MMUSIC
Internet-Draft
Intended status: Standards Track
Expires: December 14, 2011

A. Begen
Y. Cai
H. Ou
Cisco

June 12, 2011

Duplication Grouping Semantics in the Session Description Protocol
draft-begen-mmusic-redundancy-grouping-01

Abstract

Packet loss is undesirable for real-time multimedia sessions, but it is not avoidable due to congestion or other unplanned network outages. This is especially the case for IP multicast networks. 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. This document defines the semantics for grouping redundant streams in the Session Description Protocol (SDP). The semantics defined in this document are to be used with the SDP Grouping Framework [RFC5888]. SSRC-level (Synchronization Source) grouping semantics are also defined in this document for RTP streams using SSRC multiplexing.

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 December 14, 2011.

Copyright Notice

Table of Contents

1. Introduction

RTP [RFC3550] transport is widely used today for delivering real-time multimedia streams. Most of the applications also rely on IP multicast to reach many receivers efficiently.

While the combination proves successful, there does exist a weakness. As [RFC2354] noted, packet loss is not avoidable. This 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 does not provide any means for detecting loss and retransmitting packets, it leaves up to the RTP or the applications to detect and recover from the loss. For retransmission-based recovery, one example is described in [RFC4588].

In this document, we describe a technique that involves transmitting redundant streams to overcome packet loss. Variations of this technique have already been implemented and deployed today [IC2011]. We also describe the semantics needed in the Session Description Protocol (SDP) [RFC4566] to support this technique.

A work-in-progress draft specification [I-D.singh-avtcore-mprtp] proposes changes to the RTP protocol so that a single RTP session can benefit from using multiple paths between two endpoints (to increase the aggregated throughput and improve reliability). While we also discuss spatial diversity in this document, we use diverse paths solely for sending redundant streams. For our purposes, we do not require changes in the RTP protocol.

2. 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

Dual streaming refers to a technique that involves transmitting two redundant (often RTP) streams of the same content, with each stream itself capable of supporting the playback when there is no packet loss. Therefore, adding an additional 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 the technique and specification described by this document can easily be extended to support cases when more than two streams are desired. But triple, quadruple, or more, streaming is rarely used in practice.

3.1. (Routing-Plane) Identical Streams

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

- IP Source Address
- IP Destination Address

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

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

In such a scenario (where the RTP streams are routing-plane identical and share the same UDP destination port), there will be only one "m" line in the SDP description regardless of how many redundant streams are generated. Thus, the SDP Grouping Framework [RFC5888] cannot be used to indicate the grouping for the redundant streams. Instead, the ’ssrc-group’ attribute [RFC5576] with new semantics has to be used to describe the redundancy relation (See Section 4.2).

If the two routing-plane identical RTP streams were sent to different UDP destination ports, there would have been two "m" lines in the SDP description and in this case, the ’group’ attribute [RFC5888] with new semantics would have to be used to describe the redundancy relation (See Section 4.1).

3.2. Using Separate Source Interfaces

An RTP source might have multiple network interfaces associated with it and it can send two redundant streams from two separate interfaces. Such streams can be routed over diverse or identical paths depending on the routing algorithm inside the network. At the receiving end, the node responsible for duplicate suppression can look into various RTP related fields 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. The receiving host has to join each SSM session separately via Internet Group Management Protocol (IGMP) version 3 [RFC3376] or the Multicast Listener Discovery Protocol (MLD) version 2 [RFC3810]. Note that despite being transmitted in separate SSM sessions, there is still only one RTP session and the redundant streams still have to use unique SSRC identifiers.

3.3. Using Separate Destination Addresses and/or Ports

An RTP source might send the redundant streams to separate IP destination addresses and/or UDP ports. In this case, there will be multiple "m" lines in the SDP description and the ’group’ attribute [RFC5888] with new semantics will be used to describe the redundancy relation.

3.4. Dual Streaming over a Single Path or Multiple Paths

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

1. When two routing-plane identical streams are used, the two streams will have identical IP headers. This makes it impractical to forward the packets onto different paths. In order to minimize packet loss, the packets belonging to one stream are often interleaved with packets belonging to the other, and with a delay, so that if there is a packet loss, such a delay would allow the same packet from the other stream to reach the receiver because the chances that the same packet is lost in transit again is often small. This is what is also known as Time-shifted Redundancy, Temporal Redundancy or simply Delayed Duplication [I-D.begen-mmusic-temporal-interleaving] [IC2011]. This approach can be used with all three types of dual streaming described in Section 3.1, Section 3.2 and Section 3.3.

2. If the two streams have different IP headers, an additional opportunity arises in that one is able to build a network, with physically diverse paths, to deliver the two streams concurrently to the intended receivers. This reduces the delay when packet loss occurs and needs to be recovered. Additionally, it also further reduces chances for packet loss. An unrecoverable loss happens only when two network failures happen in such a way that the same packet is affected on both paths. This is referred to as Spatial Diversity or Spatial Redundancy [IC2011]. The techniques used to build diverse paths are beyond the scope of this document.

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

To summarize, dual streaming allows an application and a network to work together to provide a near zero-loss transport with a bounded or minimum delay. The additional advantage includes a predictable bandwidth overhead that is proportional to the minimum bandwidth needed for the multimedia session, but independent of the number of receivers experiencing a packet loss and requesting a retransmission. For a survey and comparison of similar approaches, refer to [IC2011].

4. Duplication Grouping

4.1. "DUP" Grouping Semantics

Each "a=group" line is used to indicate an association relationship between the redundant streams. The streams included in one "a=group" line are called a Duplication Group.

Using the framework in [RFC5888], this document defines "DUP" as the grouping semantics for redundant streams.

The "a=group:DUP" semantics MUST be used to group the redundant streams except when the streams are specified in the same media description, i.e., in the same "m" line (See Section 4.2).

The SSRC identifiers for the RTP streams that are carried in the same RTP session MUST be unique per [RFC3550]. Thus, each redundant RTP stream MUST have its own unique SSRC identifier. This way, dual streaming does not break RTCP reporting. When the redundant streams are described in separate "m" lines and the ’group’ attribute is used to describe the redundancy relation, the SSRCs for each redundant

stream MUST be announced in the SDP description using the 'ssrc' attribute [RFC5576].

4.2. DUP Grouping for SSRC-Multiplexed RTP Streams

[RFC5576] defines an SDP media-level attribute, called 'ssrc-group', for grouping the RTP streams that are SSRC multiplexed and carried in the same RTP session. The grouping is based on the SSRC identifiers. Since SSRC-multiplexed RTP streams are defined in the same "m" line, the 'group' attribute cannot be used.

This section specifies how duplication is used with SSRC-multiplexed streams using the 'ssrc-group' attribute [RFC5576].

The semantics of "DUP" for the 'ssrc-group' attribute are the same as the one defined for the 'group' attribute except that the SSRC identifiers are used to designate the duplication grouping associations: a=ssrc-group:DUP *(SP ssrc-id) [RFC5576].

4.3. SDP Offer/Answer Model Considerations

When offering duplication grouping using SDP in an Offer/Answer model [RFC3264], the following considerations apply.

A node that is receiving an offer from a sender may or may not understand line grouping. It is also possible that the node understands line grouping but it does not understand the "DUP" semantics. From the viewpoint of the sender of the offer, these cases are indistinguishable.

When a node is offered a session with the "DUP" grouping semantics but it does not support line grouping or the duplication grouping semantics, as per [RFC5888], the node responds to the offer either (1) with an answer that ignores the grouping attribute or (2) with a refusal to the request (e.g., 488 Not Acceptable Here or 606 Not Acceptable in SIP).

In the first case, the original sender of the offer must send a new offer without any duplication grouping. In the second case, if the sender of the offer still wishes to establish the session, it should retry the request with an offer without the duplication grouping. This behavior is specified in [RFC5888].

5. SDP Examples

5.1. Separate Source Interfaces

In this example, the redundant streams use the same IP destination address (232.252.0.1) but they are sourced from different addresses (198.51.100.1 and 198.51.100.2). Thus, the receiving host needs to join both SSM sessions separately.

```
v=0
o=ali 1122334455 1122334466 IN IP4 dup.example.com
s=DUP Grouping Semantics
t=0 0
m=video 30000 RTP/AVP 100 101
c=IN IP4 232.252.0.1/127
a=source-filter:incl IN IP4 232.252.0.1 198.51.100.1 198.51.100.2
a=rtpmap:100 MP2T/90000
a=rtpmap:101 MP2T/90000
a=ssrc:1000 cname:ch1@example.com
a=ssrc:1010 cname:ch1@example.com
a=ssrc-group:DUP 1000 1010
a=mid:Group1
```

Note that in actual use, SSRC values, which are random 32-bit numbers, can be much larger than the ones shown in this example. Also, note that before receiving an RTP packet for each stream, the receiver cannot know which SSRC identifier is associated with which payload type.

5.2. Separate Destination Addresses

In this example, the redundant streams have different IP destination addresses. The example shows the same UDP port number and IP source addresses, but either or both could have been different for the two streams.

```
v=0
o=ali 1122334455 1122334466 IN IP4 dup.example.com
s=DUP Grouping Semantics
t=0 0
a=group:DUP S1a S1b
m=video 30000 RTP/AVP 100
c=IN IP4 233.252.0.1/127
a=source-filter:incl IN IP4 233.252.0.1 198.51.100.1
a=rtpmap:100 MP2T/90000
a=ssrc:1000 cname:ch1@example.com
a=mid:S1a
m=video 30000 RTP/AVP 101
c=IN IP4 233.252.0.2/127
a=source-filter:incl IN IP4 233.252.0.2 198.51.100.1
a=rtpmap:101 MP2T/90000
a=ssrc:1010 cname:ch1@example.com
a=mid:S1b
```

Editor’s note: What if there are multiple streams per "m" line but grouping has to take place across "m" lines? Could we implicitly use the CNAMEs to infer the redundancy relation (note that ’ssrc-group’ attribute is media-level only)?

5.3. Delayed Duplication

In this example, the redundant streams have the same IP source and destination addresses but different UDP port numbers. Due to the same source and destination addresses, the packets in both streams will be routed over the same path. To provide resiliency against packet loss, the duplicate of an original packet is transmitted 50 ms later as indicated by the ’duplication-delay’ attribute (defined in [I-D.begen-mmusic-temporal-interleaving]).

```
v=0
o=ali 1122334455 1122334466 IN IP4 dup.example.com
s=DUP Grouping Semantics
t=0 0
a=group:DUP S1a S1b
a=duplication-delay:50
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:ch1@example.com
a=mid:S1a
m=video 40000 RTP/AVP 101
c=IN IP4 233.252.0.1/127
a=source-filter:incl IN IP4 233.252.0.1 198.51.100.1
a=rtpmap:101 MP2T/90000
a=ssrc:1010 cname:ch1@example.com
a=mid:S1b
```

6. Performance Evaluation and Reporting

Each duplicated stream has a separate (unique) SSRC identifier. Thus, individual RTCP receiver reports can be sent as usual for each of them from the receiving node that suppresses the duplicate packets. This way, the sender can be notified about the delivery performance of the individual streams.

Editor’s note: The receiving node can also produce a new XR report to report on the (loss/delay/jitter/etc.) performance of the output stream after the stream merging process.

7. Security Considerations

There is a weak threat for the receiver that the duplication grouping can be modified to indicate relationships that do not exist. Such attacks might result in failure of the duplication mechanisms, and/or mishandling of the media streams by the receivers.

In order to avoid attacks of this sort, the SDP description needs to be integrity protected and provided with source authentication. This can, for example, be achieved on an end-to-end basis using S/MIME [RFC5652] [RFC5751] when the SDP is used in a signaling packet using MIME types (application/sdp). Alternatively, HTTPS [RFC2818] or the authentication method in the Session Announcement Protocol (SAP) [RFC2974] could be used as well.

8. IANA Considerations

This document registers the following semantics with IANA in Semantics for the 'group' SDP Attribute under SDP Parameters:

Note to the RFC Editor: In the following registrations, please replace "XXXX" with the number of this document prior to publication as an RFC.

Semantics	Token	Reference
Duplication	DUP	[RFCXXXX]

This document also registers the following semantics with IANA in Semantics for the 'ssrc-group' SDP Attribute under SDP Parameters:

Token	Semantics	Reference
DUP	Duplication	[RFCXXXX]

9. Acknowledgments

The authors would like to thank Colin Perkins, Bill Ver Steeg, Dave Oran and Toerless Eckert for their inputs and suggestions.

10. References

10.1. Normative References

[I-D.begen-mmusic-temporal-interleaving] Begen, A., Cai, Y., and H. Ou, "Temporal Interleaving Attribute in the Session Description Protocol", draft-begen-mmusic-temporal-interleaving-01 (work in progress), March 2011.

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

[RFC3264] Rosenberg, J. and H. Schulzrinne, "An Offer/Answer Model with Session Description Protocol (SDP)", RFC 3264, June 2002.

[RFC3376] Cain, B., Deering, S., Kouvelas, I., Fenner, B., and A. Thyagarajan, "Internet Group Management Protocol, Version

3", RFC 3376, October 2002.

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

[RFC3810] Vida, R. and L. Costa, "Multicast Listener Discovery Version 2 (MLDv2) for IPv6", RFC 3810, June 2004.

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

[RFC5576] Lennox, J., Ott, J., and T. Schierl, "Source-Specific Media Attributes in the Session Description Protocol (SDP)", RFC 5576, June 2009.

[RFC5888] Camarillo, G. and H. Schulzrinne, "The Session Description Protocol (SDP) Grouping Framework", RFC 5888, June 2010.

10.2. Informative References

[I-D.singh-avtcore-mprtp] Singh, V., Karkkainen, T., Ott, J., Ahsan, S., and L. Eggert, "Multipath RTP (MPRTP)", draft-singh-avtcore-mprtp-01 (work in progress), March 2011.

[IC2011] Evans, J., Begen, A., Greengrass, J., and C. Filsfils, "Towards Lossless Video Transport (in submission to IEEE Internet Computing)", 2011.

[RFC2354] Perkins, C. and O. Hodson, "Options for Repair of Streaming Media", RFC 2354, June 1998.

[RFC2818] Rescorla, E., "HTTP Over TLS", RFC 2818, May 2000.

[RFC2974] Handley, M., Perkins, C., and E. Whelan, "Session Announcement Protocol", RFC 2974, October 2000.

[RFC4588] Rey, J., Leon, D., Miyazaki, A., Varsa, V., and R. Hakenberg, "RTP Retransmission Payload Format", RFC 4588, July 2006.

[RFC5652] Housley, R., "Cryptographic Message Syntax (CMS)", STD 70, RFC 5652, September 2009.

[RFC5751] Ramsdell, B. and S. Turner, "Secure/Multipurpose Internet Mail Extensions (S/MIME) Version 3.2 Message

Specification", RFC 5751, January 2010.

Authors' Addresses

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

Email: abegen@cisco.com

Yiqun Cai
Cisco
170 W. Tasman Dr.
San Jose, CA 95134
USA

Email: ycai@cisco.com

Heidi Ou
Cisco
170 W. Tasman Dr.
San Jose, CA 95134
USA

Email: hou@cisco.com

MMUSIC
Internet-Draft
Intended status: Standards Track
Expires: December 14, 2011

A. Begen
Y. Cai
H. Ou
Cisco

June 12, 2011

Delayed Duplication Attribute in the Session Description Protocol
draft-begen-mmusic-temporal-interleaving-02

Abstract

A straightforward approach to provide protection against network outages (or packet losses) with a longest duration of T time units is to simply duplicate the original packets and send each copy separated in time by at least T time units. This approach is commonly referred to as Time-shifted Redundancy, Temporal Redundancy or simply Delayed Duplication. This document defines an attribute to indicate the presence of temporally redundant media streams and the duplication delay in the Session Description Protocol.

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 December 14, 2011.

Copyright Notice

Table of Contents

1. Introduction

Consider that a media sender transmits an original source packet and transmits its duplicate after a certain delay following the original transmission. If a network outage hits the original transmission, the expectation is that the second transmission arrives at the receiver. Alternatively, the second transmission may be hit by an outage or gets dropped, and the original transmission completes successfully. On the receiver side, both transmissions can also arrive and in that case, the receiver (or the node that does the duplicate suppression) needs to identify the duplicate packets and discard them appropriately, producing a duplicate-free stream.

Delayed duplication can be used in a variety of multimedia applications where there is sufficient bandwidth for the duplicated traffic and the application can tolerate the introduced delay. One particular use case is to improve the reliability of real-time video feeds inside a core IP network [IC2011]. Compared to other popular redundancy approaches such as Forward Error Correction (FEC) [I-D.ietf-fecframe-framework] and redundant data encoding (e.g., [RFC2198]), delayed duplication is quite easy to implement since it does not require any special type of encoding or decoding.

For duplicate suppression, the receiver has to be able to identify the identical packets. This is straightforward for media packets that carry one or more unique identifiers such as the sequence number field in RTP header [RFC3550]. In non-RTP applications, the receiver can use unique sequence numbers if available or other alternative approaches to compare the incoming packets and discard the duplicate ones.

In this specification, we are not concerned about how the sender should determine the duplication delay. We are not concerned about how the receiver can suppress the duplicate packets and merge the incoming streams to produce a hopefully loss-free and duplication-free output stream (called stream merging), either. These considerations are out of the scope for this specification. Rather, we introduce a new attribute for the Session Description Protocol (SDP) [RFC4566] that indicates that the media stream is to be duplicated and sent two or more times, and also indicates the relative delay for each additional duplication.

In practice, more than two redundant streams are unlikely to be used since the additional delay and increased overhead are not easily justified. However, we define the new attribute in a general way so that it could be used with more than two redundant streams if needed. While the primary focus in this specification is the RTP-based transport, the new attribute is applicable to both RTP and non-RTP

streams.

2. 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. The 'duplication-delay' Attribute

The following ABNF [RFC5234] syntax formally describes the 'duplication-delay' attribute:

```
delaying-attribute     = "a=duplication-delay:" periods CRLF
periods                = period *( ":" period)
period                 = 1*DIGIT ; in milliseconds
```

Figure 1: ABNF syntax for the 'interleaving-period' attribute

The 'duplication-delay' attribute is defined as both a media-level and session-level attribute. It specifies the relative delay for each duplication in milliseconds (ms). If used as a media-level attribute, it MUST be used with the 'ssrc-group' attribute and "DUP" grouping semantics as defined in [I-D.begen-mmusic-redundancy-grouping]. If used as a session-level attribute, it MUST be used with 'group' attribute and "DUP" grouping semantics as defined in [I-D.begen-mmusic-redundancy-grouping].

4. SDP Examples

In the first example below, the multicast stream is duplicated with a duplication delay of 100 ms. The streams have Synchronization Sources (SSRC) of 1000 and 1010, and they are grouped together using the 'ssrc-group' attribute defined in [RFC5576]. The "DUP" grouping semantics is defined in [I-D.begen-mmusic-redundancy-grouping]. The reason for using explicit grouping is that not all the media streams in the same "m" line are necessarily duplicates of each other.

```
v=0
o=ali 1122334455 1122334466 IN IP4 dup.example.com
s=Delayed Duplication
t=0 0
m=video 30000 RTP/AVP 100 101
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=rtpmap:101 MP2T/90000
a=ssrc:1000 cname:ch1@example.com
a=ssrc:1010 cname:ch1@example.com
a=ssrc-group:DUP 1000 1010
a=duplication-delay:100
a=mid:Group1
```

Note that in actual use, SSRC values, which are random 32-bit numbers, could be much larger than the ones shown in this example. Also, note that before receiving an RTP packet for each stream, the receiver cannot know which SSRC identifier is associated with which payload type.

In the second example below, the multicast stream is duplicated twice. 50 ms after the original transmission, the first duplicate is transmitted and 100 ms after that, the second duplicate is transmitted. In other words, the same packet is transmitted three times over a period of 150 ms.

```
v=0
o=ali 1122334455 1122334466 IN IP4 dup.example.com
s=Delayed Duplication
t=0 0
m=video 30000 RTP/AVP 100 101 102
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=rtpmap:101 MP2T/90000
a=rtpmap:102 MP2T/90000
a=ssrc:1000 cname:ch1@example.com
a=ssrc:1010 cname:ch1@example.com
a=ssrc:1020 cname:ch1@example.com
a=ssrc-group:DUP 1000 1010 1020
a=duplication-delay:50:100
a=mid:Group1
```

In the third example below, the multicast UDP stream is duplicated

with a duplication delay of 50 ms. Both redundant streams are sent in the same source-specific multicast (SSM) session but they are sent to different ports. The "DUP" grouping semantics [I-D.begen-mmusic-redundancy-grouping] is used to describe the redundany relation.

```
v=0
o=ali 1122334455 1122334466 IN IP4 dup.example.com
s=Delayed Duplication
t=0 0
a=group:DUP S1a S1b
a=duplication-delay:50
m=audio 30000 udp mp4
c=IN IP4 233.252.0.1/127
a=source-filter:incl IN IP4 233.252.0.1 198.51.100.1
a=mid:S1a
m=audio 40000 udp mp4
c=IN IP4 233.252.0.2/127
a=source-filter:incl IN IP4 233.252.0.1 198.51.100.1
a=mid:S1b
```

Editor’s note: Verify the validity of the SDP description above.

5. Performance Evaluation and Reporting

Each duplicated stream has a separate (unique) SSRC identifier [I-D.begen-mmusic-redundancy-grouping]. Thus, individual RTCP receiver reports can be sent as usual for each of them from the receiving node that suppresses the duplicate packets. This way, the sender can be notified about the delivery performance of the individual streams.

Editor’s note: The receiving node can also produce a new XR report to report on the (loss/delay/jitter/etc.) performance of the output stream after the stream merging process.

6. Security Considerations

The ’duplication-delay’ attribute is not believed to introduce any significant security risk to multimedia applications. A malevolent third party could use this attribute to misguide the receiver(s) about the duplication delays and/or the number of redundant streams. For example, if the malevolent third party increases the value of the duplication delay, the receiver(s) will unnecessarily incur a longer delay since they will have to wait for the entire period. Or, if the

duplication delay is reduced by the malevolent third party, the receiver(s) might not wait long enough for the duplicated transmission and incur unnecessary packet losses. However, these require intercepting and rewriting the packets carrying the SDP description; and if an interceptor can do that, many more attacks are also possible.

In order to avoid attacks of this sort, the SDP description needs to be integrity protected and provided with source authentication. This can, for example, be achieved on an end-to-end basis using S/MIME [RFC5652] [RFC5751] when SDP is used in a signaling packet using MIME types (application/sdp). Alternatively, HTTPS [RFC2818] or the authentication method in the Session Announcement Protocol (SAP) [RFC2974] could be used as well.

7. IANA Considerations

The following contact information shall be used for all registrations in this document:

Ali Begen
abegen@cisco.com

Note to the RFC Editor: In the following, replace "XXXX" with the number of this document prior to publication as an RFC.

7.1. Registration of SDP Attributes

This document registers a new attribute name in SDP.

SDP Attribute ("att-field"):	
Attribute name:	duplication-delay
Long form:	Duplication delay for temporally redundant streams
Type of name:	att-field
Type of attribute:	Media or session level
Subject to charset:	No
Purpose:	Specifies the relative duplication delay(s) for redundant stream(s)
Reference:	[RFCXXXX]
Values:	See [RFCXXXX]

8. Acknowledgements

Authors would like to thank Colin Perkins for his suggestions and review.

9. References

9.1. Normative References

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

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

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

[I-D.begen-mmusic-redundancy-grouping] Begen, A., Cai, Y., and H. Ou, "Redundancy Grouping Semantics in the Session Description Protocol", draft-begen-mmusic-redundancy-grouping-00 (work in progress), March 2011.

9.2. Informative References

[I-D.ietf-fecframe-framework] Watson, M., Begen, A., and V. Roca, "Forward Error Correction (FEC) Framework", draft-ietf-fecframe-framework-15 (work in progress), June 2011.

[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.

[IC2011] Evans, J., Begen, A., Greengrass, J., and C. Filsfils, "Towards Lossless Video Transport (in submission to IEEE

Internet Computing)", 2011.

[RFC5652] Housley, R., "Cryptographic Message Syntax (CMS)", STD 70, RFC 5652, September 2009.

[RFC5751] Ramsdell, B. and S. Turner, "Secure/Multipurpose Internet Mail Extensions (S/MIME) Version 3.2 Message Specification", RFC 5751, January 2010.

[RFC2818] Rescorla, E., "HTTP Over TLS", RFC 2818, May 2000.

[RFC2974] Handley, M., Perkins, C., and E. Whelan, "Session Announcement Protocol", RFC 2974, October 2000.

Authors' Addresses

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

Email: abegen@cisco.com

Yiqun Cai
Cisco
170 W. Tasman Dr.
San Jose, CA 95134
USA

Email: ycai@cisco.com

Heidi Ou
Cisco
170 W. Tasman Dr.
San Jose, CA 95134
USA

Email: hou@cisco.com

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: August 19, 2011

Q. Wu
F. Xia
R. Even
Huawei

February 15, 2011

RTCP Extension for Feedback Suppression Indication
draft-ietf-avtcore-feedback-supression-rtp-00

Abstract

In a large RTP session using the RTCP feedback mechanism defined in RFC 4585, a media source or middlebox may experience transient overload if some event causes a large number of receivers to send feedback at once. This feedback implosion can be mitigated if the device suffering from overload can send a third party loss report message to the receivers to inhibit further feedback. This memo defines RTCP extensions for third party loss report, to suppress NACK and FIR feedback requests. It also defines associated SDP signalling.

Table of Contents

1. Introduction

RTCP feedback messages [RFC4585] allow the receivers in an RTP session to report events and ask for action from the media source (or a delegated feedback target defined in SSM [RFC5760]). There are cases where multiple receivers may initiate the same, or an equivalent message towards the same media source. When the receiver count is large, this behavior may cause transient overload of the media source, the network or both. This is known as a "feedback storm" or a "NACK storm". One common cause of such a feedback storm is receivers utilizing RTP retransmission [RFC4588] as a packet loss recovery technique based, sending feedback using RTCP NACK messages [RFC4585] without proper dithering of the retransmission requests.

Another use case involves video Fast Update requests. A storm of these feedback messages can occur in conversational multimedia scenarios like Topo-Video-switch-MCU [RFC5117]. In this scenario, packet loss may happen on an upstream link of an intermediate network element such as a Multipoint Control Unit(MCU). Poorly designed receivers that blindly issue fast update requests (i.e., Full Intra Request (FIR) described in [RFC5104]), can cause an implosion of FIR requests from receivers to the same media source.

RTCP feedback storms may cause short term overload and, and in extreme cases to pose a possible risk of increasing network congestion on the control channel (e.g. RTCP feedback), the data channel, or both. It is therefore desirable to provide a way of suppressing unneeded feedback.

One approach to this, suggested in [DVB-IPTV], involves sending a NACK message to the other clients (or receiver) in the same group as the sender of NACK. However sending multicast NACK to the group can not prevent large amount of unicast NACK addressed to the same media source or middlebox, for example when the NACK is used as a retransmission request [RFC4588]. Also NACK is defined as a receiver report sent from a receiver observing a packet loss, therefore it only inform others that sender of NACK detected loss while the case the sender of the feedback has received reports that the indicated packets were lost is not covered. This document specifies a new message for this function. It further is more precise in the intended uses and less likely to be confusing to receivers. It tells receivers explicitly that feedback for a particular packet or frame loss is not needed for a period of time and can provide an early indication before the receiver reacts to the loss and invokes its packet loss repair machinery.

2. Terminology

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

3. Protocol Overview

This document extends the RTCP feedback messages defined in the Audio-Visual Profile with Feedback (AVPF) and define the Third Party Loss Report message. The Third Party Loss Report message informs the receiver in the downstream path of the middlebox that the sender of the Third Party Loss Report has received reports that the indicated packets were lost and asks a receiver to not send feedback messages for particular packets (indicated by their RTP sequence numbers) independent of whether the receiver detected the packet loss or detected a need for a decoder refresh point.

In order to observe packet loss before the receivers perceive it, one or more intermediate nodes may be placed between the media source and the receivers. These intermediates are variously referred to as Distribution servers, MCUs, RTP translator, or RTP mixers, depending on the precise use case. These intermediaries monitor for packet loss upstream of themselves by checking RTP sequence numbers, just as receivers do. Upon observing (or suspecting) an upstream loss, the intermediary may send Loss Party Loss Report message towards the receivers as defined in this specification.

These intermediate nodes need to take into account such factors as the tolerable application delay, the network dynamics, and the media type. When the packet loss is detected upstream of the intermediary and additional latency is tolerable, the intermediate node may itself send a feedback message asking for the suspected lost packet or ask for the correct decoder refresh point. Because it has already provided the necessary feedback toward the source, the intermediate node can be reasonably certain that it will help the situation by sending a Third Party Loss Report message to all the relevant receivers, thereby indicating to the receivers that they should not transmit feedback messages for a period of time.

Alternatively, the media source may directly monitor the amount of feedback requests it receives, and send Third Party Loss Report messages to the receivers.

When a receiver gets such a Third Party Loss Report message, it should refrain from sending a feedback request (e.g., NACK or FIR) for the missing packets reported in the message for a period of time.

A receiver may still have sent a Feedback message according to the AVPF scheduling algorithm of [RFC4585]before receiving a Third Party Loss Report message, but further feedback messages for those sequence numbers will be suppressed by this technique for a period of time. Nodes that do not understand the Third Party Loss Report message will ignore it, and might therefore still send feedback according to the AVPF scheduling algorithm of [RFC4585]. The media source or intermediate nodes cannot assume that the use of a Third Party Loss Report message actually reduces the amount of feedback it receives.

RTCP Third Party Loss Report follows the similar format of message type as RTCP NACK. But unlike RTCP NACK, the third party loss report is defined as an indication that the sender of the feedback has received reports that the indicated packets were lost and conveys the packet receipt/loss events at the sequence number level from the middlebox to the receivers in the downstream path of middlebox while NACK [RFC4585]just indicates that the sender of the NACK observed that these packets were lost. The Third Party Loss Report message can also be generated by RTP middleboxs that has not seen the actual packet loss and sent to the corresponding receivers. Intermediaries downstream of an intermediary detecting loss obviously SHOULD NOT initiate their own additional Third Party Loss Report messages for the same packet sequence numbers. They may either simply forward the Third Party Loss Report message received from upstream, or replace it with a Third Party Loss Report message that reflects the loss pattern they have themselves seen. The Third Party Loss Report does not have the retransmission request [rfc4588] semantics.

Since Third Party Loss Report interacts strongly with repair timing, it has to work together with feedback to not adversely impact the repair of lost source packets. One example is the middle box gets the retransmitted packet by sending a NACK upstream and sent it downstream. This retransmitted packet was lost on the downstream link. In order to deal with this, the downstream receiver can start a timeout in which it expected to get a retransmission packet. When this timeout expires and there is no retransmitted packet or a new third party loss report message, it can take its normal behavior as if there is no current retransmission suppression. In some cases where the loss was detected and repair initiated much closer to the source, the delay for the receiver to recover from packet loss can be reduced through the combination of intermediary feedback to the source and Third Party Loss Report downstream. In all (properly operating) cases, the risk of increasing network congestion is decreased.

4. RTCP Feedback Report Extension

This document registers two new RTCP Feedback messages for Third Party Loss Report. Applications that are employing one or more loss-repair methods MAY use Third Party Loss Report together with their existing loss-repair methods either for every packet they expect to receive, or for an application-specific subset of the RTP packets in a session. In other words, receivers MAY ignore Third Party Loss Report messages, but SHOULD react to them unless they have good reason to still send feedback messages despite having been requested to suppress them.

4.1. Transport Layer Feedback: Third-party Loss Report

This Third Party Loss Report message is an extension to the RTCP Transport Layer Feedback Report and identified by RTCP packet type value PT=RTPFB and FMT=TBD.

The FCI field MUST contain one or more entries of transport layer third party loss Early Indication (TLLEI). Each entry applies to a different media source, identified by its SSRC.

The Feedback Control Information (FCI) for TLLEI uses the similar format of message Types defined in the section 4.3.1.1 of [RFC5104]. The format is shown in Figure 1.

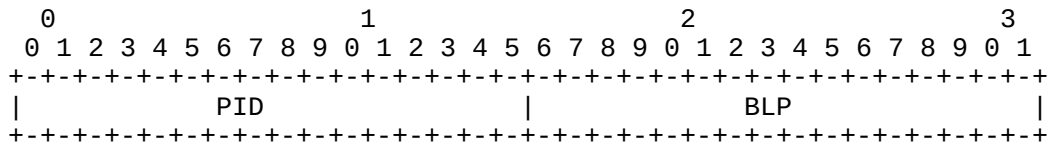


Figure 1: Message Format for the Third Party Loss Report

Packet ID (PID): 16 bits

The PID field is used to specify a lost packet. The PID field refers to the RTP sequence number of the lost packet.

bitmask of proceeding lost packets (BLP): 16 bits

The BLP allows for reporting losses of any of the 16 RTP packets immediately following the RTP packet indicated by the PID. The BLP's definition is identical to that given in [RFC4585].

4.2. Payload Specific Feedback: Third-party Loss Report

This message is an extension to the RTCP Payload Specific Feedback report and identified by RTCP packet type value PT=PSFB and FMT=TBD.

The FCI field MUST contain a Payload Specific Third Party Loss Early Indication (PSLEI) entry. Each entry applies to a different media source, identified by its SSRC.

The Feedback Control Information (FCI) for PSLEI uses the similar format of message Types defined in the section 4.3.1.1 of [RFC5104]. The format is shown in Figure 2.

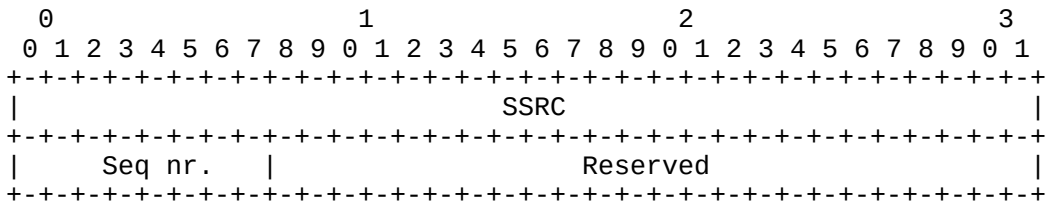


Figure 2: Message Format for the Third Party Loss Report

SSRC (32 bits):

The SSRC value of the media source that is requested to send a decoder refresh point.

Seq nr:8bits Command sequence number. The sequence number space is unique for each pairing of the SSRC of command source and the SSRC of the command target. The sequence number SHALL be increased by 1 modulo 256 for each new request.

Reserved: 24 bits

All bits SHALL be set to 0 by the media source and SHALL be ignored on reception.

5. SDP Signaling

A new feedback value "tplr" needs to be defined for the Third Party Loss Report message to be used with Session Description Protocol (SDP) [RFC4566] using the Augmented Backus-Naur Form (ABNF) [RFC4585].

The "tplr" feedback value SHOULD be used with parameters that

indicate the third party loss supported. In this document, we define two such parameter, namely:

- "tllei" denotes support of transport layer third party loss early indication (fsei).
- "pslei" denotes support of payload specific third party loss early indication.

In the ABNF for rtcp-fb-val defined in [RFC4585], there is a placeholder called rtcp-fb-id to define new feedback types. "tplr" is defined as a new feedback type in this document, and the ABNF for the parameters for tplr is defined here (please refer to section 4.2 of [RFC4585] for complete ABNF syntax).

```
      rtcp-fb-val        =/ "tplr" rtcp-fb-tplr-param
      rtcp-fb-tplr-param  = SP "tllei";transport layer third party loss early
 indication
                          / SP "pslei";payload specific third party loss earl
y indication
                          / SP token [SP byte-string]
                                    ; for future commands/indications
   byte-string = <as defined in section 4.2 of [RFC4585] >
```

Refer to Section 4.2 of [RFC4585] for a detailed description and the full syntax of the "rtcp-fb" attribute.

6. Example Use Cases

The operation of feedback suppression is similar for all types of RTP sessions and topologies [RFC5117], however the exact messages used and the scenarios in which suppression is employed differ for various use cases. The following sections outline the intended use cases of using Third Party Loss Report for feedback suppression and give an overview of the particular mechanisms.

6.1. Source Specific Multicast (SSM) use case

In SSM RTP sessions as described in [RFC5760], one or more Media Sources send RTP packets to a Distribution Source. The Distribution Source relays the RTP packets to the receivers using a source-specific multicast group.

In order to avoid the forms of Feedback implosion described in section 1,the distribution source should be told that the indicated packets were lost. How the distribution source know the indicated packets were lost is beyond of scope of this document. When upstream link or downstream aggregate link packet loss occurs, the distribution source creates a Third Party Loss Report and sent it to

all the RTP receivers, over the multicast channel. Another possibility is when there may be multiple distribution sources placed between the media source and the receivers, the upstream distribution source may inform downstream distribution sources of the detected packet loss using Third Party Loss Report messages. In response, the downstream distribution sources forward Third Party Loss Report received from upstream to all the RTP receivers, over the multicast channel. This Third Party Loss Report message tells the receivers that the sender of the third party loss report has received reports that the indicated packets were lost. The distribution source then can (optionally) ask for the lost packets from the media source on behalf of all the RTP receivers. The lost packets will either be forthcoming from distribution source, or it irretrievably lost such that there is nothing to be gained by the receiver sending a NACK to the media source.

The distribution source must be able to communicate with all group members in order for either mechanism to be effective at suppressing feedback.

As outlined in the [RFC5760], there are two Unicast Feedback models that may be used for reporting, - the Simple Feedback model and the Distribution Source Feedback Summary Model. The RTCP Feedback extension for Third Party Loss Report specified in the Section 4 of this document will work in both Feedback models. Details of operation in each are specified below.

6.1.1. Simple Feedback Model

In the simple Feedback Model, NACKs from the receiver observing the loss will be reflected to the other receivers, and there’s no need for distribution source to create the third-party loss report. The distribution source that has not seen the actual packet loss should pass through any Third Party Loss Report message it receives from the upstream direction.

This RTCP Third Party Loss Report message lets the receivers know that the sender of the Third party Loss Report has received reports that the indicated packets were lost and feedback for this packet loss is not needed and should not be sent to the media source(s). If the media source(s) are part of the SSM group for RTCP packet reflection, the Distribution Source must filter this packet out. If the media source(s) are not part of the SSM group for RTCP packets, the Distribution Source must not forward this RTCP Third Party Loss Report message to the media source(s).

6.1.2. Distribution Source Feedback Summary Model

In the distribution source feedback summary model, there may be multiple distribution sources and the Loss Detection instances are distributed into different distribution sources. In some cases, these Loss Detection instances for the same session can exist at the same time, e.g., one Loss Detection instance is implemented in the upstream distribution source A, a second Loss Detection instance for the same session is part of feedback target A and feedback target B respectively within the distribution source B. The distribution source B is placed in the path between distribution A and downstream receivers. In this section, we focus on this generic case to discuss the distribution Source Feedback Summary Model.

The distribution source A must listen on the RTP channel for data. When the distribution source A observes RTP packets from a media source are not consecutive by checking the sequence number of packets, the distribution source A generates the new RTCP Third Party Loss Report message described in the Section 4, and then send it to receivers in the downstream path via the multicast channel. Note that the distribution source A must use its own SSRC value as packet sender SSRC for transmitting the new RTCP Third Party Loss Report message.

a second detection instance within the Distribution Source B must also listen for RTCP data sent to the RTCP port. Upon receiving the RTCP Third Party Loss Report from the Distribution Source A, the distribution source B needs to check whether it sees upstream third party loss report from distribution source A reporting the same event. If the upstream Third Party Loss Report reports the different event, the distribution source B passes through any Third Party Loss Report message it receives from the upstream direction. If the same event is reported from distribution source A, the distribution source B replaces it with the summary Third Party Loss Report with the information summarization received from two loss detection instances within the Distribution Source B. In order to reduce the processing load at the distribution source, each loss detection instance may provide preliminary summarization report.

During the summary third party loss report creating, the Distribution Source B must use its own SSRC value as packet sender SSRC for transmitting summarization information and MUST perform proper SSRC collision detection and resolution.

The distribution source B may send this new RTCP summary third party loss report described in the Section 4to the group on the multicast RTCP channel and meanwhile send a packet loss request to the media source.

In some case, the distribution source B may receive RTCP NACK messages from the receivers behind the Distribution Source before the distribution source detects the packet loss which may cause potential Feedback implosion. In such case, the distribution source B may filter them out if it already detected the same loss or sent a packet loss request for the missing packet to the media source.

When the host receives the RTCP Third Party Loss Report message, if the host understands this message it will not send packet loss request (e.g., NACK) for the missing packets reported in the message. If it did not understand this new message, the host MAY send packet loss request(e.g., NACK messages) to the specified media source.

