for Handling Media Clipping at the RTP Layer

This section provides informative guideline about how media clipping is shaped and how the mixer deal with the media clipping only at the RTP layer. Dealing with the media clipping at the RTP layer just do a good quality implementation, perfectly erasing the media clipping needs more considerations at the higher layers, how the media clipping is erased at the higher layers is outside of the scope of this memo.

If the time slot for substitutive content mismatches (is shorter or longer than) the duration of the main content to be replaced, then media clipping may occur at the splicing point and thus impact the user's experience.

If the substitutive content has shorter duration from the main content, then there will be a gap in the output RTP stream. The RTP sequence number will be contiguous across this gap, but there will be an unexpected jump in the RTP timestamp. This gap will cause the receiver to have nothing to play. This is unavoidable, unless the mixer adjusts the splice in or splice out point to compensate, sending more of the main RTP stream in place of the shorter substitutive stream, or unless the mixer can vary the length of the substitutive content. It is the responsibility of the higher layer protocols to ensure that the substitutive content is of the same duration as the main content to be replaced.

If the insertion duration is longer than the reserved gap duration, there will be an overlap between the substitutive RTP stream and the main RTP stream at splicing out point. One straightforward approach is that the mixer takes an ungraceful action, terminating the splicing and switching back to main RTP stream even if this may cause media stuttering on receiver. Alternatively, the mixer may transcode the substitutive content to play at a faster rate than normal, to adjust it to the length of the gap in the main content, and generate a new RTP stream for the transcoded content. This is a complex operation, and very specific to the content and media codec used.

#### 4.4. Congestion Control Considerations

If the substitutive content has somewhat different characteristics from the main content it replaces, or if the substitutive content is encoded with a different codec or has different encoding bitrate, it might overload the network and might cause network congestion on the path between the mixer and the RTP receiver(s) that would not have been caused by the main content.

To be robust to network congestion and packet loss, a mixer that is performing splicing must continuously monitor the status of downstream network by monitoring any of the following RTCP reports that are used:

1. RTCP receiver reports indicate packet loss [RFC3550].
2. RTCP NACKs for lost packet recovery [RFC4585].
3. RTCP ECN Feedback information [I-D.ietf-avtcore-ecn-for-rtp].

Once the mixer detects congestion on its downstream link, it will treat these reports as follows:

1. If the mixer receives the RTCP receiver reports with packet loss indication, it will forward the reports to the substitutive RTP sender or the main RTP sender as described in section 4.2.
2. If mixer receives the RTCP NACK packets defined in [RFC4585] from RTP receiver for packet loss recovery, it first identifies the content category of lost packets to which the NACK corresponds. Then, the mixer will generate new RTCP NACK for the lost packets with its own SSRC, and make corresponding changes to their sequence numbers to match original, pre-spliced, packets. If the lost substitutive content comes from local media file storage, the mixer acting as substitutive RTP sender will directly fetch the lost substitutive content and retransmit it to RTP receiver. The mixer may buffer the sent RTP packets and do the

retransmission.

It is somewhat complex that the lost packets requested in a single RTCP NACK message not only contain the main content but also the substitutive content. To address this, the mixer must divide the RTCP NACK packet into two separate RTCP NACK packets: one requests for the lost main content, and another requests for the lost substitutive content.

3. If an ECN-aware mixer receives RTCP ECN feedbacks (RTCP ECN feedback packets or RTCP XR summary reports) defined in [I-D.ietf-avtcore-ecn-for-rtp] from the RTP receiver, it must process them in a similar way to the RTP/AVPF feedback packet or RTCP XR process described in section 4.2 of this memo.

These three methods require the mixer to run a congestion control loop and bitrate adaptation between itself and RTP receiver. The mixer can thin or transcode the main RTP stream or the substitutive RTP stream, but such operations are very inefficient and difficult, and bring undesirable delay. Fortunately in this memo, the mixer acting as splicer can rewrite the RTCP packets sent from the RTP receiver and forward them to the RTP sender, thus letting the RTP sender knows that congestion is being experienced on the path between the mixer and the RTP receiver. Then, the RTP sender applies its congestion control algorithm and reduces the media bitrate to a value that is in compliance with congestion control principles for the slowest link. The congestion control algorithm may be a TCP-friendly bitrate adaptation algorithm specified in [RFC5348], or a DCCP congestion control algorithms defined in [RFC5762].

If the substitutive content comes from local media file storage, the mixer must directly reduce the bitrate as if it were the substitutive RTP sender.

From above analysis, to reduce the risk of congestion and remain the bandwidth consumption stable over time, the substitutive RTP stream is recommended to be encoded at an appropriate bitrate to match that of main RTP stream. If the substitutive RTP stream comes from the substitutive RTP sender, this sender had better has some knowledge about the media encoding bitrate of main content in advance. How it knows that is out of scope in this draft.

#### 4.5. Considerations for Implementing Undetectable Splicing

If it is desirable to prevent receivers from detecting that splicing is occurring at the RTP layer, the mixer must not include a CSRC list in outgoing RTP packets, and must not forward RTCP messages from the main RTP sender or from the substitutive RTP sender. Due to the

absence of CSRC list in the output RTP stream, the RTP receiver only initiates SDP, BYE and APP packets to the mixer without any knowledge of the main RTP sender and the substitutive RTP sender.

CSRC list identifies the contributing sources, these SSRC identifiers of contributing sources are kept globally unique for each RTP session. The uniqueness of SSRC identifier is used to resolve collisions and detecting RTP-level forwarding loops as defined in section 8.2 of [RFC3550]. The absence of CSRC list in this case will create a danger that loops involving those contributing sources could not be detected. The loops could occur if either the mixer is misconfigured to form a loop, or a second mixer/translator is added, causing packets to loop back to upstream of the original mixer and hence wasting the network bandwidth. So Non-RTP means must be used to detect and resolve loops if the mixer does not add a CSRC list.

## 5. Implementation Considerations

When the mixer is used to handle RTP splicing, RTP receiver does not need any RTP/RTCP extension for splicing. As a trade-off, additional overhead could be induced on the mixer which uses its own sequence number space and timing model. So the mixer will rewrite RTP sequence number and timestamp whatever splicing is active or not, and generate RTCP flows for both sides. In case the mixer serves multiple main RTP streams simultaneously, this may lead to more overhead on the mixer.

If undetectable splicing requirement is required, CSRC list is not included in outgoing RTP packet, this brings a potential issue with loop detection as briefly described in section 4.5.

## 6. Security Considerations

The splicing application is subject to the general security considerations of the RTP specification [RFC3550].

The mixer acting as splicer replaces some content with other content in RTP packets, thus breaking any RTP level end-to-end security, such as integrity protection and source authentication. Thus any RTP level or outside security mechanism, such as IPsec or DTLS will use a security association between the splicer and the receiver. When using SRTP the splicer could be provisioned with the same security association as the main RTP sender. Using a limitation in the SRTP security services, the splicer can modify and re-protect the RTP packets without enabling the receiver to detect if the data comes from the original source or from the splicer.

Security goals to have source authentication all the way from the RTP main sender to the receiver through the splicer is not possible with splicing. The nature of this RTP service offered by a network operator employing a content splicer is that the RTP layer security relationship is between the receiver and the splicer, and between the senders and the splicer, are not end-to-end. This appears to invalidate the undetectability goal, but in the common case the receiver will consider the splicer as the main media source.

Commonly no RTP level security mechanism is employed. Instead only payload security mechanisms (e.g., ISMACryp [ISMACryp]) are used. If any payload internal security mechanisms are used, only the RTP sender and the RTP receiver can learn the security keying material generated by such internal security mechanism, in which case, any middlebox (e.g., splicer) between the RTP sender and the RTP receiver can't get such keying material, and thus fail to perform splicing. This would require a new method to be defined to make the splicer learn the security keying material, but which is out of scope of this memo.

## 7. IANA Considerations

No IANA actions are required.

## 8. Acknowledgments

The following individuals have reviewed the earlier versions of this specification and provided very valuable comments: Colin Perkins, Magnus Westerlund, Roni Even, Tom Van Caenegem, Joerg Ott, David R Oran, Cullen Jennings, Ali C Begen, Charles Eckel and Ning Zong.

## 9. 10. Appendix- Why Mixer Is Chosen

Translator and mixer both can realize splicing by changing a set of RTP parameters.

Translator has no SSRC, hence it is transparent to RTP sender and receiver. Therefore, RTP sender sees the full path to the receiver when translator is passing its content. When translator insert the substitutive content RTP sender could get a report on the path up to translator itself. Additionally, if splicing does not occur yet, translator does not need to rewrite RTP header, the overhead on translator can be avoided.

If mixer is used to do splicing, it can also allow RTP sender to

learn the situation of its content on receiver or on mixer just like translator does, which is specified in section 4.2. Compared to translator, mixer's outstanding benefit is that it is pretty straight forward to do with RTCP messages, for example, bit-rate adaptation to handle varying network conditions. But translator needs more considerations and its implementation is more complex.

From above analysis, both translator and mixer have their own advantages: less overhead or less complexity on handling RTCP. Through long and sophisticated discussion, the avtext WG members prefer less complexity rather than less overhead and incline to mixer to do splicing.

If one chooses mixer as splicer, the overhead on mixer must be taken into account even if the splicing does not occur yet.

## 10. References

### 10.1. 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.
- [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.
- [I-D.ietf-avtcore-ecn-for-rtp] Westerlund, M., "Explicit Congestion Notification (ECN) for RTP over UDP", draft-ietf-avtcore-ecn-for-rtp-08 (work in progress), May 2012.

### 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.
- [RFC5348] Floyd, S., Handley, M., Padhye, J., and J. Widmer, "TCP Friendly Rate Control (TFRC): Protocol Specification", RFC 5348, September 2008.
- [RFC5762] Perkins, C., "RTP and the Datagram Congestion Control Protocol (DCCP)", RFC 5762, April 2010.

- [SCTE30] Society of Cable Telecommunications Engineers (SCTE),  
"Digital Program Insertion Splicing API", 2009.
- [SCTE35] Society of Cable Telecommunications Engineers (SCTE),  
"Digital Program Insertion Cueing Message for Cable",  
2011.
- [ISMACryp] Internet Streaming Media Alliance (ISMA), "ISMA Encryption  
and Authentication Specification 2.0", November 2007.

