

Network Working Group
Internet-Draft
Updates: 3550 (if approved)
Intended status: Standards Track
Expires: March 19, 2014

M. Petit-Huguenin
Impedance Mismatch
G. Zorn, Ed.
Network Zen
September 15, 2013

Support for Multiple Clock Rates in an RTP Session
draft-ietf-avtext-multiple-clock-rates-10

Abstract

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

Status of This Memo

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

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

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

This Internet-Draft will expire on March 19, 2014.

Copyright Notice

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

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

Table of Contents

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

1. Introduction

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

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

This creates three problems:

- o The method used to calculate the RTP timestamp field in an RTP packet is underspecified.
- o When the same SSRC is used for different clock rates, it is difficult to know what clock rate was used for the RTP timestamp field in an RTCP 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 TC	[RFC5484]
Jitter	RSI Jitter Block	[RFC5760]
Median jitter	RSI Stats Block	[RFC5760]

Table 1

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

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

2. Terminology

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

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

3. Legacy RTP

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

3.1. Different SSRC

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

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

$$\begin{aligned} \text{timestamp} = & \text{previous_timestamp} \\ & + (\text{current_capture_time} - \text{previous_capture_time}) \\ & * \text{current_clock_rate} \end{aligned}$$

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.

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

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

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

3.2.2. Non-monotonic timestamps

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

```
timestamp = capture_time * clock_rate
```

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

The advantage with this method is that it works with the jitter calculation described in RFC 3550, as long as the correct clock rates are used. It seems that this is what most implementations are using.

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

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

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

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

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

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

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

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

4.3. RTP Receiver

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

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

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

5. Security Considerations

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, Jonathan Lennox and Magnus Westerlund for comments, suggestions and questions that helped to improve this document.

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

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

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

8. References

8.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

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

8.2. Informative References

[I-D.ietf-avt-variable-rate-audio]
Wenger, S. and C. Perkins, "RTP Timestamp Frequency for Variable Rate Audio Codecs", draft-ietf-avt-variable-rate-audio-00 (work in progress), October 2004.

[RFC2629] Rose, M.T., "Writing I-Ds and RFCs using XML", RFC 2629, June 1999.

[RFC3551] Schulzrinne, H. and S. Casner, "RTP Profile for Audio and Video Conferences with Minimal Control", STD 65, RFC 3551, July 2003.

[RFC3556] Casner, S., "Session Description Protocol (SDP) Bandwidth Modifiers for RTP Control Protocol (RTCP) Bandwidth", RFC 3556, July 2003.

[RFC3611] Friedman, T., Caceres, R., and A. Clark, "RTP Control Protocol Extended Reports (RTCP XR)", RFC 3611, November 2003.

[RFC5450] Singer, D. and H. Desineni, "Transmission Time Offsets in RTP Streams", RFC 5450, March 2009.

[RFC5484] Singer, D., "Associating Time-Codes with RTP Streams", RFC 5484, March 2009.

[RFC5760] Ott, J., Chesterfield, J., and E. Schooler, "RTP Control Protocol (RTCP) Extensions for Single-Source Multicast Sessions with Unicast Feedback", RFC 5760, February 2010.

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

Appendix A. Example Values

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

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

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

Table 2: Monotonic Timestamps

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

Table 3: Non-monotonic Timestamps

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

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

Appendix B. Using a Fixed Clock Rate

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

Appendix C. Behavior of Legacy Implementations

C.1. libccrtp 2.0.2

This library uses the formula described in Section 3.2.2.

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

C.2. libmediastreamer0 2.6.0

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

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

C.3. libpjmedia 1.0

This library uses the formula described in Section 3.2.2.

C.4. Android RTP stack 4.0.3

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

Authors' Addresses

Marc Petit-Huguenin
Impedance Mismatch

Email: petithug@acm.org

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

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

AVTEXT
Internet-Draft
Intended status: Standards Track
Expires: April 05, 2014

A. Begen
Cisco
C. Perkins
University of Glasgow
October 02, 2013

Duplicating RTP Streams
draft-ietf-avtext-rtp-duplication-04

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 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 April 05, 2014.

Copyright Notice

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

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

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

Table of Contents

1. Introduction	2
2. Terminology and Requirements Notation	3
3. Dual Streaming Use Cases	3
3.1. Temporal Redundancy	3
3.2. Spatial Redundancy	4
3.3. Dual Streaming over a Single Path or Multiple Paths	4
3.4. Requirements	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	8
5.1. RTCP Considerations	8
5.2. Signaling Considerations	8
6. Use of RTP and RTCP with Temporal and Spatial Redundancy . .	9
7. Congestion Control Considerations	9
8. Security Considerations	10
9. IANA Considerations	11
10. Acknowledgments	11
11. References	11
11.1. Normative References	11
11.2. Informative References	11
Authors' Addresses	12

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 discusses the most common use cases and explains how duplication can be achieved for RTP streams in such use cases to address the immediate market needs. In the future, if there will be a different use case, which is not covered by this document, a new specification that explains how RTP duplication should be done in such a scenario may be needed.

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 processing overhead. It is also very predictable in its overheads. Alternative approaches, for example, retransmission-based recovery [RFC4588] or Forward Error Correction [RFC6363], may be suitable in some other cases.

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 (the original plus its duplicate) 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 and the little additional protection it provides.

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 sender 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 stream and its duplicate stream can be sent to the same IP destination address and UDP destination port with a certain delay between them [I-D.ietf-mmusic-delayed-duplication]. 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 commonly called stream merging or de-duplication.

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 its

duplicate stream, and with a delay, so that if there is a packet loss, such a delay would allow the same packet from the duplicate 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.ietf-mmusic-delayed-duplication] [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 (This is often made sure through careful network design). 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].

3.4. Requirements

One of the following conditions is REQUIRED to hold in applications using this specification:

- o The original and duplicate RTP streams are carried (with their own SSRCs) in the same "m" line (There could be other RTP streams listed in the same "m" line)
- o The original and duplicate RTP streams are carried in separate "m" lines and there is no other RTP stream listed in either "m" line.

When the original and duplicate RTP streams are carried in separate "m" lines in an SDP description and if the SDP description has one or more other RTP streams listed in either "m" line, duplication grouping is not trivial and further signaling will be needed, which is left for future standardization.

4. Use of RTP and RTCP with Temporal Redundancy

To achieve temporal redundancy, the main and duplicate RTP streams SHOULD be sent using the same 5-tuple of transport protocol, source and destination IP addresses, and source and destination transport ports. Due to the possible presence of network address and port translation (NAPT) devices, load balancers, or other middleboxes, use of anything other than an identical 5-tuple might also cause spatial redundancy (which might introduce an additional delay due to the delta between the path delays), and so is NOT RECOMMENDED unless the path is known to be free of such middleboxes.

Since the main and duplicate RTP streams follow an identical path, they are part of the same RTP session. Accordingly, the sender MUST choose a different SSRC for the duplicate 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 duplicate RTP stream. The RTCP for the duplicate RTP stream is generated exactly as-if the duplicate 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 both streams, so that the receiver can synchronize them.

The main and duplicate streams are conceptually synchronized using the standard RTCP Sender Report-based mechanism, deriving a mapping between their timelines. However, the RTP timestamps and sequence numbers MUST be identical in the main and duplicate streams, making the mapping quite trivial.

Both the main and duplicate RTP streams, and their corresponding RTCP reports, will be received. If RTCP is used, receivers MUST generate RTCP reports for both the main and duplicate 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 duplicate 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 signaling is needed to indicate which stream is the duplicate (the latter can be done in-band using an RTCP extension, or out-of-band in the SDP description).

We require out-of-band signaling for both features. The required SDP attribute to signal duplication in the SDP offer/answer exchange ('duplication-delay') is defined in [I-D.ietf-mmusic-delayed-duplication]. The required SDP grouping semantics are defined in [I-D.ietf-mmusic-duplication-grouping].

In the following SDP example, a video stream is duplicated, and the main and duplicate 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:ch1a@example.com
a=ssrc:1010 cname:ch1a@example.com
a=ssrc-group:DUP 1000 1010
a=duplication-delay:50
a=mid:Ch1
```

As specified in Section 3.2 of [I-D.ietf-mmusic-duplication-grouping], it is advisable that the SSRC listed first in the "a=ssrc-group:" line (i.e., SSRC of 1000) is sent first, with the other SSRC (i.e., SSRC of 1010) being the time-delayed duplicate. This is not critical, however, and a receiving host should size its playout buffer based on the '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 signaled.

5. Use of RTP and RTCP with Spatial Redundancy

When using spatial redundancy, the duplicate RTP stream is sent 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. Thus, the SSRCs used for the main and duplicate 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 duplicate streams. An "a=group:DUP" line or "a=ssrc-group:DUP" line 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 duplicate RTP stream. The RTCP for the duplicate RTP stream is generated exactly as-if the duplicate 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 both streams, so that the receiver can synchronize them.

The main and duplicate streams are conceptually synchronized using the standard RTCP Sender Report-based mechanism, deriving a mapping between their timelines. However, the RTP timestamps and sequence numbers MUST be identical in the main and duplicate streams, making the mapping quite trivial.

Both the main and duplicate RTP streams, and their corresponding RTCP reports, will be received. If RTCP is used, receivers MUST generate RTCP reports for both the main and duplicate 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.ietf-mmusic-duplication-grouping]. In the following example, the redundant streams have different IP destination addresses. The example shows the same UDP port number and IP source address for each stream, 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 Sla Slb
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=mid:Sla
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=mid:Slb
```

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

This uses the same RTP/RTCP mechanisms from Sections Section 4 and Section 5, plus a combination of both sets of signaling.

7. Congestion Control Considerations

Duplicating RTP streams has several considerations in the context of congestion control. First of all, RTP duplication MUST NOT be used in cases where the primary cause of packet loss is congestion since duplication can make congestion only worse. Furthermore, RTP duplication SHOULD NOT be used where there is a risk of congestion upon duplicating an RTP stream. Duplication is RECOMMENDED only to be used for protection against network outages due to a temporary link or network element failure and where it is known that there is sufficient network capacity to carry the duplicated traffic. The capacity requirement constrains the use of duplication to managed networks, and makes it unsuitable for use on unmanaged public networks.

It is essential that the nodes responsible for the duplication and de-duplication are aware of the original stream's requirements and the available capacity inside the network. If there is an adaptation capability for the original stream, these nodes have to assume the same adaptation capability for the duplicated stream, too. For example, if the source doubles the bitrate for the original stream, the bitrate of the duplicate stream will also be doubled.

Depending on where de-duplication takes place, there could be different scenarios. When the duplication and de-duplication takes place inside the network before the ultimate end-points that will consume the RTP media, the whole process is transparent to these end-points. Thus, these end-points will apply any congestion control, if applicable, on the de-duplicated RTP stream. This output stream will have less losses than either of the original and duplicated stream,

and the end-point will make congestion control decisions accordingly. However, if de-duplication takes place at the ultimate end-point, this end-point MUST consider the aggregate of the original and duplicated RTP stream in any congestion control it wants to apply. The end-point will observe the losses in each stream separately, and this information can be used to fine-tune the duplication process. For example, the duplication interval can be adjusted based on the duration of a common packet loss in both streams.

8. Security Considerations

The security considerations of [RFC3550], [I-D.ietf-mmusic-delayed-duplication], [I-D.ietf-mmusic-duplication-grouping], and any RTP profiles and payload formats in use apply.

Duplication can be performed end-to-end, with the media sender generating a duplicate RTP stream, and the receiver(s) performing de-duplication. In such cases, if the original media stream is to be authenticated (e.g., using SRTP [RFC3711]) then the duplicate stream also needs to be authenticated, and duplicate packets that fail the authentication check need to be discarded.

Stream duplication and de-duplication can also be performed by in-network middleboxes. Such middleboxes will need to rewrite the RTP SSRC such that the RTP packets in the duplicate stream have a different SSRC to the original stream, and will need to generate and respond to RTCP packets corresponding to the duplicate stream. This sort of in-network duplication service has the potential to act as an amplifier for denial-of-service attacks if the attacker can cause attack traffic to be duplicated. To prevent this, middleboxes providing the duplication service need to authenticate the traffic to be duplicated as being from a legitimate source, for example using the secure RTP (SRTP) profile [RFC3711]. This requires the middlebox to be part of the security context of the media session being duplicated, so it has access to the necessary keying material for authentication. To do this, the middlebox will need to be privy to the session set-up signalling. Details of how that is done will depend on the type of signalling used (SIP, RTSP, WebRTC, etc.), and is not specified here.

Similarly, to prevent packet injection attacks, a de-duplication middlebox needs to authenticate original and duplicate streams, and ought not use non-authenticated packets that are received. Again, this requires the middlebox to be part of the security context, and have access to the appropriate signalling and keying material.

The use of the encryption features of SRTP does not affect stream de-duplication middleboxes, since the RTP headers are sent in the clear.

9. IANA Considerations

No IANA actions are required.

10. Acknowledgments

Thanks to Magnus Westerlund for his suggestions.

11. References

11.1. 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.ietf-mmusic-delayed-duplication]
Begen, A., Cai, Y., and H. Ou, "Delayed Duplication Attribute in the Session Description Protocol", draft-ietf-mmusic-delayed-duplication-02 (work in progress), May 2013.
- [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-03 (work in progress), July 2013.
- [RFC3711] Baugher, M., McGrew, D., Naslund, M., Carrara, E., and K. Norrman, "The Secure Real-time Transport Protocol (SRTP)", RFC 3711, March 2004.