6.2. Unicast based Rapid Acquisition of Multicast Stream (RAMS) use case

The typical RAMS architecture [I-D.ietf-avt-rapid-acquisition-for-rtp]may have several Burst/ Retransmission Sources(BRS) behind the multicast source (MS) These BRSes will receive the multicast SSM stream from the media source. If one of the BRSes detects packet loss (i.e., First loss in Figure 3) on its upstream link between the MS and BRS, but the others BRSes have not, as the packet loss took place on SSM tree branch that does not impact the other BRSes. In such case, the BRSes with loss detection functionality support cannot detect packet loss at their upstream link, therefore these BRSes will not create new Third Party Loss Report message and send it to receivers in their downstream path. If the BRS impacted by packet loss has loss detection support, the BRS MAY choose to create new Third Party Loss Report message and send it to the receivers in the downstream link. Note that BRS must use its own SSRC as packet sender SSRC for transmitting the feedback suppress message.

The BRS may also send a NACK upstream to request the retransmitted packet. Upon receiving the retransmitted packet, the BRS sent it downstream. Note that this retransmitted packet may get lost (i.e., second loss in the Figure 3) on the downstream link. In order to deal with this issue, the downstream receiver can start a timeout clock in which it expected to get a retransmission packet. When this timeout expires and there is no retransmitted packet or a new Third Party Loss Report message, it can take its normal behavior as if there is no current retransmission suppression in place.

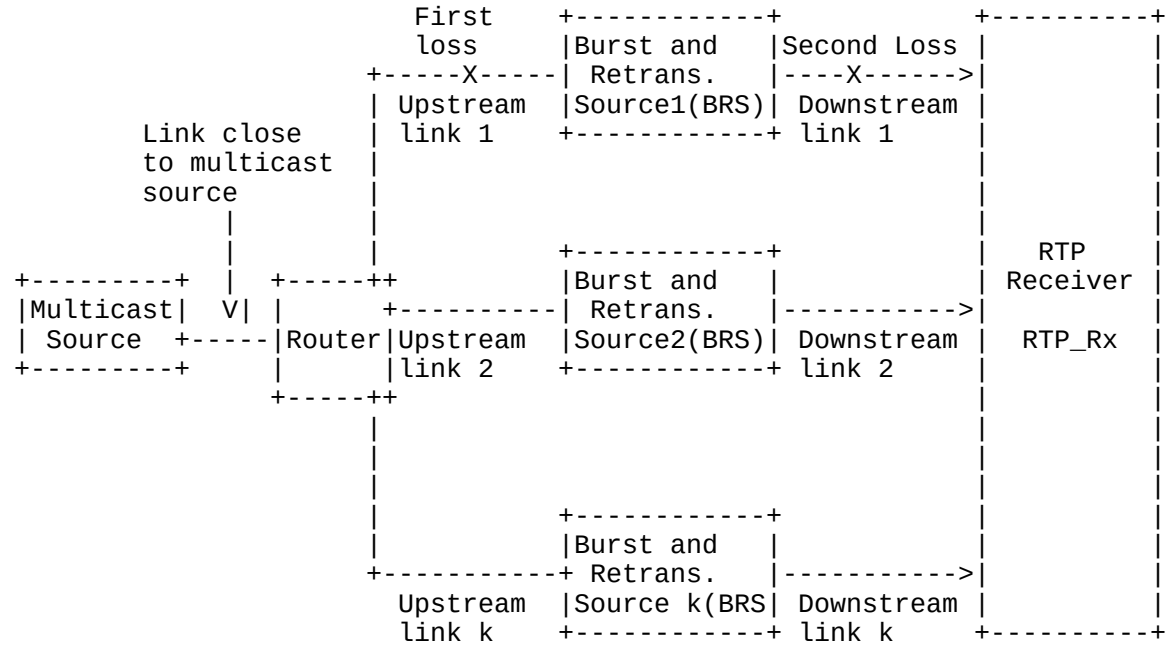


Figure 3: RAMS Use Case

6.3. RTP transport translator use case

A Transport Translator (Topo-Trn-Translator), as defined in [RFC5117] is typically forwarding the RTP and RTCP traffic between RTP clients, for example converting between multicast and unicast for domains that do not support multicast. The translator can identify packet loss from the upstream and send the Third Party Loss Report message to the unicast receivers. Note that the translator must be a participant in the session and can then use it’s own SSRC as packet sender SSRC for transmitting the Third Party Loss Report message

6.4. Multipoint Control Unit (MCU) use case

In point to multipoint topologies using video switching MCU (Topo-Video-switch-MCU) [RFC5117], the MCU typically forwards a single media stream to each participant, selected from the available input streams. The selection of the input stream is often based on voice activity in the audio-visual conference, but other conference management mechanisms (like presentation mode or explicit floor control) exist as well.

In this case the MCU may detect packet loss from the sender or may decide to switch to a new source. In both cases the receiver may

lose synchronization with the video stream and may send a FIR request. If the MCU itself can detect the mis-synchronization of the video, the MCU can send the FIR suppression message to the receivers and send a FIR request to the video source. As suggested in RFC 5117, this topology is better implemented as an Topo-mixer, in which case the mixer's SSRC is used as packet sender SSRC for transmitting Third Party Loss Report message.

7. Security Considerations

The defined messages have certain properties that have security implications. These must be addressed and taken into account by users of this protocol.

Spoofed or maliciously created feedback messages of the type defined in this specification can have the following implications:

Sending Third Party Loss Report with wrong sequence number of lost packet that makes missing RTP packets can not be compensated.

To prevent these attacks, there is a need to apply authentication and integrity protection of the feedback messages. This can be accomplished against threats external to the current RTP session using the RTP profile that combines Secure RTP [RFC3711] and AVPF into SAVPF [RFC5124].

Note that middleboxes that are not visible at the RTP layer that wish to send Third Party Loss Reports on behalf of the media source can only do so if they spoof the SSRC of the media source. This is difficult in case SRTP is in use. If the middlebox is visible at the RTP layer, this is not an issue, provided the middlebox is part of the security context for the session.

Also note that endpoints that receive a Third Party Loss Report would be well-advised to ignore it, unless it is authenticated via SRTCP or similar. Accepting un-authenticated Third Party Loss Report can lead to a denial of service attack, where the endpoint accepts poor quality media that could be repaired.

8. IANA Consideration

New feedback type and New parameters for RTCP Third Party Loss Report are subject to IANA registration. For general guidelines on IANA considerations for RTCP feedback, refer to [RFC4585].

This document assigns one new feedback type value x in the RTCP

feedback report registry to "Third Party Loss Report" with the following registrations format:

Name:	TPLR
Long Name:	Third Party Loss Report
Value:	TBD
Reference:	This document.

This document also assigns the parameter value y in the RTCP TPLR feedback report Registry to " Transport Layer Third Party Loss Early Indication ", with the following registrations format:

Name:	TLLEI
Long name:	Transport Layer Third Party Loss Early Indication
Value:	TBD
Reference:	this document.

This document also assigns the parameter value z in the RTCP TPLR feedback report Registry to "Payload Specific Third Party Loss Early Indication ", with the following registrations format:

Name:	PSLEI
Long name:	Payload Specific Third Party Loss Early Indication
Value:	TBD
Reference:	this document.

The contact information for the registrations is:

Qin Wu
sunseawq@huawei.com
101 Software Avenue, Yuhua District
Nanjing, Jiangsu 210012, China

9. Acknowledgement

The authors would like to thank David R Oran, Ali C. Begen, Colin Perkins,Tom VAN CAENEGEM, Ingemar Johansson S, Bill Ver Steeg, Jonathan Lennox, WeeSan Lee for their valuable comments and suggestions on this document.

10. References

10.1. Normative References

[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.

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

[RFC4585] Ott, J., Wenger, S., Sato, N., Burmeister, C., and J. Rey, "Extended RTP Profile for Real-time Transport Control Protocol (RTCP)-Based Feedback (RTP/AVPF)", RFC 4585, July 2006.

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

[RFC5117] Westerlund, M. and S. Wenger, "RTP Topologies", RFC 5117, January 2008.

[RFC4588] Rey, J., Leon, D., Miyazaki, A., Varsa, V., and R. Hakenberg, "RTP Retransmission Payload Format", RFC 4588, July 2006.

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

[RFC5234] Crocker, D. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", STD 68, RFC 5234, January 2008.

[RFC5104] Wenger, S., Chandra, U., Westerlund, M., and B. Burman, "Codec Control Messages in the RTP Audio-Visual Profile with Feedback (AVPF)", RFC 5104, February 2008.

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

[RFC5124] Ott, J. and E. Carrara, "Extended Secure RTP Profile for Real-time Transport Control Protocol (RTCP)-Based Feedback (RTP/SAVPF)", RFC 5124, February 2008.

10.2. Informative References

[RFC5740] Adamson, B., Bormann, C., Handley, M., and J. Macker, "NACK-Oriented Reliable Multicast (NORM) Transport Protocol", November 2009.

[DVB-IPTV]
ETSI Standard, "Digital Video Broadcasting(DVB); Transport of MPEG-2 TS Based DVB Services over IP Based Networks",

ETSI TS 102 034, V1.4.1 , August 2009.

[I-D.ietf-avt-rapid-acquisition-for-rtp]
Steeg, B., Begen, A., Caenegem, T., and Z. Vax, "Unicast-Based Rapid Acquisition of Multicast RTP Sessions", November 2010.

[I-D.hunt-avt-monarch-01]
Hunt, G. and P. Arden, "Monitoring Architectures for RTP", August 2008.

[I-D.ietf-pmol-metrics-framework-02]
Clark, A., "Framework for Performance Metric Development".

Authors' Addresses

Qin Wu
Huawei
101 Software Avenue, Yuhua District
Nanjing, Jiangsu 210012
China

Email: sunseawq@huawei.com

Frank Xia
Huawei
1700 Alma Dr. Suite 500
Plano, TX 75075
USA

Phone: +1 972-509-5599
Email: xiayangsong@huawei.com

Roni Even
Huawei
14 David Hamelech
Tel Aviv 64953
Israel

Email: even.roni@huawei.com

Audio/Video Transport Working Group
Internet-Draft
Intended status: Informational
Expires: December 18, 2011

Q. Wu, Ed.
Huawei
G. Hunt
Unaffiliated
P. Arden
BT

June 16, 2011

Monitoring Architectures for RTP
draft-ietf-avtcore-monarch-03.txt

Abstract

This memo proposes an architecture for extending RTCP with a new RTCP XR (RFC3611) block type to report new metrics regarding media transmission or reception quality, as proposed in RFC5968. This memo suggests that a new block should contain a single metric or a small number of metrics relevant to a single parameter of interest or concern, rather than containing a number of metrics which attempt to provide full coverage of all those parameters of concern to a specific application. Applications may then "mix and match" to create a set of blocks which covers their set of concerns. Where possible, a specific block should be designed to be re-usable across more than one application, for example, for all of voice, streaming audio and video.

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 December 18, 2011.

Table of Contents

1. Introduction

As more users and subscribers rely on real time application services, uncertainties in the performance and availability of these services are driving the need to support new standard methods for gathering performance metrics from RTP applications. These rapidly emerging standards, such as RTCP XR [RFC3611]and other RTCP extension to Sender Reports(SR), Receiver Reports (RR) [RFC3550]are being developed for the purpose of collecting and reporting performance metrics from endpoint devices that can be used to correlate the metrics, provide end to end service visibility and measure and monitor QoE.

However the proliferation of RTP/RTCP specific metrics for transport and application quality monitoring has been identified as a potential problem for RTP/RTCP interoperability, which attempt to provide full coverage of all those parameters of concern to a specific application. Since different applications layered on RTP may have some monitoring requirements in common, therefore these metrics should be satisfied by a common design.

The objective of this document is to define an extensible RTP monitoring framework to provide a small number of re-usable QoS/QoE metrics which facilitate reduced implementation costs and help maximize inter-operability. [RFC5968] has stated that, where RTCP is to be extended with a new metric, the preferred mechanism is by the addition of a new RTCP XR [RFC3611] block. This memo assumes that any requirement for a new metric to be transported in RTCP will use a new RTCP XR block.

2. Requirements notation

This memo is informative and as such contains no normative requirements.

In addition, the following terms are defined:

Transport level metrics

A set of metrics which characterise the three transport impairments of packet loss, packet delay, and packet delay variation. These metrics should be usable by any application which uses RTP transport.

Application level metrics

Metrics relating to QoE related parameters. These metrics are measured at the application level and focus on quality of content rather than network parameters.

End System metrics

Metrics relating to the way a terminal deals with transport impairments affecting the incident RTP stream. These may include de-jitter buffering, packet loss concealment, and the use of redundant streams (if any) for correction of error or loss.

3. RTP monitoring architecture

The RTP monitoring architecture comprises the following two key functional components shown below:

- Monitor
- Metric Block Structure

Monitor is a functional component defined in RFC3550 that acts as a source of information gathered for monitoring purposes. It may also collect statistics from multiple source, stores such information reported by RTCP XR or other RTCP extension appropriately as base metric or calculates composite metric. According to the definition of monitor in RFC3550, the end system that source RTP streams, an intermediate-system that forwards RTP packets to End-devices or a third party that does not participate RTP session (i.e., the third party monitor depicted in figure 1) can be envisioned to act as Monitor within the RTP monitoring architecture.

The Metric Block exposes real time Application Quality information in the appropriate report block format to monitor within the RTP monitoring architecture. Both the RTCP or RTCP XR can be extended to convey such information. The details on transport protocol for metric block is described in Section 3.1.

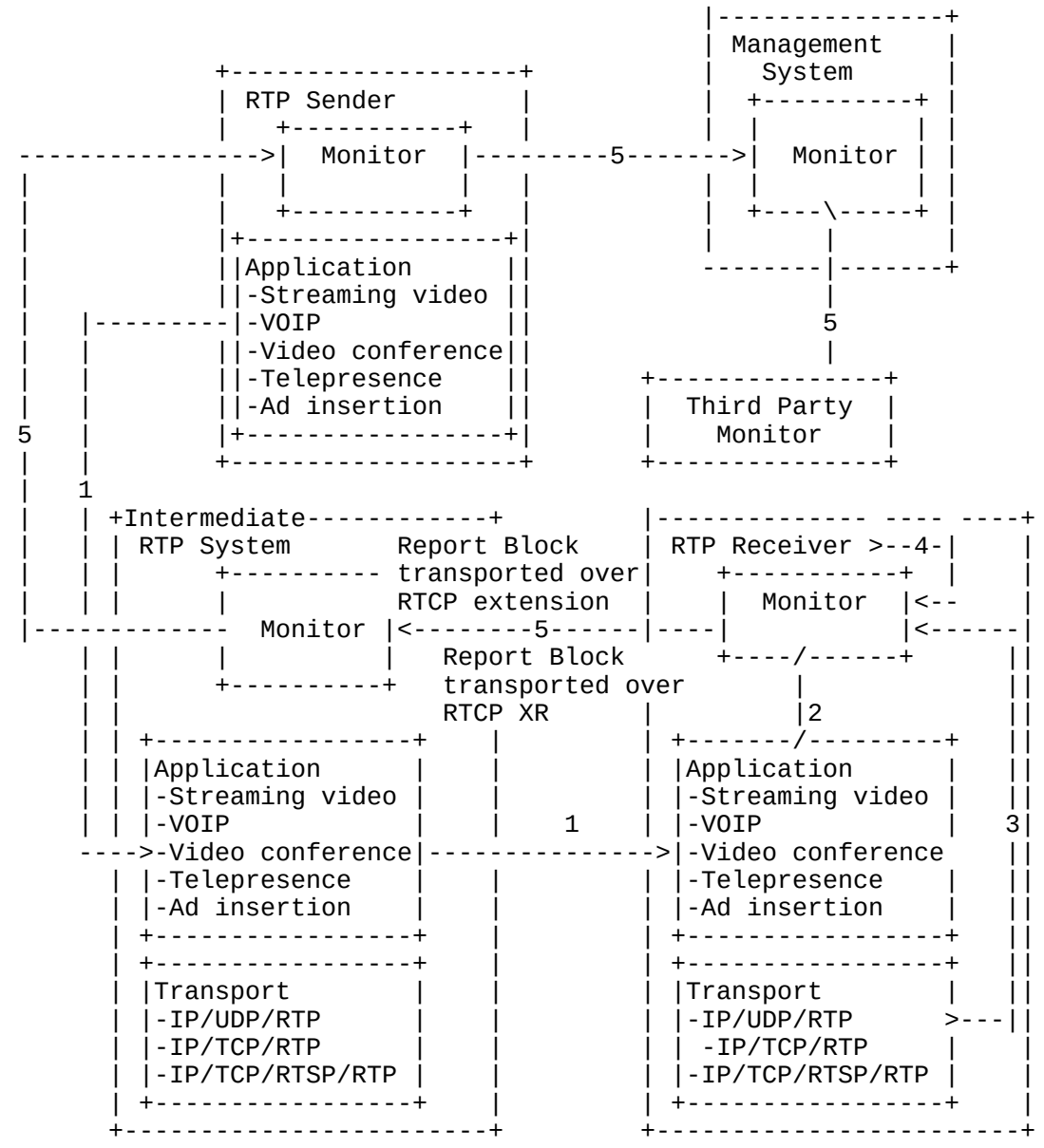


Figure 1: RTP Monitoring Architecture

1. RTP communication between real time applications

2. Application level metrics

3. Transport level metrics

4. End System metrics

5. Reporting Session- metrics transmitted over specified interfaces

3.1. RTCP Metric Block Report and associated parameters

The basic RTCP Reception Report (RR) conveys reception statistics in metric block report format for multiple RTP media streams including

- transport level statistics
- the fraction of packet lost since the last report
- the cumulative number of packets lost
- the highest sequence number received
- an estimate of the inter-arrival jitter
- and information to allow senders to calculate the network round trip time.

The RTCP XRs [RFC3611] supplement the existing RTCP packets and provide more detailed feedback on reception quality in several categories:

- Loss and duplicate RLE reports
- Packet-receipt times reports
- Round-trip time reports
- Statistics Summary Reports

There are also various other scenarios in which it is desirable to send RTCP Metric reports more frequently. The Audio/Video Profile with Feedback [RFC4585]extends the standard A/V Profile[RFC3551] to allow RTCP reports to be sent early provided RTCP bandwidth allocation is respected. There are four use cases but are not limited to:

- RTCP NACK is used to provide feedback on the RTP sequence number of the lost packets.

- RTCP XR is extended to provide feedback on multicast acquisition statistics information and parameters.
- RTCP is extended to convey requests for full intra-coded frames or select the reference picture, and signalchanges in the desired temporal/spatial trade-off and maximum media bit rate.
- RTCP or RTCP XR is extended to provide feedback on ECN statistics information.

4. Issues with reporting metric block using RTCP XR extension

Issues that have come up in the past with reporting metric block using RTCP XR extensions include (but are probably not limited to) the following:

- o Using large block. A single report block or metric is designed to contain a large number of parameters in different classes for a specific application. For example, RFC 3611 [RFC3611] defines seven report block formats for network management and quality monitoring. However some of these block types defined in [RFC3611] are only specifically designed for conveying multicast inference of network characteristics(MINC) or voice over IP (VoIP) monitoring. However different applications layered on RTP may have some monitoring requirements in common, design large block only for specific applications may increase implementation cost and minimize interoperability.

- o Identity Information duplication. Identity information is used to identify an instance of a metric block. The SSRC of the measured stream as part of the metric block is one example of Identity information. However in some cases, Identity information may be not part of metric and include information more than one in the metric block,e.g., when we set a metric interval for the session and monitor RTP packets within one or several consecutive metric interval, extra identity information is expected. In such case, if we put such extra identity information into each metric block or create standalone block containing extra identity information, there may be situations where an RTCP XR packet containing more than two metric blocks and other standalone block containing extra identity information, reports on the same streams from the same source. each block have the same identity information for measurement, if each metric block carry such duplicated data for the measurement, it leads to redundant information in this design since equivalent information is provided multiple times, once in *every* metric block and other block containing identity information. Though this ensures immunity to packet loss, the design bring more complexity and the overhead is not completely trivial.

- o Correlating RTCP XR with the non-RTP data. There may be situations where RTCP reports are sent to other participating endpoints using non-RTP protocol in a session. For example, as described in [RFC6035], the data contained in RTCP XR VoIP metrics reports [RFC3611] are forwarded to a central collection server systems using SIP. In such case, there is a large portfolio of quality parameters that can be associated with real time application,e.g., VOIP application, but only a minimal number of

parameters are included on the RTCP-XR reports. Therefore correlation between RTCP XR and non-RTP data should be concerned if administration or management systems need to rely on the mapping RTCP statistics to non-RTCP measurements to conducts data analysis and creates alerts to the users. Without such correlation, it is hardly to provide accurate measures of real time application quality with a minimal number of parameters included on the RTCP-XR reports in such case.

5. Guideline for reporting block format using RTCP XR

5.1. Using small blocks

Different applications using RTP for media transport certainly have differing requirements for metrics transported in RTCP to support their operation. For many applications, the basic metrics for transport impairments provided in RTCP SR and RR packets [RFC3550] (together with source identification provided in RTCP SDES packets) are sufficient. For other applications additional metrics may be required or at least sufficiently useful to justify the overheads, both of processing in endpoints and of increased session bandwidth. For example an IPTV application using Forward Error Correction (FEC) might use either a metric of post-repair loss or a metric giving detailed information about pre-repair loss bursts to optimise payload bandwidth and the strength of FEC required for changing network conditions. However there are many metrics available. It is likely that different applications or classes of applications will wish to use different metrics. Any one application is likely to require metrics for more than one parameter but if this is the case, different applications will almost certainly require different combinations of metrics. If larger blocks are defined containing multiple metrics to address the needs of each application, it becomes likely that many different such larger blocks are defined, which becomes a danger to interoperability.

To avoid this pitfall, this memo proposes the use of small RTCP XR metrics blocks each containing a very small number of individual metrics characterizing only one parameter of interest to an application running over RTP. For example, at the RTP transport layer, the parameter of interest might be packet delay variation, and specifically the metric "IPDV" defined by [Y1540]. See Section 6 for architectural considerations for a metrics block, using as an example a metrics block to report packet delay variation.

5.2. Reducing the identity information repetition

Any measurement must be identified. For example, an instance of a metric must be identified using information which is likely to include most of the following:

- the node at which it was measured,
- the source of the measured stream (for example, its CNAME),
- the SSRC of the measured stream,

- o the sequence number of the first packet of the RTP session,
- o the extended sequence numbers of the first packet of the current measurement interval, and the last packet included in the measurement,
- o the duration of the most recent measurement interval and
- o the duration of the interval applicable to cumulative measurements (which may be the duration of the RTP session to date).

Note that this set of information (identity information) may overlap with, but is more extensive than, that in the union of similar information in RTCP RR packets. However we can not assume that RR information is always present when XR is sent, since they may have different measurement intervals. Also the reason for the additional information carried in the XR is the perceived difficulty of "locating" the *start* of the RTP session (sequence number of 1st packet, duration of interval applicable to cumulative measurements) using only RR. Therefore this set of information can provide more detailed identity information relevant to the measurement report. However if such detailed identity information are all put into each metric and the metric are delivered in small blocks there is a danger of inefficiency arising from repeating this information in a number of metrics blocks within the same RTCP packet, in cases where the same identification information applies to multiple metrics blocks.

This memo proposes an approach to minimise the inefficiency of providing this identification information, assuming that an architecture based on small blocks means that a typical RTCP packet will contain more than one metrics block needing the same identification. The choice of identification information to be provided is discussed in [IDENTITY]. The approach is to define a stand-alone block containing only identification information within the scope of the containing RTCP XR packet. The "containing RTCP XR packet" is defined here as the RTCP XR header with PT=XR=207 defined in Section 2 of [RFC3611] and the associated payload defined by the length field of this RTCP XR header. The RTCP XR header itself includes the SSRC of the node at which all of the contained metrics were measured, hence this SSRC need not be repeated in the stand-alone identity block. A single RTCP XR packet containing multiple metric block may contain multiple identity blocks. Typically there will be one identity block per monitored source SSRC, but the use of more than one identification block for a single monitored source SSRC within a single containing RTCP XR packet is not ruled out. In order to further reduce identity information repeatition, This stand-alone block is recommended to only be exchanged occasionally, for example sent once at the start of a session.

5.3. Correlating identity information with the non-RTP data

When more than one media transport protocols are used by one application to interconnected to the same session (in gateway),e.g., one RTCP XR Packet is sent to the participating endpoints using non-RTP-based media transport in a VOIP session, one crucial factor lies in how to handle their different identities that are corresponding to different media transport.

This memo proposes an approach to facilitate the correlation of the RTCP XR identity information with other session-related non-RTP data, i.e., using SSRC of source in the identity information as the correlation tag to associate identity information with the non-RTP-based data.

An example use case is for an participant endpoint may convey a call identifier or a global call identifier associated with the XR block identity information and use SSRC of source in the identity information as the correlation tag to the call identifier. A flow measurement tool that is not call-aware then forward the subsequent metric reports along with this correlation tag which is included in the XR Block header to the network management. Network management can then use this tag to correlate this report with other diagnostic information such as call detail records.

6. An example of a metric block

This section uses the example of an existing proposed metrics block to illustrate the application of the principles set out in Section 5.1.

The example [PDV] (work in progress) is a block to convey information about packet delay variation (PDV) only, consistent with the principle that a metrics block should address only one parameter of interest. One simple metric of PDV is available in the RTCP RR packet as the "jit" field. There are other PDV metrics which may be more useful to certain applications. Two such metrics are the IPDV metric ([Y1540], [RFC3393]) and the MAPDV2 metric [G1020]. Use of these metrics is consistent with the principle in Section 5 of [RFC5968] that metrics should usually be defined elsewhere, so that RTCP standards define only the transport of the metric rather than its nature. The purpose of this section is to illustrate the architecture using the example of [PDV] (work in progress) rather than to document the design of the PDV metrics block or to provide a tutorial on PDV in general.

Given the availability of at least three metrics for PDV, there are design options for the allocation of metrics to RTCP XR blocks:

- provide an RTCP XR block per metric
- provide a single RTCP XR block which contains all three metrics
- provide a single RTCP block to convey any one of the three metrics, together with a identifier to inform the receiving RTP system of the specific metric being conveyed

In choosing between these options, extensibility is important, because additional metrics of PDV may well be standardized and require inclusion in this framework. The first option is extensible but only by use of additional RTCP XR blocks, which may consume the limited namespace for RTCP XR blocks at an unacceptable rate. The second option is not extensible, so could be rejected on that basis, but in any case a single application is quite unlikely to require transport of more than one metric for PDV. Hence the third option was chosen. This implies the creation of a subsidiary namespace to enumerate the PDV metrics which may be transported by this block, as discussed further in [PDV] (work in progress).

7. Application to RFC 5117 topologies

The topologies specified in [RFC5117] fall into two categories. The first category relates to the RTP system model utilizing multicast and/or unicast. The topologies in this category are specifically Topo-Point-to-Point, Topo- Multicast, Topo-Translator (both variants, Topo-Trn-Translator and Topo-Media-Translator, and combinations of the two), and Topo-Mixer. These topologies use RTP end systems, RTP mixers and RTP translators defined in [RFC3550]. For purposes of reporting connection quality to other RTP systems, RTP mixers and RTP end systems are very similar. Mixers resynchronize audio packets and do not relay RTCP reports received from one cloud towards other cloud(s). Translators do not resynchronize packets and SHOULD forward certain RTCP reports between clouds. In this category, the RTP system (end system, mixer or translator) which originates, terminates or forwards RTCP XR blocks is expected to handle RTCP, including RTCP XR, according to [RFC3550]. Provided this expectation is met, an RTP system using RTCP XR is architecturally no different from an RTP system of the same class (end system, mixer, or translator) which does not use RTCP XR. The second category relates to deployed system models used in many H.323 [H323] video conferences. The topologies in this category are Topo-Video-Switch-MCU and Topo-RTCP-terminating-MCU. Such topologies based on systems do not behave according to [RFC3550].

7.1. Applicability to MCU

Topo-Video-Switch-MCU and Topo-RTCP-terminating-MCU, suffer from the difficulties described in [RFC5117]. These difficulties apply to systems sending, and expecting to receive, RTCP XR blocks as much as to systems using other RTCP packet types. For example, a participant RTP end system may send media to a video switch MCU. If the media stream is not selected for forwarding by the switch, neither RTCP RR packets nor RTCP XR blocks referring to the end system’s generated stream will be received at the RTP end system. Strictly the RTP end system can only conclude that its RTP has been lost in the network, though an RTP end system complying with the robustness principle of [RFC1122] should survive with essential functions unimpaired.

7.2. Applicability to Translators

Section 7.2 of [RFC3550] describes processing of RTCP by translators. RTCP XR is within the scope of the recommendations of [RFC3550]. Some RTCP XR metrics blocks may usefully be measured at, and reported by, translators. As described in [RFC3550] this creates a requirement for the translator to allocate an SSRC for the monitor within itself so that it may populate the SSRC in the RTCP XR packet header (although the translator is not a Synchronisation Source in

the sense of originating RTP media packets). It must also supply this SSRC and the corresponding CNAME in RTCP SDES packets.

In RTP sessions where one or more translators generate any RTCP traffic towards their next-neighbour RTP system, other translators in the session have a choice as to whether they forward a translator's RTCP packets. Forwarding may provide additional information to other RTP systems in the connection but increases RTCP bandwidth and may in some cases present a security risk. RTP translators may have forwarding behaviour based on local policy, which might differ between different interfaces of the same translator.

For bidirectional unicast, an RTP system may usually detect RTCP XR from a translator by noting that the sending SSRC is not present in any RTP media packet. However even for bidirectional unicast there is a possibility of a source sending RTCP XR before it has sent any RTP media (leading to transient mis-categorisation of an RTP end system or RTP mixer as a translator), and for multicast sessions - or unidirectional/streaming unicast - there is a possibility of a receive-only end system being permanently mis-categorised as a translator sending XR report, i.e.,monitor collocated with transaltor. Hence it is desirable for a translator that sends XR to have a way to declare itself explicitly.

8. IANA Considerations

None.

9. Security Considerations

This document itself contains no normative text and hence should not give rise to any new security considerations, to be confirmed.

10. Acknowledgement

The authors would also like to thank Colin Perkins, Graeme Gibbs, Debbie Greenstreet, Keith Drage,Dan Romascanu, Ali C. Begen, Roni Even for their valuable comments and suggestions on the early version of this document.

11. Informative References

[G1020] ITU-T, "ITU-T Rec. G.1020, Performance parameter definitions for quality of speech and other voiceband applications utilizing IP networks", July 2006.

[H323] ITU-T, "ITU-T Rec. H.323, Packet-based multimedia communications systems", June 2006.

[IDENTITY] Hunt, G., "RTCP XR Report Block for Measurement Identity", ID draft-ietf-avt-rtcp-xr-meas-identity-02, May 2009.

[PDV] Hunt, G., "RTCP XR Report Block for Packet Delay Variation Metric Reporting", ID draft-ietf-avt-rtcp-xr-pdv-03, May 2009.

[RFC1122] Braden, R., "Requirements for Internet Hosts -- Communication Layers", RFC 1122, October 1989.

[RFC3393] Demichelis, C., "IP Packet Delay Variation Metric for IP Performance Metrics (IPPM)", RFC 3393, November 2002.

[RFC3550] Schulzrinne, H., "RTP: A Transport Protocol for Real-Time Applications", RFC 3550, July 2003.

[RFC3551] Schulzrinne , H. and S. Casner, "Extended RTP Profile for Real-time Transport Control Protocol (RTCP)-Based Feedback (RTP/AVPF)", RFC 3551, July 2003.

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

[RFC4585] Ott, J. and S. Wenger, "Extended RTP Profile for Real-time Transport Control Protocol (RTCP)-Based Feedback (RTP/AVPF)", RFC 4585, July 2006.

[RFC5117] Westerlund, M., "RTP Topologies", RFC 5117, January 2008.

[RFC5968] Ott, J. and C. Perkins, "Guidelines for Extending the RTP Control Protocol (RTCP)", RFC 5968, September 2010.

[RFC6035] Pendleton, A., Clark, A., Johnston, A., and H. Sinnreich, "Session Initiation Protocol Event Package for Voice Quality Reporting", RFC 6035, November 2010.

[Y1540] ITU-T, "ITU-T Rec. Y.1540, IP packet transfer and availability performance parameters", November 2007.

Appendix A. Change Log

Note to the RFC-Editor: please remove this section prior to publication as an RFC.

A.1. draft-ietf-avtcore-monarch-00

The following are the major changes compared to draft-hunt-avtcore-monarch-02:

- Move Geoff Hunt and Philip Arden to acknowledgement section.

A.2. draft-ietf-avtcore-monarch-01

The following are the major changes compared to 00:

- Restructure the document by merging section 4 into section 3.
- Remove section 4.1,section 5 that is out of scope of this document.
- Remove the last bullet in section 6 and section 7.3 based on conclusion of last meeting.
- Update figure 1 and related text in section 3 according to the monitor definition in RFC3550.
- Revise section 9 to address monitor declaration issue.
- Merge the first two bullet in section 6.
- Add one new bullet to discuss metric block association in section 6.

A.3. draft-ietf-avtcore-monarch-02

The following are the major changes compared to 01:

- Deleting first paragraph of Section 1.
- Deleting Section 3.1, since the interaction with the management application is out of scope of this draft.
- Separeate identity information correlation from section 5.2 as new section 5.3.
- Remove figure 2 and related text from section 5.2.

- Editorial changes in the section 4 and the first paragraph of section 7.

A.4. draft-ietf-avtcore-monarch-03

The following are the major changes compared to 01:

- Update bullet 2 in section 4 to explain the ill-effect of Identity Information duplication.
- Update bullet 3 in section 4 to explain why Correlating RTCP XR with the non-RTP data is needed.
- Update section 5.2 to focus on how to reduce the identity information repetition
- Update section 5.3 to explain how to correlate identity information with the non-RTP data

Authors' Addresses

Qin Wu (editor)
Huawei
101 Software Avenue, Yuhua District
Nanjing, Jiangsu 210012
China

Email: sunseawq@huawei.com

Geoff Hunt
Unaffiliated

Email: r.geoff.hunt@gmail.com

Philip Arden
BT
Orion 3/7 PP4
Adastral Park
Martlesham Heath
Ipswich, Suffolk IP5 3RE
United Kingdom

Phone: +44 1473 644192
Email: philip.arden@bt.com

AVTCORE
Internet-Draft
Intended status: Standards Track
Expires: December 3, 2011

J. Lennox
Vidyo

June 1, 2011

Encryption of Header Extensions in the Secure Real-Time Transport Protocol (SRTP)
draft-ietf-avtcore-srtp-encrypted-header-ext-00

Abstract

The Secure Real-Time Transport Protocol (SRTP) provides authentication, but not encryption, of the headers of Real-Time Transport Protocol (RTP) packets. However, RTP header extensions may carry sensitive information for which participants in multimedia sessions want confidentiality. This document provides a mechanism, extending the mechanisms of SRTP, to selectively encrypt RTP header extensions in SRTP.

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 December 3, 2011.

Copyright Notice

Table of Contents

1. Introduction

The Secure Real-Time Transport Protocol [RFC3711] specification provides confidentiality, message authentication, and replay protection for multimedia payloads sent using of the Real-Time Protocol (RTP) [RFC3550]. However, in order to preserve RTP header compression efficiency, SRTP provides only authentication and replay protection for the headers of RTP packets, not confidentiality.

For the standard portions of an RTP header, this does not normally present a problem, as the information carried in an RTP header does not provide much information beyond that which an attacker could infer by observing the size and timing of RTP packets. Thus, there is little need for confidentiality of the header information.

However, this is not necessarily true for information carried in RTP header extensions. A number of recent proposals for header extensions using the General Mechanism for RTP Header Extensions [RFC5285] carry information for which confidentiality could be desired or essential. Notably, two recent drafts ([I-D.ietf-avtext-client-to-mixer-audio-level] and [I-D.ietf-avtext-mixer-to-client-audio-level]) carry information about per-packet sound levels of the media data carried in the RTP payload, and exposing this to an eavesdropper may be unacceptable in many circumstances.

This document, therefore, defines a mechanism by which encryption can be applied to RTP header extensions when they are transported using SRTP. As an RTP sender may wish some extension information to be sent in the clear (for example, it may be useful for a network monitoring device to be aware of RTP transmission time offsets [RFC5450]), this mechanism can be selectively applied to a subset of the header extension elements carried in an SRTP packet.

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] and indicate requirement levels for compliant implementations.

3. Encryption Mechanism

Encrypted header extension elements are carried in the same manner as non-encrypted header extension elements, as defined by [RFC5285]. The (one- or two-byte) header of the extension elements is not

encrypted, nor is any of the header extension padding. If multiple different header extension elements are being encrypted, they have separate element identifier values, just as they would if they were not encrypted; similarly, encrypted and non-encrypted header extension elements have separate identifier values.

To encrypt (or decrypt) an encrypted extension header, an SRTP participant first generates a keystream for the SRTP extension header. This keystream is generated in the same manner as the encryption keystream for the corresponding SRTP payload, except the the SRTP encryption and salting keys k_e and k_s are replaced by the keys k_he and k_hs, respectively. The keys k_he and k_hs are computed in the same manner as k_e and k_s, except that the <label> values used are 0x06 for k_he and and 0x07 for k_hs. (Note that since RTP headers, including extension headers, are authenticated in SRTP, no new authentication key is needed for extension headers.)

The SRTP participant then computes an encryption mask for the header extension, identifying the portions of the header extension that are, or are to be, encrypted. This encryption mask corresponds to the entire payload of each header extension element that is encrypted. It does not include any non-encrypted header extension elements, any extension element headers, or any padding octets. The encryption mask has all-bits-1 octets (i.e., hexadecimal 0xff) for header extension octets which are to be encrypted, and all-bits-0 octets for header extension octets which are not to be.

For those octets indicated in the encryption mask, the SRTP participant bitwise exclusive-ors the header extension with the keystream to produce the ciphertext version of the header extension. Those octets not indicated in the encryption mask are left unmodified. Thus, conceptually, the encryption mask is logically ANDed with the keystream to produce a masked keystream. The sender and receiver MUST use the same encryption mask. The set of extension elements to be encrypted is communicated between the sender and the receiver using the signaling mechanisms described in Section 4.

The SRTP authentication tag is computed across the encrypted header extension, i.e., the data that is actually transmitted on the wire. Thus, header extension encryption MUST be done before the authentication tag is computed, and authentication tag validation MUST be done on the encrypted header extensions. For receivers, header extension decryption SHOULD be done only after the receiver has validated the packet’s message authentication tag.

3.1. Example Encryption Mask

If a sender wished to send a header extension containing an encrypted SMPTE timecode [RFC5484] with ID 1, a plaintext transmission time offset [RFC5450] with ID 2, and an encrypted audio level indication [I-D.ietf-avtext-client-to-mixer-audio-level] with ID 3, the plaintext RTP header extension might look like this:

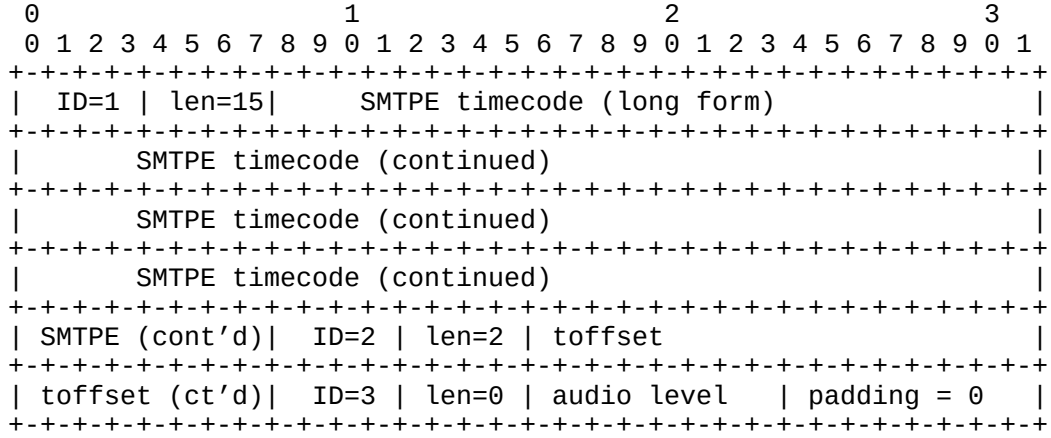


Figure 1

The corresponding encryption mask would then be:

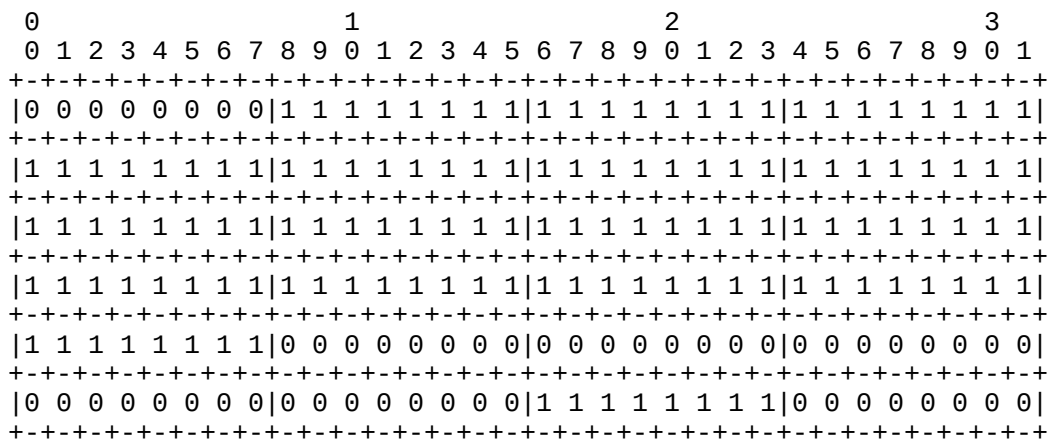


Figure 2

In the mask, the octets corresponding to the payloads of the encrypted header extension elements are set to all-1 values, and

octets corresponding to non-encrypted elements, element headers, and header extension padding are set to all-0 values.