Author's Address

Jinwei Xia  
Huawei  
Software No.101  
Nanjing, Yuhuatai District 210012  
China

Phone: +86-025-86622310  
Email: xiajinwei@huawei.com





Network Working Group  
Internet-Draft  
Intended status: Standards Track  
Expires: April 25, 2013

M. Westerlund  
B. Burman  
P. Sandgren  
Ericsson  
October 22, 2012

RTCP SDES Item SRCNAME to Label Individual Sources  
draft-westerlund-avtext-rtcp-sdes-srcname-02

## Abstract

This document defines a new SDES item called SRCNAME which uniquely identifies a single media source, like a camera or a microphone. That way anyone receiving the SDES information from a set of interlinked RTP sessions can determine which SSRCs are logically related to the same source. It can equally be used to label SSRC multiplexed related streams, such as FEC or Retransmission streams related to the original source stream in the same session. In addition the new SDES item is also defined for usage with the SDP source specific media attribute ("a=ssrc"), enabling an end-point to declare and learn the source bindings through signalling ahead of receiving RTP/RTCP packets.

## Status of this Memo

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

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

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

This Internet-Draft will expire on April 25, 2013.

## Copyright Notice

Copyright (c) 2012 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. Requirements Language . . . . .	3
3. Problem Description . . . . .	3
4. Motivation . . . . .	4
4.1. RTP SSRC . . . . .	4
4.2. RTCP SDES CNAME . . . . .	4
4.3. SDP . . . . .	5
4.4. Implicit Methods . . . . .	5
5. Proposed Solution . . . . .	6
5.1. SRCNAME Contents . . . . .	6
5.2. SRCNAME in SDES . . . . .	7
5.3. SRCNAME in SDP . . . . .	7
5.4. SRCNAME in RTP Header Extension . . . . .	8
6. SRCNAME Format . . . . .	8
7. SDES Item SRCNAME . . . . .	8
8. SRCNAME in SDP . . . . .	9
9. SRCNAME as RTP Header Extension . . . . .	10
10. Examples . . . . .	11
10.1. Simulcast . . . . .	11
10.2. SVC with multi-session transmission . . . . .	13
10.3. Retransmission . . . . .	15
10.4. Forward Error Correction . . . . .	16
11. Usage with the Offer/Answer Model . . . . .	17
12. Backward Compatibility . . . . .	17
13. IANA Considerations . . . . .	18
14. Security Considerations . . . . .	18
15. References . . . . .	19
15.1. Normative References . . . . .	19
15.2. Informative References . . . . .	19
Authors' Addresses . . . . .	21

## 1. Introduction

RTP [RFC3550] has always been a protocol that supports multiple participants, each sending their own media streams in RTP sessions. Previously, many implementations have aimed only at point to point voice over IP with a single source in each end-point. Even client implementations aimed at video conferences have often been built with the assumption around central mixers that only deliver a single media stream per media type. However, more advanced client implementations may transmit multiple streams in the same RTP session and there may be tight relations between different streams and their SSRCs. For example, a client with several cameras that uses simulcast to send streams with different encodings of the video from each camera have the need of conveying the relation of the streams to the receiver. A similar example is a client with several cameras that uses SVC multi-session transmission [RFC6190] and also here the receiver needs to know which streams relate to which video source. Other examples of tight RTP relations are a retransmission stream and its original stream, and cases of forward error correction (FEC), where a client needs to associate a number of source streams with, in general, a different number of repair streams.

## 2. 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. Problem Description

In a scenario where an endpoint needs to send several RTP media streams, in a single RTP session or spread across several RTP sessions, and where two or more of those streams are somehow related, that relation information is today not always possible to convey in a timely manner to entities (endpoints and middle nodes) that need it.

An RTP Mixer [RFC5117], on the other hand, must have all the SDP information available and can provide it to any number of participants, since there must be a mapping from the original sources to the Mixer's own streams, which are in turn distributed to all other participants. That is also true for a source projecting mixer, since there is a projection algorithm that must be made to work. It is even likely that the Mixer is allowed to provide the stream relation and impose that onto all of the clients, rather than trying to map a wide variety of different relations onto what it provides.

A single relation between two or more streams means that each stream has a certain "role" in that specific relation. A "role" is related to a specific reason to group a set of streams. The number of different grouping tags defined in various RFC for use with the SDP group attribute [RFC5888], as well as the media decoding dependency attribute [RFC5583] can be used as an indication of the different roles that may need to be described.

Those stream relational roles are typically application-specific, can sometimes be complex, and a single stream can even take on several roles. The major difference between roles is that they commonly do not share the same hierarchy root node and sometimes also middle nodes differ between roles. All roles however use the same hierarchy leaves, being the RTP media streams, but different roles may want to name leaves differently. It should be possible to express such relation structure and allow a single stream to hold several roles. It is believed to be sufficient if a single stream role can be described as being part of a relation hierarchy.

#### 4. Motivation

This section contains a brief description of existing techniques that conceivably could be used to provide information on RTP stream relations, and a motivation why those are not always sufficient. In addition, there are defined milestones for RTP stream duplication [I-D.ietf-avtext-rtp-duplication] in IETF AVTEXT and stream duplication grouping [I-D.ietf-mmusic-duplication-grouping] in MMUSIC WG that makes normative references to this document.

##### 4.1. RTP SSRC

To rely on using the same RTP Synchronization SouRCe (SSRC) for all streams related to a particular media source is many times not possible when the related streams are part of the same RTP session, since the SSRC itself is the identifier to tell the streams apart. This method is not robust against SSRC collision and potentially forces cascading SSRC changes between sessions. It does also not provide any information in how the streams are related.

##### 4.2. RTCP SDP CNAME

CNAME is not sufficient to express the necessary type of relation, although that is commonly inferred from end-points that have only one media stream per media type. The primary use of CNAME in multi-source usages is instead to indicate which end-point and what synchronization context a particular media stream relates to, and that usually means that all streams sent from a client have the same

CNAME.

#### 4.3. SDP

A common solution is to use SDP [RFC4566] attributes to convey the relation between streams. Session-multiplexed streams can be associated with an attribute that groups different SDP m-lines [RFC5888], and SSRC-multiplexed streams can be grouped at the media level for each SDP m-line [RFC5576]. For example, Forward Error Correction Grouping Semantics in the Session Description Protocol [RFC5956] uses that media level grouping with the "FEC-FR" tag to group FEC associations when the different streams from a source are SSRC-multiplexed in the same RTP session.

Using SDP attributes may work fine in the case when the receivers of the streams also get an SDP describing the bindings of all the streams, but that is not always the case. One such example is a highly dynamic conference session where a large amount of clients are communicating with each other via an RTP Translator [RFC5117]. The RTP Translator forwards all RTP and RTCP traffic from a client to all other clients and the clients can be prepared to receive any number of streams of certain specified media. When a new client joins the session, the other clients may not be notified via explicit signalling before starting to receive media streams from this new client. Such notification could for example be made through a SIP Update with a new SDP containing an explicit list of the new streams, but there are also other possibilities. The clients will instead detect the new client's streams directly via RTP and RTCP. Similar situations typically arise in multicast scenarios. In those cases, there is no way for a client or middle node to identify if and how certain streams are related to each other, since that information was only included in the SDP, if at all.

#### 4.4. Implicit Methods

RTP Retransmission Payload Format [RFC4588] describes a solution for finding the association between original streams and retransmission streams when SSRC-multiplexing is used. The association can be resolved when the receiver receives a retransmission packet matching a retransmission request sent earlier. However, the RFC states that this mechanism might fail if there are two outstanding requests for the same packet sequence number in two different original streams of a session. Therefore, to avoid ambiguity in unicast a receiver **MUST NOT** have two outstanding requests for the same packet sequence number in two different original streams before the association is resolved. For multicast, however, this ambiguity cannot be avoided and SSRC-multiplexing of original and retransmission streams is therefore prohibited in multicast. By defining a solution for one to one

mapping between an original stream and any supporting streams, this issue can be avoided in the future.

Note: This document does not update RFC 4588 to use the proposed solution, but it may be done in the future.

## 5. Proposed Solution

To enable an RTP session participant to determine the close relation of different streams without the above mentioned problems, a new method for identifying such sources is needed. This identification is called Source Name, or SRCNAME and is a unique identifier identifying a single media source, like a camera, a microphone, a particular media mix, or conceptual stream.

### 5.1. SRCNAME Contents

The basic idea is that streams with matching SRCNAME are related, similar to the idea with RTCP SDES CNAME.

It is assumed that related streams will share the same synchronization context, meaning that the SRCNAME is scoped by CNAME and need not duplicate any CNAME information.

The SRCNAME format includes "." (%x2E) as a hierarchy separator, allowing a stream to relate to another stream at a certain hierarchy level. Each hierarchy level is then a node in a hierarchy tree. For example, assume a video stream being provided in two different resolutions, named "lowres" and "hires", each being protected by a Forward Error Correction stream, with another additive FEC stream covering both resolutions. The low resolution video media stream could have a SRCNAME being "program1.video.lowres.media", and its FEC stream "program1.video.lowres.fec". By this, and although it is not a stream in itself, it is possible to use "program1.video.lowres" to refer to the set of related streams (in this case media and FEC) belonging to "lowres". If needed, it is still possible to refer to the individual, physical, streams by using one more level of the hierarchy (".media" and ".fec"). The SRCNAME for the additive FEC stream, covering both resolutions and their per-stream FEC, could be "program1.video.fec". Building on the same example, an high fidelity audio stream belonging to the above video could use an SRCNAME of "program1.audio.hifi".

Note that the hierarchy structure can be chosen entirely by the media sender, but it is anyway possible to decide stream relations, at what level the streams relate, and which other streams that are included in the relation at that level by matching SRCNAME hierarchically

left-to-right between "." hierarchy separators. The specific type of relation is not encoded into SRCNAME in any mandated way, but need to be stringently described by other means, for example SDP, and is out of scope for this specification. SRCNAME needs only express that streams are related, not exactly how the related streams should be processed together.

Note that SRCNAME need not be particularly human-readable as long as each node in the hierarchy has a tag that is unique for that CNAME context, which makes it possible to limit the SRCNAME size.

## 5.2. SRCNAME in SDES

RTP [RFC3550] defines the Source Description RTCP Packet (SDES), which contains one or more chunks, each of which is composed of SDES items describing the SSRC identified in that chunk. None of the present SDES items is, however, suitable for uniquely identifying a media source.