11.2. 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.

- [RFC6363] Watson, M., Begen, A., and V. Roca, "Forward Error Correction (FEC) Framework", RFC 6363, October 2011.
- [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

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

Email: csp@csperkins.org
URI: <http://orcid.org/0000-0002-3404-8964>

Network Working Group
Internet-Draft
Intended status: Informational
Expires: April 21, 2014

J. Lennox
Vidyo
K. Gross
AVA
S. Nandakumar
G. Salgueiro
Cisco Systems
B. Burman
Ericsson
October 18, 2013

A Taxonomy of Grouping Semantics and Mechanisms for Real-Time Transport
Protocol (RTP) Sources
draft-lennox-raiarea-rtp-grouping-taxonomy-03

Abstract

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

Status of This Memo

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

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

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

This Internet-Draft will expire on April 21, 2014.

Copyright Notice

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

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents

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

Table of Contents

1. Introduction	3
2. Concepts	4
2.1. Media Chain	4
2.1.1. Physical Stimulus	7
2.1.2. Media Capture	7
2.1.3. Raw Stream	7
2.1.4. Media Source	8
2.1.5. Source Stream	9
2.1.6. Media Encoder	9
2.1.7. Encoded Stream	10
2.1.8. Dependent Stream	10
2.1.9. Media Packetizer	10
2.1.10. Packet Stream	11
2.1.11. Media Redundancy	12
2.1.12. Redundancy Packet Stream	12
2.1.13. Media Transport	12
2.1.14. Received Packet Stream	15
2.1.15. Received Redundancy Packet Stream	15
2.1.16. Media Repair	15
2.1.17. Repaired Packet Stream	15
2.1.18. Media Depacketizer	15
2.1.19. Received Encoded Stream	15
2.1.20. Media Decoder	16
2.1.21. Received Source Stream	16
2.1.22. Media Sink	16
2.1.23. Received Raw Stream	16
2.1.24. Media Render	16
2.2. Communication Entities	17
2.2.1. End Point	17
2.2.2. RTP Session	17
2.2.3. Participant	18
2.2.4. Multimedia Session	19
2.2.5. Communication Session	19
3. Relations at Different Levels	20
3.1. Media Source Relations	20
3.1.1. Synchronization Context	20
3.1.2. End Point	21
3.1.3. Participant	22

3.1.4. WebRTC MediaStream	22
3.2. Packetization Time Relations	22
3.2.1. Single Stream Transport of SVC	23
3.2.2. Multi-Channel Audio	23
3.2.3. Redundancy Format	23
3.3. Packet Stream Relations	24
3.3.1. Simulcast	24
3.3.2. Layered Multi-Stream Transmission	25
3.3.3. Robustness and Repair	26
3.3.4. Packet Stream Separation	29
3.4. Multiple RTP Sessions over one Media Transport	30
4. Topologies and Communication Entities	30
4.1. Point-to-Point Communication	31
4.2. Central Conferencing	32
4.3. Full Mesh Conferencing	33
4.4. Source-Specific Multicast	36
5. Security Considerations	37
6. Acknowledgement	38
7. Contributors	38
8. IANA Considerations	38
9. References	38
9.1. Normative References	38
9.2. Informative References	38
Appendix A. Changes From Earlier Versions	40
A.1. Modifications Between Version -02 and -03	40
A.2. Modifications Between Version -01 and -02	40
A.3. Modifications Between Version -00 and -01	40
Authors' Addresses	41

1. Introduction

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

This document attempts to provide some clarity by reviewing the semantics of various aspects of sources in RTP. As an organizing mechanism, it approaches this by describing various ways that RTP sources can be grouped and associated together.

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

2. Concepts

This section defines concepts that serve to identify and name various transformations and streams in a given RTP usage. For each concept an attempt is made to list any alternate definitions and usages that co-exist today along with various characteristics that further describes the concept. These concepts are divided into two categories, one related to the chain of streams and transformations that media can be subject to, the other for entities involved in the communication.

2.1. Media Chain

This section contains the concepts that can be involved in taking a sequence of physical world stimulus (sound waves, photons, key-strokes) at a sender side and transport them to a receiver, which may recover a sequence of physical stimulus. This chain of concepts is of two main types, streams and transformations. Streams are time-based sequences of samples of the physical stimulus in various representations, while transformations changes the representation of the streams in some way.

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

- o A transformation may have zero or more inputs and one or more outputs.
- o A Stream is of some type.
- o A Stream has one source transformation and one or more sink transformation (with the exception of Physical Stimulus (Section 2.1.1) that can have no source or sink transformation).
- o Streams can be forwarded from a transformation output to any number of inputs on other transformations that support that type.
- o If the output of a transformation is sent to multiple transformations, those streams will be identical; it takes a transformation to make them different.
- o There are no formal limitations on how streams are connected to transformations, this may include loops if required by a particular transformation.

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

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

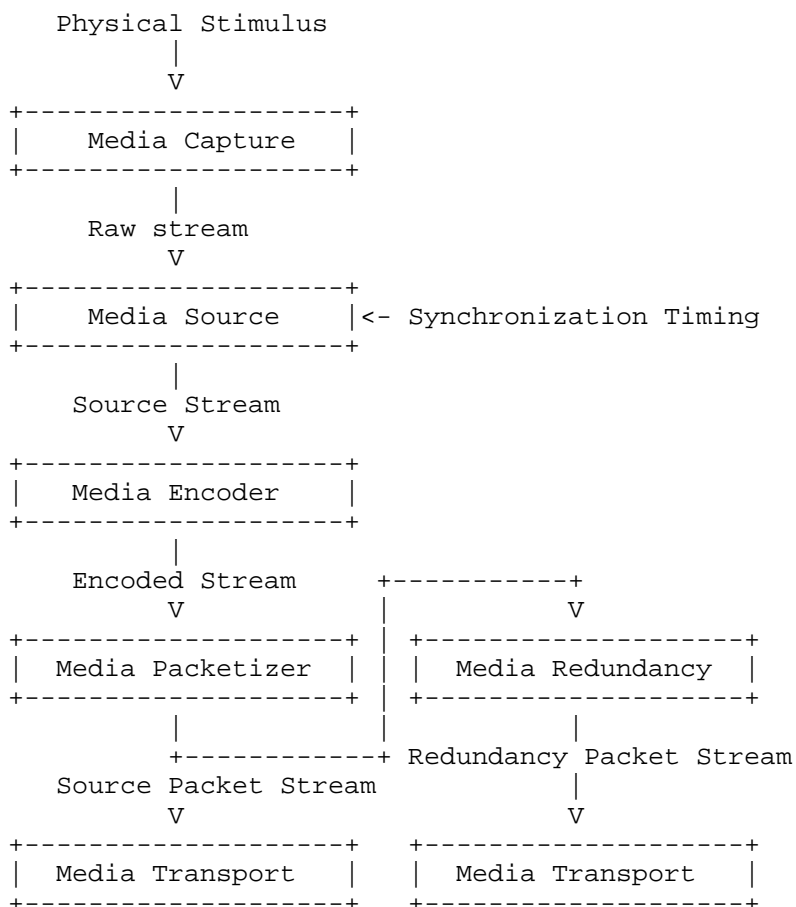
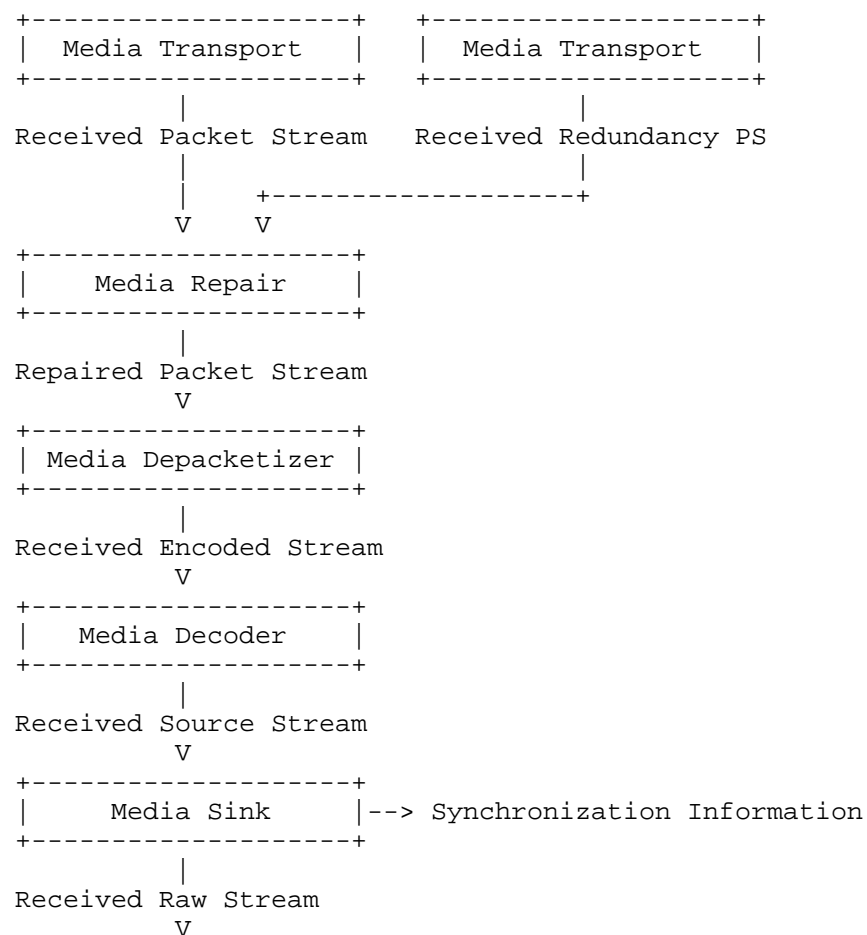


Figure 1: Sender Side Concepts in the Media Chain

In Figure 1 we have included a branched chain to cover the concepts for using redundancy to improve the reliability of the transport. The Media Transport concept is an aggregate that is decomposed below in Section 2.1.13.2.

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



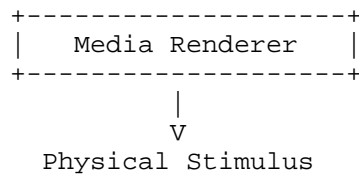


Figure 2: Receiver Side Concepts of the Media Chain

2.1.1. Physical Stimulus

The physical stimulus is a physical event that can be captured and provided as media to a receiver. This include sound waves making up audio, photons in a light field that is visible, or other excitations or interactions with sensors, like keystrokes on a keyboard.

2.1.2. Media Capture

The process of transforming the Physical Stimulus (Section 2.1.1) into captured media. The Media Capture performs a digital sampling of the physical stimulus, usually periodically, and outputs this in some representation as a Raw Stream (Section 2.1.3). This data is due to its periodical sampling, or at least being timed asynchronous events, some form of a stream of media data. The Media Capture is normally instantiated in some type of device, i.e. media capture device. Examples of different types of media capturing devices are digital cameras, microphones connected to A/D converters, or keyboards.

2.1.2.1. Alternate Usages

The CLUE WG uses the term "Capture Device" to identify a physical capture device.

WebRTC WG uses the term "Recording Device" to refer to the locally available capture devices in an end-system.

2.1.2.2. Characteristics

- o A Media Capture is identified either by hardware/manufacturer ID or via a session-scoped device identifier as mandated by the application usage.
- o A Media Capture can generate an Encoded Stream (Section 2.1.7) if the capture device support such a configuration.

2.1.3. Raw Stream

The time progressing stream of digitally sampled information, usually periodically sampled, provided by a Media Capture (Section 2.1.2).

2.1.4. Media Source

A Media Source is the logical source of a reference clock synchronized, time progressing, digital media stream, called a Source Stream (Section 2.1.5). This transformation takes one or more Raw Streams (Section 2.1.3) and provides a Source Stream as output. This output has been synchronized with some reference clock, even if just a system local wall clock.

The output can be of different types. One type is directly associated with a particular Media Capture's Raw Stream. Others are more conceptual sources, like an audio mix of multiple Raw Streams (Figure 3), a mixed selection of the three loudest inputs regarding speech activity, a selection of a particular video based on the current speaker, i.e. typically based on other Media Sources.

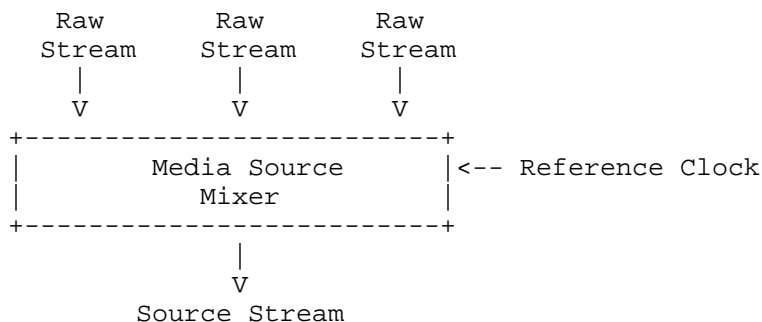


Figure 3: Conceptual Media Source in form of Audio Mixer

2.1.4.1. Alternate Usages

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

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

Typically a Media Source is mapped to a single m=line via the Session Description Protocol (SDP) [RFC4566] unless mechanisms such as

Source-Specific attributes are in place [RFC5576]. In the latter cases, an m=line can represent either multiple Media Sources, multiple Packet Streams (Section 2.1.10), or both.