4. Signaling (Setup) Information

Encrypted header extension elements are signaled in the SDP extmap attribute, using the URI "urn:ietf:params:rtp-hdrext:encrypt", followed by the URI of the header extension element being encrypted as well as any extensionattributes that extension normally takes. Thus, for example, to signal an SRTP session using encrypted SMPTE timecodes [RFC5484], while simultaneously signaling plaintext transmission time offsets [RFC5450], an SDP document could contain (line breaks added for formatting):

```
m=audio 49170 RTP/SAVP 0
a=crypto:1 AES_CM_128_HMAC_SHA1_32 \
  inline:NzB4d1BINUAvLEw6UzF3WSJ+PSdFcGdUJShpX1Zj|2^20|1:32
a=extmap:1 urn:ietf:params:rtp-hdrext:encrypt \
    urn:ietf:params:rtp-hdrext:smpte-tc 25@600/24
a=extmap:2 urn:ietf:params:rtp-hdrext:toffset
```

Figure 3

This example uses SDP Security Descriptions [RFC4568] for SRTP keying, but this is merely for illustration; any SRTP keying mechanism to establish session keys will work.

5. Security Considerations

The security properties of header extension elements protected by the mechanism in this document are equivalent to those for SRTP payloads.

The mechanism defined in this document does not provide confidentiality about which header extension elements are used for a given SRTP packet, only for the content of those header extension elements. This appears to be in the spirit of SRTP itself, which does not encrypt RTP headers. If this is a concern, an alternate mechanism would be needed to provide confidentiality.

For the two-byte-header form of header extension elements (0x100x), this mechanism does not provide any protection to zero-length header extension elements (for which their presence or absence is the only information they carry). It also does not provide any protection for the two-byte-headers' app bits (field 256, the lowest four bits of the "defined by profile" field). Neither of these features are used

in for one-byte-header form of header extension elements (0xBEDE), so these limitations do not apply in that case.

This document does not specify the circumstances in which extension header encryption should be used. Documents defining specific header extension elements should provide guidance on when encryption is appropriate for these elements.

6. IANA Considerations

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

Extension URI: urn:ietf:params:rtp-hdrext:encrypt
Description: Encrypted extension header element
Contact: jonathan@vidyo.com
Reference: RFC XXXX

(Note to the RFC-Editor: please replace "XXXX" with the number of this document prior to publication as an RFC.)

7. References

7.1. 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.

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

[RFC5285] Singer, D. and H. Desineni, "A General Mechanism for RTP Header Extensions", RFC 5285, July 2008.

7.2. Informative References

[I-D.ietf-avt-srtp-aes-gcm]
McGrew, D., "AES-GCM and AES-CCM Authenticated Encryption in Secure RTP (SRTP)", draft-ietf-avt-srtp-aes-gcm-01 (work in progress), January 2011.

[I-D.ietf-avtext-client-to-mixer-audio-level]
Lennox, J., Ivov, E., and E. Marocco, "A Real-Time Transport Protocol (RTP) Header Extension for Client-to-Mixer Audio Level Indication", draft-ietf-avtext-client-to-mixer-audio-level-01 (work in progress), March 2011.

[I-D.ietf-avtext-mixer-to-client-audio-level]
Ivov, E., Marocco, E., and J. Lennox, "A Real-Time Transport Protocol (RTP) Header Extension for Mixer-to-Client Audio Level Indication", draft-ietf-avtext-mixer-to-client-audio-level-02 (work in progress), May 2011.

[RFC4568] Andreasen, F., Baugher, M., and D. Wing, "Session Description Protocol (SDP) Security Descriptions for Media Streams", RFC 4568, July 2006.

[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.

Appendix A. Test Vectors

TODO

Appendix B. Open issues

- It is not clear how best to create the keystream for extension headers carried in SRTP packets protected with Authenticated Encryption with Associated Data (AEAD) cryptographic transforms, such as AES_GCM and AES_CCM [I-D.ietf-avt-srtp-aes-gcm]. Header extensions are already protected as ancillary data by AEAD mechanisms, and the mechanism defined in this document does not have any location to insert an additional authentication tag.

Appendix C. Changes From Earlier Versions

Note to the RFC-Editor: please remove this section prior to publication as an RFC.

C.1. Changes from draft-lennox-avtcore -00

- Published as working group item.
- Added discussion of limitations when used with the two-byte-header form of header extension elements.
- Added open issue about how to use this mechanism with Authenticated Encryption with Associated Data (AEAD) transforms.
- Updated references.

C.2. Changes from draft-lennox-avt -02

- Retargeted at AVTCORE working group.
- Updated references.

C.3. Changes From Individual Submission Draft -01

- Minor editorial changes.

C.4. Changes From Individual Submission Draft -00

- Clarified description of encryption mask creation.
- Added example encryption mask.
- Editorial changes.

Author's Address

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

Email: jonathan@vidyo.com

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: December 11, 2011

M. Ramalho, Ed.
P. Jones
Cisco Systems
N. Harada
NTT
M. Perumal
Cisco Systems
L. Miao
Huawei Technologies
June 9, 2011

RTP Payload Format for G.711.0
draft-ramalho-payload-g7110-00

Abstract

This document specifies the Real-Time Transport Protocol (RTP) payload format for ITU-T Recommendation G.711.0. ITU-T Rec. G.711.0 defines a lossless and stateless compression for G.711 packet payloads typically used in IP networks. This document also defines two storage mode formats for G.711.0. A media type registration for this RTP payload format is also included.

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 December 11, 2011.

Copyright Notice

Table of Contents

1. Introduction

The International Telecommunication Union (ITU-T) Recommendation G.711.0 [G.711.0] specifies a stateless and lossless compression for G.711 packet payloads typically used in VoIP networks. This document specifies the Real-Time Transport Protocol (RTP) RFC 3550 [RFC3550] payload format and storage modes for this compression.

2. Requirements Language

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

3. G.711.0 Codec Background

ITU-T Recommendation G.711.0 [G.711.0] is a lossless and stateless compression mechanism for ITU-T Recommendation G.711 [G.711] and thus is not a "codec" in the sense of "lossy" codecs typically carried by RTP. When negotiated end-to-end ITU-T Rec. G.711.0 is negotiated as if it were a codec, with the understanding that ITU-T Rec. G.711.0 losslessly encoded the underlying (lossy) G.711 pulse code modulation (PCM) sample representation of an audio signal. For this reason ITU-T Rec. G.711.0 will be interchangeably referred to in this document as a "lossless data compression algorithm" or a "codec", depending on context. Within this document, individual G.711 PCM samples will be referred to as "G.711 symbols" or just "symbols" for brevity.

This section describes the ITU-T Recommendation G.711 [G.711] codec, its properties, typical uses cases and its key design properties.

3.1. General Information and Use of the ITU-T G.711.0 Codec

ITU-T Recommendation G.711 is the benchmark standard for narrowband telephony. It has been successful for many decades because of its proven voice quality, ubiquity and utility. A new ITU-T recommendation, G.711.0, has been established for defining a stateless and lossless compression for G.711 packet payloads typically used in VoIP networks. ITU-T Rec. G.711.0 is also known as ITU-T Rec. G.711 Annex A [G.711-A1], as ITU-T Rec. G.711 Annex A is effectively a pointer ITU-T Rec. G.711.0. Henceforth in this document, ITU-T Rec. G.711.0 will simply be referred to as "G.711.0" and ITU-T Rec. G.711 simply as "G.711".

G.711.0 may be employed end to end; in which case the RTP payload format specification and use will be nearly identical to the G.711 RTP specification found in RFC 3550 [RFC3550]. The only significant difference other than the payload type (which will be a dynamically assigned payload type) will be the recommendation not to use Voice Activity Detection (as G.711.0 achieves its greatest compression during "VAD silence intervals"). SDP signaling elements are proposed for this use case of G.711.0 herein.

G.711.0, being both lossless and stateless, may also be employed as a LOSSLESS compression mechanism somewhere in between end systems which have negotiated use of G.711. For this case, the G.711 payloads and the corresponding G.711 RTP headers should appear to the end systems as having been transported transparently. This use case will be referred to as "G.711.0 in the Middle" and will be described in detail in Section 6 (Section 6). G.711.0, being both lossless and stateless, can be employed multiple times (e.g., on multiple,

individual hops or series of hops) of a given flow with no degradation of quality relative to end-to-end G.711. Stated another way, multiple "lossless transcodes" from/to G.711.0/G.711 do not negatively affect voice quality as may occur with lossy transcodes to/from dissimilar codecs. Since the use of G.711.0 as a compression mechanism can be used on any hop or hops of an end-to-end G.711 flow, neither Session Description Protocol (SDP) signaling elements nor G.711.0 negotiation mechanisms will be mandated in this document for this particular use case (although SDP descriptions in this document MAY be used for such G.711.0 negotiation).

Lastly, it is expected that G.711.0 will be used as an archival format for recorded G.711 streams. Therefore, a G.711.0 Storage Mode Format is also included in this document.

3.2. Key Properties of G.711.0 Design

The fundamental design of G.711.0 resulted from the desire losslessly encode and compress frames of G.711 symbols independent of what types of signals those G.711 frames contained. The primary G.711.0 use case is for G.711 encoded, zero-mean, acoustic signals (such as speech and music).

G.711.0 attributes are below:

A1 Compression for zero-mean acoustic signals: G.711.0 was designed as its primary use case for the compression of G.711 payloads which contained "speech" or other zero-mean acoustic signals. G.711.0 obtains greater than 50% average compression in service provider environments [ICASSP].

A2 Lossless for any G.711 payload: G.711.0 was designed to be lossless for any valid G.711 payload - even if the payload consisted of apparently random G.711 symbols (e.g., a modem or FAX payload). G.711.0 could be used for "aggregate 64 kbps G.711 channels" carried over IP without explicit concern if a subset of these channels happened to be carrying something other than voice or general audio. To the extent that a particular channel carried something than voice or general audio, G.711.0 ensured that it was carried losslessly, if not significantly compressed.

A3 Stateless: Compression of a frame of G.711 symbols was only to be dependent on that frame and not on any prior frame. Although greater compression is usually available by observing a longer history of past G.711 symbols, it was decided to for the compression design would be stateless to completely eliminate error propagation common in many lossy codec designs

(e.g., ITU-T Rec. G.729 [G.729], ITU-T Rec. G.722 [G.722]). That is, the decoding process need not be concerned about lost prior packets because the decompression of a given G.711.0 frame is not dependent on potentially lost prior frames. Owing to this stateless property, the frames input to the G.711.0 encoder may be changed "on-the-fly" (a 5 ms encoding could be followed by a 20 ms encoding).

A4 Self-describing: This property is defined as the ability to determine how many source G.711 samples are contained within the G.711.0 frame solely by information contained within the G.711.0 frame. Generally, the number of source G.711 symbols can be determined by decoding the initial octets of the compressed G.711.0 frame (these octets are called "prefix codes" in the standard)[ICASSP]. A G.711.0 decoder need not know what ptime is, as it is able to decompress the G.711.0 frame presented to it without signaling knowledge.

A5 Accommodate G.711 payload sizes typically used in IP: G.711 input frames of length typically found in VoIP applications represent SDP ptimes (see RFC 4566 [RFC4566]) of 5 ms, 10 ms, 20 ms, 30 ms or 40 ms. Since the dominant sampling frequency for G.711 is 8000 samples per second, G.711.0 was designed to compress G.711 input frames of 40, 80, 160, 240 or 320 samples.

A6 Bounded expansion: Since attribute A2 above requires G.711.0 to be lossless for any payload, by definition there exists at least one potential G.711 payload which must be "uncompressible". Since the quantum of compression is an octet, the minimum expansion of such an uncompressible payload was designed to be the minimum possible of one octet. Thus G.711.0 "compressed" frames can be of length one octet to X+1 octets, where X is the size of the input G.711 frame in octets. G.711.0 can therefore be viewed as a Variable Bit Rate (VBR) encoding in which the size of the G.711.0 output frame is a function of the G.711 symbols input to it.

A7 Algorithmic delay: G.711.0 was designed to have the algorithmic delay equal to the time represented by the number of samples in the G.711 input frame (i.e., no "look-ahead").

A8 Low Complexity: Less than 1.0 WMOPS average and low memory footprint (˜5k octets RAM, ˜5.7k octets ROM and ˜3.6 basic operations) [ICASSP] [G.711].

A9 Both A-law and Mu-law supported: G.711 has two operating laws, A-law and Mu-law. These two laws are also known as PCMA and PCMU in RFC 3550 [RFC3550]. The use of A-law or Mu-law should be signaled in SDP for IP applications.

These attributes generally make it trivial to compress a G.711 input frame consisting of 40, 80, 160, 240 or 320 samples. After the input frame is presented to a G.711.0 encoder, a G.711.0 "self-describing" output frame is produced. The number of samples contained within this frame is easily determined at the G.711.0 decoder by virtue of attribute A4. The G.711.0 decoder can decode the G.711.0 frame back to a G.711 frame by using only data within the G.711.0 frame.

Lastly we note that losing a G.711.0 encoded packet is identical in effect of losing a G.711 packet (when using RTP); this is because a G.711.0 payload, like the corresponding G.711 payload, is stateless. Thus, it is anticipated that existing G.711 PLC mechanisms will be employed when a G.711.0 packet is lost and an identical MOS degradation relative to G.711 loss will be achieved.

3.3. G.711 Input Frames to G.711.0 Output Frames

G.711.0 is a lossless and stateless compression of G.711 frames. The following figure depicts this where "A" is the process of G.711.0 encoding and "B" is the process of G.711.0 decoding.

1:1 Mapping from G.711 Input Frame to G.711.0 Output Frame

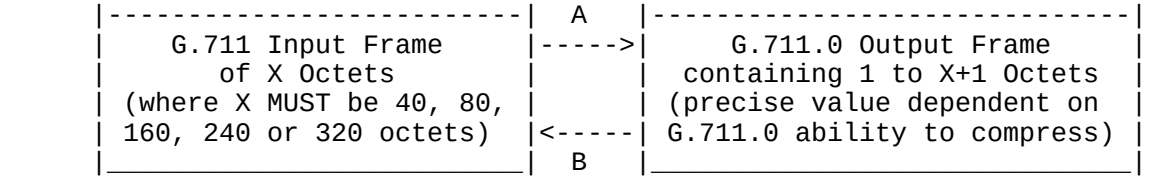


Figure 1

Note that the mapping is 1:1 (lossless) in both directions, subject to two constraints. The first constraint is that the input frame provided to the G.711.0 encoder (process "A") has a specific number of input G.711 symbols consistent with attribute A5 (40, 80, 160, 240 or 320 octets). The second constraint is that the compression law used to create the G.711 input frame (A-law or Mu-law) must be known, consistent with attribute A9.

Subject to these two constraints, the input G.711 frame is processed

by the G.711.0 encoder ("A") and produces a "self-describing" G.711.0 output frame, consistent with attribute A4. Depending on the source G.711 symbols, the G.711.0 output frame can contain anywhere from 1 to X+1 octets, where X is the number of input G.711 symbols. For virtually every use of G.711.0 resulting from zero-mean acoustic signal capture, we expect compression.

Since the G.711.0 output frame is "self-describing", a G.711.0 decoder (process "B") can losslessly reproduce the original G.711 input frame with only the knowledge of which companding law was used (A-law or Mu-law). The G.711.0 frame, being "self-describing", allows for the G.711.0 decoder ("B") to know precisely how many G.711 symbols to create.

Since G.711.0 was designed with typical G.711 payload lengths as a design constraint (attribute A5), this lossless encoding can be performed only with knowledge of the companding law being used. This information is anticipated to be signaled in SDP and will be described later in this document.

If the original inputs were known to be from a zero-mean acoustic signal coded by G.711, an intelligent G.711.0 encoder could infer the G.711 companding law in use (via G.711 input signal histogram). Likewise, an intelligent G.711.0 decoder producing G.711 from the G.711.0 frames could also infer the encoding law in use. Thus G.711.0 could be designed for use in applications that have limited stream signaling between the G.711 endpoints (i.e., they only know "G.711 at 8k sampling is being used", but nothing more). Such usage is not further described in this document. Additionally, if the original inputs were known to come from zero-mean acoustic signals, an intelligent G.711.0 encoder could tell if the G.711.0 payload had been encrypted - as the symbols would not have the distribution expected in either companding law and would appear random. Such determination is also not further discussed in this document.

It is easily seen that this process is 1:1 and that G.711.0 based lossless compression can be employed multiple times, as the original G.711 input symbols are always reproduced with 100% fidelity.

G.711.0 frames containing more source G.711 symbols compress more as a general rule, but there are exceptions. For example, an intelligent G.711.0 encoder may choose to encode 20 ms of G.711 as two individual 10 ms G.711.0 frames if a higher overall compression will result (this might occur if the first 10 ms was "silence" and two, 10 ms G.711.0 frames contained fewer octets than one 20 ms G.711.0 frame). For this reason, we will explicitly allow multiple G.711.0 encoded frames in the G.711.0 RTP payload in Section 4.2.2 (Section 3.3) below even though the usual case is anticipated to be

only one G.711.0 frame per RTP payload.

4. RTP Header and Payload

In this section we describe the precise format for G.711.0 frames carried via RTP. We begin with RTP header description relative to G.711, then provide two G.711.0 payload examples.

4.1. G.711.0 RTP Header

Relative to G.711 RTP headers, the utilization of G.711.0 does not create any special requirements with respect to the contents of the RTP packet header. The only significant difference is that the payload type (PT) RTP header field will have a value corresponding to the dynamic payload type assigned to the flow (whereas G.711 PCMU has a static PT = 0 and G.711 PCMA has a static PT = 8 [RFC3551]).

Voice Activity Detection (VAD) SHOULD NOT be used when G.711.0 is negotiated because G.711.0 obtains high compression during "VAD silence intervals" and one of the advantages of G.711.0 over G.711 with VAD is the lack of any VAD-inducing artifacts in the received signal. However, if VAD is employed, the Marker bit (M) MUST be set in the first packet of a talkspurt, that is, the first packet after a silence period which packets have not been transmitted contiguously as per rules specified in [RFC3550] for G.711 payloads.

With this introduction, the RTP packet header fields are defined as follows:

V - As per [RFC3550]

P - As per [RFC3550]

X - As per [RFC3550]

CC - As per [RFC3550]

M - As per [RFC3550]

PT- Dynamic PT assigned, consistent with MIME allocation for G711.0 defined in Media Type Definition (Section 5.1 (Section 5.1)).

SN - As per [RFC3550]

timestamp - As per [RFC3550]

SSRC - As per [RFC3550]

CSRC - As per [RFC3550]

Where V (version bits), P (padding bit), X (extension bit), CC (CSRC count), M (marker bit), PT (payload type), SN (sequence number), timestamp, SSRC (synchronizing source) and CSRC (contributing sources) are as defined in [RFC3550] and as typically used with G.711. PT (payload type) is as defined in [RFC3550].

4.2. G.711.0 RTP Payload

In this section we provide two examples for carrying G.711.0 frames in RTP payloads. The first example is used when it is desired to carry only one G.711.0 frame is in the payload. This example is a subset of the second and shown separately for clarity.

4.2.1. Single G.711.0 Frame per RTP Payload Example

This example depicts a single G.711.0 frame in the RTP payload. This is expected to be the dominant RTP payload case for G.711.0, as the G.711.0 encoding process supports the SDP packet times (ptime and maxptime, see [RFC4566]) commonly used when G.711 is transported in RTP. Additionally, as mentioned previously, larger G.711.0 frames generally compress more effectively than a multiplicity of smaller G.711.0 frames.

The following Figure illustrates the single G.711.0 frame per RTP payload case.

Single G.711.0 Frame in RTP Payload Case

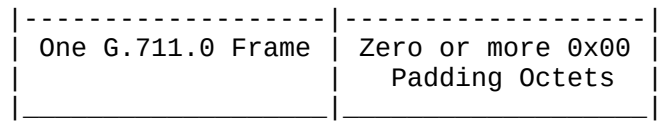


Figure 2

Encoding Process: A single G.711.0 frame is inserted into the RTP payload. The amount of time represented by the G.711 symbols compressed in the G.711.0 frame MUST correspond to the ptime signaled for applications using SDP. Although generally not desired, padding desired in the RTP payload after the G.711.0 frame MAY be created by placing one or more 0x00 octets after the G.711.0 frame. Such padding may be desired based on security considerations (see Section 11 (Section 11)).

Decoding Process: Passing the entire RTP payload to the G.711.0 decoder is sufficient for the G.711.0 decoder to create the source G.711 symbols. Any padding inserted after the G.711.0 frame (i.e., the 0x00 octets) present in the RTP payload is silently ignored by the G.711.0 decoding process. The decoding process is fully described in Section Section 4.2.3 below.

4.2.2. Multiple G.711.0 Frames per RTP Payload Example

This example depicts the case where multiple G.711.0 frames are desired in the RTP payload.

As described in Section 3.3 (Section 3.3), an "intelligent G.711.0 encoder" can decide to encode, let's say, 20 ms of G.711 symbols as two, 10 ms G.711.0 frames because a greater compression is attained for that particular 20 ms segment. Thus such "smart encoding" of such inputs is accommodated by the ability to have multiple G.711.0 frames in the RTP payload.

Note that since each G.711.0 frame is self-describing (see Attribute A4 in Section 3.2 (Section 3.2)), the individual G.711.0 frames in the RTP payload need not represent the same duration of time (i.e., a 5 ms G.711.0 frame could be followed by a 20 ms G.711.0 frame). Owing to this, the amount of time represented in the RTP payload MAY be any integer multiple of 5 ms (as 5 ms is the smallest interval of time that can be represented in a G.711.0 frame).

The following Figure illustrates the multiple G.711.0 frame per RTP payload case where the number of G.711.0 frames placed in the RTP payload is N.

Multiple G.711.0 Frames in RTP Payload Case

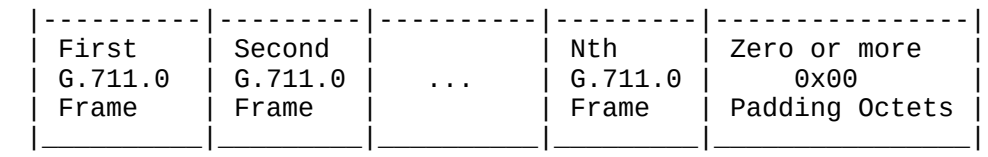


Figure 3

We note here that the individual G.711.0 frames can be, and generally are, of different lengths. The decoding process in the following section is used to determine the frame boundaries.

Encoding Process: One or more G.711.0 frames are placed in the RTP

payload simply by concatenating the G.711.0 frames together. The amount of time represented by the G.711 symbols compressed in all the G.711.0 frames in the RTP payload MUST correspond to the ptime signaled for applications using SDP. Although not generally desired, padding desired in the RTP payload SHOULD be placed after the last G.711.0 frame in the payload and MAY be created by placing one or more 0x00 octets after the last G.711.0 frame. Such padding may be desired based on security considerations (see Section 11 (Section 11)).

Decoding Process: As G.711.0 frames can be of varying length, the payload decoding process described in the following section is used to determine where the individual G.711.0 frame boundaries are.

4.2.3. G.711.0 RTP Payload Decoding Process

This decoding process is a standard part of G.711.0 bit stream decoding and is implemented in the ITU-T Rec. G.711.0 reference code.

Before describing the decoding, we note here that the largest possible G.711.0 frame is created whenever the largest number of G.711 symbols is encoded (320 from Section 3.2 (Section 3.2), property A5) and these 320 symbols are "uncompressible" by the G.711.0 encoder. In this case (via property A6 in Section 3.2 (Section 3.2)) the G.711.0 output frame will be 321 octets long. We also note that the value 0x00 chosen for the optional padding cannot be the first octet of a valid G.711.0 frame (see [G.711.0]). We also note that whenever more than one G.711.0 frame is contained in the RTP payload, the decoding of the individual G.711.0 frames will occur multiple times.

For the decoding heuristic below, let N be the number of octets in the RTP payload (i.e., excluding any RTP padding, but including any RTP payload padding), let P equal the number of RTP payload octets processed by the G.711.0 decoding process, let J represent the present G.711.0 frame being decoded, let K be the number of G.711 symbols in the output buffer, let Q be the number of octets contained in the present G.711.0 frame being processed and let "!=" represent not equal to. The keyword "STOP" is used below to indicate the end of the processing of G.711.0 frames in the RTP payload. The heuristic below assumes an output buffer for the decoded G.711 source symbols of length sufficient to accommodate the expected number of G.711 symbols and an input buffer of length 321 octets.

G.711.0 RTP Decoding Heuristic:

H1 Initialize the number of processed octets to zero (P = 0). Initialize the G.711.0 frame counter J to zero (J = 0). Initialize the counter for how many G.711 symbols are in the output buffer to zero (K = 0).

H2 Read min{320+1, (N-P)} octets into the internal buffer from the (P+1) octet of the RTP payload. We note at this point, N-P octets have yet to be processed and that 320+1 octets is the largest possible G.711.0 frame.

H3 Analyze the first octet in the internal buffer. If this octet is other than 0x00 (a padding octet), increment the G.711.0 frame counter (set J = J + 1) and continue to Step H4. Otherwise increment the processed packets counter by one (set P = P + 1). If the result of this increment results in P = N then STOP (as all RTP Payload octet have been processed), otherwise go to Step H2.

H4 Pass the internal buffer to the G.711.0 decoder. The G.711.0 decoder will read the first octet (called the "prefix code" octet in [G.711.0]) to determine the number of source G.711 samples M are contained in this G.711.0 frame.

H5 The G.711.0 decoder will produce exactly M G.711 source symbols. If J = 1, these M symbols will be the first in the output buffer and are placed at the beginning of the output buffer. If J != 1, these M symbols are concatenated with the prior symbols in the output buffer. Set K = K + M (as there are now this many G.711 source symbols in the output buffer).

H6 In process H5, the G.711.0 decoder will have consumed some number of packets, Q, in the internal buffer to produce the M G.711 symbols. Increment the number of processed octets by this quantity; that is set P = P + Q.

H7 If P < N there are more octets in the RTP payload left to process, go to Step H2. If P >= N, STOP (as all RTP payload octets have been processed).

At this point, the output buffer will contain precisely K G.711 source symbols which should correspond to the ptime signaled if SDP was used and the encoding process was without error. We also note, as an aside, that the heuristic above (and the ITU-T G.711.0 reference code) accommodates padding octets (0x00) placed anywhere in the RTP payload.

If the decoder is at a playout endpoint location, this G.711 buffer SHOULD be used in the same manner as a received G.711 payload would

have been used (passed to a playout buffer, to a PLC implementation. etc.). If not, then the instructions in Section 6 (Section 6) (G.711.0 "In The Middle") should be followed.

5. Payload Format Parameters

This section defines the parameters that may be used to configure optional features in the G.711.0 RTP transmission.

The parameters defined here as a part of the media subtype registration for the G.711.0 codec. Mapping of the parameters into Session Description Protocol (SDP) RFC 4566 [RFC4566] is also provided for those applications that use SDP.

5.1. Media Type Registration

Type name: audio

Subtype name: G7110

Required Parameters:

rate: The RTP timestamp clock rate, which is equal to the sampling rate. The typical rate is 8000, but other rates may be specified.

complaw: Indicates the companding law (A-law or mu-law) employed. The case-insensitive values are "a" or "mu".

Optional parameters:

channels: how many audio streams are represented in the G.711.0 payload - defaults to 1; stereo would be 2, etc.

[Editor's Note: We are considering specifying more than one channel for multiplexing or conference switching applications. One option for the delimiting the channels with the RTP payload would be to use one of the few not allowed G.711.0 frame prefix codes to delineate the channel data and appropriate modification to the RTP decoding heuristic in Section 4.2.3 (Section 4.2.3). Note that the channel order is already well specified in RFC 3551 [RFC3551].]

ptime, maxptime: see RFC 4566 [RFC4566]

Encoding considerations:

This media type is framed binary data (see Section 4.8 in RFC 4288 [RFC4288]) compressed as per ITU-T Rec. G.711.0.

Security considerations:

This media type does not carry active content. It does transfer compressed data. See Section 4 of RFC 4856 [RFC4856].

Interoperability considerations: none

Published specification:

ITU-T Rec. G.711.0 and RFC QQQQ.

[RFC Editor: please replace QQQQ with a reference to this RFC]

Applications that use this media type:

Audio and video streaming and conferencing tools.

Additional information: none

Person & email address to contact for further information:

Michael Ramalho <mramalho@cisco.com> or <mar42@cornell.edu>

Intended usage: COMMON

Restrictions on usage:

This media type depends on RTP framing, and hence is only defined for transfer via RTP [RFC3550]. Transport within other framing protocols is not defined at this time.

Author: Michael Ramalho

Change controller:

IETF Audio/Video Transport working group delegated from the IESG.

5.2. Mapping to SDP Parameters

The information carried in the media type specification has a specific mapping to fields in the Session Description Protocol (SDP), which is commonly used to describe RTP sessions. When SDP is used to specify sessions employing G.711.0, the mapping is as follows:

- The media type ("audio") goes in SDP "m=" as the media name.
- The media subtype ("G7110") goes in SDP "a=rtpmap" as the encoding name.

- The required parameter "rate" also goes in "a=rtpmap" as the clock rate.
- The parameters "ptime" and "maxptime" go in the SDP "a=ptime" and "a=maxptime" attributes, respectively.
- Remaining parameters go in the SDP "a=fmtp" attribute by copying them directly from the media type string as a semicolon-separated list of parameter=value pairs.

5.3. Offer/Answer Considerations

There are no special considerations when using the SDP offer/answer RFC 3264 [RFC3264] as all the SDP parameters are declaritive.

[EDITOR'S NOTE: This may change in a future revision if the channel parameter is negotiated. This could happen when the offer desires channels = N and the answerer can only support a number less than N. We would then insert an offer/answer example in a future revision of this draft.]

5.4. SDP Example

The following examples illustrate how to signal G.711.0 via SDP:

```
m=audio RTP/AVP 98

a=rtpmap:  98 G7110/8000

a=prime:  20

a=fmtp:98 complaw = mu
```

In the above example, the dynamic payload type 98 is mapped to G.711.0 via the "a=rtpmap" parameter. The packetization time (ptime) is indicated to be 20 ms of audio. The mandatory "complaw" is on the "a=fmtp" parameter line.

6. G.711.0 "In The Middle"

When G.711 has been negotiated end-to-end, G.711.0 compression can be employed by entities in the middle of the end-to-end G.711 flow as a compression mechanism. When used in this manner, it can be used with or without compression of the RTP header. In either case, the G.711 payloads AND the corresponding G.711 RTP headers MUST appear to the end systems as having been transported transparently.

6.1. G.711.0 "In The Middle" - No RTP Header Compression

This figure below illustrates how the compression could be accomplished without the RTP header compression.

G.711.0 Compression "In The Middle" - No RTP Header Compression Case

Figure 4

Figure 4 depicts G.711.0 compression in Box C and G.711.0 decompression back to G.711 in Box E. It depicts the case where only compression of the G.711 payload is desired; the RTP header (including any extensions) is simply copied with the exception that the G.711 payload type (the usual static PT of 0 or 8 is shown) is

replaced by a PT negotiated between Box C and Box E (depicted as PT = Q).

Note that if there are no hops between Box C and E (i.e, no Box D), this is equivalent to compression over a single link. The compression segment represented by Box C, Box E and Box F is labeled a "G.711.0 compression segment" in the above figure.

Since G.711.0 is a lossless and stateless compression, there can be multiple such segments between the sending and receiving endpoints (not shown).

The G.711.0 compression and decompression (Box C and E) may reside in a variety of network elements such as, but not limited to, switches, routers, middleboxes (NATs/PATs, firewalls, session border controllers, transport acceleration devices) and is purposely not specified here.

There may be many "potential G.711.0 compression/decompression points" along the end-to-end G.711 flow; the mechanisms by which certain entities determine that they should perform G.711.0-based compression and decompression are outside the scope of this document.

The method by which G.711.0 compression segment endpoints negotiate which RTP payload type (Q shown above) is to be used is outside the scope of this document, although the SDP elements described herein MAY be used.

Firewalls, NATs, SBCs, etc. that may exist in the path of the G.711.0 packets (Box D) and who may drop packets of unexpected payload types may need additional configuration and/or intelligence to let the compressed G.711.0 packets through. Mechanisms to do this are also outside the scope of this document.

[EDITOR'S NOTE: Because many boxes of this type inspect signaling to determine which RTP packets are allowed to progress, we are considering a hint to be place in the G.711 (not G.711.0) SDP that says, in essence, if you do compress this packet, please use PT = Q. In this way middleboxes may also know to pass PT = Q packets as well as PT = [0 | 8]. Such a G.711 SDP entry may look like:

Example of G.711 SDP with hint for G.711.0 PT

```
m=audio RTP/AVP 0
a=rtpmap: 0 PCMU
a=fmtp:0 G7110 = Q   <<< the G.711 SDP hint
```

Here, the last line provides the hint that this G.711 Mu-law flow may be represented anywhere on the end-to-end G.711 flow as PT = 0 (uncompressed) or PT = Q (compressed) RTP packets. Then candidate compression entities MAY choose to honor that hint in their respective dynamic payload type negotiation for G.711.0. Of course, this hint would require the registration of an optional parameter for the PCMU and PCMA media registrations. We hereby solicit feedback on this concept.]

6.2. G.711.0 "In The Middle" - With RTP Header Compression

When it is desired to compress the G.711 header as well, the G.711.0 compression segment endpoints of the previous section have further functionality by which they also compress the headers. However, this functionality is outside of the scope of this document.

We simply note here that if such functionality is employed, that the G.711 payloads AND the corresponding G.711 RTP headers MUST appear to the end systems as having been transported transparently. Such RTP header compression functionality SHOULD be stateless so as to minimize error propagation for lost packets to be consistent with G.711.0 design goal attribute A3.

6.3. G.711.0 "In The Middle" - Implications for Voice Quality and Added Delay

As described in the sections immediately previous, G.711.0 can be employed multiple times (e.g., on multiple, individual hops or series of hops) of a given G.711 flow. Owing to the stateless design of G.711.0 and any RTP header compression scheme recommended above, there is no error propagation owing to loss of a G.711.0 packet. Thus the impact of an individual packet drop of a G.711.0 RTP packet is identical to the impact of the corresponding equivalent G.711 RTP packet.

Stated another way, multiple "lossless transcodes" from/to G.711.0/ G.711 do not negatively affect voice quality as may occur with lossy transcodes to/from dissimilar codecs.

G.711.0 provides over 50% reduction in average payload size with exactly 0.0000% quality loss relative to G.711 [ICASSP].

For completeness, we note that a G.711.0 encode/decode average complexity is 1 WMOPS (see Section 3.2 (Section 3.2), attribute A8). Given such low complexity, less than 1 ms of compression/ decompression delay per each G.711.0 compression segment is expected in most implementations.

6.4. G.711.0 "In The Middle" - Multiplexing Multiple G.711 Flows

It may also be desired to multiplex the payloads of many G.711 channels into one "G.711.0 payload". As with the previous section, functionality for the mechanism used is outside the scope of this document.

[EDITOR'S NOTE: If we allow the channels parameter to be >1, then a mechanism for multiplexing the G.711.0 payloads into one RTP payload would be specified in this document. However, we don't know of a standardized use case for multiplexing a multiplicity of media streams representing multiple endpoint flows in one RTP media flow. This would be advantageous if, for example, multiple G.711 flows traverse two known endpoints and where the endpoints could "add and delete" G.711 flows "on-the-fly" (perhaps by association of the RTP CSRCs) and then fake out the endpoints with simulated RTP headers in a manner similar to WAN optimization product does today for TCP. Comments are welcome on this point.]

If such RTP multiplexing compression functionality is designed, the RTP header compression used with the G.711.0 multiplexing SHOULD be stateless so as to minimize error propagation for lost packets to be consistent with G.711.0 design goal attribute A3.

7. G.711.0 Storage Mode

There are two storage modes defined for the G.711.0; one for short recordings and one for long recordings.

For short recordings, the recommendation will be similar to many other IETF codecs (e.g., iLBC, EVRC-NW) and will fundamentally be a concatenation of received G.711.0 frames and erasure frames.

However, since G.711.0 has variable length frames, it is prudent for long recordings (over many minutes long) to use an indexing methodology that enables playout far into the recording without decoding all the G.711.0 frames since the recording began.

For either storage mode, a "G.711.0 erasure frame" is defined in the following section because ITU-T Rec. G.711.0 [G.711.0] does not define one.

7.1. G.711.0 Erasure Frame

A G.711.0 erasure frame is a representation of the amount of time in lost or not received G.711.0 frames. Lost frames are typically determined by unexpected discontinuities observed in the RTP send timestamps at the receiver.

A G.711.0 compressed frame can represent 40, 80, 160, 240 or 320 G.711 symbols (Attribute A5 of Section 3.2 (Section 3.2)). As G.711.0 also does not use the value of 0x01 for the first octet of any valid G.711.0 frame, it can be used as an identifier for an erasure frame. Thus an erasure frame is the two octet quantity shown below.

The Two Octet G.711.0 Erasure Frame Definition

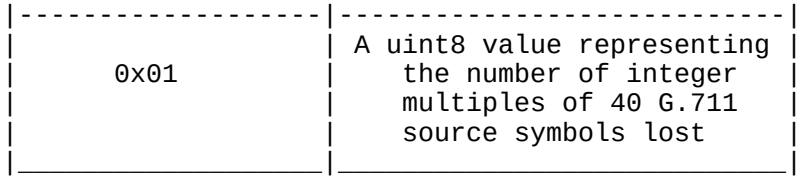


Figure 5

The value of the second octet is representative of how many G.711 source samples were lost. For example, a value of 4 implies 160 samples were lost independently of how many G.711.0 frames those 160

samples were represented by. Erasure frames themselves may be concatenated if it is desired to create one per G.711.0 frame, or one per G.711.0 RTP payload, etc. This erasure frame format can therefore represent 255*40 = 10,200 missing G.711 symbols (i.e., about 1.2 seconds of G.711 at 8000 samples per second).

The erasure frame has been designed to represent precisely what was observed at the receiver for true archival purposes (including potentially law enforcement). Therefore, the use of erasure frames for missing G.711.0 frames is RECOMMENDED for archival purposes.

If such an archival needs are not required, a storage implementation MAY provide packet loss concealment (PLC) or simply "silence" (G.711.0 analog zero representation) for the missing source G.711 data and choose not to use an erasure frame. Such PLC/silence insertion MUST represent the precise amount of time represented by the missing data to maintain synchronization with the original media.

7.2. G.711.0 Storage Mode - Short Recordings

This short recording storage format is used for storing G.711.0 encoded frames. The file begins with a magic number to identify the coder that is used. The magic number for G.711.0 A-law corresponds to the ASCII character string "#!G7110A\n", i.e., "0x23 0x21 0x47 0x37 0x31 0x31 0x30 0x41 0x0A". Likewise, the magic number for G.711.0 MU-law corresponds to the ASCII character string "#!G7110M\n", i.e., "0x23 0x21 0x47 0x37 0x31 0x31 0x4E 0x4D 0x0A". The codec data frames including any necessary erasure frames are stored in consecutive order concatenated together as shown in Section 4.2.2 (Section 4.2.2).