Therefore, we propose to define a new SDES item called the SRCNAME, which uses a unique label to identify a single media source, like a camera or a microphone. The source may also be a particular media mix or conceptual stream, such as the "most active speaker" output by a RTP mixer performing stream switching. That way, anyone receiving the SDES information from a set of interlinked RTP sessions or multiple SSRCs in the same session can determine which SSRCs are the same source. Connecting streams with SRCNAME can be done irrespective of which multiplexing type is used and it solves the problems with the current solutions described above.

## 5.3. SRCNAME in SDP

It is, however, possible that a receiver will receive the RTP streams before receiving SDES packets with all SRCNAME items and that would mean that the receiver cannot make the connections between SSRCs and SRCNAMEs when starting to receive the media. "Source-Specific Media Attributes in the Session Description Protocol (SDP)" [RFC5576] defines a way of declaring different attributes for SSRCs in each session in SDP, and if a new source attribute is added to this framework, it would be suitable for conveying the connections between SSRCs and SRCNAMEs before the media communication starts. Thus, in addition to the new SDES item we also define a new SDP source-specific media attribute called "srcname", which enables an end-point to declare and learn the source bindings ahead of receiving RTP/RTCP packets. Of course, this new SDP source attribute will not be useful for the case described above when clients did not get updates with new client's stream bindings, but it will be useful in most other cases.

#### 5.4. SRCNAME in RTP Header Extension

There is a risk that neither RTCP SDES nor SDP attributes are timely enough in cases where RTP streams are received before the SDES has arrived, in which case an RTP header extension [RFC5285] could be negotiated for use, containing a combination of CNAME and SRCNAME information. This type of rapid information synchronization through RTP header extension is similar to what is described in [RFC6051]. The RTP header extension need not be present in every RTP packet, for example only in the beginning of the stream, at key points, or periodically, according to the application's needs and as chosen by the media sender.

#### 6. SRCNAME Format

The SRCNAME MUST fulfill the requirements Section 6.5 in RTP [RFC3550] puts on SDES item values in general. These requirements is that it is a UTF-8 [RFC3629] string that have a maximum length of 255 bytes.

In addition, there are format restrictions to accommodate the relation hierarchy and multiple roles, as described by the following ABNF [RFC5234]:

```
srcname-node = 1*(%x01-09 / %x0B-0C / %x0E-2D / %x2F-FF)
               ; Same as RFC 4566 "byte-string"
               ; except for the hierarchy separator

srcname-content = srcname-node *(%x2E srcname-node)
```

Figure 1: SRCNAME Format ABNF

It is RECOMMENDED to use per communication session unique random identifiers, applying srcname-node restrictions, as srcname-node. The length of such srcname-node identifiers MAY be limited down to a single character, especially when the resulting SRCNAME has several nodes.

#### 7. SDES Item SRCNAME

Source Descriptions are a method that should work with all RTP topologies (assuming that any intermediary node is supporting this item) and existing RTP extensions. We propose to define a new SDES item called SRCNAME. That way, anyone receiving the SDES information from a set of interlinked RTP sessions or SSRCs in a single session



can determine which SSRCs are related to the same source, and at what hierarchy level.

This SRCNAME's relation to CNAME is the following. CNAME represents an end-point and a synchronization context. If the different sources identified by SRCNAMEs should be played out synchronized when receiving them in a multi-stream context, then the sources need to be in the same synchronization context. Thus in all cases, all SSRCs with the same SRCNAME will have the same CNAME. A given CNAME may contain multiple sets of sources using different SRCNAMEs.

The SDES SRCNAME item follows the same format as the other SDES items defined in RTP [RFC3550]:

```

      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
+-----+-----+-----+-----+-----+-----+-----+-----+
| SRCNAME=TBA1 |      length      | source name          | ...
+-----+-----+-----+-----+-----+-----+-----+-----+

```

Figure 2: SDES SRCNAME Format

The source name field MUST follow the above srcname-content definition. Multiple SDES SRCNAME describing different relation roles MAY be included.

When using the SRCNAME SDES item, it is equally important as CNAME. Thus SRCNAME is RECOMMENDED to be included in all full compound RTCP packets being sent. It MAY also be included in non-compound packets in cases where the implementation believes that there might be new receivers needing the information.

## 8. SRCNAME in SDP

"Source-Specific Media Attributes in the Session Description Protocol (SDP)" [RFC5576] defines a way of declaring attributes for SSRC in each session in SDP. With a new SDES item, it is possible to use this framework to define how SRCNAME can also be provided in the SDP for each SSRC in each RTP session, thus enabling an end-point to declare and learn the source bindings ahead of receiving RTP/RTCP packets.

Hence, we propose a new SDP source attribute called srcname with the following structure:

```
a=ssrc:<ssrc-id> srcname:<srcname>
```

The srcname value MUST be identical to the SRCNAME value the media sender will send in the SDES SRCNAME item in the SDES RTCP packets. Multiple srcname attributes MAY be used to describe multiple relation roles.

FormalABNF syntax [RFC5234] for the "srcname" attribute:

```
srcname-attr = "srcname:" srcname  
  
srcname = srcname-content  
  
attribute =/ srcname-attr  
; The definition of "attribute" is in RFC 4566.
```

Figure 3: SRCNAME Attribute ABNF

When used in SDP, srcname-content MUST use ISO 10646 in UTF-8 encoding, and MUST be independent of any "a=charset".

## 9. SRCNAME as RTP Header Extension

When SRCNAME information is carried as RTP header extension [RFC5285], the header extension MUST contain both CNAME and SRCNAME information, since SRCNAME is scoped by CNAME. Separate header extension identities are defined for SRCNAME and CNAME. This is motivated by the fact that a single RTP stream can have several SRCNAME, but only a single CNAME.

The RTP header extensions for CNAME and SRCNAME MAY use either one of the one-byte or two-byte header formats, depending on the CNAME and SRCNAME value size. The one-byte header SHOULD be used when the value contains at most 16 bytes. Note that the RTP header extension specification does not allow to mix one-byte and two-byte headers for the same stream, so if the value size of either SRCNAME or CNAME requires the two-byte header, the other MUST also use that header format.

The header extension payload for SRCNAME contains the srcname-content, as defined in Section 6. The header extension payload for CNAME contains the CNAME value as defined in [RFC3550]. Figures Figure 4 and Figure 5 show samples of the structure of the header extension payload for the two header formats.

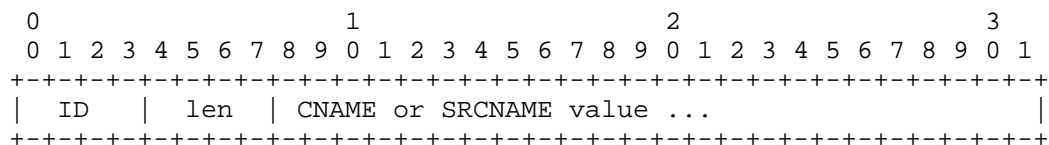


Figure 4

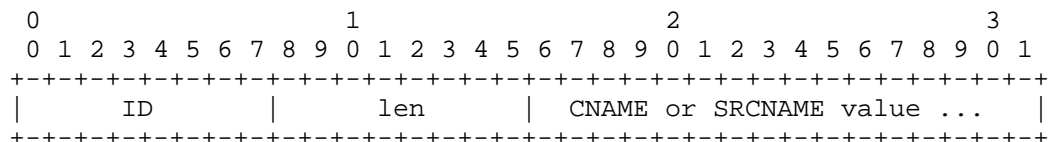


Figure 5

The URN identifiers to use with "a=extmap" SDP signaling for SRCNAME and CNAME, respectively, MUST be

```
urn:ietf:params:rtp-hdrext:srcname
urn:ietf:params:rtp-hdrext:cname
```

## 10. Examples

This section shows SDP examples of declaring the SRCNAME in SDP.

### 10.1. Simulcast

In this use case the end-point is a client with a single audio source and two video sources, and it uses simulcast for sending different encodings of the same video source. This example is based on Using Simulcast in RTP sessions [I-D.westerlund-avtcore-rtp-simulcast]. The following SDP describes this.

```
v=0
o=alice 3203093520 3203093520 IN IP4 foo.example.com
s=Simulcast enabled client
t=0 0
c=IN IP4 foo.example.com
m=audio 49200 RTP/AVP 96
a=rtpmap:96 G719/48000/2
a=ssrc:521923924 cname:alice@foo.example.com
a=ssrc:521923924 srcname:al
a=mid:1
m=video 49300 RTP/AVP 96
a=rtpmap:96 H264/90000
a=fmtp:96 profile-level-id=42c01e
a=imageattr:96 send [x=640,y=360] recv [x=640,y=360] [x=320,y=180]
a=ssrc:192392452 cname:alice@foo.example.com
a=ssrc:192392452 srcname:v1
a=ssrc:834753488 cname:alice@foo.example.com
a=ssrc:834753488 srcname:v2
a=mid:2
a=content:main
m=video 49400 RTP/AVP 97
a=rtpmap:97 H264/90000
a=fmtp:97 profile-level-id=42c00d
a=imageattr:97 send [x=320,y=180]
a=ssrc:239245219 cname:alice@foo.example.com
a=ssrc:239245219 srcname:v1
a=ssrc:734623563 cname:alice@foo.example.com
a=ssrc:734623563 srcname:v2
a=mid:3
a=sendonly
```

The audio session is proposing to use one stereo stream of G.719 and the video sessions are proposing to send two different encodings of each video source, one with the resolution 640x360 and one with 320x180. The end-point also declares the SSRCS it intends to use with bindings to CNAME and SRCNAME, enabling the receiver of the SDP to bind together the video streams that originate from the same video camera. For example, the two streams having an SRCNAME of "v1" originate from the same video camera and belong together.

The use of the srcname attribute in the SDP is optional and the information can be retrieved from RTCP reporting, but it will then not be possible to correctly relate the video sources until the first RTCP report is received.

## 10.2. SVC with multi-session transmission

Here an example is shown of a client that uses SVC with multi-session transmission as described in RTP Payload Format for Scalable Video Coding [RFC6190]. RTP Payload Format for Scalable Video Coding [RFC6190] only describes examples for a client with one video source and the decoder dependencies of the different sessions are grouped using the Session grouping DDP attribute as defined in Signaling Media Decoding Dependency in the Session Description Protocol (SDP) [RFC5583] and implicitly CNAME.