2.1.4.2. Characteristics

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

2.1.5. Source Stream

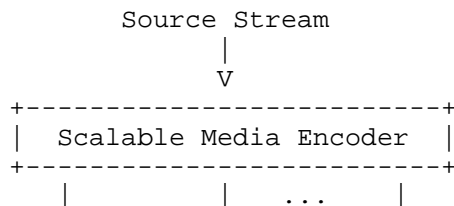
A time progressing stream of digital samples that has been synchronized with a reference clock and comes from particular Media Source (Section 2.1.4).

2.1.6. Media Encoder

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

The Media Encoder step commonly includes pre-encoding transformations, such as scaling, resampling etc. The Media Encoder can have a significant number of configuration options that affects the properties of the encoded stream. This include properties such as bit-rate, start points for decoding, resolution, bandwidth or other fidelity affecting properties. The actually used codec is also an important factor in many communication systems, not only its parameters.

Scalable Media Encoders need special mentioning as they produce multiple outputs that are potentially of different types. A scalable Media Encoder takes one input Source Stream and encodes it into multiple output streams of two different types; at least one Encoded Stream that is independently decodable and one or more Dependent Streams (Section 2.1.8) that requires at least one Encoded Stream and zero or more Dependent Streams to be possible to decode. A Dependent Stream's dependency is one of the grouping relations this document discusses further in Section 3.3.2.



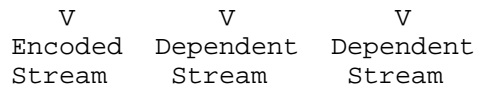


Figure 4: Scalable Media Encoder Input and Outputs

2.1.6.1. Alternate Usages

Within the SDP usage, an SDP media description (m=line) describes part of the necessary configuration required for encoding purposes.

CLUE's "Capture Encoding" provides specific encoding configuration for this purpose.

2.1.6.2. Characteristics

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

2.1.7. Encoded Stream

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

2.1.7.1. Characteristics

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

2.1.8. Dependent Stream

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

2.1.8.1. Characteristics

- o Each Dependent Stream has a set of dependencies. These dependencies must be understood by the parties in a multi-media session that intend to use a Dependent Stream.

2.1.9. Media Packetizer

The transformation of taking one or more Encoded (Section 2.1.7) or Dependent Stream (Section 2.1.8) and put their content into one or

more sequences of packets, normally RTP packets, and output Source Packet Streams (Section 2.1.10). This step includes both generating RTP payloads as well as RTP packets.

The Media Packetizer can use multiple inputs when producing a single Packet Stream. One such example is the packetization when using SVC, as in Single Stream Transport (SST) usage of the payload format both an Encoded Stream as well as Dependent Streams are packetized in a single Source Packet Stream using a single SSRC.

The Media Packetizer can also produce multiple Packet Streams, for example when Encoded and/or Dependent Streams are distributed over multiple Packet Streams, possibly in different RTP sessions.

2.1.9.1. Alternate Usages

An RTP sender is part of the Media Packetizer.

2.1.9.2. Characteristics

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

2.1.10. Packet Stream

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

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

2.1.10.1. Alternate Usages

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

RFC3550 [RFC3550] uses the terms media stream, audio stream, video stream and streams of (RTP) packets interchangeably. It defines the SSRC as the "The source of a stream of RTP packets, ..."

The equivalent mapping of a Packet Stream in SDP [RFC4566] is defined per usage. For example, each Media Description (m=line) and associated attributes can describe one Packet Stream OR properties for multiple Packet Streams OR for an RTP session (via [RFC5576] mechanisms for example).

2.1.10.2. Characteristics

- o Each Packet Stream is identified by a unique Synchronization source (SSRC) [RFC3550] that is carried in every RTP and RTP Control Protocol (RTCP) packet header in a specific RTP session context.
- o At any given point in time, a Packet Stream can have one and only one SSRC.
- o Each Packet Stream defines a unique RTP sequence numbering and timing space.
- o Several Packet Streams may map to a single Media Source via the source transformations.
- o Several Packet Streams can be carried over a single RTP Session.

2.1.11. Media Redundancy

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

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

2.1.12. Redundancy Packet Stream

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

2.1.13. Media Transport

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

2.1.13.1. Characteristics

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

2.1.13.2. Media Stream Decomposition

The Media Transport concept sometimes needs to be decomposed into more steps to enable discussion of what a sender emits that gets transformed by the network before it is received by the receiver. Thus we provide also this Media Transport decomposition (Figure 5).

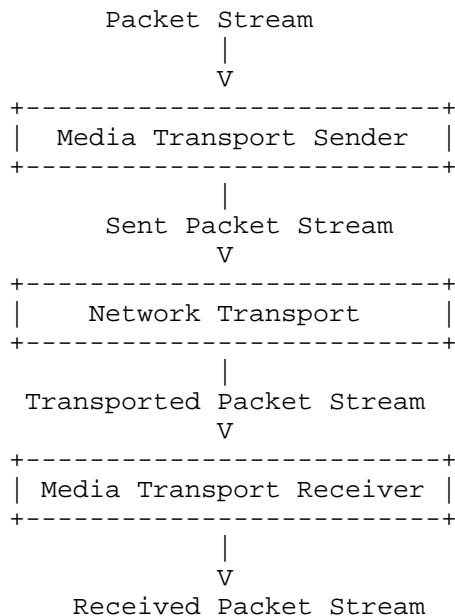


Figure 5: Decomposition of Media Transport

2.1.13.2.1. Media Transport Sender

The first transformation within the Media Transport (Section 2.1.13) is the Media Transport Sender, where the sending End-Point (Section 2.2.1) takes a Packet Stream and emits the packets onto the network using the transport association established for this Media Transport thus creating a Sent Packet Stream (Section 2.1.13.2.2). In this process it transforms the Packet Stream in several ways. First, it gains the necessary protocol headers for the transport association, for example IP and UDP headers, thus forming IP/UDP/RTP packets. In addition, the Media Transport Sender may queue, pace or otherwise affect how the packets are emitted onto the network. Thus adding delay, jitter and inter packet spacings that characterize the Sent Packet Stream.

2.1.13.2.2. Sent Packet Stream

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

2.1.13.2.3. Network Transport

Network Transport is the transformation that the Sent Packet Stream (Section 2.1.13.2.2) is subjected to by traveling from the source to the destination through the network. These transformations include, loss of some packets, varying delay on a per packet basis, packet duplication, and packet header or data corruption. These transformations produces a Transported Packet Stream (Section 2.1.13.2.4) at the exit of the network path.

2.1.13.2.4. Transported Packet Stream

The Packet Stream that is emitted out of the network path at the destination, subjected to the Network Transport's transformation (Section 2.1.13.2.3).

2.1.13.2.5. Media Transport Receiver

The receiver End-Point's (Section 2.2.1) transformation of the Transported Packet Stream (Section 2.1.13.2.4) by its reception process that result in the Received Packet Stream (Section 2.1.14). This transformation includes transport checksums being verified and if non-matching, causing discarding of the corrupted packet. Other transformations can include delay variations in receiving a packet on the network interface and providing it to the application.

2.1.14. Received Packet Stream

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

2.1.15. Received Redundancy Packet Stream

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

2.1.16. Media Repair

A Transformation that takes as input one or more Source Packet Streams (Section 2.1.10) as well as Redundancy Packet Streams (Section 2.1.12) and attempts to combine them to counter the transformations introduced by the Media Transport (Section 2.1.13) to minimize the difference between the Source Stream (Section 2.1.5) and the Received Source Stream (Section 2.1.21) after Media Decoder (Section 2.1.20). The output is a Repaired Packet Stream (Section 2.1.17).

2.1.17. Repaired Packet Stream

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

2.1.18. Media Depacketizer

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

2.1.19. Received Encoded Stream

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

2.1.20. Media Decoder

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

2.1.20.1. Alternate Usages

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

2.1.20.2. Characteristics

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

2.1.21. Received Source Stream

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

2.1.22. Media Sink

The Media Sink receives a Source Stream (Section 2.1.5) that contains, usually periodically, sampled media data together with associated synchronization information. Depending on application, this Source Stream then needs to be transformed into a Raw Stream (Section 2.1.3) that is sent in synchronization with the output from other Media Sinks to a Media Render (Section 2.1.24). The media sink may also be connected with a Media Source (Section 2.1.4) and be used as part of a conceptual Media Source.

2.1.22.1. Characteristics

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

2.1.23. Received Raw Stream

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

2.1.24. Media Render

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

2.1.24.1. Characteristics

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

2.2. Communication Entities

This section contains concept for entities involved in the communication.

2.2.1. End Point

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

2.2.1.1. Alternate Usages

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

2.2.1.2. Characteristics

End Points can be identified in several different ways. While RTCP Canonical Names (CNAMEs) [RFC3550] provide a globally unique and stable identification mechanism for the duration of the Communication Session (see Section 2.2.5), their validity applies exclusively within a Synchronization Context (Section 3.1.1). Thus one End Point can have multiple CNAMEs. Therefore, mechanisms outside the scope of RTP, such as application defined mechanisms, must be used to ensure End Point identification when outside this Synchronization Context.

2.2.2. RTP Session

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

2.2.2.1. Alternate Usages

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

2.2.2.2. Characteristics

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

2.2.3. Participant

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

2.2.3.1. Characteristics

- o A single signaling-addressable entity, using an application-specific signaling address space, for example a SIP URI.
- o A participant can have several Multimedia Sessions (Section 2.2.4).

- o A participant can have several associated transport flows, including several separate local transport addresses for those transport flows.

2.2.4. Multimedia Session

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

2.2.4.1. Alternate Usages

RFC4566 [RFC4566] defines a multimedia session as a set of multimedia senders and receivers and the data streams flowing from senders to receivers.

RFC3550 [RFC3550] defines it as set of concurrent RTP sessions among a common group of participants. For example, a video conference (which is a multimedia session) may contain an audio RTP session and a video RTP session.

2.2.4.2. Characteristics

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

2.2.5. Communication Session

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

2.2.5.1. Alternate Usages

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

2.2.5.2. Characteristics

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

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

3. Relations at Different Levels

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

3.1. Media Source Relations

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

3.1.1. Synchronization Context

A Synchronization Context defines a requirement on a strong timing relationship between the Media Sources, typically requiring alignment of clock sources. Such relationship can be identified in multiple ways as listed below. A single Media Source can only belong to a single Synchronization Context, since it is assumed that a single Media Source can only have a single media clock and requiring alignment to several Synchronization Contexts (and thus reference clocks) will effectively merge those into a single Synchronization Context.

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

3.1.1.1. RTCP CNAME

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

3.1.1.2. Clock Source Signaling

[I-D.ietf-avtcore-clksrc] provides a mechanism to signal the clock source in SDP both for the reference clock as well as the media clock, thus allowing a Synchronization Context to be defined beyond the one defined by the usage of CNAME source descriptions.

3.1.1.3. CLUE Scenes

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

3.1.1.4. Implicitly via RtcMediaStream

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

3.1.1.5. Explicitly via SDP Mechanisms

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

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

3.1.2. End Point

Some applications requires knowledge of what Media Sources originate from a particular End Point (Section 2.2.1). This can include such

decisions as packet routing between parts of the topology, knowing the End Point origin of the Packet Streams.

In RTP, this identification has been overloaded with the Synchronization Context through the usage of the source description CNAME item. This works for some usages, but sometimes it breaks down. For example, if an End Point has two sets of Media Sources that have different Synchronization Contexts, like the audio and video of the human participant as well as a set of Media Sources of audio and video for a shared movie. Thus, an End Point may have multiple CNAMEs. The CNAMEs or the Media Sources themselves can be related to the End Point.

3.1.3. Participant

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

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

3.1.4. WebRTC MediaStream

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

3.2. Packetization Time Relations

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

3.2.1. Single Stream Transport of SVC

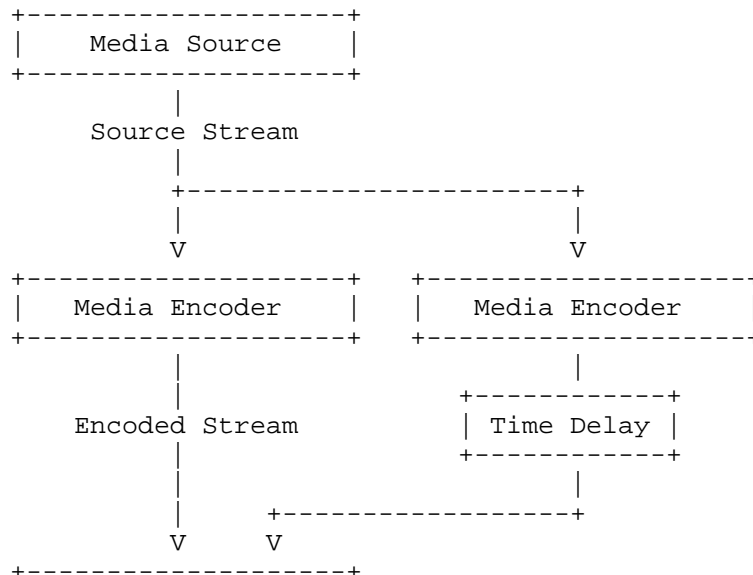
Scalable Video Coding [RFC6190] has a mode of operation where Encoded Streams and Dependent Streams from the SVC Media Encoder is grouped together in a single Source Packet Stream using the SVC RTP Payload format.