To decode the individual G.711.0 frames, a heuristic similar to the one presented in Section 4.2.2 (Section 4.2.2) modified appropriately to recognize and process erasure frames as legitimate G.711.0 frames in the recording format.

7.3. G.711.0 Storage Mode - Long Recordings

[EDITOR'S NOTE: The long recordings storage mode format is TBD. This storage mode will likely have indexing capability and metadata capabilities.]

8. Acknowledgements

There have been many people contributing to G.711.0 in the course of its development. The people listed here deserve special mention: Takehiro Moriya, Claude Lamblin, Herve Taddei, Simao Campos, Yusuke Hiwasaki, Jacek Stachurski, Lorin Netsch, Paul Coverdale, Patrick Luthi, Paul Barrett, Jari Hagqvist, Pengjun (Jeff) Huang, and Jon Gibbs.

9. Contributors

The authors thank everyone who have comtributed to this document. The people listed here deserve special mention: Ali Begen and Roni Even.

10. IANA Considerations

One media type (audio/G7110) has been defined and requires IANA registration in the media types registry. See Section 5.1 (Section 5.1)

11. Security Considerations

RTP packets using the payload format defined in this specification are subject to the security considerations discussed in the RTP specification [RFC3550], and in any appropriate RTP profile (for example RFC 3551 [RFC3551] or [RFC4585]. This implies that confidentiality of the media streams is achieved by encryption; for example, through the application of SRTP [RFC3711]. Because the data compression used with this payload format is applied end-to-end, any encryption needs to be performed after compression.

Note that the appropriate mechanism to ensure confidentiality and integrity of RTP packets and their payloads is very dependent on the application and on the transport and signaling protocols employed. Thus, although SRTP is given as an example above, other possible choices exist.

Note that end-to-end security with either authentication, integrity or confidentiality protection will prevent a network element not within the security context from performing media-aware operations other than discarding complete packets. To allow any (media-aware) intermediate network element to perform its operations, it is required to be a trusted entity which is included in the security context establishment.

G.711.0 has no known denial-of-service attacks due to decoding, as data posing as a desired G711.0 payload will be decoded into something (as per the decoding algorithm) with a finite amount of computation. This is due to the decompression algorithm having a finite worst-case processing path (no infinite computational loops are possible).

G.711.0 is a variable bit rate (VBR) audio codec. There have been recent concerns with VBR speech codecs where a passive observer can identify phrases from a standard speech corpus by means of the lengths produced by the encoder even when the payload is encrypted [IEEE]. In this paper, it was determined that some code excited linear prediction (CELP) codecs would produce discrete packet lengths for some phonemes. And furthermore with the use of appropriately designed Hidden Markov Models (HMMs) that such a system could predict phrases with unexpected accuracy. One CELP codec studied, SPEEX, had the property that it produced 21 different packet lengths in its wideband mode and that these packet lengths probabilistically mapped to phonemes that a HMM system could be trained on. In this paper it was determined that a mitigation technique would be to pad the output of the encoder with random padding lengths to the effect: 1) that more discrete payload sizes would result, and 2) that the probabilistic mapping to phonemes would become less clear. As G.711

is not a speech model based codec, neither is G.711.0. A G.711.0 encoding, during talking periods, produces frames of varying frame lengths which are not likely to have a strong mapping to phonemes. Thus G.711.0 is not expected to have this same vulnerability. It should be noted that "silence" (only one value of G.711 in the entire G.711 input frame)" or "near silence" (only a few G.711 values) is easily detectable as G.711.0 frame lengths or one or a few octets. If one desires to mitigate for silence/non-silence detection, statistically variable padding should be added to G.711.0 frames that resulted in very small G.711.0 frames (less than about 20% of the symbols of the corresponding G.711 input frame). Methods of introducing padding in the G.711.0 payloads have been provided in the G.711.0 RTP payload definitions in Sections 4.2.1 (Section 4.2.1) and 4.2.2 (Section 4.2.2).

12. References

12.1. Normative References

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

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

[RFC4288] Freed, N. and J. Klensin, "Media Type Specifications and Registration Procedures", BCP 13, RFC 4288, December 2005.

[RFC4855] Casner, S., "Media Type Registration of RTP Payload Formats", RFC 4855, February 2007.

[RFC4856] Casner, S., "Media Type Registration of Payload Formats in the RTP Profile for Audio and Video Conferences", RFC 4856, February 2007.

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

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

[RFC4585] Ott, J., Wenger, S., Sato, N., Burmeister, C., and J. Rey, "Extended RTP Profile for Real-time Transport Control Protocol (RTCP)-Based Feedback (RTP/AVPF)", RFC 4585, July 2006.

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

[RFC3264] Rosenberg, J. and H. Schulzrinne, "An Offer/Answer Model with Session Description Protocol (SDP)", RFC 3264, June 2002.

[G.711.0] ITU-T G.711.0, "Recommendation ITU-T G.711.0 - Lossless Compression of G.711 Pulse Code Modulation", September 2009.

[G.711] ITU-T G.711.0, "Recommendation ITU-T G.711 - Pulse Code Modulation (PCM) of Voice Frequencies", November 1988.

[G.711-A1] ITU-T G.711 Amendment 1, "Recommendation ITU-T G.711 Amendment 1 - Amendment 1: New Annex A on Lossless Encoding of PCM Frames", September 2009.

12.2. Informative References

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

[G.729] ITU-T G.729, "Recommendation ITU-T G.729 - Coding of speech at 8 kbit/s using conjugate-structure algebraic-code-excited linear prediction (CS-ACELP)", January 2007.

[G.722] ITU-T G.722, "Recommendation ITU-T G.722 - 7 kHz audio-coding within 64 kbit/s", November 1988.

[ICASSP] N. Harada, Y. Yamamoto, T. Moriya, Y. Hiwasaki, M. A. Ramalho, L. Netsch, Y. Stachurski, Miao Lei, H. Taddei, and Q. Fengyan, "Emerging ITU-T Standard G.711.0 - Lossless Compression of G.711 Pulse Code Modulation, International Conference on Acoustics Speech and Signal Processing (ICASSP), 2010, ISBN 978-1-4244-4244-4295-9", March 2010.

[IEEE] C.V. Wright, L. Ballard, S.E. Coull, F. Monrose, and G.M. Masson, "Spot Me if You Can: Uncovering Spoken Phrases in Encrypted VoIP Conversations, IEEE Symposium on Security and Privacy, 2008, ISBN: 978-0-7695-3168-7", May 2008.

Authors' Addresses

Michael A. Ramalho (editor)
Cisco Systems, Inc.
4563 Tuscana Drive
Sarasota, FL 34241
USA

Phone: +1 919 476 2038
Email: mramalho@cisco.com

Paul E. Jones
Cisco Systems, Inc.
7025 Kit Creek Rd.
Research Triangle Park, NC 27709
USA

Phone: +1 919 476 2048
Email: paulej@packetizer.com

Noboru Harada
NTT Communications Science Labs
3-1 Morinosato-Wakamiya
Atsugi, Kanagawa 243-0198
JAPAN

Email: harada.noboru@lab.ntt.co.jp

Muthu Arul Mozhi Perumal
Cisco Systems, Inc.
Cessna Business Park
Sarjapur-Marathahalli Outer Ring Road
Bangalore, Karnataka 560103
India

Phone: +91 9449288768
Email: mpermumal@cisco.com

Miao Lei
Huawei Technologies Co. Ltd
No. 3 Xinxi Rd.
ShangDi, HaiDian District
Beijing, Beijing 100085
Beijing

Phone: +86 1082882759
Email: lei.miao@huawei.com

AVT Core Working Group
Internet-Draft
Intended status: Experimental
Expires: January 9, 2012

V. Singh
T. Karkkainen
J. Ott
S. Ahsan
Aalto University
L. Eggert
Nokia

July 8, 2011

Multipath RTP (MPRTP)
draft-singh-avtcore-mprtp-02

Abstract

The Real-time Transport Protocol (RTP) is used to deliver real-time content and, along with the RTP Control Protocol (RTCP), forms the control channel between the sender and receiver. However, RTP and RTCP assume a single delivery path between the sender and receiver and make decisions based on the measured characteristics of this single path. Increasingly, endpoints are becoming multi-homed, which means that they are connected via multiple Internet paths. Network utilization can be improved when endpoints use multiple parallel paths for communication. The resulting increase in reliability and throughput can also enhance the user experience. This document extends the Real-time Transport Protocol (RTP) so that a single session can take advantage of the availability of multiple paths between two endpoints.

Status of this Memo

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

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

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

This Internet-Draft will expire on January 9, 2012.

Copyright Notice

Table of Contents

1. Introduction

Multi-homed endpoints are becoming common in today's Internet, e.g., devices that support multiple wireless access technologies such as 3G and Wireless LAN. This means that there is often more than one network path available between two endpoints. Transport protocols, such as RTP, have not been designed to take advantage of the availability of multiple concurrent paths and therefore cannot benefit from the increased capacity and reliability that can be achieved by pooling their respective capacities.

Multipath RTP (MPRTP) is an OPTIONAL extension to RTP [1] that allows splitting a single RTP stream into multiple subflows that are transmitted over different paths. In effect, this pools the resource capacity of multiple paths. Multipath RTCP (MPRTCP) is an extension to RTCP, it is used along with MPRTP to report per-path sender and receiver characteristics.

Other IETF transport protocols that are capable of using multiple paths include SCTP [10], MPTCP MPTCP [11] and SHIM6 [12]. However, these protocols are not suitable for realtime communications.

1.1. 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 [2].

1.2. Terminology

- Endpoint: host either initiating or terminating an RTP connection.
- Interface: logical or physical component that is capable of acquiring a unique IP address.
- Path: sequence of links between a sender and a receiver. Typically, defined by a set of source and destination addresses.
- Subflow: flow of RTP packets along a specific path, i.e., a subset of the packets belonging to an RTP stream. The combination of all RTP subflows forms the complete RTP stream. Typically, a subflow would map to a unique path, i.e., each combination of IP addresses and port pairs (5-tuple) is a unique subflow.

1.3. Use-cases

The primary use-case for MPRTP is transporting high bit-rate streaming multimedia content between endpoints, where at least one is multi-homed. Such endpoints could be residential IPTV devices that connect to the Internet through two different Internet service providers (ISPs), or mobile devices that connect to the Internet through 3G and WLAN interfaces. By allowing RTP to use multiple paths for transmission, the following gains can be achieved:

- Higher quality: Pooling the resource capacity of multiple Internet paths allows higher bit-rate and higher quality codecs to be used. From the application perspective, the available bandwidth between the two endpoints increases.

- Load balancing: Transmitting one RTP stream over multiple paths can reduce the bandwidth usage, compared to transmitting the same stream along a single path. This reduces the impact on other traffic.

- Fault tolerance: When multiple paths are used in conjunction with redundancy mechanisms (FEC, re-transmissions, etc.), outages on one path have less impact on the overall perceived quality of the stream.

A secondary use-case for MPRTP is transporting Voice over IP (VoIP) calls to a device with multiple interfaces. Again, such an endpoint could be a mobile device with multiple wireless interfaces. In this case, little is to be gained from resource pooling, i.e., higher capacity or load balancing, because a single path should be easily capable of handling the required load. However, using multiple concurrent subflows can improve fault tolerance, because traffic can shift between the subflows when path outages occur. This results in very fast transport-layer handovers that do not require support from signaling.

2. Goals

This section outlines the basic goals that multipath RTP aims to meet. These are broadly classified as Functional goals and Compatibility goals.

2.1. Functional goals

Allow unicast RTP session to be split into multiple subflows in order to be carried over multiple paths. This may prove beneficial in case of video streaming.

- Increased Throughput: Cumulative capacity of the two paths may meet the requirements of the multimedia session. Therefore, MPRTP MUST support concurrent use of the multiple paths.

- Improved Reliability: MPRTP SHOULD be able to send redundant packets or re-transmit packets along any available path to increase reliability.

The protocol SHOULD be able to open new subflows for an existing session when new paths appear and MUST be able to close subflows when paths disappear.

2.2. Compatibility goals

MPRTP MUST be backwards compatible; an MPRTP stream needs to fall back to be compatible with legacy RTP stacks if MPRTP support is not successfully negotiated.

- Application Compatibility: MPRTP service model MUST be backwards compatible with existing RTP applications, i.e., an MPRTP stack MUST be able to work with legacy RTP applications and not require changes to them. Therefore, the basic RTP APIs MUST remain unchanged, but an MPRTP stack MAY provide extended APIs so that the application can configure any additional features provided by the MPRTP stack.

- Network Compatibility: individual RTP subflows MUST themselves be well-formed RTP flows, so that they are able to traverse NATs and firewalls. This MUST be the case even when interfaces appear after session initiation. Interactive Connectivity Establishment (ICE) [3] MAY be used for discovering new interfaces or performing connectivity checks.

3. RTP Topologies

RFC 5117 [13] describes a number of scenarios using mixers and translators in single-party (point-to-point), and multi-party (point-to-multipoint) scenarios. RFC 3550 [1] (Section 2.3 and 7.x) discuss in detail the impact of mixers and translators on RTP and RTCP packets. MPRTP assumes that if a mixer or translator exists in the network, then either all of the multiple paths or none of the multiple paths go via this component.

4. MPRTP Architecture

In a typical scenario, an RTP session uses a single path. In an

MPRTP scenario, an RTP session uses multiple subflows that each use a different path. Figure 1 shows the difference.

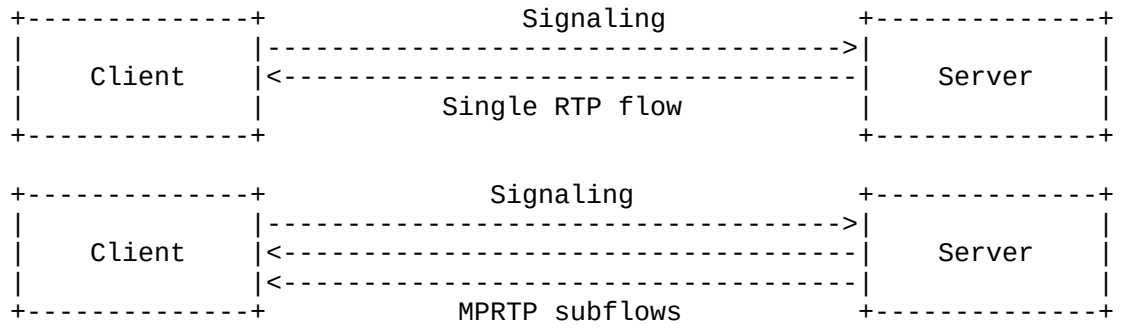


Figure 1: Comparison between traditional RTP streaming and MPRTP

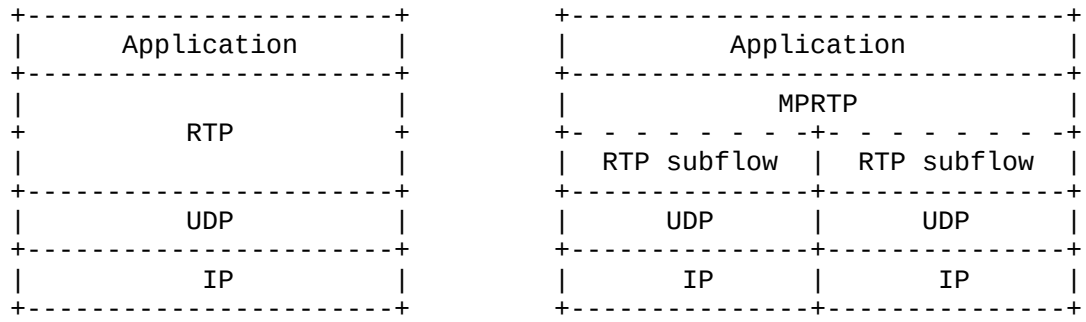


Figure 2: MPRTP Architecture

Figure 2 illustrates the differences between the standard RTP stack and the MPRTP stack. MPRTP receives a normal RTP session from the application and splits it into multiple RTP subflows. Each subflow is then sent along a different path to the receiver. To the network, each subflow appears as an independent, well-formed RTP flow. At the receiver, the subflows are combined to recreate the original RTP session. The MPRTP layer performs the following functions:

- Path Management: The layer is aware of alternate paths to the other host, which may, for example, be the peer's multiple interfaces. So that it is able to send differently marked packets along separate paths. MPRTP also selects interfaces to send and receive data. Furthermore, it manages the port and IP address pair bindings for each subflow.

- Packet Scheduling: the layer splits a single RTP flow into multiple subflows and sends them across multiple interfaces (paths). The splitting MAY BE done using different path characteristics.

- Subflow recombination: the layer creates the original stream by recombining the independent subflows. Therefore, the multipath subflows appear as a single RTP stream to applications.

4.1. Relationship of MPRTP with Session Signaling

Session signaling (e.g., SIP [14], RTSP [15]) SHOULD be done over a failover-capable or multipath-capable transport for e.g., SCTP [10] or MPTCP [11] instead of TCP or UDP.

5. Example Media Flow Diagrams

There may be many complex technical scenarios for MPRTP, however, this memo only considers the following two scenarios: 1) a unidirectional media flow that represents the streaming use-case, and 2) a bidirectional media flow that represents a conversational use-case.

5.1. Streaming use-case

In the unidirectional scenario, the receiver (client) initiates a multimedia session with the sender (server). The receiver or the sender may have multiple interfaces and both endpoints are MPRTP-capable. Figure 3 shows this scenario. In this case, host A has multiple interfaces. Host B performs connectivity checks on host A's multiple interfaces. If the interfaces are reachable, then host B streams multimedia data along multiple paths to host A. Moreover, host B also sends RTCP Sender Reports (SR) for each subflow and host A responds with a standard RTCP Receiver Report (RR) for the overall session and receiver statistics for each subflow. Host B distributes the packets across the subflows based on the individually measured path characteristics.

Alternatively, to reduce media startup time, host B may start streaming multimedia data to host A's initiating interface and then perform connectivity checks for the other interfaces. This method of updating a single path session to a multipath session is called "multipath session upgrade".

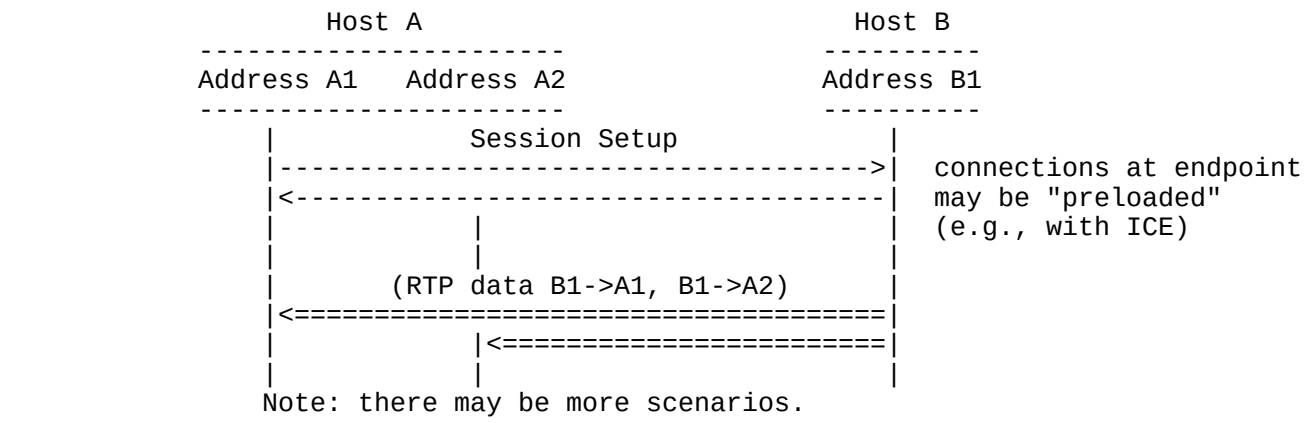


Figure 3: Unidirectional media flow

5.2. Conversational use-case

In the bidirectional scenario, multimedia data flows in both directions. The two hosts exchange their lists of interfaces with each other and perform connectivity checks. Communication begins after each host finds suitable address, port pairs. Interfaces that receive data send back RTCP receiver statistics for that path (based on the 5-tuple). The hosts balance their multimedia stream across multiple paths based on the per path reception statistics and its own volume of traffic. Figure 4 describes an example of a bidirectional flow.

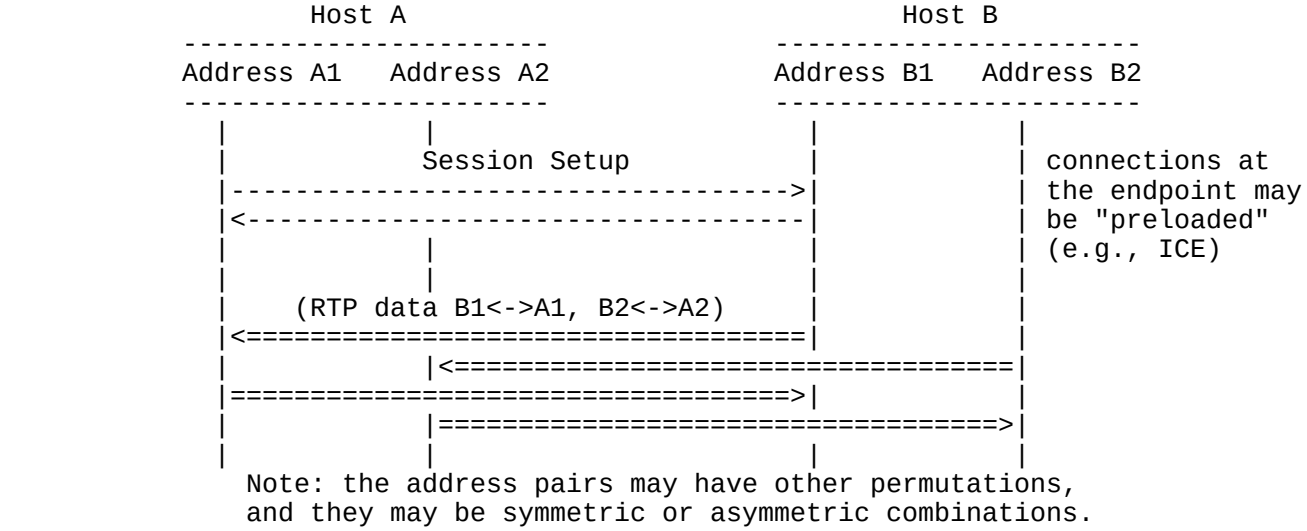


Figure 4: Bidirectional flow

5.3. Challenges with Multipath Interface Discovery

For some applications, where the user expects immediate playback, e.g., High Definition Media Streaming or IPTV, it may not be possible to perform connectivity checks within the given time bound. In these cases, connectivity checks MAY need to be done ahead of time.

[Open Issue: ICE or any other system would have to be aware of the endpoint's interfaces ahead of time].

6. MPRTP Functional Blocks

This section describes some of the functional blocks needed for MPRTP. We then investigate each block and consider available mechanisms in the next section.

1. Session Setup: Multipath session setup is an upgrade or add-on to a typical RTP session. Interfaces may appear or disappear at anytime during the session. To preserve backward compatibility with legacy applications, a multipath session MUST look like a bundle of individual RTP sessions.

2. Expanding RTP: For a multipath session, each subflow MUST look like an independent RTP flow, so that individual RTCP messages can be generated per subflow. Furthermore, MPRTP splits the single multimedia stream into multiple subflows based on path characteristics (e.g. RTT, loss-rate, receiver rate, bandwidth-delay product etc.) and dynamically adjusts the load on each link.

3. Adding Interfaces: Interfaces on the host need to be regularly discovered and signaled. This can be done at session setup and/or during the session. When discovering and receiving new interfaces, the MPRTP layer needs to select address and port pairs.

4. Expanding RTCP: MPRTP MUST recombine RTCP reports from each path to re-create a single RTCP message to maintain backward compatibility with legacy applications.

5. Maintenance and Failure Handling: In a multi-homed endpoint interfaces may appear and disappear. If this happens at the sender, it has to re-adjust the load on the available links. On the other hand, if this occurs on the receiver, then the multimedia data transmitted by the sender to those interfaces is

lost. This data may be re-transmitted along a different path i.e., to a different interface on the receiver. Furthermore, the receiver has to explicitly signal the disappearance of an interface, or the sender has to detect it. [Open Issue: What happens if the interface that setup the session disappears? does the control channel also failover? re-start the session?]

6. Teardown: The MPRTP layer releases the occupied ports on the interfaces.

7. Available Mechanisms within the Functional Blocks

This section discusses some of the possible alternatives for each functional block mentioned in the previous section.

7.1. Session Setup

MPRTP session can be set up in many possible ways e.g., during handshake, or upgraded mid-session. The capability exchange may be done using out-of-band signaling (e.g., SDP [16] in SIP [14], RTSP [15]) or in-band signaling (e.g., RTP/RTCP header extension). Furthermore, ICE [3] may be used for discovering and performing connectivity checks during session setup.

7.2. Expanding RTP

RTCP [1] is generated per media session. However, with MPRTP, the media sender spreads the RTP load across several interfaces. The media sender SHOULD make the path selection, load balancing and fault tolerance decisions based on the characteristics of each path. Therefore, apart from normal RTP sequence numbers defined in [1], the MPRTP sender MUST add subflow-specific sequence numbers to help calculate fractional losses, jitter, RTT, playout time, etc., for each path and a subflow identifier to associate the characteristics with a path. The RTP header extension for MPRTP is shown in Section 9).

7.3. Adding New Interfaces

When interfaces appear and disappear mid-session, ICE [3] may be used for discovering interfaces and performing connectivity checks. However, MPRTP may require a capability re-negotiation (using SDP) to include all these new interfaces. This method is referred to as out-of-band multipath advertisement.

Alternatively, when new interfaces appear, the interface advertisements may be done in-band using RTP/RTCP extensions. The

endpoints perform connectivity checks (see Figure 5 for more details). If the connectivity packets are received by the peers, then multimedia data can flow between the new address, port pairs.

7.4. Expanding RTCP

To provide accurate per path information an MPRTP endpoint MUST send (SR/RR) report for each unique subflow along with the overall session RTCP report. Therefore, the additional subflow reporting affects the RTCP bandwidth and the RTCP reporting interval for each subflow. RTCP report scheduling for each subflow may cause a problem for RTCP recombination and reconstruction in cases when 1) RTCP for a subflow is lost, and 2) RTCP for a subflow arrives later than the other subflows. (There may be other cases as well.)

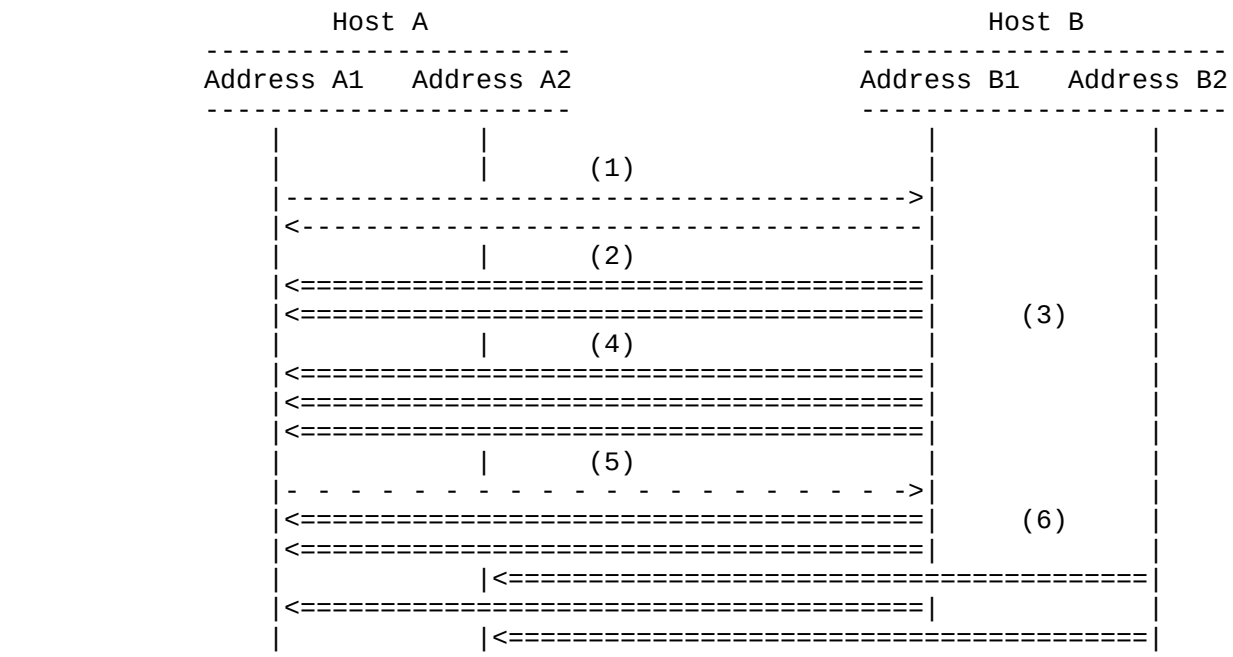
The sender distributes the media across different paths using the per path RTCP reports. However, this document doesn't cover algorithms for congestion control or load balancing.

7.5. Checking and Failure Handling

[Note: If the original interface that setup the session disappears then does the session signaling failover to another interface? Can we recommend that SIP/RTSP be run over MPTCP, SCTP].

8. MPRTP Protocol

To enable a quick start of a multimedia session, a multipath session MUST be upgraded from a single path session. Therefore, no explicit changes are needed in multimedia session setup and the session can be setup as before.



Key:
| Interface
---> Signaling Protocol
<=== RTP Packets
- -> RTCP Packet

Figure 5: MPRTP New Interface

8.1. Overview

The bullet points explain the different steps shown in Figure 5 for upgrading a standard single path multimedia session to multipath session.

(1) The first two interactions between the hosts represents the standard session setup. This may be SIP or RTSP.

(2) Following the setup, like in a conventional RTP scenario, host B using interface B1 starts to stream data to host A at interface A1.

(3) Host B is an MPRTP-capable media sender and becomes aware of another interface B2.

(4) Host B advertises the multiple interface addresses using an RTCP header extensions.

(5) Host A is an MPRTP-capable media receiver and becomes aware of another interface A2. It advertises the multiple interface addresses using an RTCP extension.

Side note, even if an MPRTP-capable host has only one interface, it SHOULD respond to the advertisement with its single interface.

(6) Each host receives information about the additional interfaces and performs the connectivity tests (not shown in figure). If the paths are reachable then the host starts to stream the multimedia content using the additional paths.

8.1.1. Subflow or Interface advertisement

To advertise the multiple interfaces, an MPRTP-capable endpoint MUST add the MPRTP Interface Advertisement defined in Figure 13 with the RTCP Sender Report (SR). Each unique address is encapsulated in an Interface Advertisement block and contains the IP address, RTP and RTCP port addresses. The Interface Advertisement blocks are ordered based on a decreasing priority level. On receiving the MPRTP Interface Advertisement, an MPRTP-capable receiver MUST respond with its own set of interfaces.

If the sender and receiver have only one interface, then the endpoints MUST respond with the default IP, RTP port and RTCP port addresses. If an endpoint receives an RTCP report without the MPRTP Interface Advertisement, then the endpoint MUST assume that the other endpoint is not MPRTP capable.

8.1.2. Path selection

After MPRTP support has been discovered and interface advertisements have been exchanged, the sender MUST initiate connectivity checks to determine which interface pairs offer valid paths between the sender and the receiver. Each combination of IP addresses and port pairs (5-tuple) is a unique subflow. An endpoint MUST associate a Subflow ID to each unique subflow.

To initiate a connectivity check, the endpoints send an RTP packet using the appropriate MPRTP extension header (See Figure 7), associated Subflow ID and no RTP payload. The receiving endpoint replies to each connectivity check with an RTCP packet with the appropriate packet type (See Figure 10) and Subflow ID. After the endpoint receives the reply, the path is considered a valid candidate for sending data. An endpoint MAY choose to do any number of

connectivity checks for any interface pairs at any point in a session.

[Open Issue: How should the endpoint adjust the RTCP Reporting interval/schedule the RTCP packet on receiving a connectivity check containing a new Subflow ID? Editor: One option is send immediately as defined in [4]. Another option is the RTCP timing defined in [17].]

8.1.3. Opening subflows

The sender MAY open any number of subflows from the set of candidate subflows after performing connectivity checks. To use the subflow, the sender simply starts sending the RTP packets with an MPRTP extension shown in Figure 8. The MPRTP extension carries a mapping of a subflow packet to the aggregate flow. Namely, sequence numbers and timestamps associated with the subflow.

An endpoint MAY use all or a subset of candidate subflows for sending media packets. To avoid redoing the connectivity checks the endpoint MAY send keep-alive MPRTP packets (see Section 9.1.3) to the passive subflows to keep the NAT bindings alive.

[Open Issue: How to differentiate between Passive and Active connections? Editor: Active paths get "regular flow" of media packets while passive paths are for failover of active paths.]

[Open Issue: How to keep a passive connection alive, if not actively used? Alternatively, what is the maximum timeout? Editor: keep-alive for ICE/NAT bindings should not be less than 15 seconds [3].]

8.2. RTP Transmission

The MPRTP layer SHOULD associate an RTP packet with a subflow based on a scheduling strategy. The scheduling strategy may either choose to augment the paths to create higher throughput or use the alternate paths for enhancing resilience or error-repair. Due to the changes in path characteristics, an MPRTP sender can change its scheduling strategy during an ongoing session. The MPRTP sender MUST also populate the subflow specific fields described in the MPRTP extension header (see Section 9.1.1).

8.3. Playout Considerations at the Receiver

A media receiver, irrespective of MPRTP support or not, should be able to playback the media stream because the received RTP packets are compliant to [1], i.e., a non-MPRTP receiver will ignore the MPRTP header and still be able to playback the RTP packets. However,

the variation of jitter and loss per path may affect proper playout. The receiver can compensate for the jitter by modifying the playout delay (i.e., by calculating skew across all paths) of the received RTP packets.

8.4. Subflow-specific RTCP Statistics and RTCP Aggregation

Aggregate RTCP provides the overall media statistics and follows the standard RTCP defined in RFC3550 [1]. However, subflow specific RTCP provides the per path media statistics because the aggregate RTCP report may not provide sufficient per path information to an MPRTP scheduler. Specifically, the scheduler should be aware of each path's RTT and loss-rate, which an aggregate RTCP cannot provide. The sender/receiver MUST use non-compound RTCP reports defined in RFC5506 [5] to transmit the aggregate and subflow-specific RTCP reports. Also, each subflow and the aggregate RTCP report MUST follow the timing rules defined in [4].

The RTCP reporting interval is locally implemented and the scheduling of the RTCP reports may depend on the the behavior of each path. For instance, the RTCP interval may be different for a passive path than an active path to keep port bindings alive. Additionally, an endpoint may decide to share the RTCP reporting bit rate equally across all its paths or schedule based on the receiver rate on each path.

8.5. RTCP Transmission

The sender sends an RTCP SR on each active path. For each SR the receiver gets, it echoes one back to the same IP address-port pair that sent the SR. The receiver tries to choose the symmetric path and if the routing is symmetric then the per-path RTT calculations will work out correctly. However, even if the paths are not symmetric, the sender would at maximum, under-estimate the RTT of the path by a factor of half of the actual path RTT.

9. Packet Formats

In this section we define the protocol structures described in the previous sections.

9.1. RTP Header Extension for MPRTP

The MPRTP header extension is used to 1) distribute a single RTP stream over multiple subflows, 2) perform connectivity checks on the advertised interfaces, and 3) keep-alive passive interfaces (paths).

The header conforms to the one-byte RTP header extension defined in [6]. The header extension contains a 16-bit length field that counts the number of 32-bit words in the extension, excluding the four-octet extension header (therefore zero is a valid length, see Section 5.3.1 of [1] for details).

To signal the use of the above RTP header extensions in SDP, the following URI MUST be used: urn:ietf:params:rtp-hdrext:mprtp.

The RTP header for each subflow is defined below:

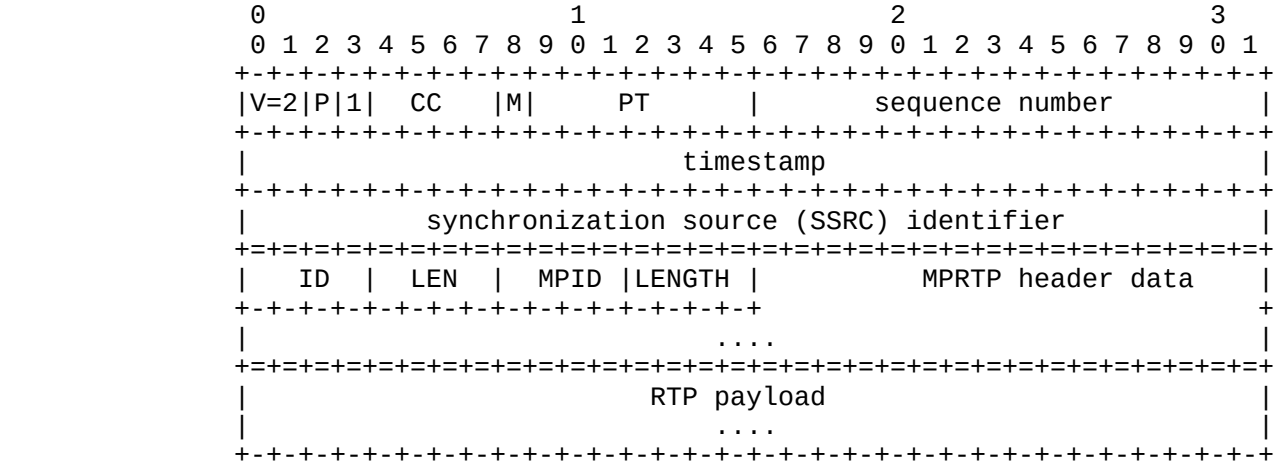


Figure 6: Generic MPRTP header extension

MPID:

The MPID field corresponds to the type of MPRTP packet. Namely:

MPID ID Value	Use
0x00	Subflow RTP Header. For this case the Length is set to 6
0x01	Connectivity Check. For this case the length is set to 0
TBD	Keep Alive Packet.

Figure 7: RTP header extension values for MPRTP (H-Ext ID)

length

The 4-bit length field is the length of extension data in bytes not including the H-Ext ID and length fields. The value zero indicates there is no data following.

MPRTP header data

Carries the MPID specific data as described in the following sub-sections.

9.1.1. MPRTP RTP Extension for a Subflow

The RTP header for each subflow is defined below:

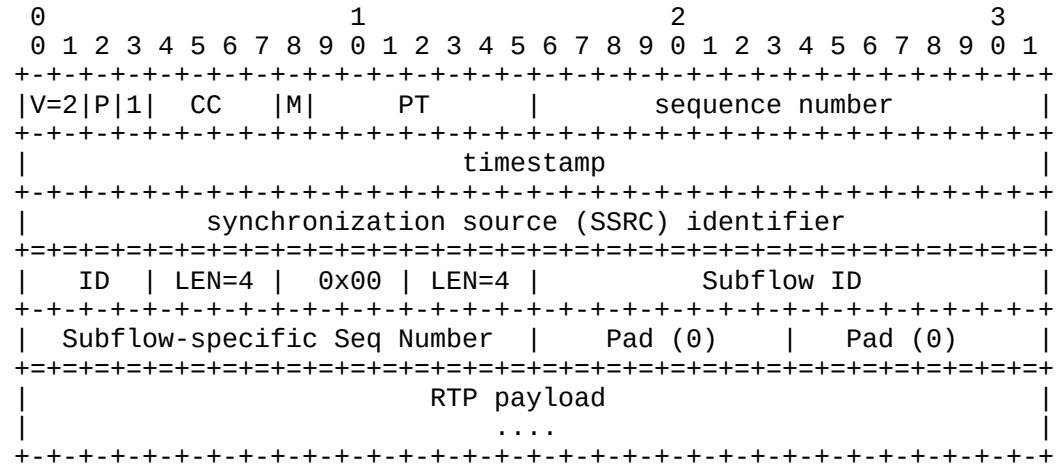


Figure 8: MPRTP header for subflow

MP ID = 0x00

Indicates that the MPRTP header extension carries subflow specific information.

length = 4

Subflow ID: Identifier of the subflow. Every RTP packet belonging to the same subflow carries the same unique subflow identifier.

Flow-Specific Sequence Number (FSSN): Sequence of the packet in the subflow. Each subflow has its own strictly monotonically increasing sequence number space.