However, if a client has two video sources and wishes to use multi-session transmission and send streams from both sources in each session, an additional grouping mechanism is needed to group the different streams in the different sessions. SRCNAME is suitable for this and here we show an example where the DDP attribute groups the different sessions and the SRCNAME is used to relate the different SSRCs in each RTP session to one of the two video sources.

```
v=0
o=bob 8473948250 8473948250 IN IP4 foo.example.com
s=SVC MST client
t=0 0
c=IN IP4 foo.example.com
a=group:DDP L1 L2 L3
m=audio 49500 RTP/AVP 96
a=rtpmap:96 G719/48000/2
a=ssrc:293848928 cname:bob@foo.example.com
a=mid:A1
m=video 20000 RTP/AVP 96
a=rtpmap:96 H264/90000
a=fmtp:96 profile-level-id=4de00a; packetization-mode=1;
  mst-mode=NI-TC; sprop-parameter-sets={sps0},{pps0};
a=ssrc:743947584 cname:bob@foo.example.com
a=ssrc:743947584 srcname:V1.L1
a=ssrc:283894947 cname:bob@foo.example.com
a=ssrc:283894947 srcname:V2.L1
a=mid:L1
m=video 20002 RTP/AVP 97
a=rtpmap:97 H264-SVC/90000
a=fmtp:97 profile-level-id=53000c; packetization-mode=1;
  mst-mode=NI-T; sprop-parameter-sets={sps1},{pps1};
a=ssrc:492784823 cname:bob@foo.example.com
a=ssrc:492784823 srcname:V1.L2
a=ssrc:892362397 cname:bob@foo.example.com
a=ssrc:892362397 srcname:V2.L2
a=mid:L2
a=depend:97 lay L1:96
m=video 20004 RTP/AVP 98
a=rtpmap:98 H264-SVC/90000
a=fmtp:98 profile-level-id=53001F; packetization-mode=1;
  mst-mode=NI-T; sprop-parameter-sets={sps2},{pps2};
a=ssrc:184562894 cname:bob@foo.example.com
a=ssrc:184562894 srcname:V1.L3
a=ssrc:305605682 cname:bob@foo.example.com
a=ssrc:305605682 srcname:V2.L3
a=mid:L3
a=depend:98 lay L1:96 L2:97
```

Thus, the client declares that it will send two video streams in each RTP session and the receiver is then able to relate the streams in the different sessions by using the SRCNAME binding, with matching (first parts of the) SRCNAME value. Without the SRCNAME binding it would not be possible for the receiver to know which streams belong to the same source. Note that the audio stream does not have an explicit srcname attribute in this example, but only relate to the

video streams through the same CNAME. Note that the last part of the SRCNAMEs in the example, ".L1", ".L2" and ".L3" are not necessary but allowed and will not impact the ability to tell that the streams belong together, since related streams have the first part in common.

### 10.3. Retransmission

This use case shows how SRCNAME can be used to connect retransmission streams to the original streams in the case of SSRC multiplexed RTP retransmission [RFC4588]. This is included to exemplify how RTP retransmission could be updated to provide explicit bindings between the source and the repair stream, but just an example and not a specification.

```
v=0
o=carol 3462534872 3462534872 IN IP4 foo.example.com
s=SSRC-multiplexed retransmission client
t=0 0
c=IN IP4 foo.example.com
m=audio 49800 RTP/AVP 96
a=rtpmap:96 G719/48000/2
a=ssrc:8372496978 cname:carol@foo.example.com
a=mid:1
m=video 49300 RTP/AVP 96 97
a=rtpmap:96 H264/90000
a=rtcp-fb:96 nack
a=fmtp:96 profile-level-id=42c01e
a=rtpmap:97 rtx/90000
a=fmtp:97 apt=96;rtx-time=200
a=ssrc:192392452 cname:carol@foo.example.com
a=ssrc:192392452 srcname:v1.o
a=ssrc:834753488 cname:carol@foo.example.com
a=ssrc:834753488 srcname:v1.r
a=ssrc:682394013 cname:carol@foo.example.com
a=ssrc:682394013 srcname:v2.o
a=ssrc:284576129 cname:carol@foo.example.com
a=ssrc:284576129 srcname:v2.r
a=mid:2
```

The client proposes to send two original video streams in the video session and a retransmission stream for each one of them. The retransmission streams are associated with the respective original stream by using matching SRCNAME and a receiver would then know which original stream a certain retransmission stream is associated with. This solves the ambiguity problem when SSRC-multiplexing is used for retransmission and it enables SSRC-multiplexing of original and retransmission streams to be used also in multicast sessions. Note that ".o" and ".r" parts of SRCNAME are not needed, but may improve

understanding of the example and will not affect the ability to match related streams.

#### 10.4. Forward Error Correction

Forward Error Correction Grouping Semantics in the Session Description Protocol [RFC5956] defines two SDP attributes for grouping the associated source and FEC-based repair streams. One can be used for grouping different RTP sessions and the other can be used for grouping SSRCs in the same RTP session, i.e. when session-respective SSRC-multiplexing is used. However, it may be advantageous to SSRC-multiplex the source streams in one RTP session and the repair streams in another since that gives a receiver the possibility to reject the repair session in case it does not support the proposed FEC. In this case, the above mentioned grouping attributes cannot be used to associate the repair streams with the respective source stream since grouping of SSRCs cannot be made across RTP sessions. The following example shows how SRCNAME can be used for that.

```
v=0
o=dave 7352395826 7352395826 IN IP4 foo.example.com
s=FEC client
t=0 0
c=IN IP4 foo.example.com
a=group:FEC-FR 2 3
m=audio 49300 RTP/AVP 96
a=rtpmap:96 G719/48000/2
a=ssrc:237847298 cname:dave@foo.example.com
a=mid:1
m=video 49200 RTP/AVP 100
a=rtpmap:100 MP2T/90000
a=ssrc:847612849 cname:dave@foo.example.com
a=ssrc:847612849 srcname:v1.o
a=ssrc:558237845 cname:dave@foo.example.com
a=ssrc:558237845 srcname:v2.o
a=exthdr:1 urn:ietf:params:rtp-hdrext:cname
a=exthdr:4 urn:ietf:params:rtp-hdrext:srcname
a=mid:2
m=application 49300 RTP/AVP 101
a=rtpmap:101 ld-interleaved-parityfec/90000
a=fmtp:101 L=5; D=10; repair-window=200000
a=ssrc:389572053 cname:dave@foo.example.com
a=ssrc:389572053 srcname:v1.r
a=ssrc:185729479 cname:dave@foo.example.com
a=ssrc:185729479 srcname:v2.r
a=exthdr:2 urn:ietf:params:rtp-hdrext:cname
a=exthdr:5 urn:ietf:params:rtp-hdrext:srcname
```



a=mid:3

In this example the client proposes to send two video streams in one session and two repair streams in the other session. The repair streams are associated with the respective video stream by using a matching SRCNAME. When receiving either this SDP, the SDES SRCNAME packets, or the SRCNAME/CNAME RTP header extensions (which are also offered), a receiver can make the connection between the source streams and the repair streams. Even a client not receiving the SDP will be able to do the association, by SRCNAME in either SDES or RTP header extension, if it has established one RTP session for receiving source streams and another for receiving repair streams. Note that ".o" and ".r" parts of SRCNAME are not needed, but may improve understanding of the example and will not affect the ability to match related streams (since they match on the highest hierarchical level).

#### 11. Usage with the Offer/Answer Model

The SDP offer/answer procedures for a=ssrc are specified in Source-Specific Media Attributes in the Session Description Protocol (SDP) [RFC5576]. The SDP offer/answer procedures for a=exthdr are specified in A General Mechanism for RTP Header Extensions [RFC5285].

#### 12. Backward Compatibility

Clients not supporting SRCNAME will not have the possibility to bind different streams to a specific media source, since they will not understand the SRCNAME SDES item or the RTP header extension. However, sending SRCNAME SDES items to a client not supporting it should not impose any problems since all clients should be prepared that new SDES items may be specified according to RTP [RFC3550].

According to the definition of SDP attributes in SDP: Session Description Protocol [RFC4566], if an attribute is received that is not understood, it MUST be ignored by the receiver. So a receiver not supporting the ssrc attribute will simply ignore it.

Source-Specific Media Attributes in the Session Description Protocol (SDP) [RFC5576] defines rules of how new source attributes should be registered, which means that a receiver supporting RFC 5576 should be prepared that new source attributes may be defined. This means that a user supporting some of the source attributes should not have any problems when the user receives an SDP with unknown source attributes.

RTP header extension will only be used when successfully negotiated in SDP, which requires support in both sender and receiver.

### 13. IANA Considerations

Following the guidelines in SDP [RFC4566], in The Session Description Protocol (SDP) Grouping Framework [RFC5888], and in RTP [RFC3550], the IANA is requested to register:

1. A new SDES item named SRCNAME, as defined in Section 7. This item needs to be assigned an identifier TBA1.
2. A new SDP source attribute named srcname, as defined in Section 8.
3. New RTP header extension URN identifiers for SRCNAME and CNAME, as defined in Section 9.

### 14. Security Considerations

The SDES or header extension SRCNAMEs being close to opaque identifiers could potentially carry additional meanings or function as overt channel. If the SRCNAME would be permanent between sessions, they have the potential for compromising the users' privacy as they can be tracked between sessions. See Guidelines for Choosing RTP Control Protocol (RTCP) Canonical Names (CNAMEs) [RFC6222] for more discussion.

A third party modification of the srcname labels either in the RTCP SDES items, in the SDP a=ssrc attribute, or in the RTP header extension can cause service disruption. By modifying labels the wrong streams could be associated, with potentially serious effects including media disruptions. If streams that are to be associated aren't associated, then another type of failures occur. To prevent modification, insertion or deletion of the srcname labels, the carrying channel needs to be protected by integrity protection and source authentication. For RTCP and RTP header extension, various solutions exist, such as SRTP [RFC3711], DTLS [RFC6347], or IPsec [RFC4301]. For protecting the SDP, the signalling channel needs to provide protection. For SIP S/MIME [RFC3261] are the ideal, and hop by hop TLS [RFC5246] provides at least some protection, although not perfect. For SDPs retrieved using RTSP DESCRIBE [RFC2326], TLS would be the RECOMMENDED solution.

### 15. References

## 15.1. 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.
- [RFC3629] Yergeau, F., "UTF-8, a transformation format of ISO 10646", STD 63, RFC 3629, November 2003.
- [RFC5234] Crocker, D. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", STD 68, RFC 5234, January 2008.
- [RFC5576] Lennox, J., Ott, J., and T. Schierl, "Source-Specific Media Attributes in the Session Description Protocol (SDP)", RFC 5576, June 2009.
- [RFC6222] Begen, A., Perkins, C., and D. Wing, "Guidelines for Choosing RTP Control Protocol (RTCP) Canonical Names (CNAMES)", RFC 6222, April 2011.