3.2.2. Multi-Channel Audio

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

3.2.3. Redundancy Format

The RTP Payload for Redundant Audio Data [RFC2198] defines how one can transport redundant audio data together with primary data in the same RTP payload. The redundant data can be a time delayed version of the primary or another time delayed Encoded stream using a different Media Encoder to encode the same Media Source as the primary, as depicted below in Figure 6.



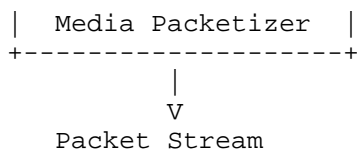


Figure 6: Concept for usage of Audio Redundancy with different Media Encoders

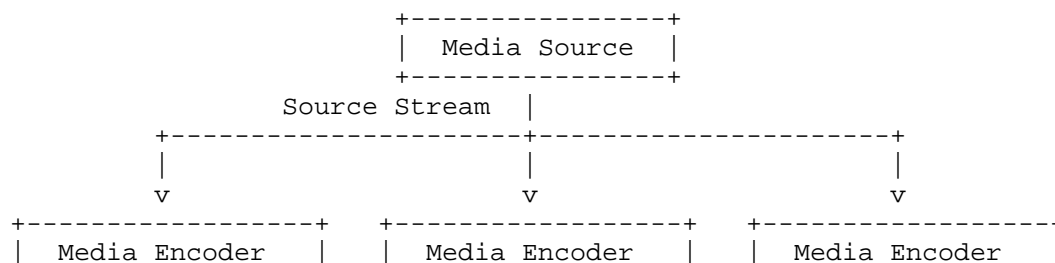
The Redundancy format is thus providing the necessary meta information to correctly relate different parts of the same Encoded Stream, or in the case depicted above (Figure 6) relate the Received Source Stream fragments coming out of different Media Decoders to be able to combine them together into a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

3.3. Packet Stream Relations

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

3.3.1. Simulcast

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



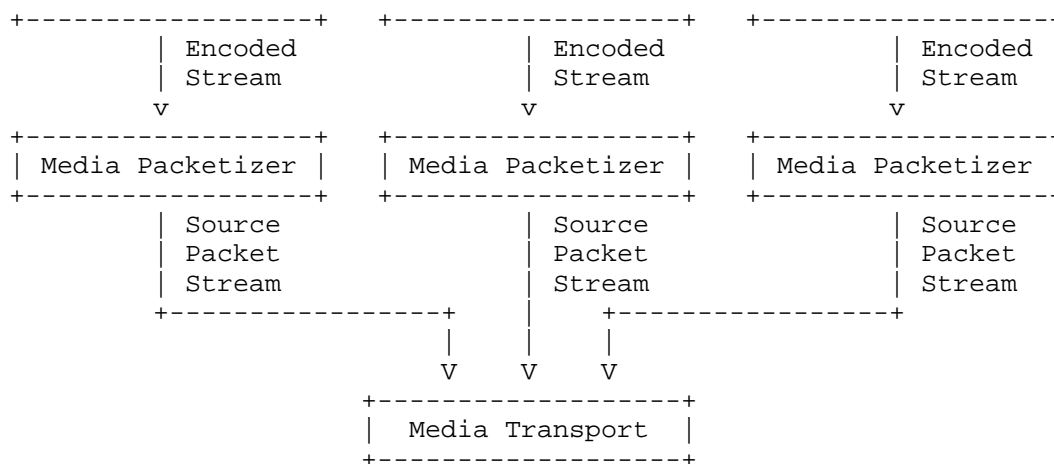


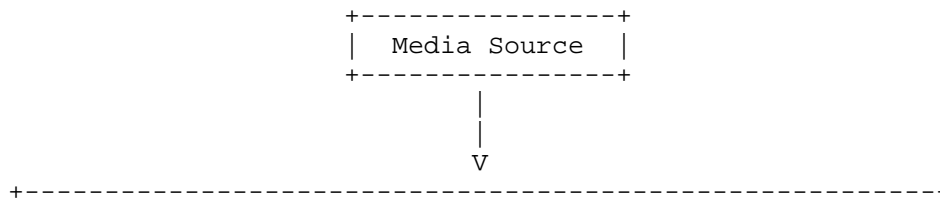
Figure 7: Example of Media Source Simulcast

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

3.3.2. Layered Multi-Stream Transmission

Multi-stream transmission (MST) is a mechanism by which different portions of a layered encoding of a Source Stream are sent using separate Packet Streams (sometimes in separate RTP sessions). MSTs are useful for receiver control of layered media.

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



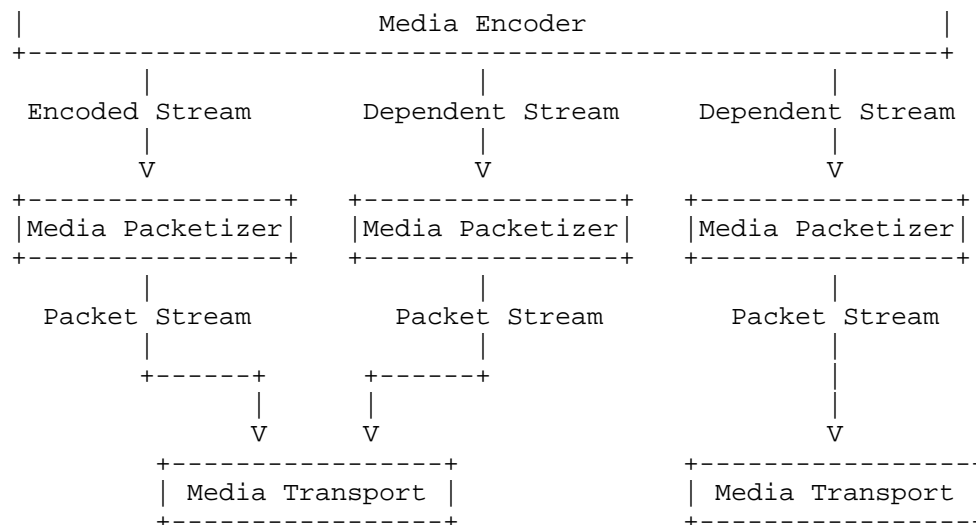


Figure 8: Example of Media Source Layered Dependency

The SVC MST relation needs to identify the common Media Encoder origin for the Encoded and Dependent Streams. The SVC RTP Payload RFC is not particularly explicit about how this relation is to be implemented. When using different RTP Sessions, thus different Media Transports, and as long as there is only one Packet Stream per Media Encoder and a single Media Source in each RTP Session, common SSRC and CNAMEs can be used to identify the common Media Source. When multiple Packet Streams are sent from one Media Encoder in the same RTP Session, then CNAME is the only currently specified RTP identifier that can be used. In cases where multiple Media Encoders use multiple Media Sources sharing Synchronization Context, and thus having a common CNAME, additional heuristics need to be applied to create the MST relationship between the Packet Streams.

3.3.3. Robustness and Repair

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

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

3.3.3.1. RTP Retransmission

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

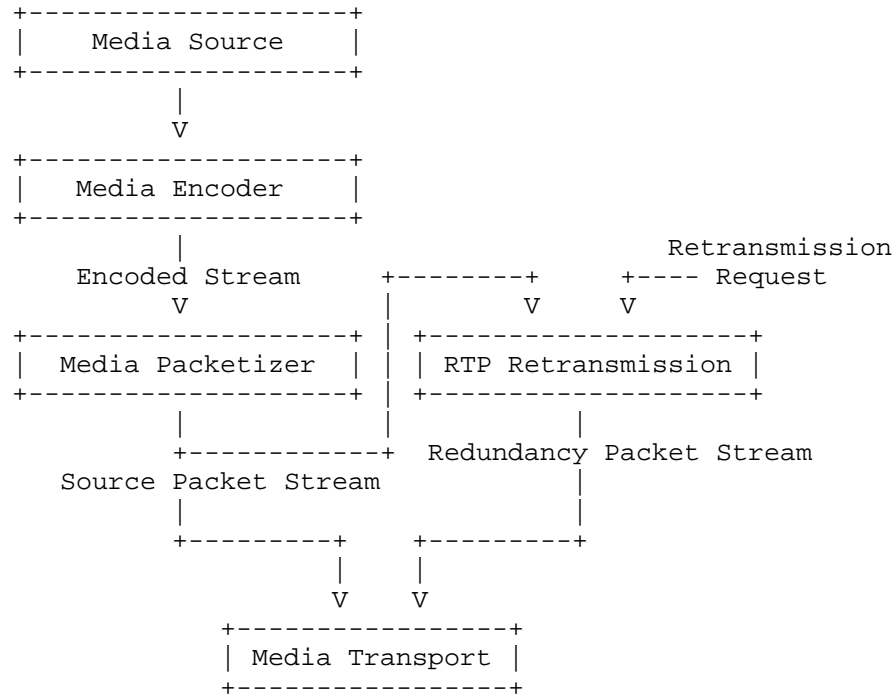


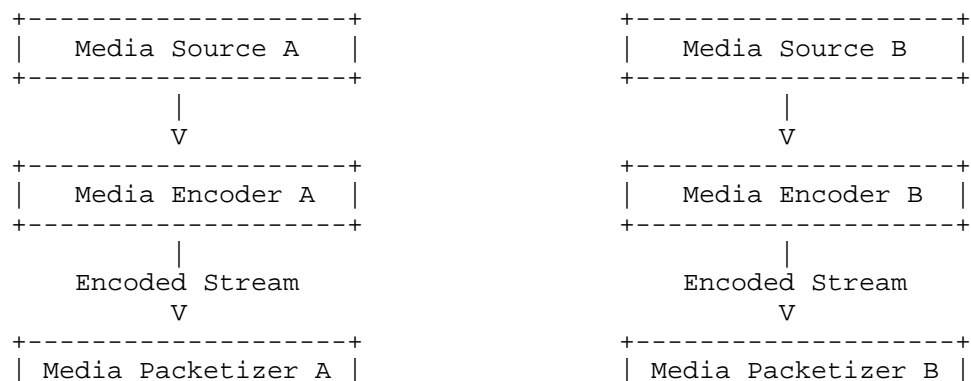
Figure 9: Example of Media Source Retransmission Flows

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

3.3.3.2. Forward Error Correction

The figure below (Figure 10) represents an example where two Media Sources' Source Packet Streams are protected by FEC. Source Packet Stream A has a Media Redundancy transformation in FEC Encoder 1. This produces a Redundancy Packet Stream 1, that is only related to Source Packet Stream A. The FEC Encoder 2, however takes two Source Packet Streams (A and B) and produces a Redundancy Packet Stream 2 that protects them together, i.e. Redundancy Packet Stream 2 relate to two Source Packet Streams (a FEC group). FEC decoding, when needed due to packet loss or packet corruption at the receiver, requires knowledge about which Source Packet Streams that the FEC encoding was based on.

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



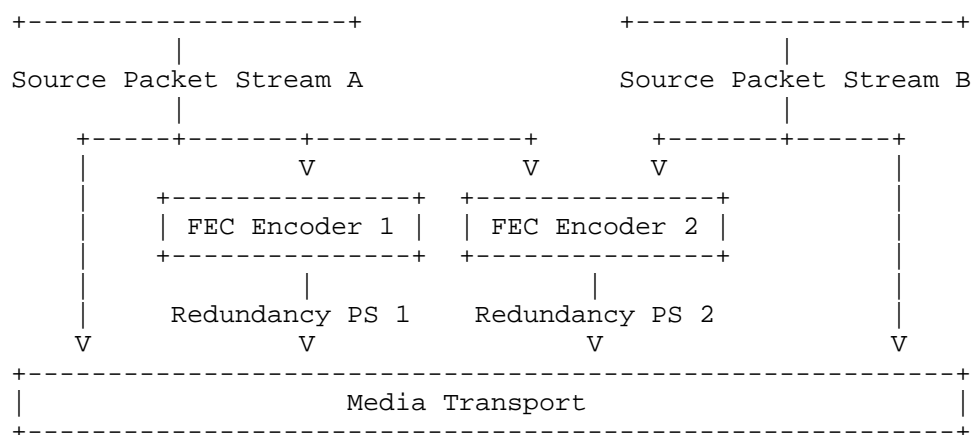


Figure 10: Example of FEC Flows

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

3.3.4. Packet Stream Separation

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

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

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

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

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

3.4. Multiple RTP Sessions over one Media Transport

[I-D.westerlund-avtcore-transport-multiplexing] describes a mechanism that allow several RTP Sessions to be carried over a single underlying Media Transport. The main reasons for doing this are related to the impact of using one or more Media Transports. Thus using a common network path or potentially have different ones. There is reduced need for NAT/FW traversal resources and no need for flow based QoS.

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

4. Topologies and Communication Entities

This Section reviews some communication topologies and looks at the relationship among the communication entities that are defined in Section 2.2. This section doesn't deal with discussions about the streams and their relation to the transport. Instead, it covers the aspects that enable the transport of those streams. For example, the Media Transports (Section 2.1.13) that exists between the End Points

(Section 2.2.1) that are part of an RTP session (Section 2.2.2) and their relationship to the Multi-Media Session (Section 2.2.4) between Participants (Section 2.2.3) and the established Communication session (Section 2.2.5) are explained.