9.1.2. MPRTP RTP Extension for Connectivity Checks

[Open Issue: What sequence number to use for the RTP session? Alternative 1: An MPRTP receiver MUST NOT give the packet with MPID=0x01 to the decoder and ignore these packets from RTCP calculation. Alternative 2: Instead of sending an RTP packet the sender transmits a modified STUN packet.]

9.1.3. MPRTP RTP Extension for Keep-alive Packets

[Editor: RTCP guidelines for keep alive packet [17] recommends multiplexing RTP and RTCP. If so, we can do the same and remove the keep alive from the list.

9.2. RTCP Extension for MPRTP (MPRTCP)

The MPRTP RTCP header extension is used to 1) provide RTCP feedback per subflow to determine the characteristics of each path, 2) advertise each interface and perform connectivity check on the other endpoint’s interfaces, and 3) to keep alive a passive connection.

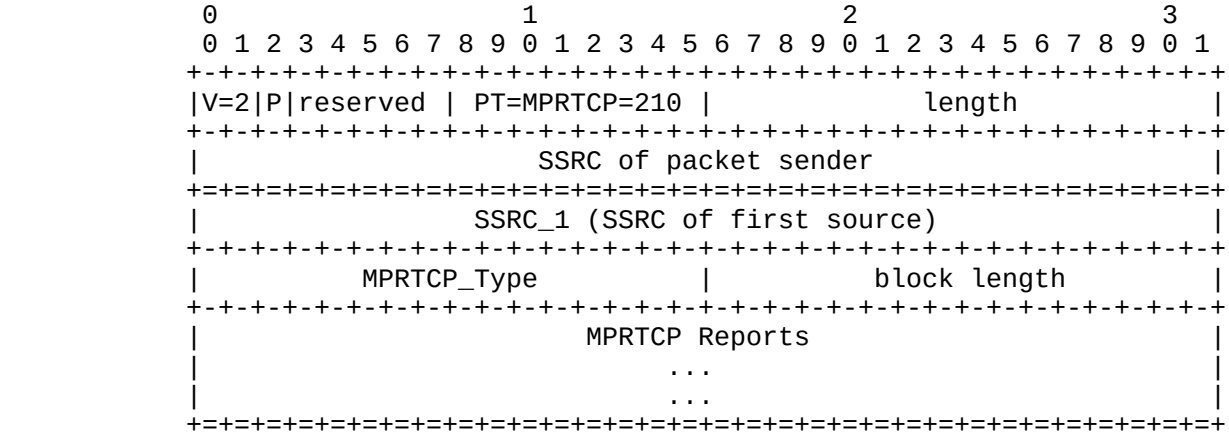


Figure 9: Generic RTCP Extension for MPRTP (MPRTCP) [appended to standard SR/RR]

MPRTCP: 8 bits

Contains the constant 210 to identify this as an Multipath RTCP packet.

length: 16 bits

As described for the RTCP packet (see Section 6.4.1 of the RTP specification [1]), the length of this is in 32-bit words minus one, including the header and any padding.

MPRTCP_Type: 16-bits

The MPRTCP_Type field corresponds to the type of MPRTP RTCP packet. Namely:

MPRTCP_Type Value	Use
0x00	Subflow Specific Report
0x01	Connectivity Check. For this case the length is set to 0
0x02	Interface Advertisement
TBD	Keep Alive Packet.

Figure 10: RTP header extension values for MPRTP (MPRTCP_Type)

block length: 16-bits

The 16-bit length field is the length of the encapsulated MPRTCP reports in 32-bit word length not including the MPRTCP_Type and length fields. The value zero indicates there is no data following.

MPRTCP Reports: variable size

Defined later in 9.2.1 and 9.3.1.

9.2.1. MPRTCP Extension for Subflow Reporting

When sending a report for a specific subflow the sender or receiver MUST add only the reports associated with that 5-tuple. Each subflow is reported independently using the following MPRTCP Feedback header.

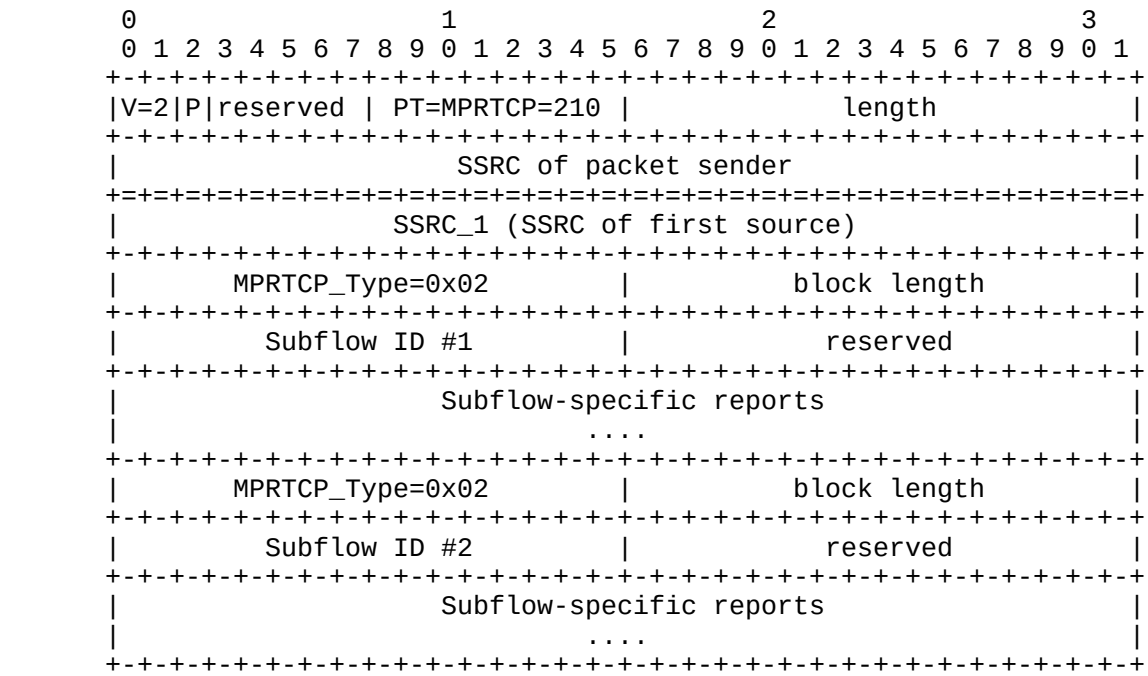


Figure 11: MPRTCP Subflow Reporting Header

MPRTCP_Type: 0x02

block length: 16-bits

The 16-bit length field is the length of the encapsulated subflow-specific reports in 32-bit word length not including the MPRTCP_Type and length fields.

Subflow ID: 16 bits

Subflow identifier is the value associated with the subflow the endpoint is reporting about. If it is a sender it MUST use the Subflow ID associated with the 5-tuple. If it is a receiver it MUST use the Subflow ID received in the Subflow-specific Sender Report.

Subflow-specific reports: variable

Subflow-specific report contains all the reports associated with the Subflow ID. For a sender, it MUST include the Subflow-specific Sender Report (SSR). For a receiver, it MUST include Subflow-specific Receiver Report (SRR). Additionally, if the

receiver supports subflow-specific extension reports then it MUST append them to the SRR.

9.2.1.1. MPRTCP for Subflow-specific SR, RR and XR

[Editor: inside the context of subflow specific reports can we reuse the payload type code for Sender Report (PT=200), Receiver Report (PT=201), Extension Report (PT=207). Transport and Payload specific RTCP messages are session specific and SHOULD be used as before.]

Example:

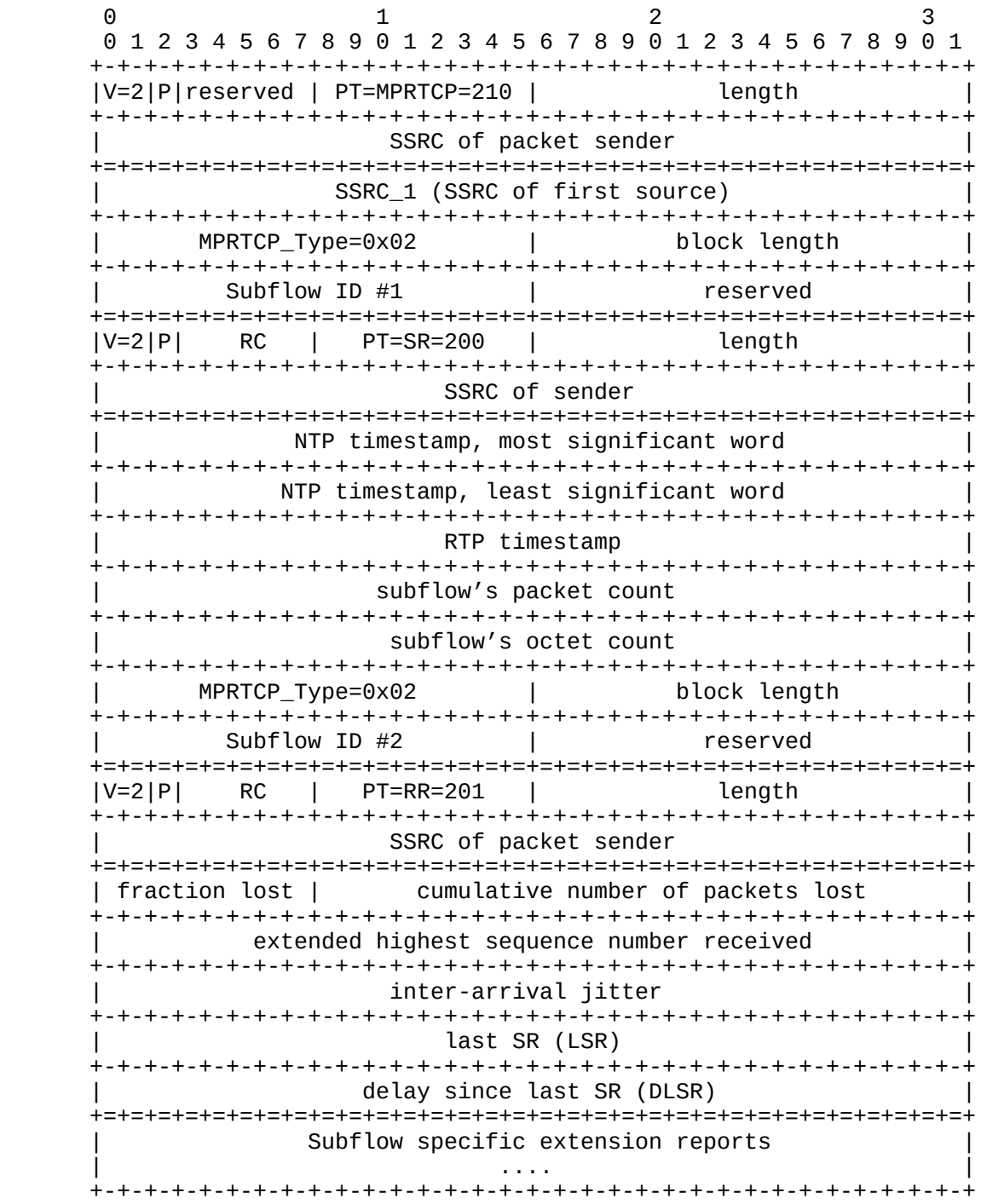

0 1 2 3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
|V=2|P|reserved | PT=MPRTCP=210 | length |
SSRC of packet sender
SSRC_1 (SSRC of first source)
MPRTCP_Type=0x02 | block length
Subflow ID #1 | reserved
|V=2|P| RC | PT=SR=200 | length |
SSRC of sender
NTP timestamp, most significant word
NTP timestamp, least significant word
RTP timestamp
subflow’s packet count
subflow’s octet count
MPRTCP_Type=0x02 | block length
Subflow ID #2 | reserved
|V=2|P| RC | PT=RR=201 | length |
SSRC of packet sender
| fraction lost | cumulative number of packets lost |
extended highest sequence number received
inter-arrival jitter
last SR (LSR)
delay since last SR (DLSR)
Subflow specific extension reports
....

Figure 12: Example of reusing RTCP SR and RR inside an MPRTCP header (Bi-directional use-case, in case of uni-directional flow the subflow will only send an SR or RR).

9.3. MPRTCP Extension for Interface advertisement

This sub-section defines the RTCP header extension for in-band interface advertisement by the receiver, instead of relying on ICE or in situations when the interface appears after SDP session establishment.

The interface advertisement SHOULD immediately follow the Receiver Report. If the Receiver Report is not present, then it MUST be appended to the Sender Report.

The endpoint MUST advertise all its interfaces when a new interface appears. Furthermore, an endpoint MUST advertise all its interfaces when it receives an Interface Advertisement.

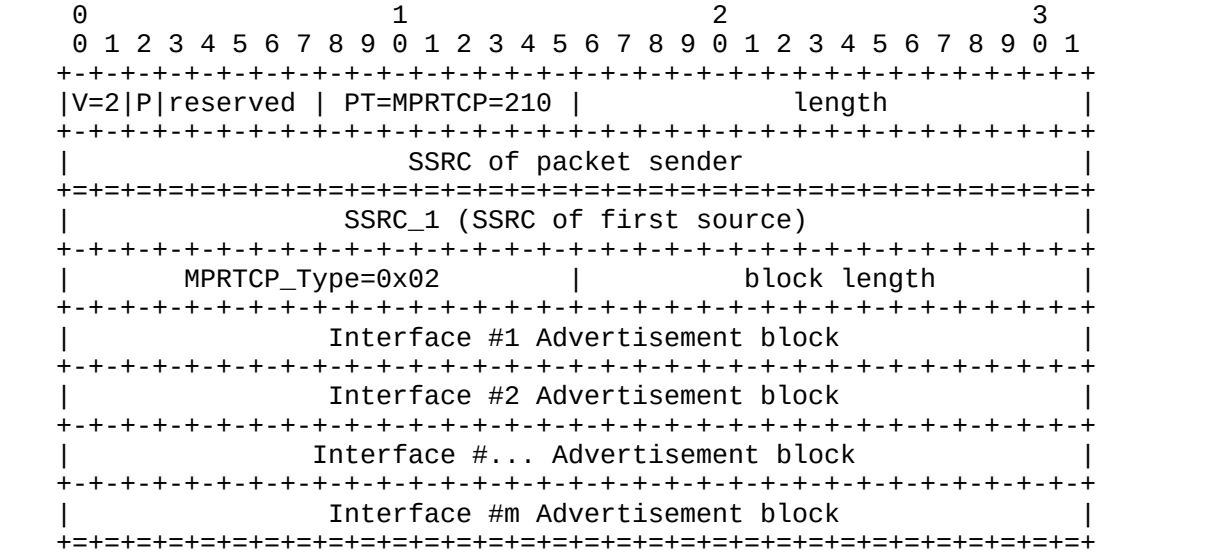


Figure 13: MPRTP Interface Advertisement. (appended to SR/RR)

MPRTCP_Type: 0x00

block length: 16-bits

The 16-bit length field is the length of the encapsulated interface advertisement blocks in 32-bit word length not including the MPRTCP_Type and length fields.

Interface Advertisement block: variable size

Defined later in 9.3.1.

9.3.1. Interface Advertisement block

This block describes a method to represent IPv4, IPv6 and generic DNS-type addresses in a block format. It is based on the sub-reporting block in [7].

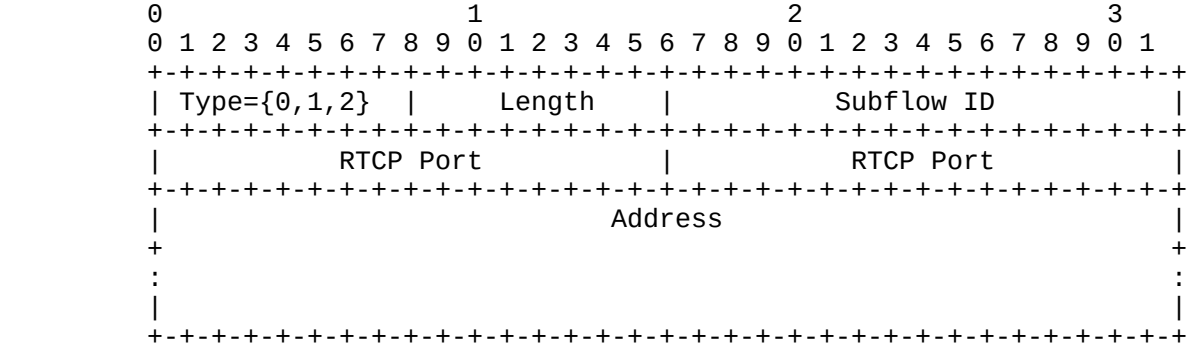


Figure 14: Interface Advertisement block during path discovery

Type: 8 bits

The Type corresponds to the type of address. Namely:

0: IPv4 address

1: IPv6 address

2: DNS name

Length: 8 bits

The length of the Interface Advertisement block in bytes.

For an IPv4 address, this should be 9 (i.e., 5 octets for the header and 4 octets for IPv4 address).

For an IPv6 address, this should be 21.

For a DNS name, the length field indicates the number of octets making up the string plus the 5 byte header.

RTP Port: 2 octets

The port number to which the sender sends RTP data. A port number of 0 is invalid and MUST NOT be used.

RTCP Port: 2 octets

The port number to which receivers send feedback reports. A port number of 0 is invalid and MUST NOT be used.

Address: 4 octets (IPv4), 16 octets (IPv6), or n octets (DNS name)

The address to which receivers send feedback reports. For IPv4 and IPv6, fixed-length address fields are used. A DNS name is an arbitrary-length string. The string MAY contain Internationalizing Domain Names in Applications (IDNA) domain names and MUST be UTF-8 [8] encoded.

10. RTCP Timing reconsiderations for MPRTCP

MPRTP endpoints MUST conform to the timing rule imposed in [4], i.e., the total RTCP rate between the participants MUST NOT exceed 5% of the media rate. For each endpoint, a subflow MUST send the aggregate and subflow-specific report. The endpoint SHOULD schedule the RTCP reports for the active subflows based on the share of the transmitted or received bit rate to the average media bit rate, this method ensures fair sharing of the RTCP bandwidth. Alternatively, the endpoint MAY schedule the reports in round-robin.

11. SDP Considerations

The packet formats specified in this document define extensions for RTP and RTCP. The use of MPRTP is left to the discretion of the sender and receiver.

11.1. Signaling MPRTP capability in SDP

A participant of a media session MAY use SDP to signal that it supports MPRTP. Not providing this information may/will make the sender or receiver ignore the header extensions. However, MPRTP MAY be used by either sender or receiver without prior signaling.

```
mprtp-attrib = "a=" "mprtp" [
        SP mprtp-optional-parameter]
        CRLF   ; flag to enable MPRTP
```

The literal 'mprtp' MUST be used to indicate support for MPRTP. Generally, senders and receivers MUST indicate this capability if they support MPRTP and would like to use it in the specific media session being signaled. They can then use the in-band signaling to exchange the additional interfaces and to perform connectivity checks.

11.2. Signaling MPRTP Header Extension in SDP

To signal the use of MPRTP header extensions in SDP, the following URI MUST be used: urn:ietf:params:rtp-hdrext:mprtp. Legacy RTP (non-MPRTP) clients MUST ignore this header extension, but continue to parse and decode the packet.

11.3. MPRTP using preloaded interfaces from ICE (out-band signaling)

If the endpoints intend to use ICE [3] for discovering new interfaces and running connectivity checks then these endpoints MUST NOT use in-band interface advertisement and the associated connectivity checks, as described in Section 9.1.2 (MPID=0x01) and Section 9.3 (MPRTCP_Type=0x01, 0x02).

Using ICE for MPRTP is a two step process:

1. Advertising ICE candidates: in the initial INVITE the endpoints exchange candidates, as defined in ICE [3]. Thereafter the endpoints run connectivity checks

2. Advertising MPRTP interfaces: When a sufficient number of connectivity checks succeed the endpoint MUST send an updated offer containing their interfaces.

This process enables the participants to use MPRTP capabilities from the start of a media session.

11.3.1. "interface" attribute

The interface attribute is only a media-level attribute. It is used to describe one of the many network addresses of an endpoint.

The syntax of the interface attribute is defined using the following Augmented BNF, as defined in [9]. It reuses the RFC4566 [18] definitions of unicast-address, port, token, SP, and CRLF.

```
mprtp-optional-parameter = mprtp-interface
                ; other optional parameters may be added later

mprtp-interface = "interface" ":" counter SP unicast-address
                    ":" rtp_port "/" rtcp_port
                     *(SP interface-description-extension)

counter = 1*DIGIT
rtp_port = port    ;port from RFC4566
rtcp_port = port   ;port from RFC4566
```

<mprtp-interface>: specifies one of the unicast IP address, the RTP and RTCP port numbers of the endpoint. The unicast address in the first interface line MUST match the connection address (’c=’ line). The RTP and RTCP ports in the first interface line MUST match with the RTP and RTCP ports in the associated ’m=’ line. The counter should start at 1 and increment with each additional interface. Multiple interface lines MUST be ordered in a decreasing priority level as is the case with the Interface Advertisement blocks in in-band signaling (See Figure 13).

<unicast-address>: is taken from RFC5245 [3]. It is one of the IP addresses of the endpoint allows the use of IPv4 addresses, IPv6 addresses and Fully Qualified Domain Names (FQDN). An endpoint MUST only include the IP address for which the connectivity checks have succeeded.

<port>: is from RFC4566 [18]. It is the RTP or RTCP port associated the unicast address for which the connectivity checks have succeeded.

<counter>: is a monotonically increasing positive integer starting at 1. The counter SHOULD reset for each media line.

The ’mprtp-interface’ can be extended using the ’interface-description-extension’ parameter. An endpoint MUST ignore any extensions it does not understand.

11.3.2. Example

The ABNF grammar is illustrated by means of an example:

```
v=0
o=alice 2890844526 2890844527 IN IP4 192.0.2.1
s=
c=IN IP4 192.0.2.1
t=0 0
m=video 49170 RTP/AVP 98
a=rtpmap:98 H264/90000
a=fmtp:98 profile-level-id=42A01E;
a=mprtp interface:1 192.0.2.1:49170/49171    ;primary interface
a=mprtp interface:2 192.1.2.1:51372/51373    ;additional interface
```

11.4. Increased Throughput

The MPRTP layer MAY choose to augment paths to increase throughput. If the desired media rate exceeds the current media rate, the endpoints MUST renegotiate the application specific ("b=AS:") [18] bandwidth.

11.5. Offer/Answer Examples

This section shows examples of SDP offer and answer for in-band and out-band signaling.

11.5.1. In-band signaling

The following offer/answer shows that both the endpoints are MPRTP capable and SHOULD use in-band signaling for exchanging interfaces and for performing connectivity checks. If candidate addresses are included in the initial INVITE then the connectivity checks are performed by ICE and the out-band signaling is preferred.

Offer:

```
  v=0
  o=alice 2890844526 2890844527 IN IP4 192.0.2.1
  s=
  c=IN IP4 192.0.2.1
  t=0 0
  m=video 49170 RTP/AVP 98
  a=rtpmap:98 H264/90000
  a=fmtp:98 profile-level-id=42A01E;
  a=mprtp
```

```
   Answer:
     v=0
     o=bob 2890844528 2890844529 IN IP4 192.0.2.2
     s=
     c=IN IP4 192.0.2.2
     t=0 0
     m=video 4000 RTP/AVP 98
     a=rtpmap:98 H264/90000
     a=fmtp:98 profile-level-id=42A01E;
     a=mprtp
```

11.5.2. Out-band signaling

In this example, the offerer advertises two interfaces and the answerer responds with a single interface description. The endpoint MAY use one or both paths depending on the end-to-end characteristics of each path.

```
   Offer:
     v=0
     o=alice 2890844526 2890844527 IN IP4 192.0.2.1
     s=
     c=IN IP4 192.0.2.1
     t=0 0
     m=video 49170 RTP/AVP 98
     a=rtpmap:98 H264/90000
     a=fmtp:98 profile-level-id=42A01E;
     a=mprtp interface:1 192.0.2.1:49170/49171
     a=mprtp interface:2 192.1.2.1:51372/51373


   Answer:
     v=0
     o=bob 2890844528 2890844529 IN IP4 192.0.2.2
     s=
     c=IN IP4 192.0.2.2
     t=0 0
     m=video 4000 RTP/AVP 98
     a=rtpmap:98 H264/90000
     a=fmtp:98 profile-level-id=42A01E;
     a=mprtp interface:1 192.0.2.2:4000/4001
```

12. IANA Considerations

TBD

13. Security Considerations

TBD

All drafts are required to have a security considerations section. See RFC 3552 [19] for a guide.

14. Acknowledgements

Varun Singh, Saba Ahsan, and Teemu Karkkainen are supported by Trilogy (http://www.trilogy-project.org), a research project (ICT-216372) partially funded by the European Community under its Seventh Framework Program. The views expressed here are those of the author(s) only. The European Commission is not liable for any use that may be made of the information in this document.

The authors would also like acknowledge the contribution of Ralf Globish and Thomas Schierl for providing the input and text for the MPRTP interface advertisement in SDP.

15. References

15.1. Normative References

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

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

[3] Rosenberg, J., "Interactive Connectivity Establishment (ICE): A Protocol for Network Address Translator (NAT) Traversal for Offer/Answer Protocols", RFC 5245, April 2010.

[4] Ott, J., Wenger, S., Sato, N., Burmeister, C., and J. Rey, "Extended RTP Profile for Real-time Transport Control Protocol (RTCP)-Based Feedback (RTP/AVPF)", RFC 4585, July 2006.

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

[6] Singer, D. and H. Desineni, "A General Mechanism for RTP Header Extensions", RFC 5285, July 2008.

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

[8] Yergeau, F., "UTF-8, a transformation format of ISO 10646", STD 63, RFC 3629, November 2003.

[9] Crocker, D. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", STD 68, RFC 5234, January 2008.

15.2. Informative References

[10] Stewart, R., "Stream Control Transmission Protocol", RFC 4960, September 2007.

[11] Ford, A., Raiciu, C., Handley, M., Barre, S., and J. Iyengar, "Architectural Guidelines for Multipath TCP Development", RFC 6182, March 2011.

[12] Nordmark, E. and M. Bagnulo, "Shim6: Level 3 Multihoming Shim Protocol for IPv6", RFC 5533, June 2009.

[13] Westerlund, M. and S. Wenger, "RTP Topologies", RFC 5117, January 2008.

[14] 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.

[15] Schulzrinne, H., Rao, A., Lanphier, R., Westerlund, M., and M. Stiemerling, "Real Time Streaming Protocol 2.0 (RTSP)", draft-ietf-mmusic-rfc2326bis-27 (work in progress), March 2011.

[16] Rosenberg, J. and H. Schulzrinne, "An Offer/Answer Model with Session Description Protocol (SDP)", RFC 3264, June 2002.

[17] Marjou, X. and A. Sollaud, "Application Mechanism for Keeping Alive the NAT Mappings Associated with RTP / RTP Control Protocol (RTCP) Flows", RFC 6263, June 2011.

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

[19] Rescorla, E. and B. Korver, "Guidelines for Writing RFC Text on Security Considerations", BCP 72, RFC 3552, July 2003.

Authors' Addresses

Varun Singh
Aalto University
School of Science and Technology
Otakaari 5 A
Espoo, FIN 02150
Finland

Email: varun@comnet.tkk.fi
URI: http://www.netlab.tkk.fi/~varun/

Teemu Karkkainen
Aalto University
School of Science and Technology
Otakaari 5 A
Espoo, FIN 02150
Finland

Email: teemuk@comnet.tkk.fi

Joerg Ott
Aalto University
School of Science and Technology
Otakaari 5 A
Espoo, FIN 02150
Finland

Email: jo@comnet.tkk.fi

Saba Ahsan
Aalto University
School of Science and Technology
Otakaari 5 A
Espoo, FIN 02150
Finland

Email: sahsan@cc.hut.fi

Lars Eggert
Nokia Research Center
P.O. Box 407
Nokia Group 00045
Finland

Phone: +358 50 48 24461
Email: lars.eggert@nokia.com
URI: http://research.nokia.com/people/lars_eggert

AVTCORE
Internet-Draft
Intended status: Standards Track
Expires: November 27, 2011

T. VanCaenegem
Alcatel-Lucent
B. VerSteeg
A. Begen
Cisco

May 26, 2011

Retransmission for Source-Specific Multicast (SSM) Sessions draft-vancaenegem-avtcore-retransmission-for-ssm-00

Abstract

This document describes RTP retransmission for source-specific multicast (SSM) architectures with unicast feedback. RTP payload format for retransmissions has been defined in RFC 4588, whereas the RTP profile an RTP receiver could use to issue RTP Control Protocol (RTCP) feedback messages and the format of these feedback messages have been defined in RFC 4585. RFC 5760 defines the operation for SSM architectures with unicast feedback.

First, we document potential issues that could arise when providing a retransmission service using RTP retransmission (RFC 4588 and RFC 4585) for RFC 5760 architectures based on rules described in the relevant RFCs. We then present solutions that allow to avoid unnecessary feedback suppression, provide enhanced retransmission services and address congestion control for retransmission in an SSM environment. These solutions specify certain rules that apply to the distribution source, feedback target(s), retransmission source(s) and RTP receivers.

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 November 27, 2011.

Table of Contents

1. Introduction

The RTP toolkit is used to deliver a variety services over IP multicast. [RFC5760] defines a source-specific multicast (SSM) architecture with one or several Media Senders, a Distribution Source (DS) (sourcing the SSM), one or several Feedback Targets (FT) that may or may not be co-joint with the DS and RTP receivers (RTP_Rx) that send unicast feedback to a FT. This document explores the complications one might face when adding one or more Retransmission Sources (RSo) in this architecture for reliable multicast distribution.

In this specification, we assume that FT and RSo logical entities are combined in a single physical entity and they share state. In the remainder of the text, the term Retransmission Server (RS) is used whenever appropriate, to refer to this single physical entity. As in [I-D.ietf-avt-rapid-acquisition-for-rtp], the RS receives the RTP packets sent in an SSM session (as a normal RTP_Rx), and buffers these packets for a certain time for retransmission purposes.

The term RAMS (Rapid Acquisition of Multicast Sessions) will be used to refer to the protocol and architecture defined in [I-D.ietf-avt-rapid-acquisition-for-rtp]. The RAMS specification presents a method in which an RTP_Rx can rapidly acquire reference information transported in an SSM session. This is enabled by having the RTP_Rx retrieve a unicast burst of data from an RS, formatted as RTP retransmission packets prior to joining the SSM session. While the present document will include some requirements and observations expressed in [I-D.ietf-avt-rapid-acquisition-for-rtp], it will particularly focus on the impact of packet loss events experienced in an SSM distribution, and how the elements in the SSM architecture interact best to ensure impacted RTP_Rx's are provided with appropriate RTP retransmissions.

A packet loss event in an SSM session, will in general trigger one or multiple interactions between the RTP_Rx and the FT/RS/DS entities:

- The RTP_Rx sending an RTCP NACK message to the FT ([RFC4585])
- The RTP_Rx receiving RTP retransmission(s) from the RS
- The RTP_Rx receiving an RTCP feedback message from the DS

This document describes the behavior for all entities defined in an SSM architecture with unicast feedback as described in [RFC5760] (DS, FT, RS and RTP_Rx) addressing feedback implosion (avoidance), retransmission efficacy and congestion concerns.

One application that can benefit from this document is IPTV. In many IPTV deployments, streams are transported over SSM. Transporting this data over SSM enables a packet loss recovery by means of RTP retransmissions and/or forward error correction (FEC) [I-D.ietf-fecframe-dvb-al-fec]. This document focuses on retransmission. Note that Digital Video Broadcasting (DVB) forum has specified an RTP retransmission solution for its managed Linear Media Broadcast (LMB) service, and which references most of the relevant RTP specifications. An overview of this solution is provided in Section 9.

2. 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. Definitions

This document uses some acronyms and definitions which have been introduced [I-D.ietf-avt-rapid-acquisition-for-rtp] as the baseline. Some of these definitions have been slightly retailored to fit the restricted scope of this document.

(Primary) SSM (or Multicast) Session: The multicast session to which RTP_Rx's can join at a random point in time. A primary SSM session can carry multiple RTP streams.

Primary Multicast RTP Session: The multicast RTP session that is of interest to an RTP receiver. It comprises the primary multicast RTP stream(s) and an associated unicast RTCP feedback stream to a Feedback Target.

Primary Multicast (RTP) Streams: The RTP stream(s) carried in the primary multicast RTP session.

Source Filtering Group Management Protocol (SFGMP): Following the definition in [RFC4604], SFGMP refers to the Internet Group Management Protocol (IGMP) version 3 [RFC3376] and the Multicast Listener Discovery Protocol (MLD) version 2 [RFC3810] in the IPv4 and IPv6 networks, respectively. In the remainder of this document, SFGMP will refer to any group management protocol that has Join and Leave functionalities.

Feedback Target (FT): Unicast RTCP feedback target as defined in

[RFC5760]. FT_Ap denotes a specific feedback target running on a particular address and port.

Retransmission Packet: An RTP packet that is formatted as defined in Section 4 of [RFC4588]. The payload of a retransmission packet comprises the retransmission payload header followed by the payload of the original RTP packet.

(Unicast) Retransmission RTP Session: The unicast RTP session used to send one or more unicast retransmission RTP streams and their associated RTCP messages. This session is setup as described in [I-D.ietf-avtcore-ports-for-ucast-mcast-rtp].

Retransmission Server (RS): The RTP/RTCP endpoint that can generate the retransmission packets.

New definitions introduced in this document are:

Solicited Retransmission: Retransmission packet(s) that are received by an RTP_Rx which have been explicitly requested by this RTP_Rx by means of one or more RTCP NACKs.

Unsolicited Retransmission: Retransmission packet(s) that are received by an RTP_Rx which have not been explicitly requested by this RTP_Rx by means of an RTCP NACK.

Feedback Implosion/Storm: The situation where multiple RTP receivers start sending (quasi-) simultaneously (RTCP) feedback, often triggered by an upstream packet loss event in the SSM tree impacting a large set of RTP receivers.

4. Feedback Suppression and Retransmission for SSM: Applicable Specifications

In this section we first look at the various specifications in place that describe the architecture for SSM with unicast feedback and determine how feedback suppression and retransmission are accomplished.

4.1. RTCP Feedback Suppression in RFC 4585

[RFC4585] defines an RTP profile for an RTP receiver that wishes to provide fast feedback by means of RTCP feedback messages, such as is applicable to an RTP retransmission scenario. [RFC4585] defines the timing rules by which such RTCP feedback messages can be transmitted both in unicast and multicast/multi-party sessions, and it also defines the general format of RTCP feedback messages, as well as some

specific feedback messages. One of the transport-layer feedback messages that is particularly relevant here is the RTCP NACK message through which an RTP receiver can report missing packets. The missing packets are identified by means of an SSRC identifier (associated with the primary multicast stream for which the DS uses this SSRC as media sender) and an RTP sequence number (SN). The RS receiving such an RTCP NACK message can respond by sending retransmission packets whose format has been described in [RFC4588]. An important aspect covered by [RFC4585] is feedback suppression for RTP receivers involved in a multi-party session in order to avoid feedback implosion. To that extent, the RTP receiver waits for a (short) random dithering interval to check whether it sees a feedback message from any other RTP receiver reporting the same event. If such a feedback message is received, the RTP receiver refrains from sending the feedback message and continues to follow the regular RTCP transmission schedule. Note that a feedback message from one RTP receiver is typically not visible to other RTP receivers, so the FT might need to provide information to the RTP receivers. We will investigate the implications of this feedback suppression rule for SSM architectures with unicast feedback as defined in [RFC5760], where one or multiple RS's are deployed. To that extent we first provide an overview of how RTCP feedback messages transmitted by the RTP receivers are handled in the SSM architecture as specified by [RFC5760].

4.2. RTCP Feedback Message Distribution in RFC 5760

Two models in terms of DS behavior have been defined in [RFC5760]:

- In Feedback Summary Model, the unicast RTCP receiver report messages from the RTP_Rx's are by default aggregated at the DS and their information is transmitted as Receiver Summary Information (RSI) messages in the SSM session. The RTCP feedback messages are by default terminated by the DS. However, the DS might also aggregate or forward RTCP feedback messages and transmit them in the SSM session, when this is explicitly signaled. Note that from the RTP perspective, the DS is an RTP_Rx generating its own RTCP receiver reports as well as other RTCP packets and sending them to the receiver group and media senders.

- In Simple Feedback Model, the DS reflects all RTCP messages (including RTCP feedback) received in unicast via the FT from the RTP_Rx's. Note that many network topologies have a high degree of fanout, and also have a constrained link between the FT and the RTP_Rx's, so that reflecting all of the feedback messages is often not feasible.

The communication between DS and disjoint FT(s) occurs as follows:

[RFC5760] indicates that for the Simple Feedback Model where the FT(s) are disjoint from the DS, the FT must forward all RTCP packets to the DS.

[RFC5760] indicates the following for the Feedback Summary Model where the FT(s) are disjoint from the DS:

- The FTs may simply forward all RTCP packets incoming from the RTP_Rx's to the DS.
- The FTs may also perform aggregation of incoming RTCP packets and send only aggregated information to the DS.
- If the FT performs summarization functions, it must also act as a receiver and choose a unique SSRC for its own reporting towards the DS.

4.3. RAMS

Assume now that the FT is combined with an RSo, constituting together the RS, which provides retransmissions in response to unicast RTCP NACK messages received from RTP_Rx's. Such architecture is described in [I-D.ietf-avt-rapid-acquisition-for-rtp] where the FT is co-joint with a Burst/Retransmission Source (BRS).

Figure 1 shows the main entities involved in SSM with retransmission enabled. They are

- Distribution Source
- Feedback Target (FT)
- Retransmission Source (RSo)
- RTP Receiver (RTP_Rx)

The figure also shows the various RTP/RTCP flows and associated sessions. The difference with RAMS is that the Burst/Retransmission Source (BRS) is replaced with a Retrasmission Source (RSo). Another difference - not shown in the figure - is that next to the primary multicast session, there might also be a (source-specific) multicast retransmission session that can be sourced by the same DS entity (acting as an RS itself) or by the same RS that provides the unicast retransmissions. This is discussed in more detail in Section 6.3. The unicast retransmission session must be established as per [I-D.ietf-avtcore-ports-for-ucast-mcast-rtp]. Note that when RAMS and retransmissions are provided for the same SSM session, a single unicast retransmission session is established. The RAMS

specification puts a constraint on the RS/FT, stating that an FT (identified by means of IP address and port, and referred to as FT_Ap) must be associated with only a single RTP session. This constraint was put in place, because an RTP_Rx using the RAMS service may not necessarily know the media sender SSRC a priori. However, an RTP_Rx using the retransmission service can of course learn the SSRC by inspecting the SSRC identifier field in the RTP packets, and furthermore the SSRC field needs to be correctly filled out in the RTCP NACK message. Thus, while we do not have this constraint on FTs when only the retransmission service is enabled, the contraint still applies when both RAMS and retransmission services are enabled.

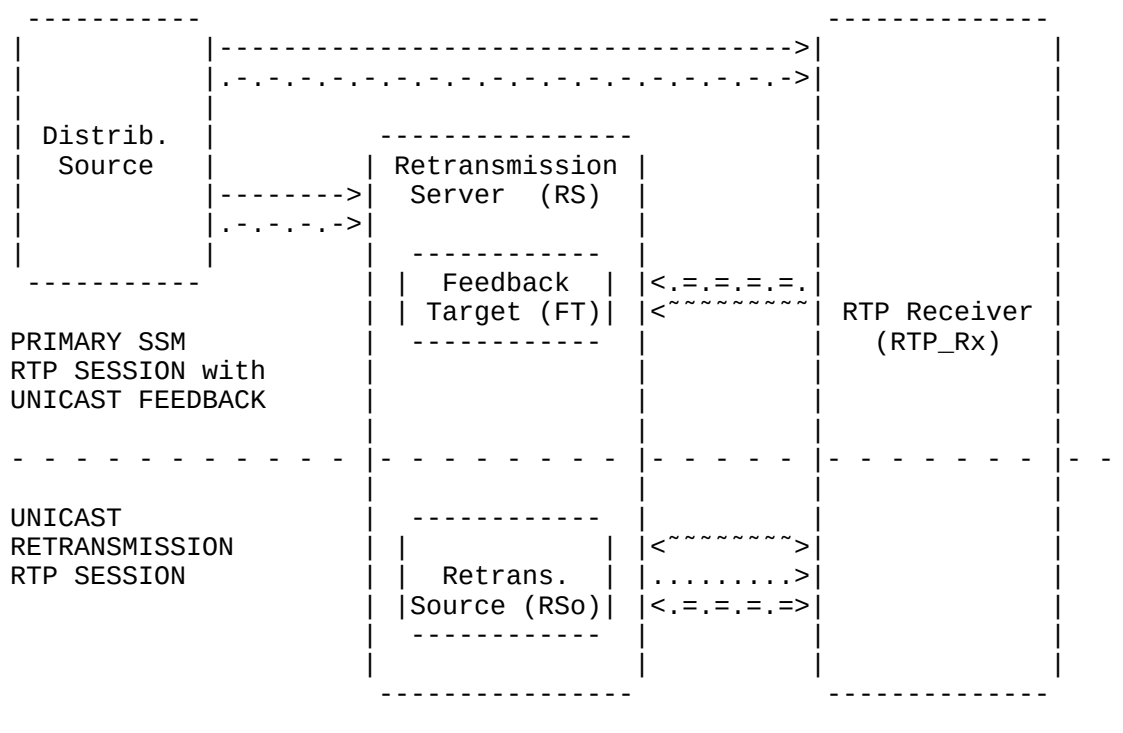


Figure 1: Flow diagram for SSM with retransmission

5. Pitfalls When Combining RTCP Feedback Handling and Retransmission Using RFC 5760 Rules

In this section, we investigate a number of scenarios in terms of SSM topology and behavior of the DS with respect to the forwarding/ termination of RTCP feedback messages received from RTP_Rx's. In the first two scenarios, we assume there is only one FT. Note that the DS could be joint with the FT or disjoint from the FT.