## 15.2. Informative References

- [I-D.ietf-avtext-rtp-duplication]  
Begen, A. and C. Perkins, "Duplicating RTP Streams", draft-ietf-avtext-rtp-duplication-00 (work in progress), July 2012.
- [I-D.ietf-mmusic-duplication-grouping]  
Begen, A., Cai, Y., and H. Ou, "Duplication Grouping Semantics in the Session Description Protocol", draft-ietf-mmusic-duplication-grouping-00 (work in progress), October 2012.
- [I-D.westerlund-avtcore-rtp-simulcast]  
Westerlund, M., Burman, B., Lindqvist, M., and F. Jansson, "Using Simulcast in RTP sessions", draft-westerlund-avtcore-rtp-simulcast (work in progress), October 2011.
- [RFC2326] Schulzrinne, H., Rao, A., and R. Lanphier, "Real Time Streaming Protocol (RTSP)", RFC 2326, April 1998.
- [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.

- [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.
- [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.
- [RFC5117] Westerlund, M. and S. Wenger, "RTP Topologies", RFC 5117, January 2008.
- [RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, August 2008.
- [RFC5285] Singer, D. and H. Desineni, "A General Mechanism for RTP Header Extensions", RFC 5285, July 2008.
- [RFC5583] Schierl, T. and S. Wenger, "Signaling Media Decoding Dependency in the Session Description Protocol (SDP)", RFC 5583, July 2009.
- [RFC5888] Camarillo, G. and H. Schulzrinne, "The Session Description Protocol (SDP) Grouping Framework", RFC 5888, June 2010.
- [RFC5956] Begen, A., "Forward Error Correction Grouping Semantics in the Session Description Protocol", RFC 5956, September 2010.
- [RFC6051] Perkins, C. and T. Schierl, "Rapid Synchronisation of RTP Flows", RFC 6051, November 2010.
- [RFC6190] Wenger, S., Wang, Y., Schierl, T., and A. Eleftheriadis, "RTP Payload Format for Scalable Video Coding", RFC 6190, May 2011.
- [RFC6347] Rescorla, E. and N. Modadugu, "Datagram Transport Layer Security Version 1.2", RFC 6347, January 2012.

Authors' Addresses

Magnus Westerlund  
Ericsson  
Farogatan 6  
SE-164 80 Kista  
Sweden

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

Bo Burman  
Ericsson  
Farogatan 6  
SE-164 80 Kista  
Sweden

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

Patrik Sandgren  
Ericsson  
Farogatan 6  
SE-164 80 Kista  
Sweden

Phone: +46 10 717 97 41  
Email: patrik.sandgren@ericsson.com



Network Working Group  
Internet-Draft  
Intended status: Standards Track  
Expires: April 25, 2013

A. Akram  
B. Burman  
D. Grondal  
M. Westerlund  
Ericsson AB  
October 22, 2012

RTP Media Stream Pause and Resume  
draft-westerlund-avtext-rtp-stream-pause-03

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 adding a group of new real-time feedback messages used to pause and resume RTP data streams.

Status of this Memo

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

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

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

This Internet-Draft will expire on April 25, 2013.

Copyright Notice

Copyright (c) 2012 IETF Trust and the persons identified as the document authors. All rights reserved.

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

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

## Table of Contents

1. Introduction . . . . .	4
2. Definitions . . . . .	5
2.1. Abbreviations . . . . .	5
2.2. Terminology . . . . .	5
2.3. Requirements Language . . . . .	6
3. Use Cases . . . . .	6
3.1. Point to Point . . . . .	6
3.2. RTP Mixer to Media Sender . . . . .	7
3.3. Media Receiver to RTP Mixer . . . . .	8
3.4. Media Receiver to Media Sender Across RTP Mixer . . . . .	8
4. Design Considerations . . . . .	9
4.1. Real-time Nature . . . . .	9
4.2. Message Direction . . . . .	9
4.3. Apply to Individual Sources . . . . .	9
4.4. Consensus . . . . .	9
4.5. Acknowledgements . . . . .	10
4.6. Retransmitting Requests . . . . .	10
4.7. Sequence Numbering . . . . .	10
5. Relation to Other Solutions . . . . .	11
5.1. Signaling Technology Performance Comparison . . . . .	11
5.2. SDP "inactive" Attribute . . . . .	19
5.3. CCM TMMBR / TMMBN . . . . .	19
5.4. Media Stream Selection . . . . .	20
5.5. Media Source Selection in SDP . . . . .	21
5.6. Codec Operation Point . . . . .	21
5.7. Conclusion . . . . .	22
6. Solution Overview . . . . .	22
6.1. Expressing Capability . . . . .	23
6.2. Requesting to Pause . . . . .	23
6.3. Media Sender Pausing . . . . .	24
6.4. Requesting to Resume . . . . .	25
7. Participant States . . . . .	26
7.1. Playing State . . . . .	27
7.2. Pausing State . . . . .	27
7.3. Paused State . . . . .	28
7.3.1. RTCP BYE Message . . . . .	28
7.3.2. SSRC Time-out . . . . .	28
7.4. Local Paused State . . . . .	29



8. Message Format . . . . .	29
9. Message Details . . . . .	31
9.1. PAUSE . . . . .	31
9.2. PAUSED . . . . .	32
9.3. RESUME . . . . .	33
9.4. REFUSE . . . . .	34
9.5. Transmission Rules . . . . .	34
10. Signalling . . . . .	35
10.1. Offer-Answer Use . . . . .	37
10.2. Declarative Use . . . . .	38
11. Examples . . . . .	39
11.1. Offer-Answer . . . . .	39
11.2. Point-to-Point Session . . . . .	40
11.3. Point-to-multipoint using Mixer . . . . .	43
11.4. Point-to-multipoint using Translator . . . . .	45
12. IANA Considerations . . . . .	48
13. Security Considerations . . . . .	49
14. Acknowledgements . . . . .	49
15. References . . . . .	49
15.1. Normative References . . . . .	49
15.2. Informative References . . . . .	50
Authors' Addresses . . . . .	52

## 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 and demands control over 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, which either uses logic based on voice activity, other measurements, user input over proprietary interfaces, or Media Stream Selection [I-D.westerlund-dispatch-stream-selection] 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 supports pausing and resuming an incoming data 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 should thus be defined as a Codec Control Message (CCM) extension.

## 2. Definitions

### 2.1. Abbreviations

RTP: Real-time Transport Protocol

RTCP: Real-time Transport Control Protocol

SSRC: Synchronization Source

CSRC: Contributing Source

FB: Feedback

AVPF: Audio-Visual Profile with Feedback

FMT: Feedback Message Type

PT: Payload Type

CCM: Codec Control Messages

MCU: Multipoint Control Unit

### 2.2. Terminology

In addition to following, the definitions from RTP [RFC3550], AVPF [RFC4585] and CCM [RFC5104] also apply in this document.

**Feedback Messages:** CCM [RFC5104] categorised 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.

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

**Sender:** The RTP entity that sends an RTP data stream.

**Receiver:** The RTP entity that receives an RTP data stream.

**Mixer:** The intermediate RTP node which receives a data stream from different nodes, combines them to make one stream and forwards to destinations, in the sense described in Topo-Mixer of RTP Topologies [RFC5117].

**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).

### 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 has one or more SSRCs.



Figure 1: Point to Point

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

RTCWEB WG's use case and requirements document [I-D.ietf-rtcweb-use-cases-and-requirements] defines the following

API requirement derived in the most basic usage "Simple Video Communication Service" (Section 4.2.1 in [I-D.ietf-rtcweb-use-cases-and-requirements]) :

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.

The PAUSED indication in this document can be used to indicate to the media receiver that the stream delivery is deliberately paused due to a sender side mute operation.

### 3.2. RTP Mixer to Media Sender

One of the most commonly used topologies in centralized conferencing is based on the RTP Mixer. 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 having its' own SSRCs 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 media stream the Mixer provides can be an actual media mixing of multiple media sources, but it might also be as simple as selecting one of the underlying sources based on some Mixer policy or control signalling.

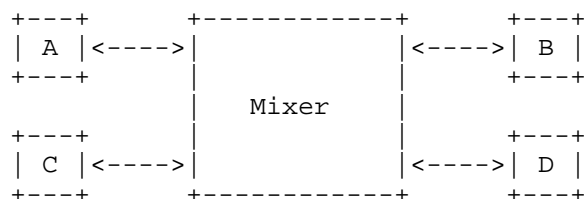


Figure 2: RTP Mixer

The media streams being 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 senders 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 media selection could also be under the user's control using a protocol like Media Stream Selection [I-D.westerlund-dispatch-stream-selection]. The media

streams may also be simulcasted or scalable encoded (for Multi-Stream Transmission), thus providing multiple versions that can be delivered by the media sender. These examples indicate that there are numerous reasons why a particular media 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 media 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 media stream from being delivered to it. It also needs to be able to resume delivery with minimal delay.

### 3.3. 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.

### 3.4. 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.

First of all, the Mixer might not include CSRC in its stream indications. Secondly, 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. Thirdly, the end-point cannot determine if a given media stream is still needed by the RTP Mixer to deliver to another session participant.

In addition, pause is only part of the semantics when it comes to selecting media streams. As can be seen in MESS [I-D.westerlund-dispatch-stream-selection], it can be beneficial to have both include and exclude semantics. In addition, substitution and possibility to control in what local RTP media stream the selected remote RTP media stream is to be provided gives richer functionality.

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. Indication that an RTP media stream is paused is the responsibility of the RTP media stream sender.

##### 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 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, 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. This error condition, a negative acknowledgement, is 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, it 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

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 [3GPP.36.201], 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 [3GPP.25.308] (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.