4.1. Point-to-Point Communication

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

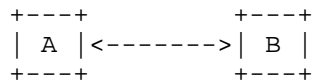


Figure 11: Point to Point Communication

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

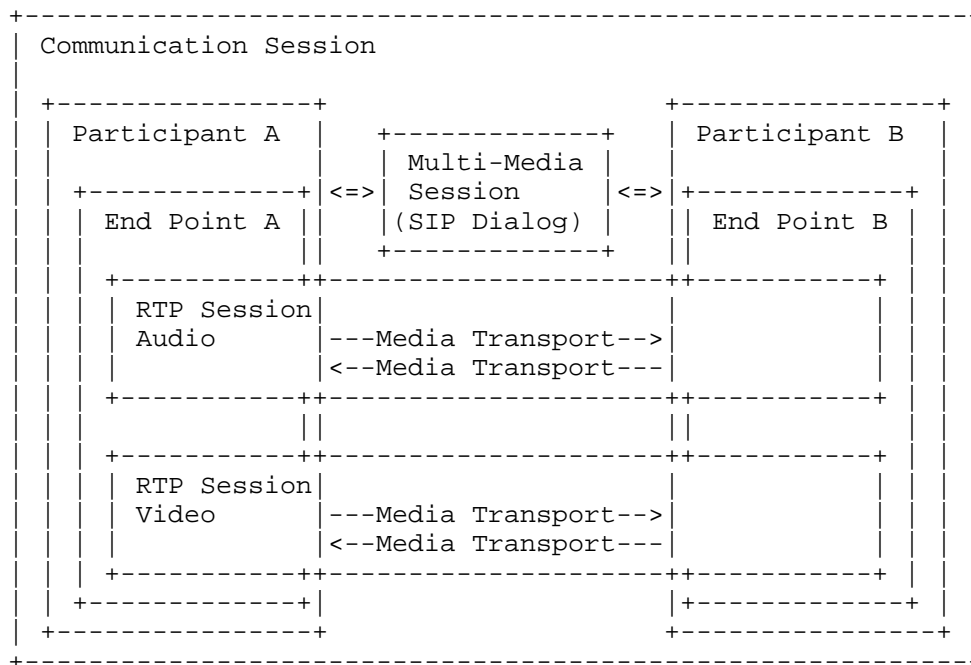


Figure 12: Point to Point Communication Session with two RTP Sessions

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

4.2. Central Conferencing

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

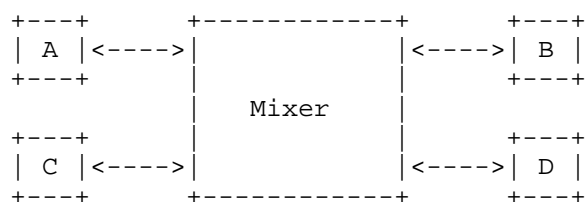
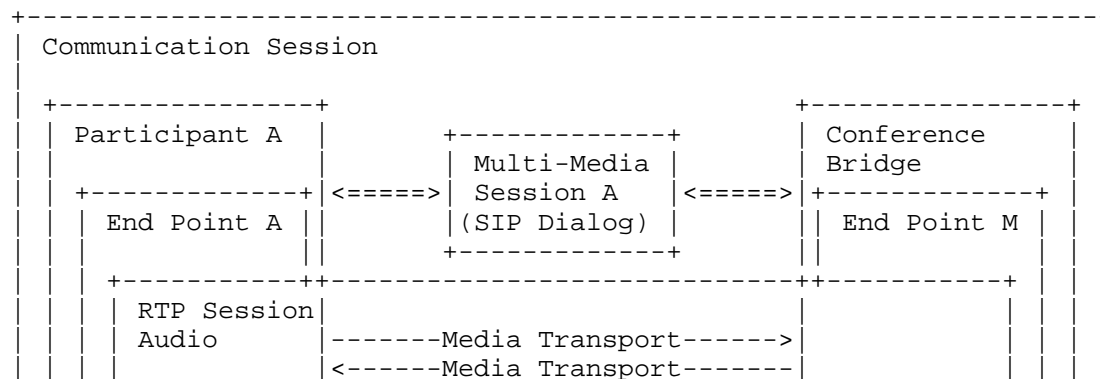


Figure 13: Centralized Conferencing using an RTP Mixer

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



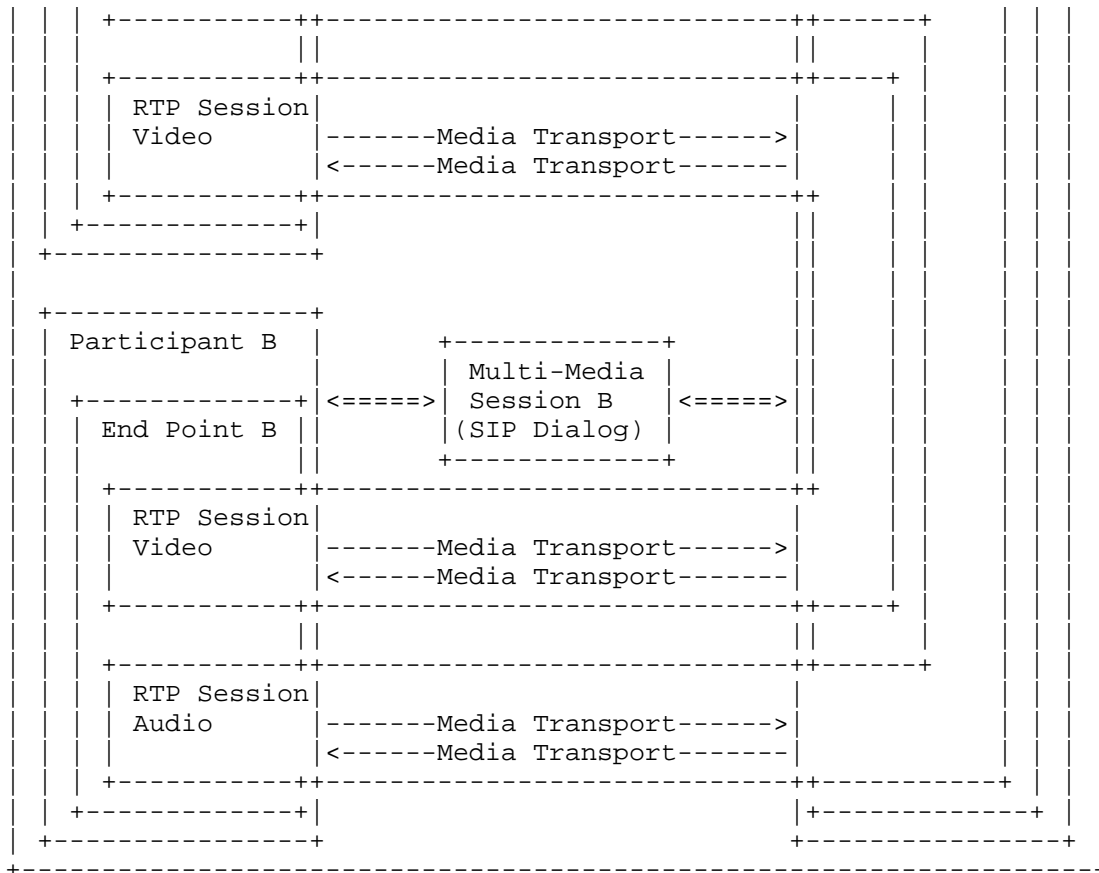


Figure 14: Central Conferencing with Two Participants A and B communicating over a Conference Bridge

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

4.3. Full Mesh Conferencing

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

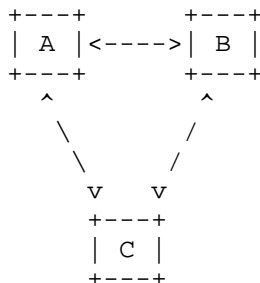
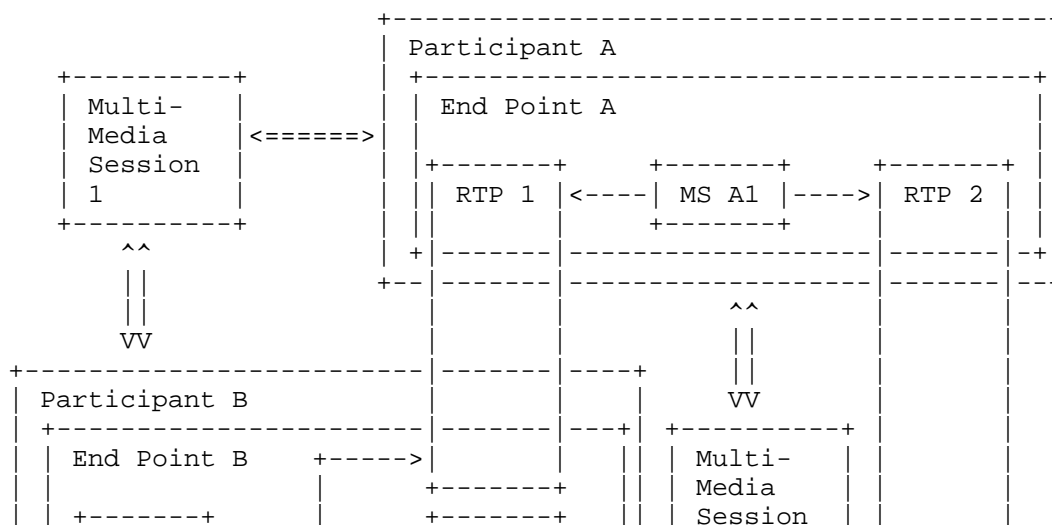


Figure 15: Full Mesh Conferencing with three Participants A, B and C

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



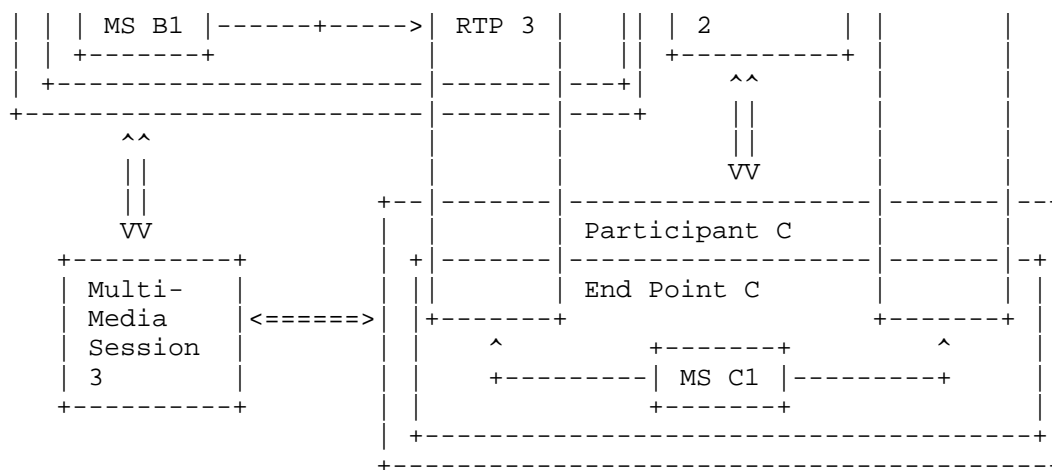
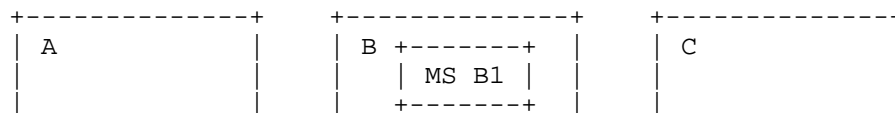


Figure 16: Full Mesh Conferencing between three Participants A, B and C

For the sake of clarity, Figure 16 above does not include all these concepts. The Media Sources (MS) from a given End Point is sent to the two peers. This requires encoding and Media Packetization to enable the Packet Streams to be sent over Media Transports in the context of the RTP sessions depicted. The RTP sessions 1, 2, and 3 are independent, and established in the context of each of the Multi-Media Sessions 1, 2 and 3. The joint communication session the full figure represents (not shown here as it was Figure 14 in order to save space), however, combines the received representations of the peers' Media Sources and plays them back.

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



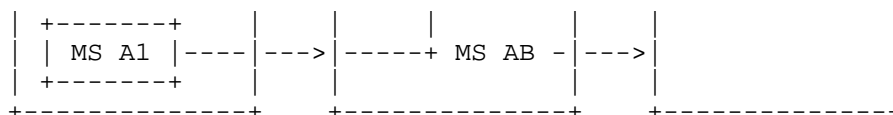
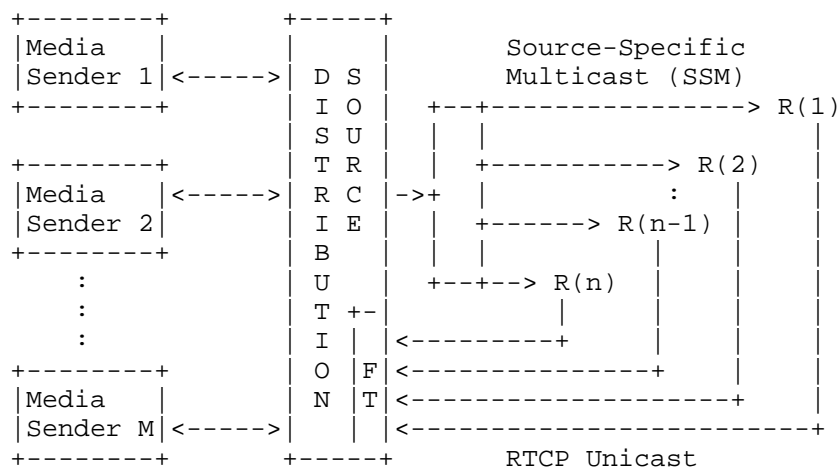


Figure 17: Mixing Three Party Communication Session

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

4.4. Source-Specific Multicast

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



FT = Feedback Target

Figure 18: Source-Specific Multicast Communication Topology

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

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

To improve the readability, Figure 18 intentionally hides the details of the various entities. Expanding on this, one can think of Media Senders being part of one or more Multi-Media Sessions grouped under a Communication Session. The Media Sender in this scenario refers to the Media Packetizer transformation Section 2.1.9. The Packet Stream generated by such a Media Sender can be part of its own RTP Session or can be multiplexed with other Packet Streams within an End Point. The latter case requires careful consideration since the re-distributed RTCP packets now correspond to a single RTP Session Context across all the Media Senders.