5.1. Case of DS Forwarding/Reflecting All RTCP NACKs

The forwarding of RTCP feedback messages is one option allowed for by the Feedback Summary Model of [RFC5760], where reflection is what the DS must do based on the Simple Feedback Model of RFC 5760. In this case, RTP_Rx's applying the [RFC4585]suppression rule will result in the following: when an RTP_Rx X reports a packet loss to the FT by sending an RTCP NACK, this message is distributed, via the FT and, by the DS via the SSM to all other RTP_Rx's. If an RTP_Rx Y, different from RTP_Rx X, detected the same packet loss, but prior to sending out a NACK, this receiver Y gets the NACK message originally sent by X, it will refrain from sending a NACK, following [RFC4585]. The forwarding/reflection of RTCP NACK messages effectively results in feedback suppression, but obviously this is at the expense of limited visibility to the RS on which RTP_Rx suffered packet loss. In general this will result in a decreased quality at the service layer for any RTP_Rx X, because the SSM receiver does not even get an opportunity to report its packet loss by means of an RTCP NACK. Note that for packet losses upstream of the DS and which subsequently impact all RTP_Rx's, the DS does not need to be informed by each individual RTP_Rx about this packet loss. When the DS is capable of recovering the lost packet in due time, it may send an unsolicited retransmission to all RTP_Rx's. The reduced visibility to the RS of packet losses taking place downstream of the DS, could be compensated for by having the RS sending an unsolicited retransmission, whenever an RTCP NACK is received. This will increase the chance on network congestion and will also consume RS processing resources when retransmissions are provided over unicast. The conclusion is that reflection/forwarding of RTCP NACK messages by the DS is not (always) the "right" thing to do.

5.2. Case of DS Terminating All RTCP NACKs

Not forwarding and hence terminating RTCP feedback messages is the alternative option for a DS behaving in [RFC5760] Feedback Summary Model. In this case every RTP_Rx that detects a packet loss event, will also transmit an RTCP NACK (assuming no congestion is sensed by the receiver, see Section 6.4.2). We look at two example scenarios resulting in different consequences: As a first example: two or

multiple RTP_Rx's detect the same packet loss, which was caused by two or multiple un-related packet loss events taking place (quasi-) simultaneously. This could be caused by link transmission errors, such as on xDSL links or in cable networks. In this case, the RS will receive the NACKs from all impacted receivers, and can act accordingly by sending a unicast (solicited) retransmission to the impacted receivers. As a second example, take the case considered previously where a packet loss event took place on the link between a media sender and the DS. Because, even though the transmissions could be dispersed in a time frame Delta T, when a very large number of SSM receivers are impacted by the single packet loss event, a feedback implosion might occur. This may load the network link(s) downstream of the RS and the RS itself, beyond its processing capabilities. Clearly, this is a case where the feedback suppression rule of [RFC4585] would be useful. The conclusion is that termination of RTCP NACK messages by the DS is not (always) the "right" thing to do.

5.3. Case of Multiple FTs

A specific case of SSM with unicast feedback, is where there are multiple FTs, disjoint from the DS. Similar as before, in the considered architectures, each FT is combined with a retransmission source, constituting a retransmission server. Note that the RS are generally not positioned in the direct SSM path between the DS and the RTP_Rx's, i.e., an RS is typically a leaf of the SSM tree. This architecture provides a scalable solution for SSM with a large population of receivers, because it is able to distribute RTCP feedback processing loads across different entities in different parts of the network. It is an architecture that is well suited for IPTV networks, where the DS is the head-end sourcing the SSM that carries video broadcast streams over IP. Note that we again assume that the RTP_Rx's behave compliant to [RFC4585], inclusive conformance to the feedback suppression rule. The previous discussions on feedback suppression (or absence of feedback suppression) and its consequences on retransmissions for the SSM with single FT topology remains applicable and valid for the SSM with multiple (disjoint) FTs topology. A distributed FT architecture brings in an additional aspect that should be addressed, and a dedicated example packet loss event use case visualizes this. The considered topology is a DS with two disjoint FT/RS entities, FT/RS 1 and FT/RS 2 , where each FT receives RTCP messages from a separate group of RTP_Rx's. The assumption is that

- An RTP packet (with RTP sequence number N) in the primary SSM got dropped in the network upstream of the FT 1 and upstream of the SSM receivers that report to FT 1.

- The FT 2 and the SSM receivers reporting to FT 2 are not impacted by the packet loss event impacting FT 1. This is because the FT 2 and its reporting SSM receivers do not get the original SSM packets from the DS via the router where the packet loss impacting FT 1 took place.

The FT 1 and its reporting SSM receivers experience a situation that is discussed in a previous section. Because the packet loss event impacts all SSM receivers reporting to FT 1, it is important that those receivers in general suppress sending an RTCP NACK. Hence having the FT 1 forwarding the first received RTCP NACK(s) from an RTP_Rx to the DS which then reflects/forwards the NACK over the original SSM, is the correct thing to do from that point of view. However, the reflection /forwarding of the NACK by the DS means that also the RTP_Rx's reporting to the FT 2 will suppress sending an RTCP NACK for packet N in the SSM. As a consequence, a packet loss event involving an RTP packet with SN N impacting a single RTP_Rx reporting to FT 2 / RS 2 -e.g. due to a link transmission error- will not be reported by this RTP_Rx to RS 2.

This means there is a discrepancy between the network reach of the suppression (covering all SSM receivers) and the actual network domain that was commonly impacted by the packet loss. The RS 2 will in general not know whether there are any SSM receivers (reporting to FT 2) that missed RTP packet with RTP SN N. Note also that an unsolicited retransmission by RS 1 optionally following the loss detection for packet with SN N -can remain confined to the subdomain impacted by the loss, when the FT is co-located with the RS (using either unicast retransmission sessions or a multicast retransmission session sourced by the RS).

SSM with multiple disjoint unicast FTs hence may result in efficient feedback storm suppression across all RTP_Rx's, but this also prevents any RTP_Rx from sending an RTCP NACK for detected packet loss, even when no feedback storm was imminent for the subdomain covered by a particular FT. A solution for maximizing the retransmission service fulfillment may be for the DS to also act as RS and always send retransmissions requested by a particular FT, over a separate retransmission SSM to all RTP_Rx's. However, this unnecessarily loads the network and requires all the SSM RTP receivers to join both a primary SSM and the multicast retransmission session.

5.4. Takeaways

The general conclusion from these three scenarios is that feedback suppression enabled by having the DS forwarding/reflecting RTCP Messages received from the RTP_Rx's or (disjoint) FTs is sometimes

the right thing to do, but sometimes feedback suppression is not appropriate and interferes in a negative way with the packet loss reporting and retransmission service.

6. Rules for SSM with Retransmissions

6.1. Feedback Suppression Behavioral Requirements for DS, FT and RTP_Rx's

The combination of the [RFC4585] suppression rule imposed on RTP_Rx's and the behavior of the DS and the FT(s) -when disjoint from the DS- as specified in [RFC5760], does result in non-ideal situations when deploying an SSM RTP session with Retransmission Server(s), especially for a large group of receivers. In this section we provide some rules for DS and FT on the one hand and RTP_Rx's on the other hand, which impose a deviation respectively from rules defined in [RFC5760] and [RFC4585] , and which results in:

- All RTP_Rx's offering the opportunity to report detected packet loss by means of RTCP NACK when there is no risk on feedback implosion
- RTCP NACK suppression (only) when RTCP NACK implosion is imminent
- A better retransmission efficacy

The rules apply to the transmission and handling of RTCP NACK messages by RTP_Rx's and FT, but similar rules may be applicable for other RTCP feedback messages that may or may not trigger dedicated service actions.

Note that in the context of this document, the FT is part of a Retransmission Server, which is an RTP_Rx on its own. In the remainder of this text, the term 'RTP_Rx' always refers to any RTP receiver reporting to a FT/RS and which does not host a FT/RS functionality, unless mentioned otherwise.

The rules are:

1. A FT/RS that is disjoint from the DS MUST NOT forward nor reflect RTCP NACK messages received from the RTP_Rx's, to the DS.

   This behavior for a FT operating in an SSM enabled with retransmission is different from the behavior prescribed by [RFC5760] for a FT with a primary SSM operating in Simple Feedback mode.This rule will prevent feedback suppression at the RTP_Rx's directly triggered by RTCP feedback messages sent by

other RTP_Rx's.

2. A disjoint FT MAY send either an RTCP NACK or a 3rd party loss RTCP feedback message, each time using its own SSRC, to the DS.

   Note : The third party loss message format is defined in [I-D.ietf-avtcore-feedback-supression-rtp].

   This behavior for a FT operating in an SSM enabled with retransmission is different from the behavior prescribed by [RFC5760] for a FT with a primary SSM operating in Simple Feedback mode.

   This rule will enable feedback suppression in the group of RTP_Rx's reporting to this FT (see recommendation 5).

   The FT/RS will send an RTCP NACK or 3rd party loss message depending on the location of the packet loss event:

   * The FT/RS SHOULD send an RTCP NACK to the DS -using its own SSRC- when it detected packet loss itself (with the FT/RS acting as an RTP_Rx).

   * The FT/RS SHOULD send an RTCP NACK or a third party loss message to the DS -using its own SSRC- when it did not detect packet loss itself, but there is a risk on feedback implosion.

   Feedback implosion risk detection by the FT without detecting directly packet loss, can be based on monitoring the amount of arriving RTCP NACKs from RTP_Rx's reporting the same packet loss, and setting a threshold as function of the known amount of RTP SSM receivers reporting to this FT/RS.

3. A DS MUST forward/reflect any RTCP NACK or RTCP third party loss feedback message received from FT entities on the primary SSM RTP session

4. When capable of detecting packet loss in the RTP streams received from the media senders, the DS MUST send an RTCP NACK using its own SSRC on the primary SSM RTP session and to the media senders

5. An RTP_Rx receiving a third party loss message or RTCP NACK in the primary SSM RTP session MUST conform to the following behavior:

   If the SSRC identifier in the 'packet sender SSRC' field in the received RTCP feedback message matches either

* the SSRC of the (primary SSM) DS (this is the 'media sender SSRC')

* or the SSRC that is used by the FT/RS entity to which the RTP_Rx reports (see next section)

the RTP_Rx MUST follow the feedback suppression rule defined in [RFC4585], i.e. it abstains from sending an RTCP NACK for the same SN.

If there is no match with these SSRC values, the RTP_Rx SHOULD NOT apply the [RFC4585] feedback suppression rule

This behavior is different from [RFC4585], and prevents feedback suppression when there is no risk on feedback implosion.

If the RTP_Rx does not know the value of the SSRC that is used by the FT/RS entity -in its role towards the DS- to which the receiver reports, it MAY still send its own RTCP feedback message, when having detected the same packet loss, but respecting the [RFC4585]timing rules.

RTP_Rx's that do not know the value of the SSRC that is used by its FT/RS entity in its primary SSM role and abstain from sending an RTCP NACK message when receiving such a message, implement feedback suppression and behave as defined in [RFC4585].

The next section describes how an RTP_Rx knows the value of the SSRC that is used by the FT/RS entity to which the receiver reports

6.2. Informing RTP_Rx's with the SSRC Value of FT/RS

There are different SSRC values that can be associated to the FT/RS:

- As described in [RFC5760], in summary feedback model, the FT must act as a receiver and choose a unique SSRC for its own reporting towards the Distribution Source. If a FT acts strictly according to the Simple Feedback Model of [RFC5760], it does not need a unique SSRC. However, with the recommendation '2' of the previous section, the FT acts rather as translator when sending its own RTCP NACK or Third Party Loss message to the DS. The FT selects a SSRC value in its role as translator between the RTP_Rx's and the DS.

- The FT is part of the RS and the RS will typically also act as RTP_Rx, being the most efficient way for obtaining and caching copies of RTP packets that are transmitted down the primary SSM

and providing retransmission from that RS. The RS selects a SSRC value in its role as RTP_Rx as well, when reporting to the DS. Nothing prevents this SSRC value to be the same as the SSRC value used by the FT entity.

- o An RS also takes a role as RTP sender towards each of the RTP_Rx's that established a unicast retransmission session. In this unicast retransmission session, the SSRC used by the RS is the same as the SSRC value used by the DS in the primary SSM RTP session, as per [RFC4588].

Note that the CNAME of the FT/RS associated with all these SSRC identifier values is one and the same.

It is the SSRC value used by the FT acting as receiver/translator in the SSM with unicast feedback topology, that will be present in the 'Packet Sender SSRC' field of the RTCP NACK or Third Party Loss message forwarded by the DS to the RTP_Rx's. There can be several ways by which an RTP_Rx learns this SSRC value:

1. In-band : applies only for the Feedback Summary Model, where by means of a new RSI "FT SSRC list" message, the DS provides a listing of all deployed FT_Ap's with the corresponding SSRC for each of these FTs/RSs. FTs/RSs are identified by means of their IPv4/IPv6 address, optional port, and optional CNAME. The DS can learn the SSRC value chosen by a FT/RS by inspecting RTCP messages received from the FT. The DS sends regularly the RSI "FT SSRC list" message down the primary SSM, to give every RTP_Rx joining the SSM at a random time a chance to learn this SSRC value.

2. Application-specific : a specific extension can be defined for the RAMS-I message that signals the SSRC value associated with the FT in its role as reporter to the DS. This RAMS-I message is transferred in the Burst/Retransmission Session.

3. Out-of-band for either the Feedback Summary or Simple Feedback Model of SSM operation, by advertising the FT's SSRC as a media attribute for the FT in the SSM RTP session description [RFC5576].

The out-of-band signaling mechanism requires the application signaling to know/learn the SSRCs deployed by the FTs prior to signaling this information to the RTP_Rx's.

6.3. Unsolicited Retransmissions

The method of providing an (unsolicited) RTP retransmission packet to a set of RTP_Rx's enables loss recovery for packet loss events impacting a large set of receivers without risk for feedback implosion. Unsolicited retransmissions can be provided over each of the established unicast retransmission sessions or only once in a single multicast retransmission session, sourced either by the DS or by the RS. Providing (unsolicited) retransmissions over a separate multicast retransmission RTP session has the following advantages:

- The retransmission packet is transmitted by the DS/RS only once, saving both network and server resources.
- The multicast retransmission packet is less likely to be blocked by any intermediate gateway or a middlebox on the path between the DS/RS and RTP_Rx.
- An RTP_Rx that joins the multicast retransmission RTP session via SFGMP, implicitly indicates it is willing to accept unsolicited retransmissions.

The rules related to the feedback suppression in Section 6.1 do not impose an RS/DS implementation to provide (support for) unsolicited retransmissions. In this document, we assume that whenever the FT/RS/DS supports solicited retransmission and it actively suppresses feedback by sending a RTCP NACK or 3rd party loss message, it might try to provide one or more subsequent unsolicited retransmissions. However, as it is the case with any RTP retransmission, such unsolicited retransmissions may or may not arrive at their destinations.

The requirements for unsolicited retransmissions are:

1. Unsolicited retransmissions MAY be provided over a separate multicast retransmission that is sourced by the RS that also acts as retransmission server for the unicast retransmission sessions or by the DS that also sources the primary SSM session.
2. An RS SHOULD only send unsolicited retransmissions when it knows with a minimum certainty that the receiver has implemented feedback suppression for this specific packet (unicast retransmission) or a majority of the receivers have implemented feedback suppression (multicast retransmission).

The same retransmission of a packet originally transmitted in the primary SSM can be sent in both a unicast retransmission session and multicast retransmission session. The RTP_Rx will handle these two

(solicited or unsolicited) packets as duplicate packets.

Editor's note: Should we define a means by which an RTP_Rx can indicate it does not want to receive unsolicited retransmissions in the unicast retransmission session?

6.4. Congestion Control Considerations

6.4.1. Congestion Control in RFC 4585 and in RAMS

The RAMS document indicates that an RTP SSM distribution equipped with a retransmission/burst server can work reliably in both engineered and best-effort networks. However, there is a difference in the core operation of RAMS and the core operation of a Retransmission-enabled SSM in terms of congestion risk and avoidance:

1. In RAMS, retransmission packets, constituting a burst from the RAMS server, are requested by an RTP_Rx prior to the moment the receiver joins the SSM session. The Retransmission server bursts the retransmission packets to the receiver and instructs the receiver when to join the SSM, which will take place near the end of this burst. The congestion control for the bursting process can hence be decoupled from the SSM distribution (and any congestion control taking place on the SSM). RAMS discusses in more detail how to limit the risk on congestion for the RAMS bursts, especially in a best effort environment.

2. When using retransmission service for an SSM, retransmission packets are transmitted to and received by an RTP_Rx, while the RTP_Rx also receives simultaneously the SSM RTP session. When used in best effort networks, packet losses can be caused by congestion or (access) transmission link errors. If packets in the SSM are dropped because of congestion, there is a good chance that a subsequent retransmission will actually exaggerate the congestion situation (this will depend on the relative positioning of the congestion, the RS and the RTP_Rx in the considered SSM topology).

[RFC4588] section 7 states: 'In a best-effort environment, the sender SHOULD NOT send retransmission packets without reducing the packet rate and bitrate of the original stream (for example, by encoding the data at a lower rate).' In the context of SSM, some RTP_Rx's may observe packet loss (caused by congestion), other receivers receiving the same SSM may not. As the DS transmits a single original stream to all SSM receivers, the recommendation from [RFC4588] for the DS to adapt its packet rate whenever an RTP_Rx is reporting packet loss, will impact all SSM receivers. This would make SSM with retransmission service practically infeasible in best

effort networks. It is also important to note that the retransmission server/sender may be different from the DS sending the original stream, especially in large-scale SSM applications. Hence, to a large extent the SSM path from DS to an SSM receiver may be decoupled from the network path between RS and SSM receiver. I.e. it is quite possible that the congestion on the SSM does not impact the network path taken by the retransmission packet(s). For these reasons, a different recommendation is in place in terms of congestion handling when retransmission is applied in SSM with unicast feedback.

6.4.2. Congestion Handling for SSM with Retransmissions

In this section we formulate a set of rules for the RS in order to be cautious that retransmissions will not contribute to enhanced congestion situations on the SSM path between DS and RTP_Rx, but without imposing an explicit role to the RS to assist in (helping to) relieve any (potential) congestion between DS and RTP_Rx. This results in a congestion control mechanism for an RS operating in an SSM deployment that is decoupled from the behavior of the DS which may use its own congestion control.

Note: in this section only congestion on the path from the RS towards the RTP_Rx is considered. Congestion on the upstream path towards the FT/RS has been discussed in the context of feedback implosion risk and avoidance in the previous sections.

Case: Unmanaged Networks

The general requirement is that for unmanaged networks, an RTP retransmission server for an SSM RTP session SHOULD keep track, on a per RTP_Rx basis, of the recent retransmission request pattern for that RTP_Rx. The RS then behaves as follows:

- Initially and when an RTP_Rx sends a retransmission request for the first time to the FT/RS, the RS SHOULD assume there is no congestion between the DS and the RTP_Rx, unless the RS has information deduced or received not by way of RTCP NACKs through which it can assume the SSM path is likely congested.

  Such non-RTCP feedback information can be information coming from RTCP RR, other RTCP messages or any out-of-band signaling from the receiver or other network elements.

- An RS MAY respond to a retransmission request with one or more retransmission packets when the path from the DS to the RTP_Rx originating the request is assumed to be congestion free.

- When an RS observes an increasing amount of retransmission requests from a single receiver, while having serviced all or most retransmission requests from that receiver, or the server receives information in a way different from RTCP NACKs which predicts or indicates that congestion may likely occur, it SHOULD stop responding to retransmission requests from that receiver. Equally, the RS SHOULD NOT send unicast unsolicited retransmissions to such receiver.

- When no other information is available, an RS MAY assume absence of congestion for an RTP_Rx when it has not received retransmission requests for a minimum duration of time.

- An RS sending unsolicited retransmissions over a dedicated retransmission multicast SHOULD only do so when it knows this will not cause or contribute to congestion on the primary SSM path for the large majority of primary SSM receivers.

Detailed algorithms by which a RS can declare an RTP_Rx in a congestion state, or free from congestion, are not elaborated further upon in this document.

Note 1: an RS may correlate information it has on RTP_Rx's to make a judgement on congestion or congestion-free state of the SSM paths to other RTP_Rx's using the same RS.

Note 2: any congestion handling policy is orthogonal to any other policy maintained at the RS having to do with retransmission service access control e.g. using access control lists, IP anti-spoofing measures, authentication, etc.

Note 3: an algorithm similar to the one described above for congestion handling, but to handle a denial of service attack MAY be considered for the RS implementation, next to the other security considerations related to retransmission as discussed in section 12 in [RFC4588].

Equivalent requirements are formulated for an RTP_Rx in an unmanaged network, where retransmission is enabled:

- An RTP_Rx which senses that the net throughput of the combination of original RTP packets and retransmission packets decreases over time with increasing amount of retransmission requests and subsequent retransmissions, SHOULD cease making retransmission requests.

- An RTP_Rx may re-start sending out retransmission requests once the original SSM session is received with close-to-zero packet

loss, and where any optional sporadic packet loss can be assumed to be attributed to link transmission errors.

Case: Managed Networks

For managed networks congestion should not happen on the SSM path or otherwise be a rare event. [RFC4588] states : 'If enhanced service is used, it should be made sure that the total bitrate and packet rate do not exceed that of the requested service. It should be further monitored that the requested services are actually delivered.'

Note: The total bitrate and packet rate is the original data (or primary data) and retransmission data.

This means that the network is engineered to make sure that the combined SSM packets and retransmission packets on the network paths to any RTP_Rx will not be impacted by congestion.

An alternative approach is to engineer the network for the congestion-free transmission of the primary SSM data, where-as retransmissions are delivered without dedicated bandwidth engineering provisions or otherwise with limited engineering provisions, in a sense that up to a certain maximum rate and up to certain maximum burst size, the retransmission traffic can be delivered on a congestion-free path.

In the engineered case for SSM where retransmissions are not taken into account for the bandwidth provisioning, the chance on having congestion in the retransmission session for SSM enabled with retransmission-only service is expected to be smaller compared to an SSM enabled with RAMS service. This is because a single RAMS-R request results in a burst of packets in the retransmission session whereas retransmissions requests typically results in the transmission of only one or several retransmission packets, with retransmission requests being sporadic.

6.5. New RSI Message and RAMS-I TLV Extension

6.5.1. RSI Message 'FT SSRC List'

TBD.

6.5.2. RAMS-I TLV Extension 'FT SSRC'

TBD.

7. Security Considerations

Editor's note: A subset of the security aspects discussed in the RAMS specification also applies to this document. TBC.

8. IANA Considerations

TBC.

9. RTP Retransmission Service in DVB IPTV

RTP retransmission for linear IPTV streams has been specified in Annex D of [DVB-IPTV]. This ETSI document, produced by the DVB Technical Module on IP-Infrastructure (TM-IPI) specifies the interface of a managed IPTV service client, commonly called the Home Network End Device or HNED in DVB IPTV terms. The DVB IPTV RTP retransmission solution in Annex D references [RFC3550], [RFC4585], [RFC4588] and the [RFC5760]. Its solution is hence very well aligned with the IETF RTP specifications and addresses feedback implosion and feedback suppression in a similar way as described in this document:

- The DVB Linear Media Broadcast (LMB) services are delivered over an RTP SSM that operates according to the summary feedback model defined in [RFC5760]
- RTCP NACK messages transmitted by HNEDs towards the FT('s) are not forwarded over the primary RTP SSM.
- An HNED enables feedback suppression when receiving an RTCP Feedforward message from the network. This RTCP Feedforward (FF) message has the same format as an RTCP NACK message.
- The RTCP FF message stems from a 'so-called' upstream reporter (having its own SSRC identifier, signaled to the HNEDs in a DVB IPTV specific way) , which represents the Retransmission Server in its role as primary RTP_Rx. Alternatively the RTCP FF message is transmitted by the RS acting as RTP translator when the RS assumes a packet loss event took place that is impacting many HNEDs, but not the RS (as RTP_Rx) itself.
- This RTCP FF message may be sent in the primary RTP SSM but may also be sent in a dedicated RTP SSM sourced by the Retransmission Server.
- In order to better control and allow the Retransmission Server to better recognize potential feedback implosion situations, an

optional feature has been defined. A subset of RTP_Rx can act as immediate reporters (which do not follow the dithering transmission timing rules from [RFC4585]), and will send an RTCP NACK message immediately upon packet loss detection. Whether an HNED acts as immediate reporter or not is determined by whether its randomly chosen SSRC identifier value maps to a pre-defined SSRC mask/pattern that is signaled to all HNEDs.

10. Contributors

TBC.

11. Acknowledgments

TBC.

12. References

12.1. 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.

[RFC3376] Cain, B., Deering, S., Kouvelas, I., Fenner, B., and A. Thyagarajan, "Internet Group Management Protocol, Version 3", RFC 3376, October 2002.

[RFC3810] Vida, R. and L. Costa, "Multicast Listener Discovery Version 2 (MLDv2) for IPv6", RFC 3810, June 2004.

[RFC4585] Ott, J., Wenger, S., Sato, N., Burmeister, C., and J. Rey, "Extended RTP Profile for Real-time Transport Control Protocol (RTCP)-Based Feedback (RTP/AVPF)", RFC 4585, July 2006.

[RFC4588] Rey, J., Leon, D., Miyazaki, A., Varsa, V., and R. Hakenberg, "RTP Retransmission Payload Format", RFC 4588, July 2006.

[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.

[RFC4604] Holbrook, H., Cain, B., and B. Haberman, "Using Internet Group Management Protocol Version 3 (IGMPv3) and Multicast Listener Discovery Protocol Version 2 (MLDv2) for Source-Specific Multicast", RFC 4604, August 2006.

[I-D.ietf-avtcore-ports-for-ucast-mcast-rtp] Begen, A., Wing, D., and T. VanCaenegem, "Port Mapping Between Unicast and Multicast RTP Sessions", draft-ietf-avtcore-ports-for-ucast-mcast-rtp-02 (work in progress), April 2011.

[I-D.ietf-avt-rapid-acquisition-for-rtp] Steeg, B., Begen, A., Caenegem, T., and Z. Vax, "Unicast-Based Rapid Acquisition of Multicast RTP Sessions", draft-ietf-avt-rapid-acquisition-for-rtp-17 (work in progress), November 2010.

[I-D.ietf-avtcore-feedback-supression-rtp] Wu, W., Xia, F., and R. Even, "RTCP Extension for Third-party Loss Report", draft-ietf-avtcore-feedback-supression-rtp-03 (work in progress), May 2011.

12.2. Informative References

[RFC5576] Lennox, J., Ott, J., and T. Schierl, "Source-Specific Media Attributes in the Session Description Protocol (SDP)", RFC 5576, June 2009.

[I-D.ietf-fecframe-dvb-al-fec] Begen, A. and T. Stockhammer, "Guidelines for Implementing DVB-IPTV Application-Layer Hybrid FEC Protection", draft-ietf-fecframe-dvb-al-fec-04 (work in progress), December 2009.

[DVB-IPTV] "ETSI TS 102034 Digital Video Broadcasting (DVB); Transport of MPEG-2 TS Based DVB Services over IP Based Networks", August 2009.

Authors' Addresses

Tom VanCaenegem
Alcatel-Lucent
Copernicuslaan 50
Antwerpen, 2018
Belgium

Email: Tom.Van_Caenegem@alcatel-lucent.com

Bill VerSteeg
Cisco
5030 Sugarloaf Parkway
Lawrenceville, GA 30044
USA

Email: billvs@cisco.com

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

Email: abegen@cisco.com

Network Working Group M. Westerlund
Internet-Draft B. Burman
Intended status: Standards Track Ericsson
Expires: January 5, 2012 July 4, 2011

RTP Multiple Stream Sessions and Simulcast
draft-westerlund-avtcore-multistream-and-simulcast-00

Abstract

RTP has always been a protocol that supports multiple participants each sending their own media streams in an RTP session. Unfortunately many implementations aimed only at point to point voice over IP with a single source in each end-point. Even client implementations aimed at video conferences have often been built with the assumption around central mixers that only deliver a single media stream per media type. Thus any application that wants to allow for more advance usage where multiple media streams are sent and received by an end-point has a problem with legacy. This issue is analyzed, and RTP clarifications and signalling extensions are proposed to handle this issue. A related issue is how to perform simulcast, in the meaning of sending multiple encodings or representations of the same media source, when using RTP for media transport. This is further analyzed and possible solutions discussed and we arrive at a conclusion for session multiplexing of simulcast versions. We also found a number of related issues when having multiple streams and simulcast.

Table of Contents

1. Introduction

This document looks at the issues of non basic usage of RTP where there is multiple media sources sent over an RTP session. This include multiple sources from the same end-point, multiple end-points each having a source, or due to an application that needs multiple encodings of a particular source. As will be shown these issues are interrelated and need a common discussion to ensure consistency.

After presenting the usages and the found issues the document goes on to discuss ways of solving the issues. These include both clarifications to the basic RTP behaviors and signalling extensions to be able to setup these session, also in the presence of legacy systems that are not assumed to have full support for multiple media streams within an RTP session.

This document proposes several general mechanisms that could be used independently in other use cases. We foresee that those proposals would in the end become independent but related documents in the relevant WGs of AVTCORE, AVTEXT and MMUSIC. However, at this stage when all these ideas are introduced we find it more useful to keep them together to ensure consistency and to make any relations clear, hopefully making it easier to find and resolve any issues in the area of multiple streams and simulcast.

1.1. Multiple Streams

RTP sessions are a concept which most fundamental part is a SSRC space. This space can encompass a number of network nodes and interconnect transport flows between these nodes. Each node may have zero, one or more source identifiers (SSRCs) used to either identify a real media source such as a camera or a microphone, a conceptual source, like the most active speaker selected by a RTP mixer that switches between incoming media streams based on the media stream or additional information, or simply as an identifier for a receiver that provides feedback and reports on reception. There are also RTP nodes, like translators that are manipulating, data, transport or session state without making their presence aware to the other session participants.

RTP was designed with multiple participants in a session from the beginning. This was not restricted to multicast as many believe but also unicast using either multiple transport flows below RTP or a network node that redistributes the RTP packets, either unchanged in the form of a transport translator (relay) or modified in an RTP mixer. In addition a single end-point may have multiple media sources of the same media type, like cameras or microphones.

However, the most common use cases has been point to point Voice over IP (VoIP) or streaming applications where there has commonly not been more than one media source per end-point. Even in conferencing applications, especially voice only, the conference focus or bridge has provided a single stream being a mix of the other participants to each participant. Thus there has been perceived little need for handling multiple SSRCs in implementations. This has resulted in an installed legacy base that isn't fully RTP specification compliant and will have different issues if they receive multiple SSRCs of media, either simultaneously or in sequence. These issues will manifest themselves in various ways, either by software crashes, or simply in limited functionality, like only decoding and playing back the first or latest SSRC received and discarding any other SSRCs.

The signalling solutions around RTP, especially SDP based, hasn't considered the fundamental issues around RTP session's theoretical support of up to 4 billion plus sources all sending media. No end-point has infinite processing resources to decode and mix any number of sources with media. In addition the memory for storing related state, especially decoder state is limited, and the network bandwidth to receive multiple streams is also limited. Today, the most likely limitations are processing and network bandwidth, although for some use cases memory or other limitations may exist. The point is that a given end-point will have some limitations in the number of streams it simultaneously can receive, decode and playback. These limitations needs to be possible to expose and enabling the session participants to take them into account.

In similar ways there is a need for an end-point to express if it intends to produce one or more media stream. Todays SDP signalling support for this is basically the directionality attribute which indicates an end-point intend to send media or not. No indication of how many media streams.

Taking these things together there exist a clear need to enable the usage of multiple simultaneous media streams within an RTP session in a way that allows a system to take legacy implementations into account in addition to negotiate the actual capabilities around the multiple streams in an RTP session.

In addition to address the above set of issues we will also identify a number of issues related to multiple streams that should be addressed in the most suitable way. These include both obscurities in the RTP specification and short-comings in various signalling mechanisms that are exposed by multi-stream use cases.

1.2. Simulcast

Simulcast is the act of simultaneously sending multiple different versions of a media content. This can be done in several ways and for different purposes. This document focuses on the case where one wants to provide multiple different encodings towards a intermediary so that the intermediary can select which version to forward to other participants in the session. More discussion on the different ways of doing simulcast, which is the focus of this document in "Simulcast Usage and Applicability" (Section 3).

The different versions of a source content that can be simulcasted and that are considered in this document are:

Bit-rate: The primary difference is the amount of bits spent to encode the source and thus primarily affects the media signal to noise ratio (SNR).

Codec: Different media codecs are used to ensure that different receivers that do not have a common set of decoders can decode at least one of the versions. This includes codec configuration options that aren't compatible, like video encoder profiles, or the capability of receiving the transport packetization.

Sampling: Different sampling of media, in spatial as well as in temporal domain, may be used to suit different rendering capabilities or needs at receiving endpoints, as well as a method to achieve different bit-rates. For video streams, spatial sampling affects image resolution, and temporal sampling affects video framerate. For audio, spatial sampling relates to the number of audio channels, and temporal sampling affects audio bandwidth.

Different applications will have different reasons for providing a single media source in different versions. And as soon as an application have need for multiple versions for some reason, a potential need for simulcast is created. This need can arise even in media codecs that have scalability features built in to solve a set of variations.

The purpose of this document is to find the most suitable solution for the non-trivial variants of simulcast. To determine this, an analysis of different ways of multiplexing the different encodings are discussed in Section 6. Following the presentation of the alternatives, an analysis is performed in Section 7 on how different aspects like RTP mechanisms, signaling possibilities, and network features are affected by the alternatives.

The document ends with a recommendation for which solution is the most suitable and indicates what standardization work should be done if the WG agrees on the analysis and the suitability to define how simulcast should be done.

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

The following terms and abbreviations are used in this document:

Encoding: A particular encoding is the choice of the media encoder (codec) that has been used to compress the media, the fidelity of that encoding through the choice of sampling, bit-rate and other configuration parameters.

Different encodings: An encoding is different when some parameter that characterize the encoding of a particular media source has been changed. Such changes can be one or more of the following parameters; codec, codec configuration, bit-rate, sampling.

3. Simulcast Usage and Applicability

This section discusses different usage scenarios the term simulcast may refer to, and makes it clear which of those this document focuses on. It also reviews why simulcast and scalable codecs can be a useful combination.

3.1. Simulcasting to RTP Mixer

The usage here is in a multi-party session where one uses one or more central nodes to help facilitate the media transport between the session participants. Thus, this targets the RTP topology defined in [RFC5117] of RTP Mixer (Section 3.4: Topo-Mixer). This usage is one which is targeted for further discussion in this document.

Simulcasting different media encodings of video that has both different resolution and bit-rate is highly applicable to video conferencing scenarios. For example an RTP mixer selects the most active speaker and sends that participant’s media stream as a high

resolution stream to a receiver and in addition provides a number of small resolution video streams of any additional participants, thus enabling the receiving user to both see the current speaker in high quality and monitor the other participants. The active speaker gets a different combination of streams as it has limited use to get back the streams itself is sending. Thus, there can be several different combinations of high resolution and low resolution video in use simultaneously; requiring both a high and low resolution video from some sources at the same time.

For example, to provide both high and low resolution from an RTP Mixer there exist these potential alternatives:

Simulcast: The client sends one stream for the low resolution and another for the high resolution.

Scalable Video Coding: Using a video encoder that can provide one media stream that is both providing the high resolution and enables the mixer to extract a low resolution representation that has lower bit-rate than the full stream version.

Transcoding in the Mixer: The client transmits a high resolution stream to the RTP Mixer, which performs a transcoding to a lower resolution version of the video stream that is forwarded to the ones that need it.

The Transcoding requires that the mixer has sufficient amounts of transcoding resources to produce the number of low resolution versions required. This may in worst case be that all participants' streams needs transcoding. If the resources are not available, a different solution needs to be chosen.

The scalable video encoding requires a more complex encoder compared to non-scalable encoding. Also, if the resolution difference is big, the scalable codec may in fact be only marginally more bandwidth efficient, between the encoding client and the mixer, than a simulcast that sends the resolutions in separate streams, assuming equivalent video quality. At the same time, with scalable video encoding, the transmission of all but the lowest resolution will definitely consume more bandwidth from the mixer to the other participants than a non-scalable encoding, again assuming equivalent video quality.

Simulcasting has the benefit that it is conceptually simple. It enables use of any media codec that the participants agree on, allowing the mixer to be codec-agnostic. Considering today's video encoders, it is less bit-rate efficient in the path from the sending client to the mixer but more efficient in the mixer to receiver path

compared to Scalable Video Coding.

3.1.1. Simulcast Combined with Scalable Encoding

Scalable codecs are often used in arguments to motivate why simulcast isn't needed. A single media encoding that is sent as one joint media stream or divided up in base layers and enhancement layers over multiple transport is sufficient to achieve the desired functionality. As explained above in reality scalable codec is often not more efficient, especially in the path from the mixer to the receiver.

There are however, good reasons to combine simulcast with scalable encoding. By using simulcast to cover encoding variations where the scalable codec least efficient one can optimize the efficiency of the complete system. So a low number of simulcast working points, where each working point is in its turn a scalable codec configuration providing medium and/or fine grained scalability allowing a mixer to further tune the bit-rate to the available towards particular receivers using a combination of selecting simulcast versions and the number of extensions layers from that source.

A good example of this usage would be to send video encoded using SVC, where each simulcast version is a different resolution, and each SVC media stream uses temporal scalability and SNR scalability within that single media stream. If only resolution and temporal variations are needed, this can be implemented using H.264, as each simulcast version provides the different resolution, and each media stream within a simulcast encoding has temporal scalability using no-reference frames.

3.2. Simulcasting to Consuming End-Point

This usage is based on an RTP Transport Translator (Section 3.3: Topo-Trn-Translator) [RFC5117]. The transport translator functions as a relay and transmits all the streams received from one participant to all the other participants. In this case, one would do downlink simulcasting such that all receivers would receive all the versions. However, this clearly increases the bit-rate consumed on the paths to the client. The only benefit for the receiving client would be reduced decoding complexity when needing to only display a low resolution version. Otherwise a single stream application which only transmits the high resolution stream would allow the receiver to decode it and scale it down to the needed resolution.

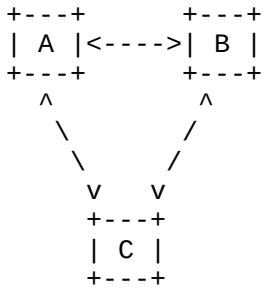
The usage of transport translator and simulcast becomes efficient if one allows each receiving client to control the relay to indicate

which version it wants to receive. However such a usage of RTP has some potential issues with RTCP. From the sending end-point it will look like the transmitted stream isn't received by a receiver that is known to receive other streams from the sender. Thus some consideration and mechanism are needed to support such a use case so that it doesn't break RTCP reception reporting.

This document will continue to consider this case but with less emphasis than on the RTP mixer case.

3.3. Same Encoding to Multiple Destinations

One interpretation of simulcast is when one encoding is sent to multiple receivers. This is well supported in RTP by simply copying all outgoing RTP and RTCP traffic to several transport destinations as long as the intention is to create a common RTP session. As long as all participants do the same, a full mesh is constructed and everyone in the multi party session has a similar view of the joint RTP session. This is analog to an Any Source Multicast (ASM) session but without the traffic optimization as multiple copies of the same content is likely to have to pass over the same link.