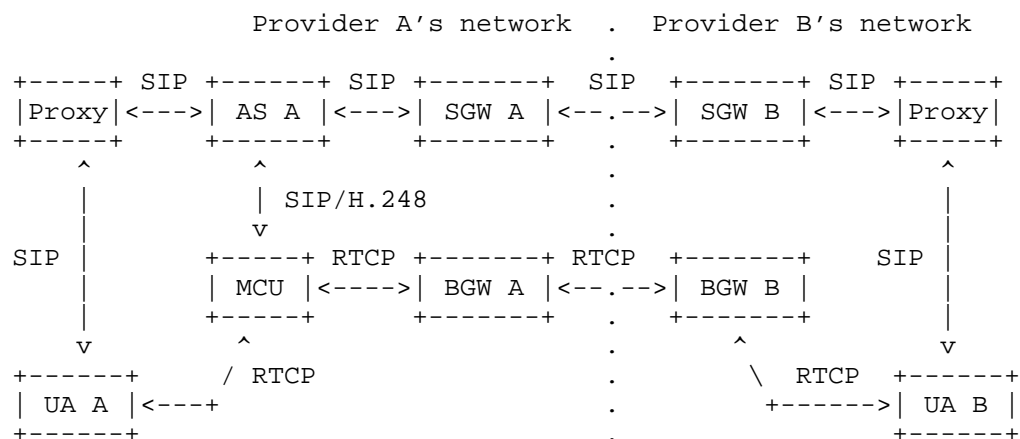


Figure 3: Comparison Signaling Topology

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 [3GPP.26.114], 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.

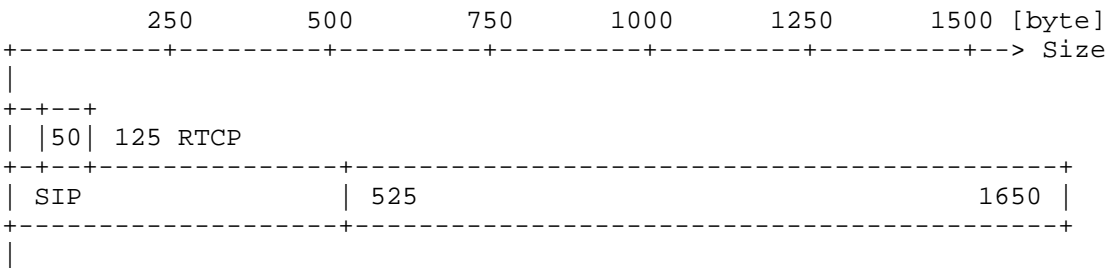


Figure 4: Message Size Comparison

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.

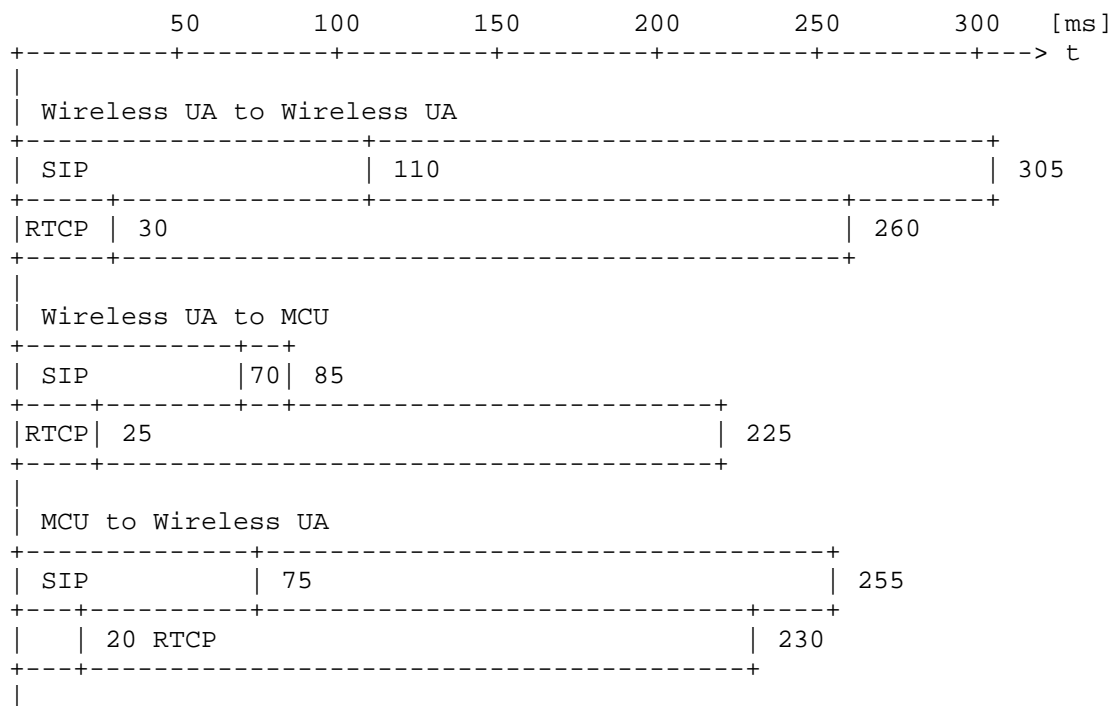


Figure 5: Mobile Access Transmission Delay Comparison

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 3.3. 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.

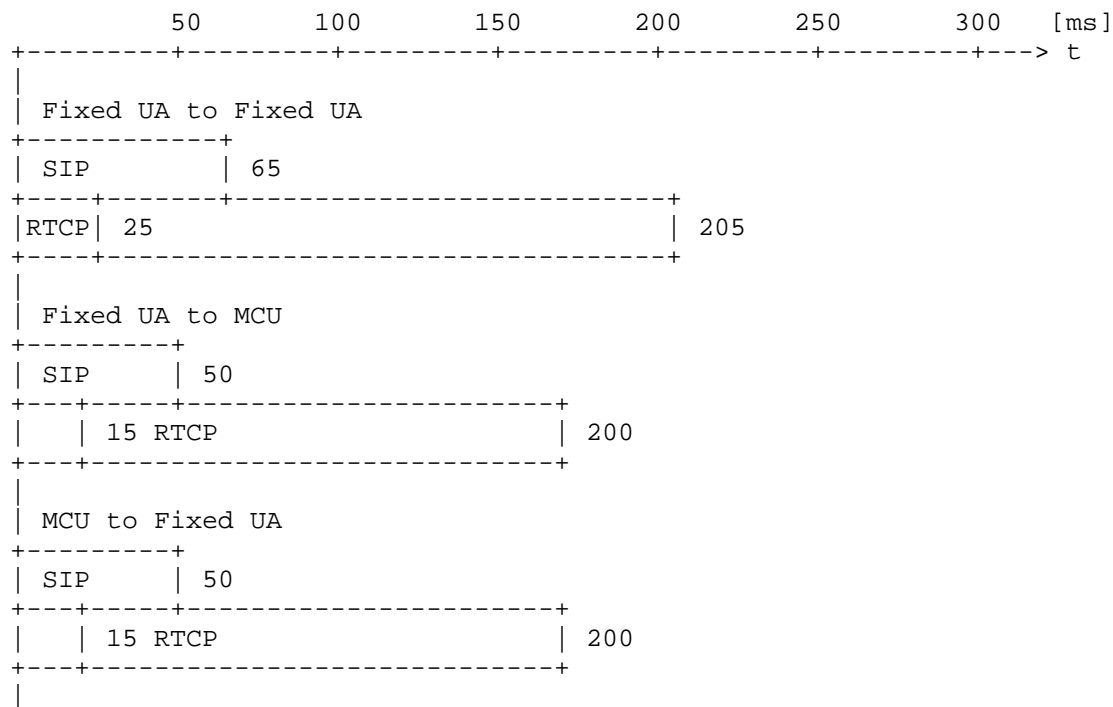


Figure 6: Fixed Access Transmission Delay Comparison

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 reconsideration. 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 [RFC3356]. 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. 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.

## 5.3. 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 seemingly provide some of the necessary functionality, using a bitrate value of 0 as PAUSE request and PAUSED indication, respectively. 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).

The defined semantics for TMMBR differ significantly from what is required for PAUSE. 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. The TMMBR semantics are the opposite; the first media receiver that sends TMMBR 0 will pause the stream for all receivers. This does not matter in the point-to-point case since there is only a single receiver, but will not provide the desired functionality in other cases.

A possibly even more significant aspect is the guard period present in TMMBR when increasing the bandwidth. The equivalent of a RESUME request using TMMBR would be to send a TMMBR message with a non-zero value, likely at the level of maximum for the session. However, 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. This applies independently of topology, i.e. also for a point to point session. Compared to the proposed solution, this adds several delays. First, there is the delay between the media sender receiving the TMMBR until it can send a TMMBN, then there is the above delay for the guard period before the media sender resumes transmission. This delay before resuming transmission is the most time critical operation in this solution, making use of TMMBR according to the defined semantics infeasible in practice.

A TMMBN message could arguably be used as an acknowledgement (Section 4.5) of either PAUSE or RESUME (depending on zero or non-zero bitrate parameter), but will not be able to provide any sequence number functionality (Section 4.7) and will thus risk mis-interpretation due to race situations.

#### 5.4. Media Stream Selection

The Media Stream Selection draft [I-D.westerlund-dispatch-stream-selection] includes a functionality that allows to include and exclude a specified stream from a received set of streams, which arguably gives similar results as pausing a stream. The functionality described in that specification is mainly transport agnostic, but the proposed implementation is to extend BFCP [RFC4582], which would likely give a performance somewhere in between RTCP and SIP.

The semantics differ between exclude / include and pause / resume for a stream in topologies other than point-to-point. For example, in RTP Receiver to Mixer (Section 3.3), 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 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 stream (CSRC) to be included in the mix again.

It would in fact be possible to use media stream selection for SSRC from the mixer itself, not CSRC, to achieve pause and resume functionality when UA and mixer are connected point-to-point, but that would not work with Translator or in multipoint, and neither would it provide any RTP level indication that the stream is paused.

In the Mixer to RTP sender (Section 3.2) case, the topology is sufficiently similar to point-to-point to make it possible to use Media Stream Selection for pause and resume functionality, but would still not provide any RTP level indication that the stream is paused.

In topologies where the same RTP media stream is received by several receivers, Media Stream Selection does not provide any functionality to achieve consensus (Section 4.4) and will need modification to be possible to use.

#### 5.5. Media Source Selection in SDP

There is also a similar 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 5.2). This suffers partly from the same mis-match in semantics and lack of functionality for consensus as the section above (Section 5.4), and would likely also suffer from lower real-time performance (Section 4.1), especially when implementing necessary signaling to reach consensus (Section 4.4).

#### 5.6. Codec Operation Point

The draft on Codec Operation Point (COP) [I-D.westerlund-avtext-codec-operation-point] includes functionality to request a stream to be encoded at a certain bandwidth, including 0. That could also be used to pause and resume a stream. Since that draft is also based on CCM [RFC5104], the performance should be very similar to this specification and it should be possible to achieve sufficiently low signaling delay (Section 4.1).