5. Security Considerations

This document simply tries to clarify the confusion prevalent in RTP taxonomy because of inconsistent usage by multiple technologies and protocols making use of the RTP protocol. It does not introduce any new security considerations beyond those already well documented in the RTP protocol [RFC3550] and each of the many respective specifications of the various protocols making use of it.

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

6. Acknowledgement

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

The authors would also like to acknowledge the insights, guidance and contributions of Magnus Westerlund, Roni Even, Paul Kyzivat, Colin Perkins, Keith Drage, and Harald Alvestrand.

7. Contributors

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

8. IANA Considerations

This document makes no request of IANA.

9. References

9.1. Normative References

- [RFC3550] Schulzrinne, H., Casner, S., Frederick, R., and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications", STD 64, RFC 3550, July 2003.
- [UML] Object Management Group, "OMG Unified Modeling Language (OMG UML), Superstructure, V2.2", OMG formal/2009-02-02, February 2009.

9.2. Informative References

- [I-D.ietf-avtcore-clksrc] Williams, A., Gross, K., Brandenburg, R., and H. Stokking, "RTP Clock Source Signalling", draft-ietf-avtcore-clksrc-07 (work in progress), October 2013.
- [I-D.ietf-clue-framework] Duckworth, M., Pepperell, A., and S. Wenger, "Framework for Telepresence Multi-Streams", draft-ietf-clue-framework-11 (work in progress), July 2013.

- [I-D.ietf-mmusic-sdp-bundle-negotiation]
Holmberg, C., Alvestrand, H., and C. Jennings,
"Multiplexing Negotiation Using Session Description
Protocol (SDP) Port Numbers", draft-ietf-mmusic-sdp-
bundle-negotiation-05 (work in progress), October 2013.
- [I-D.ietf-rtcweb-overview]
Alvestrand, H., "Overview: Real Time Protocols for Brower-
based Applications", draft-ietf-rtcweb-overview-08 (work
in progress), September 2013.
- [I-D.westerlund-avtcore-transport-multiplexing]
Westerlund, M. and C. Perkins, "Multiple RTP Sessions on a
Single Lower-Layer Transport", draft-westerlund-avtcore-
transport-multiplexing-06 (work in progress), August 2013.
- [RFC2198] Perkins, C., Kouvelas, I., Hodson, O., Hardman, V.,
Handley, M., Bolot, J., Vega-Garcia, A., and S. Fosse-
Parisis, "RTP Payload for Redundant Audio Data", RFC 2198,
September 1997.
- [RFC2974] Handley, M., Perkins, C., and E. Whelan, "Session
Announcement Protocol", RFC 2974, October 2000.
- [RFC3264] Rosenberg, J. and H. Schulzrinne, "An Offer/Answer Model
with Session Description Protocol (SDP)", RFC 3264, June
2002.
- [RFC3551] Schulzrinne, H. and S. Casner, "RTP Profile for Audio and
Video Conferences with Minimal Control", STD 65, RFC 3551,
July 2003.
- [RFC4566] Handley, M., Jacobson, V., and C. Perkins, "SDP: Session
Description Protocol", RFC 4566, July 2006.
- [RFC4588] Rey, J., Leon, D., Miyazaki, A., Varsa, V., and R.
Hakenberg, "RTP Retransmission Payload Format", RFC 4588,
July 2006.
- [RFC4867] Sjoberg, J., Westerlund, M., Lakaniemi, A., and Q. Xie,
"RTP Payload Format and File Storage Format for the
Adaptive Multi-Rate (AMR) and Adaptive Multi-Rate Wideband
(AMR-WB) Audio Codecs", RFC 4867, April 2007.
- [RFC5109] Li, A., "RTP Payload Format for Generic Forward Error
Correction", RFC 5109, December 2007.

- [RFC5404] Westerlund, M. and I. Johansson, "RTP Payload Format for G.719", RFC 5404, January 2009.
- [RFC5576] Lennox, J., Ott, J., and T. Schierl, "Source-Specific Media Attributes in the Session Description Protocol (SDP)", RFC 5576, June 2009.
- [RFC5760] Ott, J., Chesterfield, J., and E. Schooler, "RTP Control Protocol (RTCP) Extensions for Single-Source Multicast Sessions with Unicast Feedback", RFC 5760, February 2010.
- [RFC5888] Camarillo, G. and H. Schulzrinne, "The Session Description Protocol (SDP) Grouping Framework", RFC 5888, June 2010.
- [RFC5905] Mills, D., Martin, J., Burbank, J., and W. Kasch, "Network Time Protocol Version 4: Protocol and Algorithms Specification", RFC 5905, June 2010.
- [RFC6190] Wenger, S., Wang, Y., Schierl, T., and A. Eleftheriadis, "RTP Payload Format for Scalable Video Coding", RFC 6190, May 2011.
- [RFC6222] Begen, A., Perkins, C., and D. Wing, "Guidelines for Choosing RTP Control Protocol (RTCP) Canonical Names (CNAMES)", RFC 6222, April 2011.

Appendix A. Changes From Earlier Versions

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

A.1. Modifications Between Version -02 and -03

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

A.2. Modifications Between Version -01 and -02

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

A.3. Modifications Between Version -00 and -01

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

Authors' Addresses

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

Email: jonathan@vidyo.com

Kevin Gross
AVA Networks, LLC
Boulder, CO
US

Email: kevin.gross@avanw.com

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

Email: snandaku@cisco.com

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

Email: gsalguei@cisco.com

Bo Burman
Ericsson
Farogatan 6
SE-164 80 Kista
Sweden

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

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: April 24, 2014

M. Westerlund
B. Burman
Ericsson
October 21, 2013

RTCP Source Description Item SRCNAME to Label Individual Media Sources
draft-westerlund-avtext-rtcp-sdes-srcname-03

Abstract

This document defines a new Source Description (SDS) item called SRCNAME, which uniquely identifies a single media source, like a camera or a microphone. It also enables identification of the encoding to support when multiple ones are produced. That way anyone receiving the SDS information from a set of interlinked RTP sessions can determine which SSRCs are logically related to the same media source and encoding. In addition the new SDS item is also defined for usage with both a header extension and with the SDP source specific media attribute ("a=ssrc"). Enabling an end-point to receive the SRCNAME with the relevant RTP packets, as well as RTCP, or learn the source bindings through signalling, ahead of receiving RTP and 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 24, 2014.

Copyright Notice

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

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents

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

Table of Contents

1. Introduction	3
2. Definitions	3
2.1. Requirements Language	3
2.2. Terminology	3
3. Motivation	4
4. Solution	6
4.1. SRCNAME Format	6
4.2. SDP Item SRCNAME	7
4.3. SRCNAME in SDP	7
4.4. SRCNAME as RTP Header Extension	8
5. Usage with the Offer/Answer Model	8
6. Backward Compatibility	9
7. Relation to Application Token	9
8. IANA Considerations	10
9. Security Considerations	11
10. References	11
10.1. Normative References	11
10.2. Informative References	12
Authors' Addresses	13

1. Introduction

This specification defines a new RTP/RTCP [RFC3550] Source Description (SDES) item called Source Name (SRCNAME). There exist different use cases, including simulcast and scalable encoding, where a sender transmit multiple RTP packet streams containing full or partial encodings of the same media source. This include multiple independent encodings, where it is desirable to identify the different encodings. These different packet streams needs to be correctly associated with media sources and encodings in an receiver so that they correctly use the packet streams.

The proposed solution provides the RTP packet streams (SSRCs) with identities for both the media source and the specific encoding. The identification is done by creating a RTCP SDES item, SRCNAME, by combing a media source identifier and an encoding identifier separated by a full stop ("."). The SRCNAME can be sent periodically in RTCP SDES packets to enable joiners to receive the information within some time period from when they join. The SRCNAME is also proposed to be sent in an RTP header extension for SDES items [I-D.westerlund-avtext-sdes-hdr-ext] when it is desirable to speed up reception. For example by transmitting the SRCNAME in the first N RTP packets when a new SSRC joins an RTP session. Finally the SRCNAME can be associated with the SSRC in signalling, and source specific attribute is provided for this purpose.

2. Definitions

2.1. Requirements Language

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

2.2. Terminology

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

- o Media Source
- o Packet Stream
- o Media Encoder

- o Encoded Stream
- o Dependent Stream
- o Participant
- o End point

3. Motivation

In RTP Applications where an end-point has more than one Media Source in a particular RTP session there can exist need to provide these media sources with an identifier. One reason is to be able to explicitly track it across any SSRC collisions with resulting SSRC changes. Another reason is when there exist multiple RTP Packet Streams (SSRC) associated with that particular media source. Especially in RTP sessions where multiple media sources are simultaneously transmitted. This document focus on the cases that results in multiple packet streams due to the encoding process.

Simulcast [I-D.westerlund-avtcore-rtp-simulcast] as referred to in this document is the process when communication participant provides a media source in multiple encodings using multiple media encoders with different configurations. These different encoded streams are then simultaneously transmitted using RTP to a receiver or a group of receivers. The receiver(s) need two things; First to determine which of the received packet streams (SSRCs) carries which media source, and thus determining the different media sources and secondly what alternative representations of each media source that exist. This can be accomplished using an identifier to refer to a particular encoding of a media source.

Scalable encoding is performed by some few media encoders, with the prime example being H.264 Scalable Video Codec [RFC6190]. A scalable media codec produces one or more base layers, i.e. an encoded stream, and additionally one or more enhancement layers that are dependent on the base layer as well as selected other enhancement layers, these called dependent streams. The encoded and dependent streams can be sent using multiple RTP packet streams, called multi-stream transmission (MST). Thus explicit information are required for which media source a particular packet stream (SSRC) are containing, independent if it is the encoded or dependent stream. In cases where one uses multiple base layers, the encoding identifier can be used to provide RTP/RTCP level identification of the sub-groups of packet streams that form an independent dependency tree. The detailed dependency information between the encoded streams and dependent streams are present in meta data information objects (SEI messages)

that are included inside the RTP payloads for SVC [RFC6190].

By providing media source and encoding identity information on RTP and RTCP level we enable or improve usages that prior has been impractical or sub-optimal:

- a. A multi-party sessions where the media sources dynamically join and leave and the central media node is source projection mixer. A large conference with some participant churn, in this case to rely solely on a signalling based solution can be problematic, as each signalling session between the conference and all the participants needs to be updated, for example using SIP, each time a participant joins or leaves. Thus enabling RTP/RTCP level information enables the joining participant's flows to be explicitly indicated as new media sources and alternative representations on RTP/RTCP level and thus correctly handled.
- b. Multicast or broadcast situations where session configuration information is provided ahead of the session, and the exact set of media sources and their identifies can't be determined and assigned ahead of time.
- c. To optimize the away the need for buffering or holding transmission in centralized mixer cases when there is some delay on the signalling channel. When a media source is added and the information is provided using signalling only, then a receiver that hasn't gotten the signalling yet, needs to either buffer or discard received media until the signalling arrives, alternatively, the sender needs to hold the transmission until the receiver have confirmed reception of signalling.
- d. By providing this information in the RTP/RTCP also enables third party monitoring of the RTP/RTCP streams to work better as the stream relations are made clear.

It is important to note that a particular RTP packet stream's role in a communication application can be quite independent to which media source and the particular encoding the packet stream is. Although the media source and encoding is sufficient information in some use cases, there are other cases where additional information about the current role of packet stream or set of streams are required. Further discussion of this in Section 7.

SRCNAME extends and complement the existing solutions using SDP Media Description grouping [RFC5888], or SSRC grouping within a Media Description in SDP [RFC5576] or implicit or heuristic based mapping of packet streams between or within RTP sessions. SRCNAME enables explicit identity information at RTP/RTCP level in a form that are

unique across the whole communication session, usable to create relationships on RTP/RTCP level independent if one or more RTP session is used, independent on how the packet streams are distributed over those RTP sessions and how many media sources an end-point have.

4. Solution

This section defines the SRCNAME identifier format and its usage as RTCP SDES item [RFC3550], registers it as an SDES item possible to use in the RTP header extension for SDES items [I-D.westerlund-avtext-sdes-hdr-ext], and in a source specific SDP attribute [RFC5576] as well.

4.1. 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] text string that have a maximum length of 255 bytes.

In addition, there are format restrictions to accommodate the separation of the Media Source ID and the encoding ID part, as described by the following ABNF [RFC5234]:

```
media-source-id = 1*(%x01-09 / %x0B-0C / %x0E-1F / %x21-2D / %x2F-FF)
encoding-id     = media-source-id *(%x2E media-source-id)
                  ; Same as RFC 4566 "byte-string"
                  ; except for space and the "." separator
```

```
srcname-content = media-source %x2E encoding-id
```

Figure 1: SRCNAME Format ABNF

Note, the format do allow multiple "." separators, but only as part of the encoding ID.