Full Mesh / Multi-unicast

As this type of simulcast is analog to ASM usage and RTP has good support for ASM sessions, no further consideration for this case is done.

3.4. Different Encoding to Independent Destinations

Another alternative interpretation of simulcast is with multiple destinations, where each destination gets a specifically tailored version, but where the destinations are independent. A typical example for this would be a streaming server distributing the same live session to a number of receivers, adapting the quality and resolution of the multi-media session to each receiver's capability and available bit-rate. This case can be solved in RTP by having

independent RTP sessions between the sender and the receivers. Thus this case is not considered further.

4. Multiple Streams Issues

This section attempts to go a bit more in depth around the different issues when using multiple media streams in an RTP session to make it clear that although in theory multi-stream applications should already be possible to use, there are good reasons to create extensions for signalling. In addition, the RTP specification could benefit from clarifications on how certain mechanisms should be working when an RTP session contains more than two SSRCs.

4.1. Legacy behaviors

It is a common assumption among many applications using RTP that they don’t have a need to support more than one incoming and one outgoing media stream per RTP session. For a number of applications this assumption has been correct. For VoIP and Streaming applications it has been easiest to ensure that a given end-point only receives and/or sends a single stream. However, they should support a source switching SSRC, e.g due to collision.

Some RTP extension mechanisms require the RTP stacks to handle additional SSRCs, like SSRC multiplexed RTP retransmission [RFC4588]. However, that still has only required handling a single media decoding chain.

However, there are applications that clearly can benefit from receiving and using multiple media streams simultaneously. A very basic case would be T.140 conversational text, which is both low bandwidth and where there is no simple method for mixing multiple sources of text that is supposed to be transmitted and displayed as you type. An RTP session that contains more than 2 SSRC actively sending media streams has the potential to confuse a legacy client in various ways:

1. The receiving client needs to handle receiving more than one stream simultaneously rather than replacing the already existing stream with the new one.

2. Be capable of decoding multiple streams simultaneously

3. Be capable of rendering multiple streams simultaneously

These applications may be very similar to existing one media stream applications at signalling level. To avoid connecting two different

implementations, one that is built to support multiple streams and one that isn't, it is important that the capabilities are signalled. It is also the legacy that makes us use a basic assumption in the solution. Anyone that doesn't explicitly indicate capability to receive multiple media streams is assumed to only handle a single media, to avoid affecting legacy clients.

4.2. Receiver Limitations

An RTP end-point that intends to process the media in an RTP session needs to have sufficient resources to receive and process all the incoming streams. It is extremely likely that no receiver is capable to handle the theoretical upper limit of an RTP session when it comes to more than 4 billion media sources. Instead, one or more properties will limit the end-points' capabilities to handle simultaneous media streams. These properties are for example memory, processing, network bandwidth, memory bandwidth, or rendering estate to mention a few possible limitations.

We have also considered the issue of how many simultaneous non-active sources an end-point can handle. We cannot see that inactive media sending SSRCs result in significant resource consumption and there should thus be no need to limit them.

A potential issue that needs to be acknowledged is where a limited set of simultaneously active sources varies within a larger set of session members. As each media decoding chain may contain state, it is important that this type of usage ensures that a receiver can flush a decoding state for an inactive source and if that source becomes active again it does not assume that this previous state exists.

Thus, we see need for a signalling solution that allows a receiver to indicate its upper limit in terms of capability to handle simultaneous media streams. We see little need for an upper limitation of RTP session members. Applications will need to have some considerations around how they use codecs.

4.3. Transmission Declarations

In an RTP based system where an end-point may either be legacy or has an explicit upper limit in the number of simultaneous streams, one will encounter situations where the end-point will not receive all simultaneous active streams in the session. Instead the end-points or central nodes, like RTP mixers, will provide the end-point with a selected set of streams based on various metrics, such as most active, most interesting, or user selected. In addition, the central node may combine multiple media streams using mixing or composition

into a new media stream to enable an end-point to get a sufficient source coverage in the session, despite existing limitations.

For such a system to be able to correctly determine the need for central processing, the capabilities needed for such a central processing node, and the potential need for an end-point to do sender side limitations, it is necessary for an end-point to declare how many simultaneous streams it may send. Thus, enabling negotiation of the number of streams an end-point sends.

4.4. RTP and RTCP Issues

This section details a few RTP and RTCP issues identified in implementation work for supporting multiple streams.

4.4.1. Multiple Sender Reports in Compound

One potential interoperability issue is inclusion of multiple Sender Report blocks in the same RTCP compound packet. The RTP specification isn't clear if such stacking is allowed or not. Thus there might be RTCP receivers that might not correctly handle such message. There is also an uncertainty how one should calculate the RTCP transmission intervals in such cases.

4.4.2. Cross reporting within an end-point

When an end-point has more than one SSRC and sends media using them, a question arises if the different SSRCs needs to report on each other despite being local. It can be argued that it is needed due to that it might not be fully visible for any external observer that they are actually sent from the same end-point. Thus by reporting on each other there are no holes in the connectivity matrix between all sending SSRCs and all known SSRCs.

4.4.3. Which SSRC is providing feedback

When one has multiple SSRCs on an end-point and needs to send RTCP feedback messages some considerations around which SSRC is used as the source and if that is consistently used or not, may be needed.

4.5. SDP Signalling Issues

An existing issue with SDP is that the bandwidth parameters aren't specified to take asymmetric conditions into account. This becomes especially evident when we start using multiple streams in an RTP session. Such a use case can easily result in that an end-point maybe receive 5 streams of Full High Definition (HD) video but only sends one Standard Definition (SD) video stream. Thus easily having

a 10:1 asymmetry in bit-rate.

If one uses the current SDP bandwidth parameters then one likely needs to set the session bandwidth to the sum of the most consuming direction. This can result in that there is no way of negotiating an upper bound for the lower band-width direction media stream(s). In addition, an end-point may conclude that it can't support the bit-rate despite being capable of actually receiving the media streams being sent. Thus making clear what bandwidth limitations a single stream has compared to the whole RTP session is important.

In the cases there is QoS, either by end-point reservation or done by systems like IMS, the requested bandwidth based on the signalled value will not represent what is actually needed.

Asymmetry in itself also create an issue, as RTCP bandwidth may be derived from the session bandwidth. It is important that all end-points have a common view on what the RTCP bandwidth is. Otherwise if the bandwidth values are more than 5 times different, an end-point with the high bandwidth value may time out an end-point that has a low value as it's minimal reporting interval can become more than 5 times longer than for the other nodes.

5. Multi-Stream Extensions

5.1. Signaling Support for Multi-Stream

There is a need to signal between RTP sender and receiver how many simultaneous RTP streams can be handled. The number of RTP streams that can be sent from a client should not have to match the number of streams that can be received by the same client. A multi-stream capable RTP sender MUST be able to adapt the number of sent streams to the RTP receiver capability.

For this purpose and for use in SDP, two new media-level SDP attributes are defined, max-send-ssrc and max-recv-ssrc, which can be used independently to establish a limit to the number of simultaneously active SSRCs for the send and receive directions, respectively. Active SSRCs are the ones counted as senders according to RFC3550, i.e. they have sent RTP packets during the last two regular RTCP reporting intervals.

The syntax for the attributes are in ABNF [RFC5234]:

```
max-ssrc = "a=" ("max-send-ssrc:" / "max-recv-ssrc:") PT 1*WSP limit
PT = "*" / 1*3DIGIT
limit = 1*8DIGIT
```

```
; WSP and DIGIT defined in [RFC5234]
```

A payload-agnostic upper limit to the total number of simultaneous SSRC that can be sent or received in this RTP session is signaled with a * payload type. A value of 0 MAY be used as maximum number of SSRC, but it is then RECOMMENDED that this is also reflected using the sendonly or recvonly attribute. There MUST be at most one payload-agnostic limit specified in each direction.

A payload-specific upper limit to the total number of simultaneous SSRC in the RTP session with that specific payload type is signaled with a defined payload type (static, or dynamic through rtpmap). Multiple lines with max-send-ssrc or max-recv-ssrc attributes specifying a single payload type MAY be used, each line providing a limitation for that specific payload type. Payload types that are not defined in the media block MUST be ignored.

If a payload-agnostic limit is present in combination with one or more payload-specific ones, the total number of payload-specific SSRCs are additionally limited by the payload-agnostic number. When there are multiple lines with payload-specific limits, the sender or receiver MUST be able to handle any combination of the SSRCs with different payload types that fulfill all of the payload specific limitations, with a total number of SSRCs up to the payload-agnostic limit.

When max-send-ssrc or max-recv-ssrc are not included in the SDP, it MUST be interpreted as equivalent to a limit of one, unless sendonly or recvonly attributes are specified, in which case the limit is implicitly zero for the corresponding unused direction.

5.1.1. Declarative Use

When used as a declarative media description, the specified limit in max-send-ssrc indicates the maximum number of simultaneous streams of the specified payload types that the configured end-point may send at any single point in time. Similarly, max-recv-ssrc indicates the maximum number of simultaneous streams of the specified payload types that may be sent to the configured end-point. Payload-agnostic limits MAY be used with or without additional payload-specific limits.

5.1.2. Use in Offer/Answer

When used in an offer, the specified limits indicates the agent’s intent of sending and/or capability of receiving that number of simultaneous SSRC. The answerer MUST reverse the directionality of recognized attributes such that max-send-ssrc becomes max-recv-ssrc

and vice versa. The answerer SHOULD decrease the offered limit in the answer to suit the answering client's capability. A sender MUST NOT send more simultaneous streams of the specified payload type than the receiver has indicated ability to receive, taking into account also any payload-agnostic limit.

In case an answer fails to include any of the limitation attributes, the agent MUST be interpreted as capable of supporting only a single stream in the direction for which attributes are missing. If the offer lacks attributes it MUST be assumed that the offerer only supports a single stream in each direction. In case the offer lack both max-send-ssrc and max-recv-ssrc, they MUST NOT be included in the answer.

5.1.3. Examples

The SDP examples below are not complete. Only relevant parts have been included.

```
m=video 49200 RTP/AVP 99
a=rtpmap:99 H264/90000
a=max-send-ssrc:* 2
a=max-recv-ssrc:* 4
```

An offer with a stated intention of sending 2 simultaneous SSRCs and a capability to receive 4 simultaneous SSRCs.

```
m=video 50324 RTP/AVP 96 97
a=rtpmap:96 H264/90000
a=rtpmap:97 H263-2000/90000
a=max-recv-ssrc:96 2
a=max-recv-ssrc:97 5
a=max-recv-ssrc:* 5
```

An offer to receive at most 5 SSRC, at most 2 of which using payload type 96 and the rest using payload type 97. By not including "max-send-ssrc" the value is implicitly set to 1.

```
m=video 50324 RTP/AVP 96 97 98
a=rtpmap:96 H264/90000
a=rtpmap:97 H263-2000/90000
a=max-recv-ssrc:96 2
a=max-recv-ssrc:97 3
a=max-recv-ssrc:98 5
a=max-recv-ssrc:* 5
```

An offer to receive at most 5 SSRC, at most 2 of which using payload type 96, and at most 3 of which using payload type 97, and at most 5

using payload type 98. Permissible payload type combinations include those with no streams at all for one or more of the payload types, as well as a total number of SSRC less than 5, e.g. two SSRC with PT=96 and three SSRC with PT=97, or one SSRC with PT=96, one with PT=97 and two with PT=98.

5.2. Asymmetric SDP Bandwidth Modifiers

To resolve the issues around bandwidth, we propose new SDP bandwidth modifiers that supports directionality, possibility for payload specific values and clear semantics. A common problem for all the current SDP bandwidth modifiers is that they use a single bandwidth value without a clear specification. Uncertainty in how the bandwidth value is derived creates uncertainty on how bursty a media source can be.

Thus, we do consider what the design criteria are prior to providing a proposal for new SDP bandwidth attribute.

5.2.1. Design Criterias

The current b= SDP bandwidth syntax is very limited and only allows the following format:

```
bandwidth-fields  = *(%x62 "=" bwtype ":" bandwidth CRLF)
bwtype            = token
bandwidth         = 1*DIGIT
```

Thus we will need to specify a new SDP bandwidth attribute as that allows syntax of more complexity.

The functionalities we see from the new bandwidth attribute are the following:

Directionality: We need to be able to have different sets of attribute values depending on direction.

Bandwidth semantics: A semantics identifier so that new semantics can be defined in the future for other needed semantics. This part of the b= has been a very successful design feature. We do perceive a need for both single stream limitations and limitations for the aggregate of all streams in one direction.

Payload specific: The possibility to specify different bandwidth values for different RTP Payload types. This as some codecs have different characteristics and one may want to limit a specific codec and payload configuration to a particular bandwidth. Especially combined with codec negotiation there is a need to

express intentions and limitations on usage for that particular codec. In addition, payload agnostic information is also needed.

Bandwidth specification method: To have a clear specification of what any bit-rate values mean we propose that Token bucket parameters should be used, i.e. bucket depth and bucket fill rate, where appropriate for the semantics. If single values are to be specified, a clear definition on how to derive that value must be specified, including averaging intervals etc.

We will use these design criteria next in an actual proposal.

5.2.2. Attribute Specification

We define a new SDP attribute ("a=") as the bandwidth modifier line syntax can’t support the requirements and nor can it be changed in an interoperable way. Thus we define the "a=bw" attribute. This attribute is structured as follows. After the attribute name there is a directionality parameter, followed by a scope parameter and then a bandwidth semantics tag. The semantics tag defines what value(s) that follow and their interpretation.

The attribute is designed so that multiple instances of the line will be necessary to express the various bandwidth related configurations that are desired.

Scopes and semantics can be extended in the future at any point. To ensure that an end-point using SDP either in Offer/Answer or declarative truly understands these extensions, a required-prefix indicator ("!") can be added prior to any scope or semantics parameter.

5.2.2.1. Attribute Definition

The ABNF [RFC5234] for this attribute is the following:

```
bw-attrib     = "a=bw:" direction SP [req] scope SP
                [req] semantics ":" values
direction     = "send" / "recv" / "sendrecv"
scope         = payloadType / scope-ext
payloadType   = "PT=" ("*" / PT-value-list)
PT-value-list = PT-value *(";" PT-Value)
PT-value      = 1*3DIGIT
req           = "!"
semantics     = "SMT" / "AMT" / semantics-ext
values        = token-bucket / value-ext
token-bucket  = "tb=" br-value ":" bs-value
br-value      = 1*15DIGIT ; Bucket Rate
bs-value      = 1*15DIGIT ; Bucket Size

semantics-ext = token ; As defined in RFC 4566
scope-ext     = 1*VCHAR ; As defined in RFC 4566
value-ext     = 0*(WSP / VCHAR)
```

The a=bw attribute defines three possible directionalities:

send: In the send direction for SDP Offer/Answer agent or in case of declarative use in relation to the device that is being configured by the SDP.

recv: In the receiving direction for the SDP Offer/Answer agent providing the SDP or in case of declarative use in relation to the device that is being configured by the SDP.

sendrecv: The provided bandwidth values applies equally in send and recv direction, i.e. the values configures the directions symmetrically.

The Scope indicates what is being configured by the bandwidth semantics of this attribute line. This parameter is extensible and we begin with defining two different scopes based on payload type:

Payload Type: The bandwidth configuration applies to one or more specific payload type values.

PT=*: Applies independently of which payload type is being used.

This specification defines two semantics which are related. The Stream Maximum Token bucket based value (SMT) and the Aggregate Maximum Token bucket based value (AMT). Both semantics represent the bandwidth consumption of the stream or the aggregate as a token bucket. The token bucket values are the token bucket rate and the token bucket size, represented as two integer numbers. It is an open question exactly what this token bucket is measuring, if it is RTP

payload only, like TIAS, or if it includes all headers down to the IP level as most of the other bandwidth modifiers do.

The definition of the semantics in more detail are:

SMT: The maximum intended or allowed bandwidth usage for each individual source (SSRC) in an RTP session as specified by a token bucket. The token bucket values are the token rate in bits per second and the bucket size in bytes. This semantics may be used both symmetrically or in a particular direction. It can be used either to express the maximum for a particular payload type or for any payload type (PT=*).

AMT: The maximum intended or allowed bandwidth usage for sum of all sources (SSRC) in an RTP session according to the specified directionality as specified by a token bucket. The token bucket values are the token rate in bits per second and the bucket size in bytes. Thus if using the sendrecv directionality parameter, both send and receive streams SHALL be included in the generated aggregate. If only a send or recv, then only the streams present in that direction are included in the aggregate. It can be used either to express the maximum for a particular payload type or for any payload type (PT=*).

5.2.2.2. Offer/Answer Usage

The offer/answer negotiation is done for each bw attribute line individually with the scope and semantics immutable. If an answerer would like to add additional bw configurations using other directionality, scope, and semantics combination, it may add them.

An agent responding to an offer will need to consider the directionality and reverse them when responding to media streams using unicast. If the transport is multicast the directionality is not affected.

For media stream offers over unicast with directionality send, the answerer will reverse the directionality and indicate its reception bandwidth capability, which may be lower or higher than what the sender has indicated as its intended maximum.

For media stream offers over unicast with directionality receive, these do indicate an upper limit, the answerer will reverse the directionality and may only reduce the bandwidth when producing the answer indicating the answerer intended maximum.

[Need to define how the required "!" prefix is used in Offer/Answer]

5.2.2.3. Declarative Usage

In declarative usage the SDP attribute is interpreted from the perspective of the end-point being configured by the particular SDP. An interpreter MAY ignore a=bw attribute lines that contains unknown scope or semantics that does not start with the required ("!") prefix. If a "required" prefix is present at an unknown scope or semantics, the interpreter SHALL NOT use this SDP to configure the end-point.

5.2.2.4. Example

Declarative example with stream asymmetry.

```
m=video 50324 RTP/AVP 96 97 98
a=rtpmap:96 H264/90000
a=rtpmap:97 H263-2000/90000
a=rtpmap:98 MP4V-ES/90000
a=max-recv-ssrc:96 2
a=max-recv-ssrc:* 5
a=bw:send pt=* SMT:tb=1200000:16384
a=bw:recv pt=96 SMT:tb=1500000:16384
a=bw:recv pt=97:98 SMT:tb=2500000:16384
a=bw:recv pt=* AMT:tb=8000000:65535
```

In the above example the outgoing single stream is limited to bucket rate of 1.2 Mbps and bucket size of 16384 bytes. The up to 5 incoming streams can in total use maximum 8 Mbps bucket rate and with a bucket size of 65535 bytes. However, the individual streams maximum rate is depending on payload type. Payload type 96 (H.264) is limited to 1.5 Mbps with a bucket size of 16384 bytes, while the Payload types 97 (H.263) and 98 (MPEG-4) may use up top 2.5 Mbps with a bucket size of 16384 bytes.

5.3. Binding SSRCs Across RTP Sessions

When an end-point transmits multiple sources in the same RTP session there may be tight relations between two different media types and their SSRCs, for example a microphone and a camera that is co-located are tightly related. CNAME is not sufficient to express this relation although it is commonly inferred from end-points that has only one media stream per media type. CNAME primary use in multi-source usages is to indicate which end-point and what synchronization context a particular media stream relates to.

To enable a RTP session participant to determine that close binding across multiple sessions, despite the end-point sending multiple SSRCs a new method for identifying such sources are needed. We are

not relying on using the same SSRC in all sessions for a particular media source as it is not robust against SSRC collision and forces potentially cascading SSRC changes between sessions.

5.3.1. SDES Item SRCNAME

Source Descriptions are a method that should work with all RTP topologies (assuming that any intermediary node is supporting this item) and existing RTP extensions. Thus we propose one defines a new SDES item called the SRCNAME which identifies with an unique identifier a single multi-media source, like a camera and a co-located microphone, or a truly individual media source such as a camera. That way any one receiving the SDES information from a set of interlinked RTP sessions can determine which are the same source.

We proposes that the SRCNAME would commonly be per communication session unique random identifiers generated according to "Guidelines for Choosing RTP Control Protocol (RTCP) Canonical Names (CNAMEs)" [RFC6222] with the addition that a local counter enumerating the sources on the host also are concatenated to the key in step 4 prior to calculating the hash.

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

5.3.2. SRCNAME in SDP

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

6. Simulcast Alternatives

Simulcast is the act of sending multiple alternative encodings of the same underlying media source. When transmitting multiple independent flows that originate from the same source, it could potentially be done in several different ways in RTP. The below sub-sections describe potential ways of achieving flow de-multiplexing and identification of which streams are alternative encodings of the same

source.

In the below descriptions we also include how this interacts with multiple sources (SSRCs) in the same RTP session for other reasons than simulcast. So multiple SSRCs may occur for various reasons such as multiple participants in multipoint topologies such as multicast, transport relays or full mesh transport simulcasting, multiple source devices, such as multiple cameras or microphones at one end-point, or RTP mechanisms in use, such as RTP Retransmission [RFC4588].

6.1. Payload Type Multiplexing

Payload multiplexing uses only the RTP payload type to identify the different alternatives. Thus all alternative streams would be sent in the same RTP session using only a single SSRC per actual media source. So when having multiple SSRCs, each SSRC would be unique media sources or RTP mechanism-related SSRC. Each RTP payload type would then need to both indicate the particular encoding and its configuration in addition to being a stream identifier. When considering a mechanism like RTP retransmission using SSRC multiplexing, an SSRC may either be a media source with multiple encodings as provided by the payload type, or a retransmission packet as identified also by the payload type.

As some encoders, like video, produce large payloads one can not expect that multiple payload encodings can fit in the same RTP packet payload. Instead a payload type multiplexed simulcast will need to send multiple different packets with one version in each packet or sequence of packets.

6.2. SSRC Multiplexing

The SSRC multiplexing idea is based on using a unique SSRC for each alternative encoding of one actual media source within the same RTP session. The identification of how flows are considered to be alternative needs an additional mechanism, for example using SSRC grouping [RFC5576] and a new SDES item such as SRCNAME proposed in Section 5.3.1 with a semantics that indicate them as alternatives of a particular media source. When one have multiple actual media sources in a session, each media source will use a number of SSRCs to represent the different alternatives it produces. For example, if all actual media sources are similar and produce the same number of simulcast versions, one will have n*m SSRCs in use in the RTP session, where n is the number of actual media sources and m the number of simulcast versions they can produce. Each SSRC can use any of the configured payload types for this RTP session. All session level attributes and parameters which are not source specific will apply and must function with all the alternative encodings intended

to be used.

6.3. Session Multiplexing

Session multiplexing means that each different version of an actual media source is transmitted in a different RTP session, using whatever session identifier to de-multiplex the different versions. This solution needs explicit session grouping [RFC5888] with a semantics that indicate them as alternatives. When there are multiple actual media sources in use, the SSRC representing a particular source will be present in the sessions for which it produces a simulcast version. It is also important to identify the SSRCs in the different sessions that are alternative encodings to each other, this can be accomplished using the same SSRC and/or a new SDES item identifying the media source across the session as the proposed SRCNAME SDES item (Section 5.3.1). Each RTP session will have its own set of configured RTP payload types where each SSRC in that session can use any of the configured ones. In addition all other attributes for sessions or sources can be used as normal to indicate the configuration of that particular alternative.

7. Simulcast Evaluation

This chapter evaluates the different multiplexing strategies in regard to several aspects.

7.1. Effects on RTP/RTCP

This section will be oriented around the different multiplexing mechanisms.

7.1.1. Payload Type Multiplexing

The simulcast solution needs to ensure that the negative impact on RTP/RTCP is minimal and that all the features of RTP/RTCP and its extensions can be used.

Payload type multiplexing for purposes like simulcast has well known negative effects on RTP. The basic issue is that all the different versions are being sent on the same SSRC, thus using the same timestamp and sequence number space. This has many effects:

1. Putting restraint between media encoding versions. For example, media encodings that uses different RTP timestamp rates cannot be combined as the timestamp values needs to be the same across all versions of the same media frame. Thus they are forced to use the same rate. When this is not possible, Payload Type

Multiplexing cannot be used.

2. Most RTP payload formats that may fragment a media object over multiple packets, like parts of a video frame, needs to determine the order of the fragments to correctly decode them. Thus it is important that one ensure that all fragments related to a frame or a similar media object are transmitted in sequence and without interruptions within the object. This can relatively simple be solved by ensuring that each version is sent in sequence.

3. Some media formats require uninterrupted sequence number space between media parts. These are media formats where any missing RTP sequence number will result in decoding failure or invoking of a repair mechanism within a single media context. The text/T140 payload format [RFC4103] is an example of such a format. These formats will be impossible to simulcast using payload multiplexing.

4. Sending multiple versions in the same sequence number space makes it more difficult to determine which version a packet loss may relate to. If one uses RTP Retransmission [RFC4588] one can ask for the missing packet. However, if the missing packet(s) do not belong to the version one is interested in, the retransmission request was in fact unnecessary.

5. The current RTCP feedback mechanisms are built around providing feedback on media streams based on stream ID (SSRC), packet (sequence numbers) and time interval (RTP Timestamps). There is almost never a field for indicating which payload type one is reporting on. Thus giving version specific feedback is difficult.

6. The current RTCP media control messages [RFC5104] specification is oriented around controlling particular media flows, i.e. requests are done addressing a particular SSRC. Thus such mechanisms needs to be redefined to support payload type multiplexing.

7. The number of payload types are inherently limited. Accordingly, using payload type multiplexing limits the number of simulcast streams and does not scale.

7.1.2. SSRC Multiplexing

As each version of the source has its own SSRC and thus explicitly unique flows, the negative effects above (Section 7.1.1) are not present for SSRC multiplexed simulcast.

The SSRC multiplexing of simulcast version requires a receiver to know that one is expected to only decode one of the versions and need not decode all of them simultaneously. This is currently a missing functionality as SDES CNAME cannot be used. The same CNAME has to be used for all flows connected to the same end-point and location. A clear example of this could be video conference where an end-point has 3 video cameras plus an audio mix being captured in the same room. As the media has a common timeline, it is important to be able to indicate that through the CNAME. Thus one cannot use CNAME to indicate that multiple SSRCs with the same CNAME are different versions of the same source. New semantics are required.

When one has all the versions in the same RTP session going to an RTP mixer and the mixer chooses to switch from forwarding one of the versions to forwarding another version, this creates an uncertainty in which SSRC one should use in the CSRC field (if used). As one is still delivering the same original source, such switch appears questionable to a receiver not having enabled simulcast in the direction to itself. Depending on what solution one chooses, one gets different effects here. If the CSRC is changed, then any message ensuring binding will need to be forwarded by the mixer, creating legacy issues. It has not been determined if there are downsides to not showing such a switch.

The impact of SSRC collisions on the SSRC multiplexing will be highly depending on what method is used to bind the SSRCs that provide different versions. Upon a collision and a forced change of the SSRC, a media sender will need to re-establish the binding to the other versions. By doing that, it will also likely be explicit when it comes to what the change was.

7.1.3. Session Multiplexing

Also session multiplexing does not have any of the negative effects that payload type multiplexing has (Section 7.1.1). As each flow is uniquely identified by RTP Session and SSRC, one can control and report on each flow explicitly. The great advantage of this method is that each RTP session appears just like if simulcast is not used thus minimal issues in RTP and RTCP including any extensions.

One potential downside of session multiplexing is that it becomes impossible without defining new RTCP message types to do truly synchronized media requests where one request goes to version A of source and another to version B of the same source. Due to the RTP session separation, one will be forced to send different RTCP packets to the different RTP session contexts, thus losing the ability to send two different RTCP packets in the same compound packet and RTP session context. This can be a minor inconvenience.

Using the same SSRC in all the RTP sessions allows for quick binding between the different versions. It also enables an RTP mixer that forwards one version to seamlessly decide to forward another version in a RTP session to a session participant that is not using simulcast in the direction from the mixer to the participant.

An SSRC collision forces a sender to change its SSRC in all sessions. Thus the collision-induced SSRC change may have bigger impact, as it affects all versions rather than a single version. But on the positive side, the binding between the versions will be immediate, rather than requiring additional signaling.

7.2. Signaling Impact

The method of multiplexing has significant impact on signaling functionality and how to perform it, especially if SDP [RFC4566] and SDP Offer/Answer [RFC3264] is used.

7.2.1. Negotiating the use of Simulcast

There will be a need for negotiating the usage of simulcast in general. For payload type multiplexing, one will need to indicate that different RTP payload types are intended as different simulcast versions. One likely has standalone SDP attributes that indicate the relation between the payload types, as one needs unique payload type numbers for the different versions. Thus, this increases the number of payload types needed within an RTP session. In worst case this may become a restriction as only 128 payload types are possible. This limitation is exacerbated if one uses solutions like RTP and RTCP multiplexing [RFC5761] where a number of payload types are blocked due to the overlap between RTP and RTCP.

SSRC multiplexing will likely use a standalone attribute to indicate the usage of simulcast. In addition, it may be possible to use a mechanism in SDP that binds the different SSRCs together. The first part is non-controversial. However the second one has significant impact on the signaling load in sessions with dynamic session participation. As each new participant joins a multiparty session, the existing participants that need to know the binding will need to receive an updated list of bindings. If that is done in SIP and SDP offer answer, a SIP re-Invite is required for each such transaction, invoking all the SIP nodes related to invites, and in systems like IMS also a number of policy nodes. If a receiver is required, which is likely, to receive the SSRC bindings prior to being able to decode any new source, then the signaling channel may introduce additional delay before a receiver can decode the media.

Session multiplexing results in one media description per version.

It will be necessary to indicate which RTP sessions are in fact simulcast versions. For example, using a Media grouping semantics specific for this. Each of these sessions will be focused on the particular version they intend to transport.

Legacy fallback, the impact on an end-point that isn't simulcast enabled, also needs to be considered. For a payload type multiplex solution, a legacy end-point that doesn't understand the indication that different RTP payload types are for different purpose may be slightly confused by the large amount of possibly overlapping or identical RTP payload types. In addition, as payload multiplexing isn't backwards compatible within a single media stream, the signalling needs to ensure that such a legacy client doesn't join a session using simulcast.

For an SSRC multiplexed session, a legacy end-point will ignore the SSRC binding signaling. From its perspective, this session will look like an ordinary session and it will setup to handle all the versions simultaneously. Thus, a legacy client is capable of decoding and rendering a simulcast enabled RTP session, but it will consume more resources and result in a duplication of the same source.

For session multiplexing, a legacy end-point will not understand the grouping semantic. It might either understand the grouping framework and thus determine that they are grouped for some purpose, or not understand grouping at all and then the offer simply looks like several different media sessions. This enables a simple fallback solution to exclude a legacy client from all simulcast versions except one, whichever is most suitable for the application.

7.2.2. Bandwidth negotation

The payload type multiplexed session cannot negotiate bandwidth for the individual versions without extensions. The regular SDP bandwidth attributes can only negotiate the overall bandwidth that all versions will consume. This makes it difficult to determine that one should drop one or more versions due to lack of bandwidth between the peers.

SSRC multiplexing suffers the same issues as payload type multiplexing, unless additional signaling (SSRC level attributes) is added.

Session multiplexing can negotiate bandwidth for each individual version and determine to exclude a particular version, and have the full knowledge on what it excludes to avoid consuming an excessive amount of bandwidth.

7.2.3. Negotation of media parameters

The negotiation and setting of the media codec, the codec parameters and RTP payload parameters for the payload type multiplexing is possible for each individual version as each has a unique payload type. The same is true for the session multiplexing where each version negotiates the parameters in the context of it's RTP session. The SSRC multiplexed version would need additional signaling to enable a binding between the payload types and which versions they are used for. Otherwise, the RTP payload types are negotiated without any context of which version intends to use which payload type.

However, the above assumes that there are no issues with defining different payload types for different alternative encodings. If that is not possible or it is intended to use the same payload type for multiple encodings, then additional signalling becomes necessary which isn't possible for payload multiplexing. For SSRC multiplexing, this signalling needs to redefine already existing session attributes, like imageattr [RFC6236] to have a per-SSRC scope. Session multiplexing can use existing attributes as they automatically get per-encoding scope thanks to the session multiplexing.

7.2.4. Negotation of RTP/RTCP Extensions

When one negotiates or configures the existing RTP and RTCP extensions, that can be done on either session level or in direct relation to one or several RTP payload types. They are not negotiated in the context of an SSRC. Thus payload type multiplexing will need to negotiate any session level extensions for all the versions without version specific consideration, unless extensions are deployed. It can also negotiate payload specific versions at a version individual level. SSRC multiplexing cannot negotiate any extension related to a certain version without extensions. Session multiplexing will have the full freedom of negotiating extensions for each version individually without any additional extensions.

7.3. Network Aspects

The multiplexing choice has impact on network level mechanisms.

7.3.1. Quality of Service

When it comes to Quality of Service mechanisms, they are either flow based or marking based. RSVP [RFC2205] is an example of a flow based mechanism, while Diff-Serv [RFC2474] is an example of a Marking based one. If one uses a marking based scheme, the method of multiplexing

will not affect the possibility to use QoS. However, if one uses a flow based one, there is a clear difference between the methods. Both Payload Type and SSRC multiplexing will result in all versions being part of the same 5-tuple (protocol, source address, destination address, source port, destination port) which is the most common selector for flow based QoS. Thus, separation of the level of QoS between versions is not possible. That is however possible if one uses session based multiplexing, where each different version will be in a different RTP context and thus commonly being sent over different 5-tuples.

7.3.2. NAT Traversal

Both the payload and SSRC multiplexing will have only one RTP session, not introducing any additional NAT traversal complexities compared to not using simulcast and only have a single version. The session multiplexing is using one RTP session per simulcast version. Thus additional lower layer transport flows will be required unless an explicit de-multiplexing layer is added between RTP and the transport protocol.

Below we analyzed and comment on the impact of requiring more underlying transport flows in the presence of NATs and Firewalls:

End-Point Port Consumption: A given IP address only has 65536 available local ports per transport protocol for any consumer of ports that exist on the machine. This is normally never an issue for a end-user machine. It can become an issue for servers that have large number of simultaneous flows. However, if the application uses ICE, which authenticated STUN requests, a server can serve multiple end-point from the same local port, and use the whole 5-tuple (source and destination address, source and destination port, protocol) as identifier of flows after having securely bound them to end-points using the STUN request. Thus in theory the minimal number of media server ports needed are the maximum number of simultaneous RTP sessions a single end-point may use, when in practice implementation will probably benefit from using more.

NAT State: If an end-point is behind a NAT each flow it generates to an external address will result in a state on that NAT. That state is a limited resource, either from memory or processing stand-point in home or SOHO NATs, or for large scale NATs serving many internal end-points, the available external ports run-out. We see this primarily as a problem for larger centralized NATs where end-point independent mapping do require each flow mapping to use one port for the external IP address. Thus affecting the maximum aggregation of internal users per external IP address.

However, we would like to point out that a real-time video conference session with audio and video are likely using less than 10 UDP flows, it is not like certain web applications that can result that 100+ TCP flows are opened to various servers from a single browser instance.

NAT Traversal taking additional time: When doing the NAT/FW traversal it takes additional time. And it takes time in a phase of communication between accepting to communicate and the media path being established which is fairly critical. The best case scenario for how much extra time it can take following the specified ICE procedures are: 1.5*RTT + Ta*(Additional_Flows-1), where Ta is the pacing timer, which ICE specifies to be no smaller than 20 ms. That assumes a message in one direction, and then an immediate triggered check back. This as ICE first finds one candidate pair that works prior to establish multiple flows. Thus, there are no extra time until one has found a working candidate pair. Based on that working pair the extra time it takes, is what it takes to in parallel establish the additional flows which in most case are 2-3 additional flows.

NAT Traversal Failure Rate: Due to that one need more than a single flow to be established through the NAT there is some risk that one succeed in establishing the first flow but fails with one or more of the additional flows. The risk that this happens are hard to quantify. However, that risk should be fairly low as one has just prior successfully established one flow from the same interfaces. Thus only rare events as NAT resource overload, or selecting particular port numbers that are filtered etc, should be reasons for failure.

As most simulcast solutions will anyway not use a very large number of simulcast versions due to the cost in encoding resources etc. one can discuss if the extra transport flows are a significant cost. We perceive the cost as low, if others are concluding that the cost is higher, a more generalized mechanism for multiplexing RTP sessions onto the same underlying transport flow should be considered.

7.4. Summary

It is quite clear from the analysis that payload type multiplexing is not at all a realistic option for using simulcast. It has many issues, especially on RTP/RTCP level. Thus, we will not consider it a viable solution in further discussions below.

Both SSRC and session multiplexing are viable to use. However, session multiplexing provides increased flexibility in usage, better support for network QoS, signalling flexibility, and support compared

to SSRC multiplexing, without defining additional extensions. Session multiplexing does however require additional NAT/FW pinholes to be opened or some other solution to allow multiple RTP sessions to share the same transport flow, but that is anyway something that already happens in today's applications.

The authors consider the impact on the signalling one of the most significant issues when it comes to SSRC multiplexing. For many use cases, selecting SSRC multiplexing will require us to define numerous signalling mechanisms to support binding such properties to specific SSRCs or encoding groups. This signalling already exists today for non simulcast RTP sessions or for simulcast in a session multiplexing context.

Session multiplexing is in the authors view clearly the best choice and is therefore recommended to be pursued as the single solution for simulcast.

8. Simulcast Extensions

This section discusses various extensions that either are required or could provide system performance gains if they where specified.

8.1. Signalling Support for Simulcast

To enable the usage of simulcast using session multiplexing some minimal signalling support is required. That support is discussed in this section. First of all, there is need for a mechanism to identify the RTP sessions carrying simulcast alternatives to each other. Secondly, a receiver needs to be able to identify the SSRC in the different sessions that are of the same media source but in different encodings.

Beyond the necessary signalling support for simulcast we look at some very useful optimizations in regards to the transmission of media streams and to help RTP mixers to select which stream alternatives to deliver to a specific client, or request a client to encode in a particular way.

8.1.1. Grouping Simulcast RTP Sessions

The proposal is to define a new grouping semantics for the session groupings framework [RFC5888]. There is a need to separate the semantics of intent to send simulcast streams from the capability to recognize and receive them. For that reason two new simulcast grouping tags are defined, "SimulCast Receive" (SCR) and "SimulCast Send" (SCS). They both act as an indicator that session level

simulcast is occurring and which sets of RTP sessions that carries simulcast alternatives to each other.

The grouping semantics SCR and SCS SHOULD be combined with the SDP attributes "a=max-send-ssrc" and "a=max-recv-ssrc" Section 5.1 to indicate the number of simultaneous streams of each encoding that may be sent or capable of receiving.

8.1.1.1. Declarative Use

When used as a declarative media description, SCR indicates the configured end-points required capability to recognize and receive a specified set of RTP streams as simulcast streams. In the same fashion, SCS request the end-point to send a specified set of RTP streams as simulcast streams. SCR and SCS MAY be used independently and at the same time and they need not specify the same or even the same number of RTP sessions in the group.

8.1.1.2. Offer/Answer Use

When used in an offer, SCS indicates the SDP providing agent’s intent of sending simulcast, and SCR indicates the agent’s capability of receiving simulcast streams. SCS and SCR MAY be used independently and at the same time and they need not specify the same or even the same number of RTP sessions in the group. The answerer MUST change SCS to SCR and SCR to SCS in the answer, given that it has and wants to use the corresponding (reverse) capability. An answerer not supporting the SCS or SCR direction, or not supporting SCS or SCR grouping semantics at all, will remove that grouping attribute altogether, according to [RFC5888]. An offerer that receives an answer indicating lack of simulcast support in one or both directions, where SCR and/or SCS grouping are removed, MUST NOT use simulcast in the non-supported direction(s).

8.1.2. Binding SSRCs Across RTP Sessions

When one performs simulcast, a transmitting end-point will for each actual media source have one SSRC in each session for which it currently provides an encoding alternative. As a receiver or a mixer will receive one or more of these, it is important that any RTP session participant beyond the sender can explicitly identify which SSRCs in the set of RTP sessions providing a simulcast service for a particular media type that originate from the same media source and thus belong together in the simulcast.

To accomplish this we extend the usage of SRCNAME as defined in Section 5.3.1. Within a particular media type the different RTP session carrying the different encodings will have the same SRCNAME

identifier. That way even if multiple encodings or representations are produced, any one receiving the SDES information from a set of interlinked RTP sessions can determine which are the same source.

8.2. Mixer Requests of Client streams

To increase the efficiency of simulcast systems, it is highly desirable that an RTP middlebox can signal to the client encoding and transmitting the streams if a particular stream is currently needed or not. This needs to be a quick and media plane oriented solution as it changes based on for example the user's speech activity or the user's selection in the user interface. Although several SIP and SDP-based methods would be possible, the required responsiveness suggests use of TMMBR from [RFC5104] with a bandwidth value of 0 to temporarily pause a certain SSRC and re-establishing transmission through TMMBR with a non-zero value.