The message semantics of COP (Section 4.4) also suits the purpose of this specification, and it would be possible to use COP for the sole purpose of controlling bandwidth in a way that effectively constitutes pause and resume. COP also has a functionality that this specification does not, in that it can set the bandwidth individually for sub-streams within a single SSRC, for example in scalable Single Stream Transmission (SST).

The authors however believe that there will be applications that can make good use of pause and resume functionality, but that will not be able to motivate a COP implementation, not even just the 'bitrate' Parameter Type (support for each Parameter Type is individually negotiable in COP). It is thus believed that pause and resume functionality is motivated as a separate specification.

## 5.7. 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 this 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.

## 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. A single Feedback message specification is used. 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).

### 6.2. Requesting to Pause

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

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.

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 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) for the stream. The same condition can be caused by the RTP media stream sender having received a disapproving RESUME for the PAUSE request, but that the

PAUSE sender did not receive the RESUME and may instead think that the PAUSE was lost. In both cases, the PAUSE request can be re-transmitted using the same PauseID.

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. 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 at any time, subject to AVPF timing rules.

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.

## 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.



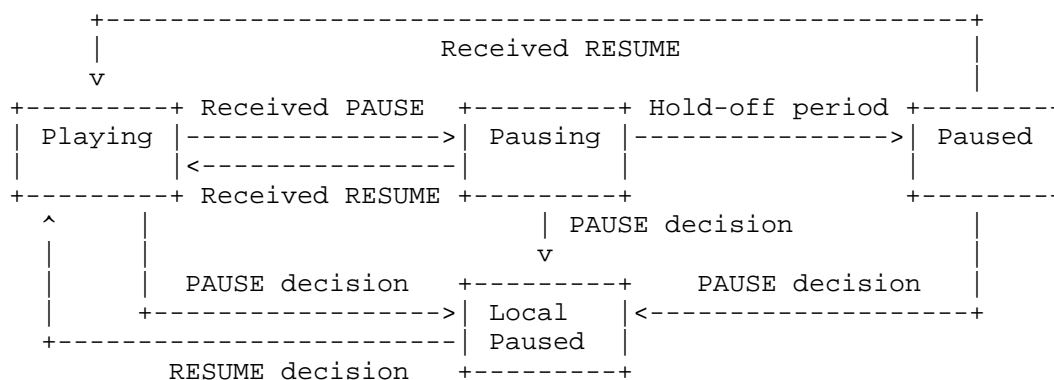


Figure 7: RTP Pause States

### 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:

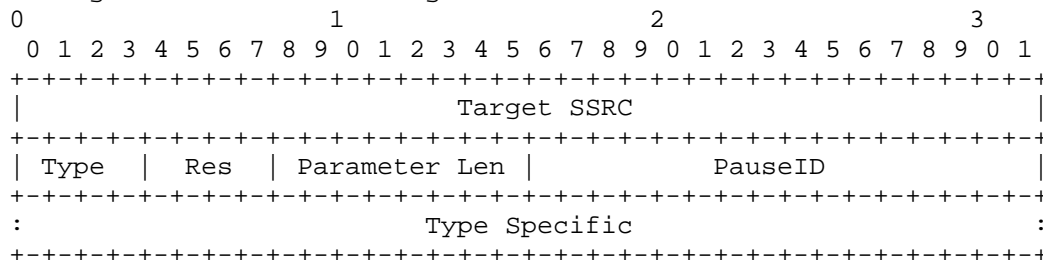


Figure 8: Syntax of FCI Entry in the PAUSE and RESUME message

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.

### 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.

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 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.

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 9: 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 10: 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 11: 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.

## 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.

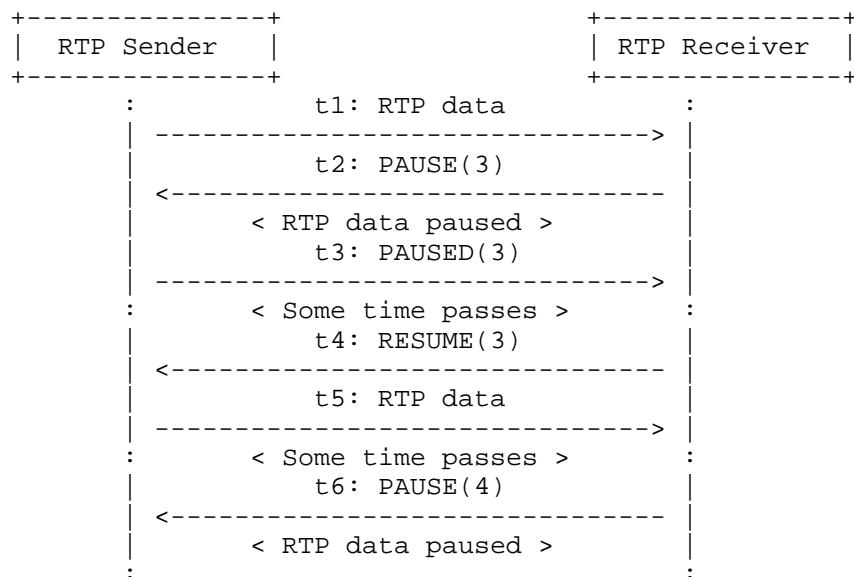


Figure 12: Pause and Resume Operation in Point-to-Point

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

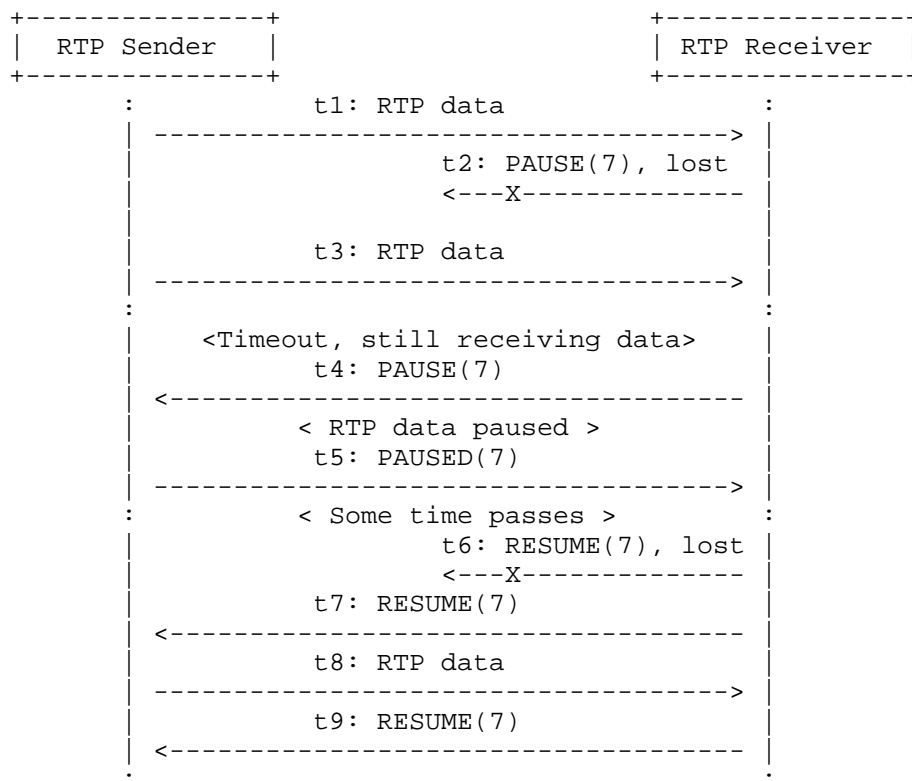


Figure 13: Pause and Resume Operation With Messages Lost

Figure 13 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.

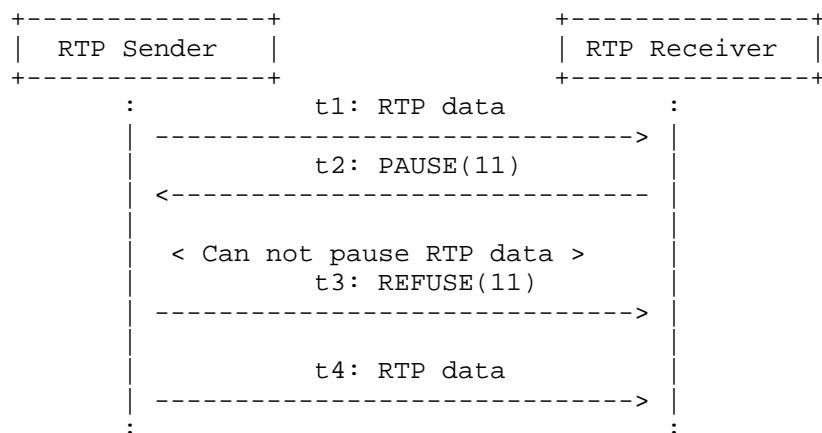


Figure 14: Pause Request is Refused in Point-to-Point

In Figure 14, 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.

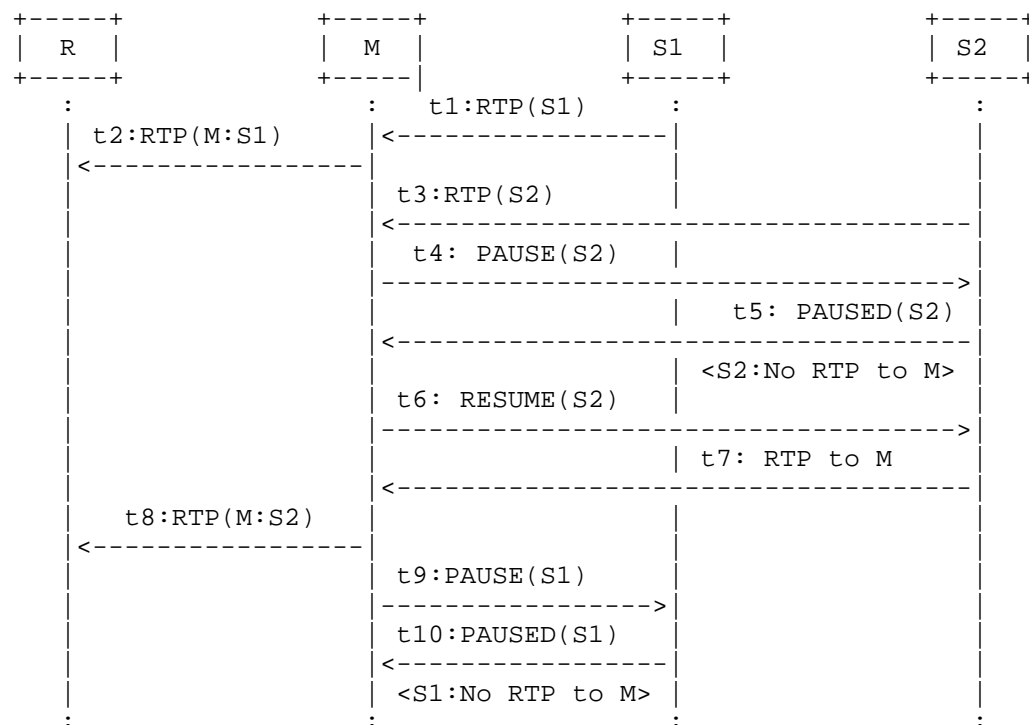


Figure 15: Pause and Resume Operation for a Voice Activated Mixer

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.