The media source identifier is identifying a media source (as defined by section 2.1.4 of [I-D.lennox-raiarea-rtp-grouping-taxonomy]). Each media source ID MUST be unique when combined with the CNAME. Note that if one intended to byte compare the combination of CNAME and media-source-id then one need to pad the CNAME to full 255 bytes with a common pattern prior to concatenation and comparison.

The encoding-id identifies a particular media encoder (Section 2.1.6 in [I-D.lennox-raiarea-rtp-grouping-taxonomy]) and its set of

produced encoded or dependent streams (as defined per section 2.1.7 and 2.1.8 in [I-D.lennox-raiarea-rtp-grouping-taxonomy] respectively). The encoding-id MUST be unique in the context of the CNAME and the media source ID.

By require uniqueness scoped by CNAME we simplify the creation of unique identifiers and reduce the overhead for the inclusion of SRCNAME. As the CNAME defines the scope of a single synchronization context, commonly a single host will be responsible for assigning media source and encoding ID to media sources and their encodings. A common case will be for having a single character media source ID followed by stop and then another single character encoding ID, e.g. "a.2".

4.2. SDES Item SRCNAME

Distributing the SRCNAME using a RTCP Source Descriptions (SDES) item are a method that should work with all RTP topologies (assuming that any intermediary node is supporting this item) and existing RTP extensions. Thus, a new SDES item called SRCNAME are defined. That way, anyone receiving the SDES information from a set of interlinked RTP sessions or SSRCs in a single session can determine the SRCNAME associated with each SSRC.

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=TBAl |      length      | source name                      ...
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+

```

Figure 2: SDES SRCNAME Format

The source name field MUST follow the above (Section 4.1) srcname-content definition.

When using the SRCNAME SDES item, it is of equally importance with 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.

4.3. SRCNAME in SDP

"Source-Specific Media Attributes in the Session Description Protocol (SDP)" [RFC5576] defines a way of declaring attributes for SSRC in

each RTP 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 define 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.

Formal ABNF 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".

4.4. SRCNAME as RTP Header Extension

In cases when timely deliver of the SRCNAME is required, for example when adding a new SSRC to an RTP session, or when new receiver joins a multiparty RTP session, then the SRCNAME can be included in the RTP header extension for SDES items [I-D.westerlund-avtext-sdes-hdr-ext].

The RTP header extension for SDES items [I-D.westerlund-avtext-sdes-hdr-ext] is functioning for any SDES item, but do require new SDES items to register its URN identifier. This is done below in the IANA section (Section 8).

5. 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].

6. 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 a=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.

7. Relation to Application Token

There exists a proposal for an application token SDES item [I-D.even-mmusic-application-token], who's purpose is to map SSRCs to application purposes or usages of the RTP packet stream. In this section the similarities and differences are discussed to arrive at the conclusion that for a number of cases both will be required to enable powerful applications.

The APPID is flexible in that it allows applications or specific usage of RTP to define how they map the APPID tokens to particular purpose or usages of the streams. This is clearly intended to provide flexibility. For example one APPID tokens can have meanings such as Presentation stream, main talker, video thumbnail number 3, FEC stream for Audio etc. Such roles can be transient in their behavior. For example main talker is a role that moves around in a multiparty communication session based on who is the current speaker, based on voice activity, or a conference management interface. Thus, the APPID token for this role will be moved between different SSRCS. This is in strong contrast with SRCNAME which identifies a particular media source and encoding. That is not expected to move around, other than in cases of SSRC collisions, when they enable tracking

across this event. RTP Mixers that perform mixes or switching between input sources, are themselves having conceptual media sources, which will have stable identities.

A case that makes it clear that SRCNAME identification may benefit from having additional role tokens is the case of having a source projection mixer using simulcast from clients to mixers. From the perspective of a receiver, there will be multiple SSRCS visible for a particular media source, but the source projection mixer will select a sub-set of all potential streams to deliver. A given sub-setting is to only deliver one representation of each media source to the receiver. During a multiparty conference where a main speaker is shown larger at the receiver, and other participants are shown smaller, the mixer may due to congestion be forced to switch representation of the main speaker. If the role would be strictly associated with the encoding representation then main speakers video may for example be reduced in display size. If instead it is explicitly indicated using APPID the receiving application would continue to show the main speaker as a larger display area, despite the reduced quality to ensure the user continuously to understand that this is still the main talker.

Where SRCNAME provides stable identification that a SSRC is associated with a media source and particular encoding of that media source, the APPID can function as a complement when needed to provide explicit indication of the current role and intended of application usage of a SSRC.

8. 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 4.2. This item needs to be assigned an identifier TBA1.
2. A new SDP source attribute named srcname, as defined in Section 4.3.
3. New RTP header extension URN identifiers for SRCNAME, as defined in Section 4.4.

9. Security Considerations

The SDP item 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) [RFC7022] for more discussion.

A third party modification of the srcname labels either in the RTCP SDP 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.

10. References

10.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC3550] Schulzrinne, H., Casner, S., Frederick, R., and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications", STD 64, RFC 3550, July 2003.
- [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.

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

10.2. Informative References

- [I-D.even-mmusic-application-token]
Even, R., Lennox, J., and Q. Wu, "The Session Description Protocol (SDP) Application Token Attribute", draft-even-mmusic-application-token-01 (work in progress), September 2013.
- [I-D.lennox-raiarea-rtp-grouping-taxonomy]
Lennox, J., Gross, K., Nandakumar, S., Salgueiro, G., and B. Burman, "A Taxonomy of Grouping Semantics and Mechanisms for Real-Time Transport Protocol (RTP) Sources", draft-lennox-raiarea-rtp-grouping-taxonomy (work in progress), October 2013.
- [I-D.westerlund-avtcore-rtp-simulcast]
Westerlund, M. and B. Burman, "Using Simulcast in RTP sessions", draft-westerlund-avtcore-rtp-simulcast (work in progress), October 2013.
- [I-D.westerlund-avtext-sdes-hdr-ext]
Westerlund, M., Burman, B., and R. Even, "RTP Header Extension for RTCP Source Description Items", draft-westerlund-avtext-sdes-hdr-ext (work in progress), October 2013.
- [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.

- [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.
- [RFC5888] Camarillo, G. and H. Schulzrinne, "The Session Description Protocol (SDP) Grouping Framework", RFC 5888, June 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

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: April 24, 2014

M. Westerlund
B. Burman
Ericsson
R. Even
Huawei Technologies
M. Zanaty
Cisco Systems
October 21, 2013

RTP Header Extension for RTCP Source Description Items
draft-westerlund-avtext-sdes-hdr-ext-01

Abstract

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

Status of This Memo

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

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

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

This Internet-Draft will expire on April 24, 2014.

Copyright Notice

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

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents

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

Table of Contents

1. Introduction	2
2. Definitions	3
2.1. Requirements Language	3
2.2. Terminology	3
3. Motivation	3
4. Specification	4
4.1. SDES Item Header Extension	5
4.1.1. One-Byte Format	5
4.1.2. Two-Byte Format	5
4.2. Usage of the SDES Item Header Extension	6
4.2.1. One or Two Byte Headers	6
4.2.2. MTU and Packet Expansion	6
4.2.3. Transmission Considerations	7
4.2.4. Different Usages	8
4.2.5. SDES Items in RTCP	8
5. IANA Considerations	9
6. Security Considerations	10
7. Acknowledgements	10
8. References	10
8.1. Normative References	10
8.2. Informative References	11
Authors' Addresses	11

1. Introduction

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

First, some requirements language is defined. The following section motivates why this header extension is sometimes required or at least provides a significant improvement compared to waiting for regular RTCP packet transmissions of the information. This is followed by a specification of the header extension. Next, a sub-space of the

header-extension URN is defined to be used for existing and future SDES items, and the existing SDES items are registered.

2. Definitions

2.1. Requirements Language

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

2.2. Terminology

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

Media Source

Packet Stream

Media Encoder

Encoded Stream

Participant

3. Motivation

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

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

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

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

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

Some new SDES items are currently proposed, which can all benefit from timely delivery:

SRCNAME: This is a media source and encoding identifier to enable support for simulcast and improve some scalable encoding usages [I-D.westerlund-avtext-rtcp-sdes-srcname]. This SDES item could be used both for new sources and late joiners.

APPID: This SDES item provides an application specific identifier dynamically assigned to a particular packet stream. The intention is to provide a receiver with information about the current role of the received packet stream or its usage in an application [I-D.even-mmusic-application-token]. Thus a particular ID can be reassigned many times during the lifetime of an RTP session. This puts additional timing requirements, not only for new sources and late joiners, but also whenever the Application token is reassigned to another stream.

Based on the above, there appear to be good reasons why an RTP header extension for SDES items is worthwhile to pursue.

4. Specification

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

4.1. SDES Item Header Extension

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

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

4.1.1. One-Byte Format

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

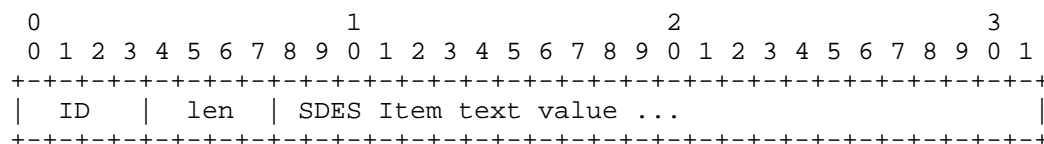


Figure 1

4.1.2. Two-Byte Format

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

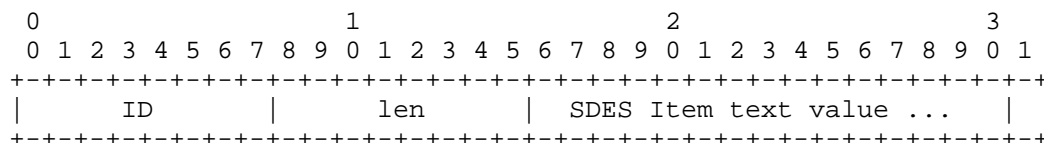


Figure 2

4.2. Usage of the SDES Item Header Extension

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

4.2.1. One or Two Byte Headers

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

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

4.2.2. MTU and Packet Expansion

The RTP packet size will clearly increase when they include the header extension. How much depends on which header extensions and their data parts. The SDES items can vary in size. There are also some use-cases which require transmitting multiple SDES items in the same packet to ensure that all relevant data reaches the receiver. An example of that is when you need both the CNAME, a SRCNAME and an APPID plus the rapid time synchronization extension from RFC 6051. Such a combination is quite likely to result in at least 16+3+1+8 bytes of data plus the headers, which will be another 8 bytes for one-byte headers, thus in total 36 bytes.

The packet expansion can cause an issue when it cannot be taken into account when producing the RTP payload. Thus an RTP payload that is created to meet a particular IP level Maximum Transmission Unit

(MTU), taking the addition of IP/UDP/RTP headers into account but excluding RTP header extensions suddenly exceeds the MTU, resulting in IP fragmentation. IP fragmentation is known to negatively impact the loss rate due to middleboxes unwilling or not capable of dealing with IP fragments.

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

4.2.3. Transmission Considerations

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

1. The probability for packet loss and burst loss determine how many repetitions of the header extensions will be required to reach a targeted delivery probability, and if burst loss is likely what dispersion would be needed to avoid getting multiple header extensions lost in a single burst.
2. How early the SDES item information is needed, from the first received RTP data or only after some set of packets are received, can guide if the header extension(s) should be in all of the first N packets or be included only once per set of packets, for example once per video frame.
3. The use of RTP level robustness mechanisms, such as RTP retransmission [RFC4588], or Forward Error Correction, e.g., [RFC5109] may treat packets differently from a robustness perspective, and SDES header extensions should be added to packets that get a treatment corresponding to the relative importance of receiving the information.

In summary, the number of header extension transmissions should be tailored to a desired probability of delivery taking the receiver population size into account. For the very basic case, N repetitions of the header extensions should be sufficient, but may not be optimal. N is selected so that probability of delivery of at least one out of the N reaches the target value when calculating $1 - P^N$, where P is the probability of packet loss. For point to point or

small receiver populations, it might also be possible to use feedback, such as RTCP, to determine when the information in the header extensions has likely reached all receivers.

4.2.4. Different Usages

4.2.4.1. New SSRC

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

4.2.4.2. Late Joiner

In a multi-party RTP session where one or a small number of receivers join a session where the majority of receivers already have all necessary information, the use of header extensions to deliver relevant information should be tailored to reach the new receivers. The trigger to send header extensions can for example either be RTCP from new receiver(s) or an explicit request like the Rapid Resynchronisation Request defined in [RFC6051].

4.2.4.3. Information Change

In cases when the SDES item text value is changed and the new SDES information is tightly coupled to and thus needs to be synchronized with a related change in the RTP stream, use of a header extension is far superior to RTCP SDES. In this case it is equal or even more important with timely SDES information than in the case of new SSRCS (Section 4.2.4.1). Continued use of the old SDES information can lead to really undesired effects in the application. Application Token [I-D.even-mmusic-application-token] would be one such case. Thus, header extension transmission strategies with high probability of delivery should be chosen.

4.2.5. SDES Items in RTCP

As this RTP header extensions information, i.e. SDES Items can and will be sent also in RTCP it is worth some reflections on this interaction. There also exist the possibility to schedule a non-regular RTCP packet transmission containing important SDES items if one uses a RTP/AVPF based RTP profile. Depending on which mode ones RTCP feedback transmitter is working on extra RTCP packets may be sent as immediate or early packets, enabling more timely deliver of SDES information.

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

5. IANA Considerations

This IANA section firstly proposes to:

- o Reserve the SDES item RTP header extension defined in this document for use with current and future SDES items.
- o Register and assign the URN sub-space "urn:ietf:params:rtp-hdrext:sdes:" in the RTP Compact Header Extensions registry.

The reason to require registering a URN within that sub-space is that the name represent an RTCP Source Description item, where a specification is strongly recommended. The formal policy is maintained from the main space, i.e. Expert Review.

Secondly, it is proposed that all the current existing SDES items are registered for usage in the RTP Compact Header Extensions registry :

URN	SDES Item	Reference
=====	=====	=====
urn:ietf:params:rtp-hdrext:sdes:cname	CNAME	[RFC3550]
urn:ietf:params:rtp-hdrext:sdes:name	NAME	[RFC3550]
urn:ietf:params:rtp-hdrext:sdes:email	EMAIL	[RFC3550]
urn:ietf:params:rtp-hdrext:sdes:phone	PHONE	[RFC3550]
urn:ietf:params:rtp-hdrext:sdes:loc	LOC	[RFC3550]
urn:ietf:params:rtp-hdrext:sdes:tool	TOOL	[RFC3550]
urn:ietf:params:rtp-hdrext:sdes:note	NOTE	[RFC3550]
urn:ietf:params:rtp-hdrext:sdes:priv	PRIV	[RFC3550]
urn:ietf:params:rtp-hdrext:sdes:h323-caddr	H323-CADDR	[Vineet_Kumar]
urn:ietf:params:rtp-hdrext:sdes:apsi	APSI	[RFC6776]

6. Security Considerations

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

The above security concerns may have to be put in relation to needs of third party monitoring. In RTP sessions where any type of confidentiality protection is enabled, the SDES item header extensions SHOULD also be protected per default. This implies that to provide confidentiality, users of SRTP need to implement encrypted header extensions per [RFC6904]. Commonly, it is expected that the same security level is applied both RTCP packets carrying SDES items, as a RTP header extension containing a SDES item. If the security level is different it is important to consider the security properties as the worst in each aspect for the different configurations.

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

7. Acknowledgements

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

8. References

8.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC3550] Schulzrinne, H., Casner, S., Frederick, R., and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications", STD 64, RFC 3550, July 2003.

- [RFC5285] Singer, D. and H. Desineni, "A General Mechanism for RTP Header Extensions", RFC 5285, July 2008.
- [RFC6904] Lennox, J., "Encryption of Header Extensions in the Secure Real-time Transport Protocol (SRTP)", RFC 6904, April 2013.

8.2. Informative References

- [I-D.even-mmusic-application-token]
Even, R., Lennox, J., and Q. Wu, "The Session Description Protocol (SDP) Application Token Attribute", draft-even-mmusic-application-token-01 (work in progress), September 2013.
- [I-D.lennox-raiarea-rtp-grouping-taxonomy]
Lennox, J., Gross, K., Nandakumar, S., and G. Salgueiro, "A Taxonomy of Grouping Semantics and Mechanisms for Real-Time Transport Protocol (RTP) Sources", draft-lennox-raiarea-rtp-grouping-taxonomy-03 (work in progress), October 2013.
- [I-D.westerlund-avtext-rtcp-sdes-srcname]
Westerlund, M., "RTCP Source Description Item SRCNAME to Label Individual Media Sources", draft-westerlund-avtext-rtcp-sdes-srcname-03 (work in progress), October 2013.
- [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.
- [RFC6051] Perkins, C. and T. Schierl, "Rapid Synchronisation of RTP Flows", RFC 6051, November 2010.
- [RFC6776] Clark, A. and Q. Wu, "Measurement Identity and Information Reporting Using a Source Description (SDES) Item and an RTCP Extended Report (XR) Block", RFC 6776, October 2012.
- [RFC7022] Begen, A., Perkins, C., Wing, D., and E. Rescorla, "Guidelines for Choosing RTP Control Protocol (RTCP) Canonical Names (CNAMEs)", RFC 7022, September 2013.

Authors' Addresses

Magnus Westerlund
Ericsson
Farogatan 6
SE-164 80 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

Roni Even
Huawei Technologies
Tel Aviv
Israel

Email: roni.even@mail01.huawei.com

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

Email: mzanaty@cisco.com

AVTEXT Working Group
INTERNET-DRAFT
Intended Status: Standards Track
Expires: April 20, 2014

J. Xia
R. Huang
Huawei
L. Deng
China Mobile
October 17, 2013

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

Abstract

Content splicing is a process that replaces the content of a main multimedia stream with other multimedia content, and delivers the substitutive multimedia content to the receivers for a period of time. The RTP mixer is designed to handle RTP splicing in [RFC6828], but how the RTP mixer knows when to start and end the splicing is still unspecified.

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

Status of this Memo

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

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

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

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

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

Copyright and License Notice

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

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

Table of Contents

1	Introduction	3
1.1	Terminology	3
2	Overview of RTP Splicing Notification	4
3	Conveying Splicing Interval in RTP/RTCP extensions	5
3.1	RTP Header Extension	5
3.2	RTCP Splicing Notification Message	6
4	Reducing Splicing Latency	7
5	Failure Cases	7
6	SDP Signaling	8
7	Security Considerations	9
8	IANA Considerations	9
8.1	RTCP Control Packet Types	9
8.2	RTP Compact Header Extensions	10
9	Acknowledges	10
10	References	10
10.1	Normative References	10
10.2	Informative References	11
	Authors' Addresses	11

1 Introduction

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

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

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

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

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

1.1 Terminology

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

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

Splicing Interval:

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

2 Overview of RTP Splicing Notification

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

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

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

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

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

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

3 Conveying Splicing Interval in RTP/RTCP extensions

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

3.1 RTP Header Extension

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

One variant is defined for this header extension. It carries the 7 octets splicing-out NTP timestamp (lower 24-bit part of the Seconds of a NTP-format timestamp and the 32 bits of the Fraction of a NTP-format timestamp as defined in [RFC5905]), followed by the 8 octets splicing-in NTP timestamp (64-bit NTP-format timestamp as defined in [RFC5905]). The top 8 bits of the splicing-out NTP timestamp are referred from the top 8 bits of the splicing-in NTP timestamp, under the consumption that the splicing-out time is after the splicing-in time, and the splicing interval is less than 2^{25} seconds, this order allows full resolution for splicing-in NTP timestamp while keeping 4 octets alignment.

The format is shown in Figures 1.

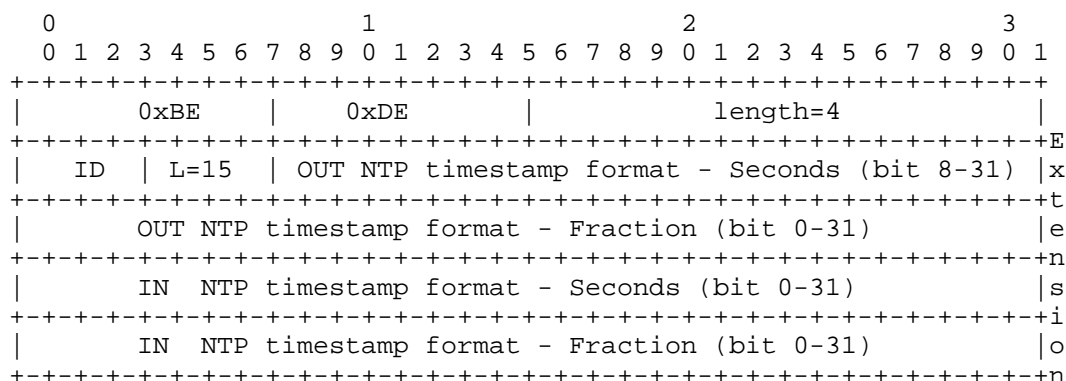


Figure 1: Sample hybrid NTP Encoding Using the One-Byte Header Format

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

remain RTP packets unmarked.

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

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

3.2 RTCP Splicing Notification Message

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

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

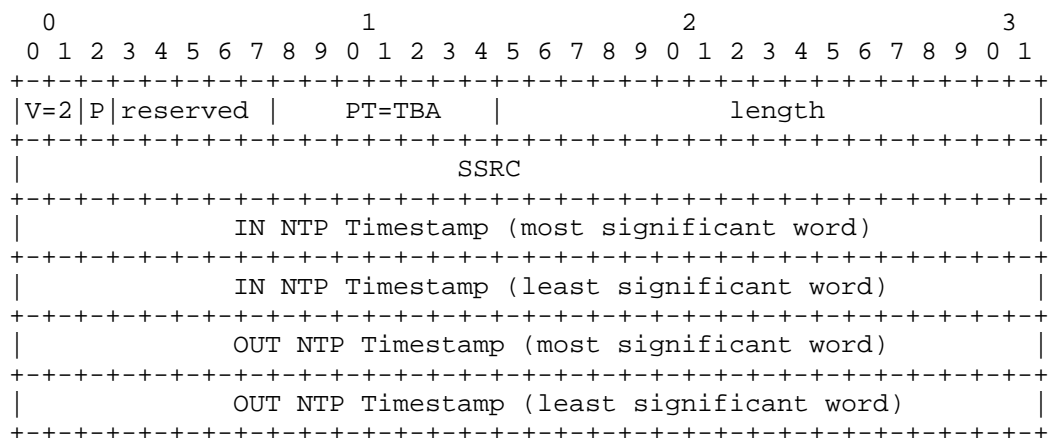


Figure 2: RTCP Splicing Notification Message

The RSI packet includes the following fields:

Length: 16 bits

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

SSRC: 32 bits

The SSRC of the Main RTP Sender.

Timestamp: 64 bits

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

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

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

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

4 Reducing Splicing Latency

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

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

5 Failure Cases

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

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

If the mixer does not get the Splicing Interval when the splicing starts, it will still output the main content to the downstream receivers and forward the RTCP RR packets sent from downstream receivers to the main RTP sender. In such case, the main RTP sender can learn the splicing failed.

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

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

6 SDP Signaling

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

This document also reuses the Flow Identification (FID) semantics defined in SDP Grouping Framework [RFC5888] to represent the relationship between the main RTP stream and the substitutive RTP stream.

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

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

```
a=mid: 1
m= video 30001 RTP/AVP 100
i=Substitutive RTP Stream
c=IN IP4 233.252.0.2/127
a=sendonly
a=mid: 2
```

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

The mixer receiving the SDP message above receives one MPEG2-TS stream (payload 100) from the main RTP sender (with multicast destination address of 233.252.0.1) on port 30000, and/or receives another MPEG2-TS stream from the substitutive RTP sender (with multicast destination address of 233.252.0.2) on port 30001. But at a particular point in time, the mixer only selects one stream and output the content from the chosen stream to the downstream receivers.

7 Security Considerations

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

The RTP header extension defined in Section 4.1 include two NTP-format timestamps. In the Secure Real-time Transport Protocol (SRTP)[RFC3711], RTP header extensions are authenticated but not encrypted. A malicious endpoint could choose to set the values in this header extension falsely, so as to falsely claim the splicing time.

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

8 IANA Considerations

8.1 RTCP Control Packet Types

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

Name: SNM

Long name: Splicing Notification Message

Value: TBA

Reference: This document

8.2 RTP Compact Header Extensions

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

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

Description: Splicing Interval

Contact: Jinwei Xia <xiajinwei@huawei.com>

Reference: This document

9 Acknowledges

TBD

10 References

10.1 Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC3550] Schulzrinne, H., Casner, S., Frederick, R., and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications", STD 64, RFC 3550, July 2003.
- [RFC4566] Handley, M., Jacobson, V., and C. Perkins, "SDP: Session Description Protocol", RFC 4566, July 2006.
- [RFC5285] Singer, D. and H. Desineni, "A General Mechanism for RTP Header Extensions", RFC 5285, July 2008.
- [RFC5888] Camarillo, G. and H. Schulzrinne, "The Session Description Protocol (SDP) Grouping Framework", RFC 5888, June 2010.
- [RFC5905] Mills, D., Martin, J., Ed., Burbank, J., and W. Kasch,

"Network Time Protocol Version 4: Protocol and Algorithms Specification", RFC 5905, June 2010.

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

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

10.2 Informative References

[RFC3711] Baugher, M., McGrew, D., Naslund, M., Carrara, E., and K. Norrman, "The Secure Real-time Transport Protocol (SRTP)", RFC 3711, March 2004.

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

[RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 5226, May 2008.

[RFC5506] Johansson, I. and M. Westerlund, "Support for Reduced-Size Real-Time Transport Control Protocol (RTCP): Opportunities and Consequences", RFC 5506, April 2009.

[RFC6285] Ver Steeg, B., Begen, A., Van Caenegem, T., and Z. Vax, "Unicast-Based Rapid Acquisition of Multicast RTP Sessions", RFC 6285, June 2011.

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

[SCTE35] Society of Cable Telecommunications Engineers (SCTE), "Digital Program Insertion Cueing Message for Cable", 2011.

Authors' Addresses

Jinwei Xia
Huawei

Email: xiajinwei@huawei.com

Rachel Huang

INTERNET DRAFT

<Document Title>

<Issue Date>

Huawei

Email: rachel.huang@huawei.com

Lingli Deng
China Mobile

Email: denglingli@chinamobile.com

<>

Expires <Expiry Date>

[Page 12]