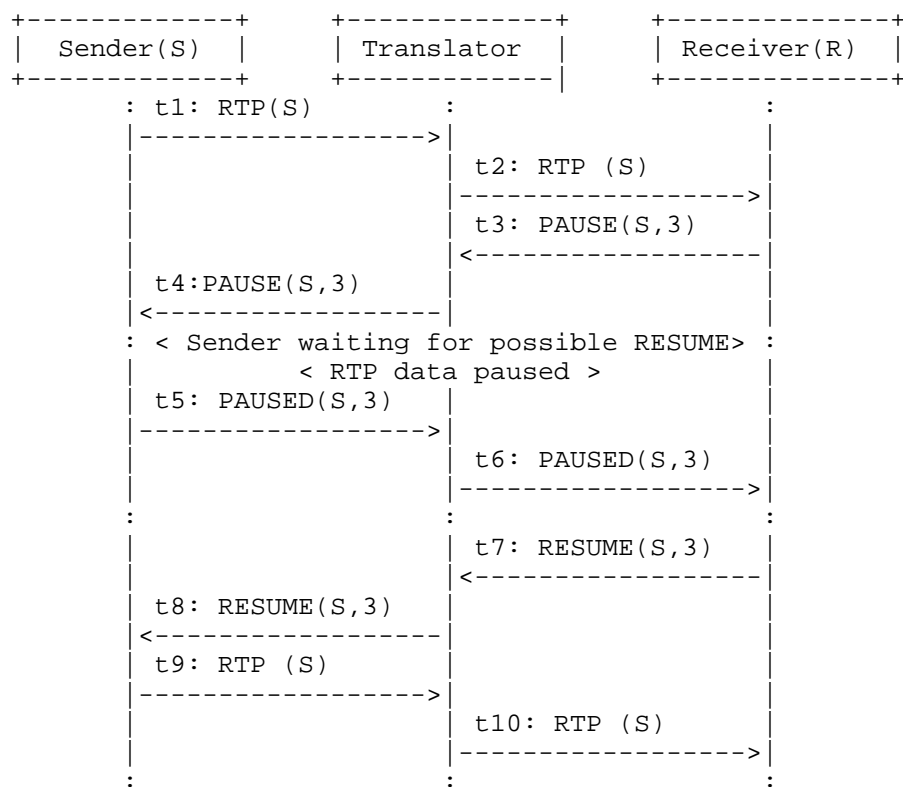


Figure 16: Pause and Resume Operation Between Two Participants Using a Translator

Figure 16 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.

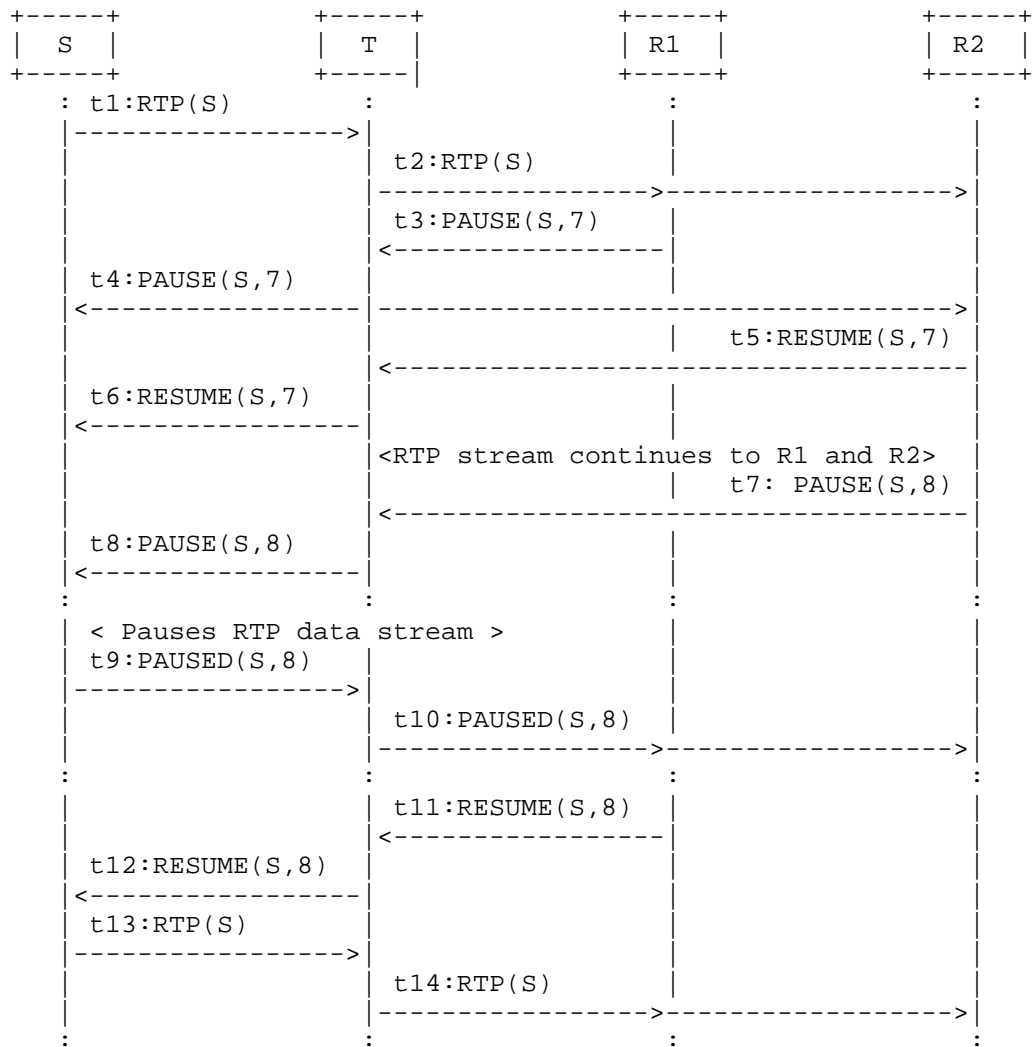


Figure 17: Pause and Resume Operation Between One Sender and Two Receivers Through Translator

Figure 17 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 the source streams which MAY result in Denial of Service (DoS). An Authentication protocol MAY save from this attack.
3. Man-in-Middle Attack (MiMT) - The pausing and resuming of the RTP source is prone to a Man-in-Middle attack. The public key authentication May be used to prevent MiMT.

### 14. Acknowledgements

### 15. References

#### 15.1. 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.

## 15.2. Informative References

- [3GPP.25.308]  
3GPP, "High Speed Downlink Packet Access (HSDPA); Overall description; Stage 2", 3GPP TS 25.308 10.6.0, December 2011.
- [3GPP.26.114]  
3GPP, "IP Multimedia Subsystem (IMS); Multimedia telephony; Media handling and interaction", 3GPP TS 26.114 10.4.0, June 2012.
- [3GPP.36.201]  
3GPP, "Evolved Universal Terrestrial Radio Access (E-UTRA); LTE physical layer; General description", 3GPP TS 36.201 10.0.0, December 2010.
- [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-09 (work in progress), June 2012.
- [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-04 (work in progress), March 2012.
- [I-D.westerlund-avtcore-rtp-simulcast]  
Westerlund, M., Burman, B., Lindqvist, M., and F. Jansson, "Using Simulcast in RTP sessions", draft-westerlund-avtcore-rtp-simulcast (work in progress), October 2011.
- [I-D.westerlund-avtext-codec-operation-point]  
Westerlund, M., Burman, B., and L. Hamm, "Codec Operation Point RTCP Extension", draft-westerlund-avtext-codec-operation-point-00 (work in progress), March 2012.
- [I-D.westerlund-dispatch-stream-selection]  
Grondal, D., Westerlund, M., and B. Burman, "Media Stream



- Selection (MESS)",  
draft-westerlund-dispatch-stream-selection-00 (work in progress), October 2011.
- [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.
- [RFC3356] Fishman, G. and S. Bradner, "Internet Engineering Task Force and International Telecommunication Union - Telecommunications Standardization Sector Collaboration Guidelines", RFC 3356, August 2002.
- [RFC4566] Handley, M., Jacobson, V., and C. Perkins, "SDP: Session Description Protocol", RFC 4566, July 2006.
- [RFC4582] Camarillo, G., Ott, J., and K. Drage, "The Binary Floor Control Protocol (BFCP)", RFC 4582, November 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.
- [RFC5117] Westerlund, M. and S. Wenger, "RTP Topologies", RFC 5117, January 2008.
- [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.

Authors' Addresses

Azam Akram  
Ericsson AB  
Farogatan 6  
SE - 164 80 Kista,  
Sweden

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

Bo Burman  
Ericsson AB  
Farogatan 6  
SE - 164 80 Kista,  
Sweden

Phone: +46107141311  
Fax: +46107175550  
Email: bo.burman@ericsson.com  
URI: www.ericsson.com

Daniel Grondal  
Ericsson AB  
Farogatan 6  
SE - 164 80 Kista,  
Sweden

Phone: +46107147505  
Fax: +46107175550  
Email: daniel.grondal@ericsson.com  
URI: www.ericsson.com

Magnus Westerlund  
Ericsson AB  
Farogatan 6  
SE- Kista 164 80,  
Sweden

Phone: +46107148287  
Fax:  
Email: [magnus.westerlund@ericsson.com](mailto:magnus.westerlund@ericsson.com)  
URI: [www.ericsson.com](http://www.ericsson.com)