8.3. Client to Mixer and Mixer to Client limiations

When a client has known limitations, for example based on local display layout between sources or if there is a better combination of streams from the available set of different encodings, then it is desirable to make these limitations known to the mixer delivering the streams. These limitations are also clearly dynamic, as sources may come or leave the session, making it prefer a different layout with another set of limitations in the delivered streams.

The Codec Control Messages in [RFC5104] defines some controls. However, with the addition of simulcast and scalable video there are more parameters that would be desired to control in a way similar to the Temporary Maximum Media Stream Bit Rate (TMMBR) messages, beyond just bit-rate. Factors such as largest image dimension and frame rate will also be needed, for example. In the context of simulcast, one also needs to consider if a limitation is not specific to an SSRC, but rather which encoding and scalability variation is most suitable from a particular media source (SRCNAME).

Thus we propose that new RTCP messages are defined to temporarily limit media source with respect to a combination of media stream properties such as for example bit-rate, frame-rate, image resolution, and audio channels. Such a message should be flexible enough to allow for additional limitation attributes.

8.4. Multiplexing Multiple RTP Sessions on Single Flow

It should be considered for RTP in non-legacy cases if multiple RTP sessions could be multiplexed in a standardized way on top of a single transport layer flow. That way the cost of opening additional

transport flows and the needed NAT/FW traversal would be avoided. We acknowledge that this has impact on use cases using a flow based QoS mechanism that needs differentiated service levels between sessions. Such a mechanism should thus be optional to use, but as there is likely a general interest in such a mechanism, work on this should be started.

8.5. Examples

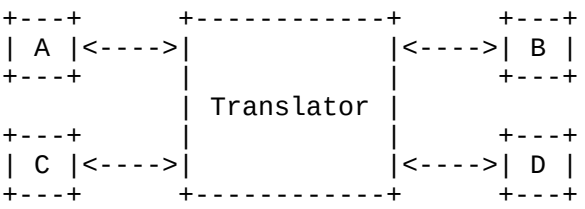
This section contains some SDP examples combining the proposals in this document to accomplish actual usages. We have skipped both NAT traversal tools as well as using the AVPF RTP profile [RFC4585] and Codec Control Messages [RFC5104] to save space in the SDPs, they are bulky enough. However, all these tools are likely to be part of a real SDP.

8.5.1. Multi-stream Signaling

This section contains examples of signalling for an application using multiple streams within an RTP session in two different contexts. In both these cases, the end-point that is involved in the signalling receives multiple streams, while only in the second case will the end-point transmit multiple streams.

8.5.1.1. Local Rendering in Video Conference Client

This example assumes a transport translator that enables the end-point to receive multiple streams from the other participants without using multiple destinations on transport level.



Four-party Translator-based Conference

Example of Media plane for RTP transport translator based multi-party conference with 4 participants.

Client A (Alice) in above figure is a desktop video conference client with a single camera and microphone. It uses a central transport translator to relay its media streams to the other participants, and in the same way it receives media streams from all other participants from the relay. This enables the client to locally render and

present other participants in a layout selected by the local client.

The network path between client A and the translator has certain known limitations, leading to a client needing to express its upper bounds in simultaneous streams that can be supported. That allows the conference server to know when it needs to tell the media plane relay to change its behavior from relaying to switching the media streams.

Alice invites herself into the conference by sending the following SDP offer:

```
v=0
o=alice 2890844526 2890842807 IN IP4 192.0.2.156
s=Multi stream Invite
c=IN IP4 192.0.2.156
b=AS:3530
t=0 0
m=audio 49200 RTP/AVP 96 97 9 8
b=AS:1450
a=rtpmap:96 G719/48000/2
a=rtpmap:97 G719/48000
a=rtpmap:9 G722/8000
a=rtpmap:8 PCMA/8000
a=bw:send pt=96 SMT:tb=128800:1500
a=bw:send pt=97 SMT:tb=64800:1500
a=bw:send pt=8;9 SMT:tb=64000:1500
a=bw:recv pt=* AMT:tb=1288000:1500
a=max-recv-ssrc:* 10
a=ssrc:834512974 cname:alice@foo.example.com
m=video 49300 RTP/AVP 96
b=AS:2080
a=rtpmap:96 H264/90000
a=fmtp:96 profile-level-id=42c01e
a=imageattr:* send [x=640,y=360] recv [x=640,y=360] [x=320,y=180]
a=bw:send pt=96 SMT:tb=500000:8192
a=bw:recv pt=96 SMT:tb=500000:8192
a=max-recv-ssrc:* 4
a=ssrc:451297483 cname:alice@foo.example.com
a=content:main
```

Alice Offer for a Multi-stream Conference

In the above SDP, Alice proposes one audio and one video RTP session. The audio session has 4 payload types being configured and the different payload configurations also show Alice’s intentions of their different bandwidth usage. For the audio receive direction, Alice accepts an aggregate bandwidth of 1288 kbps with a 1500 byte

bucket depth. This is sufficient bandwidth for 10 simultaneous streams. This limit of up to 10 streams being received is additionally indicated on SSRC level using the a=max-recv-ssrc attribute. The send limitation is implicitly set to one by excluding the a=max-send-ssrc attribute. Alice also declares the cname for the SSRC she intends to use.

The video session has only a single payload format using H.264. The configured profile and level is sufficient to support multiple resolutions of interest for the application. Alice indicates the intention to send 640x360 resolution and requests to receive either 640x360 or 320x180. The bandwidth for the video is expressed as the same 500 kbps upper limit in both send and receive directions, with an 8192 bytes bucket depth. There is no explicit limitation on the aggregate bandwidth. Alice does however express that she cannot handle receiving more than 4 simultaneous active SSRCs, so there is an implicit limit.

The application server controlling the conference receives the Offer and constructs a response based on knowledge about the conference and the available translator.

```
v=0
o=server 39451234544 39451234578 IN IP4 198.51.100.2
s=Multi stream Alice Answer
c=IN IP4 198.51.100.43
b=AS:2950
t=0 0
m=audio 49200 RTP/AVP 96 97 9
b=AS:870
a=rtpmap:96 G719/48000/2
a=rtpmap:97 G719/48000
a=rtpmap:9 G722/8000
a=bw:recv pt=96 SMT:tb=128800:1500
a=bw:recv pt=97 SMT:tb=64800:1500
a=bw:recv pt=9 SMT:tb=64000:1500
a=bw:send pt=* AMT:tb=500000:1500
a=max-send-ssrc:* 6
a=ssrc:239245219 cname:bob@foo.example.com
a=ssrc:986545121 cname:dave@foo.example.com
a=ssrc:2199983234 cname:fred@foo.example.com
m=video 49300 RTP/AVP 96
b=AS:2080
a=rtpmap:96 H264/90000
a=fmtp:96 profile-level-id=42c01e
a=imageattr:* recv [x=640,y=360] send [x=640,y=360] [x=320,y=180]
a=bw:recv pt=96 SMT:tb=500000:8192
a=bw:send pt=96 SMT:tb=500000:8192
a=max-send-ssrc:* 4
a=ssrc:924521923 cname:bob@foo.example.com
a=ssrc:654512198 cname:dave@foo.example.com
a=ssrc:3234219998 cname:fred@foo.example.com
a=content:main
```

SDP Answer to Alice from application server

The application server accepts both audio and video RTP sessions. It removed the a-law PCM format as it isn’t needed in this conference. It also reduces the number of simultaneous streams that may occur to 6 by setting the a=max-send-ssrc attribute to 6. The aggregate bandwidth that the client may receive, i.e. what the server declares as send, is limited down 500 kbps with a bucket depth of 1500 bytes. The SSRC values and their CNAMEs from the 3 already connected clients, bob, dave and fred are also included.

The video session is accepted as is, indicated by reversing the directions on the parts that indicates direction in the bw attribute and the imageattr. The max-recv-ssrc is changed to max-send-ssrc to indicate that there may be up to 4 simultaneous sources from the translator down to alice. The SSRCs and the corresponding CNAMEs are

also declared for video allowing for audio and video to be bound together, enabling synchronization before receiving the first RTCP sender reports.

8.5.1.2. Multiple Sources from Telepresence Room

In this use case Alice is an end-point which is a telepresence room. It has 3 cameras to cover different parts of the room's table. It also has directional microphones for each camera sector, such that it requests to send 3 streams of audio to maintain audio to screen bindings. If this is not possible, a stereo field sound mix can be provided instead that covers all three cameras.

Alice communicates directly with another single telepresence room end-point, Bob, but with only 2 cameras and microphones. However, Bob can receive 3 simultaneous streams and can use them in the local playout layout.

Alice invites herself into the conference by sending the following SDP offer:

```
v=0
o=alice 2890844526 2890842807 IN IP4 192.0.2.156
s=Telepresence Alice Invite
c=IN IP4 192.0.2.156
b=AS:8965
t=0 0
m=audio 49200 RTP/AVP 97 96
b=AS:725
a=rtpmap:96 G719/48000/2
a=rtpmap:97 G719/48000
a=bw:send pt=96 SMT:tb=128800:1500
a=bw:send pt=97 SMT:tb=64800:1500
a=bw:recv pt=* AMT:tb=644000:1500
a=max-recv-ssrc:* 5
a=max-send-ssrc:97 3
a=max-send-ssrc:96 1
a=ssrc:239245219 cname:alice@foo.example.com
a=ssrc:239245219 srcname:a3:d3:4b:f1:22:12
a=ssrc:986545121 cname:alice@foo.example.com
a=ssrc:986545121 srcname:12:3f:ab:d2:ec:32
a=ssrc:2199983234 cname:alice@foo.example.com
a=ssrc:2199983234 srcname:7f:12:db:87:2d:52
m=video 49300 RTP/AVP 96
b=AS:8240
a=rtpmap:96 H264/90000
a=fmtp:96 profile-level-id=42c01e
a=imageattr:* send [x=1280,y=720] recv [x=1280,y=720]
a=bw:send pt=96 SMT:tb=2500000:8192
a=bw:recv pt=96 SMT:tb=3000000:8192
a=bw:send pt=* AMT:tb=8000000:16384
a=max-recv-ssrc:* 5
a=max-send-ssrc:* 3
a=ssrc:245219239 cname:alice@foo.example.com
a=ssrc:245219239 srcname:a3:d3:4b:f1:22:12
a=ssrc:545121986 cname:alice@foo.example.com
a=ssrc:545121986 srcname:12:3f:ab:d2:ec:32
a=ssrc:199983234 cname:alice@foo.example.com
a=ssrc:199983234 srcname:7f:12:db:87:2d:52
a=content:main
```

Telepresence room Offer for a point to point session

Alice invites Bob into a session where Alice proposes one audio and one video RTP session, both with multiple streams. The audio session is proposing to use 3 mono streams of G.719 (pt=97) as being more prioritized than a single stereo G.719 (pt=96). It also states that it is willing to accept up to 5 simultaneous audio streams from Bob independent of payload type. The end-point also declares the SSRC it

intends to use with bindings to CNAME and SRCNAME, enabling Bob to bind together the audio and the video streams that come from the same part of the conference table.

The video session only configures H.264 payload format and states that it intends to send 1280x720 resolution and requests to receive the same. Alice also states that she will put the upper limit of the streams it sends to 2500 kbps with 8192 bytes bucket depth, while it will accept to receive individual streams that are up to 3000 kbps with 8192 bytes bucket depth. However, it also promises to limit the aggregate to no more than 8000 kbps and 16384 of bucket depth for the combination of all three streams it intends to send. Alice is willing to receive up to 5 streams of video simultaneous. Also here Alice informs Bob of the SSRC and their bindings to CNAME and SRCNAME.

Bob process this invite and constructs a SDP answer to be delivered to Alice. As Bob only has two cameras and microphones it will indicate this from its side. However, it is capable of receiving Alice 3 streams without any issues.

```
v=0
o=bob 2890847754 28908477889 IN IP4 198.51.100.21
s=Telepresence Bob Response
c=IN IP4 198.51.100.21
b=AS:8528
t=0 0
m=audio 49200 RTP/AVP 97 96
b=AS:288
a=rtpmap:96 G719/48000/2
a=rtpmap:97 G719/48000
a=bw:send pt=96 SMT:tb=128800:1500
a=bw:send pt=97 SMT:tb=64800:1500
a=bw:send pt=* AMT:tb=136000:1500
a=bw:recv pt=* AMT:tb=240000:1500
a=max-recv-ssrc:* 3
a=max-send-ssrc:97 2
a=max-send-ssrc:96 1
a=ssrc:52037639 cname:bob@foo.example.com
a=ssrc:52037639 srcname:37:ee:ca:38:01:3c
a=ssrc:820545843 cname:bob@foo.example.com
a=ssrc:820545843 srcname:20:85:17:48:75:a4
m=video 49300 RTP/AVP 96
b=AS:8240
a=rtpmap:96 H264/90000
a=fmtp:96 profile-level-id=42c01e
a=imageattr:* send [x=1280,y=720] recv [x=1280,y=720]
a=bw:recv pt=96 SMT:tb=2500000:8192
a=bw:send pt=96 SMT:tb=3000000:8192
a=bw:send pt=* AMT:tb=6000000:16384
a=bw:recv pt=* AMT:tb=8000000:16384
a=max-recv-ssrc:* 3
a=max-send-ssrc:* 2
a=ssrc:911548031 cname:bob@foo.example.com
a=ssrc:911548031 srcname:37:ee:ca:38:01:3c
a=ssrc:586599792 cname:bob@foo.example.com
a=ssrc:586599792 srcname:20:85:17:48:75:a4
a=content:main
```

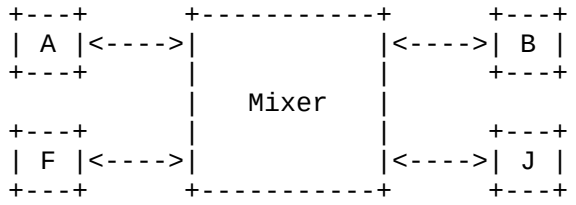
Telepresence room Answer for a point to point session

So Bob accepts the audio codec configurations but changes the aggregate bandwidths to what it is going to send itself and creates a limitation for Alice based on three mono streams. It confirms the number of streams Alice intends to be sending by including a=max-recv-ssrc:* 3. It also declares that it intends to send either two mono or one stereo stream. Bob also provides its configuration for SSRC and their mapping of CNAME and SRCNAME.

For video it is very similar, the number of streams Bob intends to send is stated as 2 and it also accept the 3 streams Alice intended to send in the max-recv-ssrc attribute. The bandwidth for these streams is accepted as suggested by Bob, keeping the upper limit for the individual streams at 3000 kbps and 8192 bytes depth. It also adds a total in Bob send direction that is twice the individual streams. It also confirms Alice's limitation for the aggregate. Finally the SSRCs for video are also declared and their bindings to CNAME and SRCNAME.

8.5.2. Simulcast Signaling

This example is for a case of client to video conference service using a centralized media topology with an RTP mixer. Alice, Bob calls into a conference server for a conference call with audio and video to the RTP mixer, these clients being capable to send a few video simulcast versions. The conference server also dials out to Fred, which is a legacy client resulting in fallback behavior. When dialing out to Joe more success is achieved as Joe is a client similar to Alice.



Four-party Mixer-based Conference

Example of Media plane for RTP mixer based multi-party conference with 4 participants.

8.5.2.1. Alice: Desktop Client

Alice is calling in to the mixer with an audiovisual single stream desktop client, only adding capability to send simulcast, announce SRCNAME and use of the new directional bandwidth attribute from Section 5.2 compared to a legacy client. The offer from Alice looks like

```
v=0
o=alice 2362969037 2362969040 IN IP4 203.0.113.156
s=Simulcast enabled Desktop Client
t=0 0
c=IN IP4 203.0.113.156
b=AS:825
a=group:SCS 2 3
m=audio 49200 RTP/AVP 96 97 9 8
b=AS:145
a=rtpmap:96 G719/48000/2
a=rtpmap:97 G719/48000
a=rtpmap:9 G722/8000
a=rtpmap:8 PCMA/8000
a=bw:send pt=96 SMT:tb=128800:1500
a=bw:send pt=97 SMT:tb=64800:1500
a=bw:send pt=8;9 SMT:tb=64000:1500
a=bw:recv pt=* AMT:tb=128800:1500
a=ssrc:521923924 cname:alice@foo.example.com
a=ssrc:521923924 srcname:a3:d3:4b:f1:22:12
a=mid:1
m=video 49300 RTP/AVP 96
b=AS:520
a=rtpmap:96 H264/90000
a=fmtp:96 profile-level-id=42c01e
a=imageattr:* send [x=640,y=360] recv [x=640,y=360] [x=320,y=180]
a=bw:send pt=96 SMT:tb=500000:8192
a=bw:recv pt=96 SMT:tb=500000:8192
a=ssrc:192392452 cname:alice@foo.example.com
a=ssrc:192392452 srcname:a3:d3:4b:f1:22:12
a=mid:2
a=content:main
m=video 49400 RTP/AVP 96
b=AS:160
a=rtpmap:96 H264/90000
a=fmtp:96 profile-level-id=42c00d
a=imageattr:96 send [x=320,y=180]
a=bw:send pt=96 SMT:tb=128000:4096
a=bw:recv pt=96 SMT:tb=128000:4096
a=ssrc:239245219 cname:alice@foo.example.com
a=ssrc:239245219 srcname:a3:d3:4b:f1:22:12
a=mid:3
a=sendonly
```

Alice Offer for a Simulcast Conference

As can be seen from the SDP, Alice has a simulcast-enabled client and offers two different session-multiplexed simulcast versions sent from her single camera, indicated by the SCS grouping tag and the two

media ID's (2 and 3). The first video version with media ID 2 prefers 360p resolution (signaled via imageattr) and the second video version with media ID 3 prefers 180p resolution. The first video media line also acts as the single receive video (making media line sendrecv), while the second video media line is only related to simulcast transmission and is thus offered sendonly. The two simulcast encoding streams and its related audio stream are bound together using SRCNAME SDES item. We also declare the end-point CNAME as all sources belong to the same synchronization context.

Alice uses the a=bw attribute defined in this document, but also uses the less exact, legacy b-line for interoperability. For video in this example, the client offers to send and receive a bandwidth lower than the video codec level maximum, which could for example have been set via some client or user preference, based on known transport limitations or knowledge what bandwidth is reasonable from a quality perspective given that specific codec at the proposed image resolution. The bitrates given in this example are supposed to be aligned with Section 5.2 and are thus based on the RTP payload level, but could also be designed based on another network layer according to the discussion in that section.

8.5.2.2. Bob: Telepresence Room

Bob is calling in to the mixer with a telepresence client that has capability for both sending multi-stream, receiving and local rendering of those multiple streams, as well as sending simulcast versions of the uplink video. More specifically, in this example the client has three cameras, each being sent in three different simulcast versions. In the receive direction, up to two main screens can show video from a (multi-stream) conference participant being active speaker, and still more screen estate can be used to show videos from up to 16 other conference listeners. Each camera has a corresponding (stereo) microphone that can also be negotiated down to mono by removing the stereo payload type from the answer.

```
v=0
o=bob 129384719 9834727 IN IP4 203.0.113.35
s=Simulcast enabled Multi stream Telepresence Client
t=0 0
c=IN IP4 203.0.113.35
b=AS:6035
a=group:SCS 2 3 4
m=audio 49200 RTP/AVP 96 97 9 8
b=AS:435
a=rtpmap:96 G719/48000/2
a=rtpmap:97 G719/48000
a=rtpmap:9 G722/8000
```

```
   a=rtpmap:8 PCMA/8000
   a=max-send-ssrc:* 3
   a=max-recv-ssrc:* 3
   a=bw:send pt=96 SMT:tb=128800:1500
   a=bw:send pt=97 SMT:tb=64800:1500
   a=bw:send pt=8;9 SMT:tb=64000:1500
   a=bw:send pt=* AMT:tb=386400:1500
   a=bw:recv pt=* AMT:tb=386400:1500
   a=ssrc:724847850 cname:bob@foo.example.com
   a=ssrc:724847850 srcname:37:ee:ca:38:01:3c
   a=ssrc:2847529901 cname:bob@foo.example.com
   a=ssrc:2847529901 srcname:20:85:17:48:75:a4
   a=ssrc:57289389 cname:bob@foo.example.com
   a=ssrc:57289389 srcname:1e:23:97:ab:9e:0c
   a=mid:1
   m=video 49300 RTP/AVP 96
   b=AS:4500
   a=rtpmap:96 H264/90000
   a=fmtp:96 profile-level-id=42c01f
   a=imageattr:* send [x=1280,y=720] recv [x=1280,y=720]
        [x=640,y=360] [x=320,y=180]
   a=max-send-ssrc:96 3
   a=max-recv-ssrc:96 2
   a=bw:send pt=96 SMT:tb=1500000:16384
   a=bw:send pt=* AMT:tb=4500000:16384
   a=bw:recv pt=96 SMT:tb=1500000:16384
   a=bw:recv pt=* AMT:tb=3000000:16384
   a=ssrc:75384768 cname:bob@foo.example.com
   a=ssrc:75384768 srcname:37:ee:ca:38:01:3c
   a=ssrc:2934825991 cname:bob@foo.example.com
   a=ssrc:2934825991 srcname:20:85:17:48:75:a4
   a=ssrc:3582594238 cname:bob@foo.example.com
   a=ssrc:3582594238 srcname:1e:23:97:ab:9e:0c
   a=mid:2
   a=content:main
   m=video 49400 RTP/AVP 96
   b=AS:1560
   a=rtpmap:96 H264/90000
   a=fmtp:96 profile-level-id=42c01e
   a=imageattr:* send [x=640,y=360]
   a=max-send-ssrc:96 3
   a=bw:send pt=96 SMT:tb=500000:8192
   a=ssrc:1371234978 cname:bob@foo.example.com
   a=ssrc:1371234978 srcname:37:ee:ca:38:01:3c
   a=ssrc:897234694 cname:bob@foo.example.com
   a=ssrc:897234694 srcname:20:85:17:48:75:a4
   a=ssrc:239263879 cname:bob@foo.example.com
   a=ssrc:239263879 srcname:1e:23:97:ab:9e:0c
```

```
a=mid:3
a=sendonly
m=video 49500 RTP/AVP 96
b=AS:420
a=rtpmap:96 H264/90000
a=fmtp:96 profile-level-id=42c00d
a=imageattr:96 send [x=320,y=180]
a=max-send-ssrc:96 3
a=bw:send pt=96 SMT:tb=128000:4096
a=ssrc:485723998 cname:bob@foo.example.com
a=ssrc:485723998 srcname:37:ee:ca:38:01:3c
a=ssrc:2345798212 cname:bob@foo.example.com
a=ssrc:2345798212 srcname:20:85:17:48:75:a4
a=ssrc:1295729848 cname:bob@foo.example.com
a=ssrc:1295729848 srcname:1e:23:97:ab:9e:0c
a=mid:4
a=sendonly
m=video 49600 RTP/AVP 96 97 98
b=AS:2600
a=rtpmap:96 H264/90000
a=fmtp:96 profile-level-id=42c01f
a=imageattr:96 recv [x=1280,y=720]
a=rtpmap:97 H264/90000
a=fmtp:97 profile-level-id=42c01e
a=imageattr:97 recv [x=640,y=360]
a=rtpmap:98 H264/90000
a=fmtp:98 profile-level-id=42c00d
a=imageattr:98 recv [x=320,y=180]
a=max-recv-ssrc:96 1
a=max-recv-ssrc:97 4
a=max-recv-ssrc:98 16
a=max-recv-ssrc:* 16
a=bw:recv pt=96 SMT:tb=1500000:16384
a=bw:recv pt=97 SMT:tb=500000:8192
a=bw:recv pt=98 SMT:tb=128000:4096
a=bw:recv pt=* AMT:tb=2500000:16384
a=mid:5
a=recvonly
a=content:alt
```

Bob Offer for a Multi-stream and Simulcast Telepresence Conference

Bob has a three-camera, three-screen, simulcast-enabled client with even higher performance than Alice’s and can additionally support 720p video, as well as multiple receive streams of various resolutions. The client implementor has thus decided to offer three simulcast streams for each camera, indicated by the SCS grouping tag and the three media ID’s (2, 3, and 4) in the SDP.

The first video media line with media ID 2 indicates the ability to send video from three simultaneous video sources (cameras) through the max-send-ssrc attribute with value 3. This media line is also marked as the main video by using the content attribute from [RFC4796]. Also the receive direction has declared ability to handle multiple video sources, and in this example it is 2. The interpretation of content:main for those two streams in the receive direction is that the client expects and can present (in prime position) at most two main (active speaker) video streams from another multi-camera client.

The second and third video media lines with media ID 3 and 4 are the sendonly simulcast streams. They can implicitly through the grouping be interpreted as also being content:main for the send direction, but is not marked as such since multiple media blocks with content:main could be confusing for a legacy client.

The fourth video media line with media ID 5 is recvonly and is marked with content:alt. That media line should, as was intended for that content attribute value, receive alternative content to the main speaker, such as "audience". In a multi-party conference, that could for example be the next-to-most-active speakers. The SDP describes that those streams can be presented in a set of different resolutions, indicated through the different payload types. The maximum number of streams per payload type is indicated through the max-recv-ssrc attribute. In this example, at most one stream can have payload type 96, preferably 720p, as indicated by the related imageattr line. Similarly, at most 4 streams can have payload type 97, preferably using 360p resolution, and at most 16 streams can have payload type 98, preferably of 180p resolution. In any case, there must never be more than 16 simultaneous streams of any payload type, but combinations of payload types may occur, such as for example two streams using payload type 97 and 8 streams using payload type 98.

To be able to relate the three cameras with the three microphones, all media lines that send audio or video use the ssrc attribute from [RFC5576], specifying the same SRCNAME from Section 5.3.2 for the audio and video versions that belong together. The use of this attribute is optional and the information can be retrieved from RTCP reporting, but it will then not be possible to correctly relate audio and video sources until the first RTCP report is received and participants may then seemingly make uncorrelated moves between screens and/or speakers when adjusting possible false correlation assumptions.

The legacy bandwidth reflects only the bandwidth in the receive direction, while the new bw attribute is very specific per direction and per media stream. We do note that the offered bandwidth for

transmission express as AS on session level woad be 6985. It is unclear what is the correct interpretation of the legacy bandwidth when there is bandwidth asymmetry.

The answer from a simulcast-enabled RTP mixer to this last SDP could look like:

```
v=0
o=server 238947290 239573929 IN IP4 198.51.100.2
s=Multi stream and Simulcast Telepresence Bob Answer
c=IN IP4 198.51.100.43
b=AS:7065
a=group:SCR 2 3 4
m=audio 49200 RTP/AVP 96
b=AS:435
a=rtpmap:96 G719/48000/2
a=max-send-ssrc:96 3
a=max-recv-ssrc:96 3
a=bw:send pt=96 SMT:tb=128800:1500
a=bw:recv pt=96 SMT:tb=128800:1500
a=bw:send pt=* AMT:tb=386400:1500
a=bw:recv pt=* AMT:tb=386400:1500
a=ssrc:4111848278 cname:server@conf1.example.com
a=ssrc:4111848278 srcname:87:e9:19:29:c1:bb
a=ssrc:835978294 cname:server@conf1.example.com
a=ssrc:835978294 srcname:1f:83:b3:85:62:7a
a=ssrc:2938491278 cname:server@conf1.example.com
a=ssrc:2938491278 srcname:99:76:b4:bb:90:52
a=mid:1
m=video 49300 RTP/AVP 96
b=AS:4650
a=rtpmap:96 H264/90000
a=fmtp:96 profile-level-id=42c01f
a=imageattr:* send [x=1280,y=720] [x=640,y=360] [x=320,y=180]
     recv [x=1280,y=720]
a=max-recv-ssrc:96 3
a=max-send-ssrc:96 2
a=bw:recv pt=96 SMT:tb=1500000:16384
a=bw:recv pt=* AMT:tb=4500000:16384
a=bw:send pt=96 SMT:tb=1500000:16384
a=bw:send pt=* AMT:tb=3000000:16384
a=ssrc:2938746293 cname:server@conf1.example.com
a=ssrc:2938746293 srcname:87:e9:19:29:c1:bb
a=ssrc:1207102398 cname:server@conf1.example.com
a=ssrc:1207102398 srcname:1f:83:b3:85:62:7a
a=mid:2
a=content:main
m=video 49400 RTP/AVP 96
```

```
   b=AS:1560
   a=rtpmap:96 H264/90000
   a=fmtp:96 profile-level-id=42c01e
   a=imageattr:* recv [x=640,y=360]
   a=max-recv-ssrc:96 3
   a=bw:recv pt=96 SMT:tb=500000:8192
   a=mid:3
   a=recvonly
   m=video 49500 RTP/AVP 96
   b=AS:420
   a=rtpmap:96 H264/90000
   a=fmtp:96 profile-level-id=42c00d
   a=imageattr:96 recv [x=320,y=180]
   a=max-recv-ssrc:96 3
   a=bw:recv pt=96 SMT:tb=128000:4096
   a=mid:4
   a=recvonly
   m=video 49600 RTP/AVP 96 97 98
   b=AS:2600
   a=rtpmap:96 H264/90000
   a=fmtp:96 profile-level-id=42c01f
   a=imageattr:96 send [x=1280,y=720]
   a=rtpmap:97 H264/90000
   a=fmtp:97 profile-level-id=42c01e
   a=imageattr:97 send [x=640,y=360]
   a=rtpmap:98 H264/90000
   a=fmtp:98 profile-level-id=42c00d
   a=imageattr:98 send [x=320,y=180]
   a=max-send-ssrc:96 1
   a=max-send-ssrc:97 4
   a=max-send-ssrc:98 8
   a=max-send-ssrc:* 8
   a=bw:send pt=96 SMT:tb=1500000:16384
   a=bw:send pt=97 SMT:tb=500000:8192
   a=bw:send pt=98 SMT:tb=128000:4096
   a=bw:send pt=* AMT:tb=2500000:16384
   a=ssrc:2981523948 cname:server@conf1.example.com
   a=ssrc:2938237 cname:server@conf1.example.com
   a=ssrc:1230495879 cname:server@conf1.example.com
   a=ssrc:74835983 cname:server@conf1.example.com
   a=ssrc:3928594835 cname:server@conf1.example.com
   a=ssrc:948753 cname:server@conf1.example.com
   a=ssrc:1293456934 cname:server@conf1.example.com
   a=ssrc:4134923746 cname:server@conf1.example.com
   a=mid:5
   a=sendonly
   a=content:alt
```

Server Answer for Bob Multi-stream and Simulcast Telepresence Conference

In this SDP answer, the grouping tag is changed to SCR, confirming that the sent simulcast streams will be received. The directionality of the streams themselves as well as the directionality of multi-stream and bandwidth attributes are changed. Note that the session level legacy bandwidth can be calculated more correctly with support from the bw attribute in the offer than would have been the case if only legacy media level bandwidth was present. Bandwidth bucket size can be adjusted down between the offer and the answer for streams sent from the answerer, indicating a more strict constant bitrate than really needed. The bucket size can be adjusted up or down for streams received by the answerer, indicating a more strict or flexible bitrate constraint, respectively, for the receiver compared to what the sender offered. The number of allowed streams in the content:alt video session has been reduced to 8 in the answer from 16 offered.

Note that the two video sources in the media block with mid:2 correspond to the two first audio sources (matching SRCNAME). The last audio source correspond to all video sources in the media block with mid:5, however SRCNAME can not be used to perform this binding as its semantic doesn’t match.

8.5.2.3. Fred: Dial-out to Legacy Client

Fred has a simple legacy client that know nothing of the new signaling means discussed in this document. In this example, the multi-stream and simulcast aware RTP mixer is calling out to Fred. Even though it is never actually sent, this would be Fred’s offer SDP, should he have called in. It is included here to improve the reader’s understanding of Fred’s response to the conference SDP.

```
v=0
o=fred 82342187 237429834 IN IP4 192.0.2.213
s=Legacy Client
t=0 0
c=IN IP4 192.0.2.213
m=audio 50132 RTP/AVP 9 8
a=rtpmap:9 G722/8000
a=rtpmap:8 PCMA/8000
m=video 50134 RTP/AVP 96 97
b=AS:405
a=rtpmap:96 H264/90000
a=fmtp:96 profile-level-id=42c00c
a=rtpmap:97 H263-2000/90000
a=fmtp:97 profile=0;level=30
```

Legacy Client Hypothetical Offer

Fred would offer a single mono audio and a single video, each with a couple of different codec alternatives.

The same conference server as in the previous example is calling out to Fred, offering the full set of multi-stream and simulcast features, with maximum stream and bandwidth limits based on what the server itself can support.

```
v=0
o=server 323439283 2384192332 IN IP4 198.51.100.2
s=Multi stream and Simulcast Dial-out Offer
c=IN IP4 198.51.100.43
b=AS:7065
a=group:SCR 2 3 4
m=audio 49200 RTP/AVP 96 97 9 8
b=AS:435
a=rtpmap:96 G719/48000/2
a=rtpmap:97 G719/48000
a=rtpmap:9 G722/8000
a=rtpmap:8 PCMA/8000
a=max-send-ssrc:* 4
a=max-recv-ssrc:* 3
a=bw:send pt=96 SMT:tb=128800:1500
a=bw:send pt=97 SMT:tb=64800:1500
a=bw:send pt=8;9 SMT:tb=64000:1500
a=bw:send pt=* AMT:tb=515200:1500
a=bw:recv pt=* AMT:tb=386400:1500
a=ssrc:3293472833 cname:server@conf1.example.com
a=ssrc:3293472833 srcname:28:23:54:39:7a:0e
a=ssrc:1734728348 cname:server@conf1.example.com
a=ssrc:1734728348 srcname:83:88:be:19:a6:15
a=ssrc:1054453769 cname:server@conf1.example.com
a=ssrc:1054453769 srcname:76:91:cc:23:02:68
a=ssrc:3923447729 cname:server@conf1.example.com
a=ssrc:3923447729 srcname:be:73:a6:03:00:82
a=mid:1
m=video 49300 RTP/AVP 96
b=AS:4650
a=rtpmap:96 H264/90000
a=fmtp:96 profile-level-id=42c01f
a=imageattr:* send [x=1280,y=720] [x=640,y=360] [x=320,y=180]
    recv [x=1280,y=720]
a=max-recv-ssrc:96 3
a=max-send-ssrc:96 3
a=bw:recv pt=96 SMT:tb=1500000:16384
a=bw:recv pt=* AMT:tb=4500000:16384
```

```
   a=bw:send pt=96 SMT:tb=1500000:16384
   a=bw:send pt=* AMT:tb=4500000:16384
   a=ssrc:78456398 cname:server@conf1.example.com
   a=ssrc:78456398 srcname:28:23:54:39:7a:0e
   a=ssrc:3284726348 cname:server@conf1.example.com
   a=ssrc:3284726348 srcname:83:88:be:19:a6:15
   a=ssrc:2394871293 cname:server@conf1.example.com
   a=ssrc:2394871293 srcname:76:91:cc:23:02:68
   a=mid:2
   a=content:main
   m=video 49400 RTP/AVP 96
   b=AS:1560
   a=rtpmap:96 H264/90000
   a=fmtp:96 profile-level-id=42c01e
   a=imageattr:* recv [x=640,y=360]
   a=max-recv-ssrc:96 3
   a=bw:recv pt=96 SMT:tb=500000:8192
   a=mid:3
   a=recvonly
   m=video 49500 RTP/AVP 96
   b=AS:420
   a=rtpmap:96 H264/90000
   a=fmtp:96 profile-level-id=42c00d
   a=imageattr:96 recv [x=320,y=180]
   a=max-recv-ssrc:96 3
   a=bw:recv pt=96 SMT:tb=128000:4096
   a=mid:4
   a=recvonly
   m=video 49600 RTP/AVP 96 97 98
   b=AS:2600
   a=rtpmap:96 H264/90000
   a=fmtp:96 profile-level-id=42c01f
   a=imageattr:96 send [x=1280,y=720]
   a=rtpmap:97 H264/90000
   a=fmtp:97 profile-level-id=42c01e
   a=imageattr:97 send [x=640,y=360]
   a=rtpmap:98 H264/90000
   a=fmtp:98 profile-level-id=42c00d
   a=imageattr:98 send [x=320,y=180]
   a=max-send-ssrc:96 1
   a=max-send-ssrc:97 4
   a=max-send-ssrc:98 8
   a=max-send-ssrc:* 8
   a=bw:send pt=96 SMT:tb=1500000:16384
   a=bw:send pt=97 SMT:tb=500000:8192
   a=bw:send pt=98 SMT:tb=128000:4096
   a=bw:send pt=* AMT:tb=2500000:16384
   a=ssrc:2342872394 cname:server@conf1.example.com
```

```
a=ssrc:1283741823 cname:server@conf1.example.com
a=ssrc:3294823947 cname:server@conf1.example.com
a=ssrc:1020408838 cname:server@conf1.example.com
a=ssrc:1999343791 cname:server@conf1.example.com
a=ssrc:2934192349 cname:server@conf1.example.com
a=ssrc:2234347728 cname:server@conf1.example.com
a=ssrc:3224283479 cname:server@conf1.example.com
a=mid:5
a=sendonly
a=content:alt
```

Server Dial-out Offer with Multi-stream and Simulcast

The answer from Fred to this offer would look like:

```
v=0
o=fred 9842793823 239482793 IN IP4 192.0.2.213
s=Legacy Client Answer to Server Dial-out
t=0 0
c=IN IP4 192.0.2.213
m=audio 50132 RTP/AVP 9
b=AS:80
a=rtpmap:9 G722/8000
m=video 50134 RTP/AVP 96
b=AS:405
a=rtpmap:96 H264/90000
a=fmtp:96 profile-level-id=42c00c
m=video 0 RTP/AVP 96
m=video 0 RTP/AVP 96
m=video 0 RTP/AVP 96
```

Legacy Client Answer to Server Dial-out

as can be seen from the hypothetical offer, Fred does not understand any of the multistream or simulcast attributes, and does also not understand the grouping framework. Thus, all those lines are removed from the answer SDP and any surplus video media blocks except for the first are rejected. The media bandwidth are adjusted down to what Fred actually accepts to receive.

8.5.2.4. Joe: Dial-out to Desktop Client

This example is almost identical to the one above, with the difference that the answering end-point has some limited simulcast and multi-stream capability. As above this is the offer SDP that Joe would have used, should he have called in.

```
v=0
```

```
   o=joe 82342187 237429834 IN IP4 192.0.2.213
   s=Simulcast and Multistream enabled Desktop Client
   t=0 0
   c=IN IP4 192.0.2.213
   b=AS:985
   a=group:SCS 2 3
   m=audio 49200 RTP/AVP 96 97 9 8
   b=AS:145
   a=rtpmap:96 G719/48000/2
   a=rtpmap:97 G719/48000
   a=rtpmap:9 G722/8000
   a=rtpmap:8 PCMA/8000
   a=bw:send pt=96 SMT:tb=128800:1500
   a=bw:send pt=97 SMT:tb=64800:1500
   a=bw:send pt=8;9 SMT:tb=64000:1500
   a=bw:recv pt=* AMT:tb=128800:1500
   a=ssrc:1223883729 cname:joe@foo.example.com
   a=ssrc:1223883729 srcname:12:88:07:cf:81:65
   a=mid:1
   m=video 49300 RTP/AVP 96
   b=AS:520
   a=rtpmap:96 H264/90000
   a=fmtp:96 profile-level-id=42c01e
   a=imageattr:96 send [x=640,y=360] recv [x=640,y=360] [x=320,y=180]
   a=bw:send pt=96 SMT:tb=500000:8192
   a=bw:recv pt=96 SMT:tb=500000:8192
   a=ssrc:3842394823 cname:joe@foo.example.com
   a=ssrc:3842394823 srcname:12:88:07:cf:81:65
   a=mid:2
   a=content:main
   m=video 49400 RTP/AVP 96
   b=AS:160
   a=rtpmap:96 H264/90000
   a=fmtp:96 profile-level-id=42c00d
   a=imageattr:96 send [x=320,y=180]
   a=bw:send pt=96 SMT:tb=128000:4096
   a=bw:recv pt=96 SMT:tb=128000:4096
   a=ssrc:1214232284 cname:joe@foo.example.com
   a=ssrc:1214232284 srcname:12:88:07:cf:81:65
   a=mid:3
   a=sendonly
   m=video 49300 RTP/AVP 96
   b=AS:320
   a=rtpmap:96 H264/90000
   a=fmtp:96 profile-level-id=42c00c
   a=imageattr:96 recv [x=320,y=180]
   a=max-recv-ssrc:* 2
   a=bw:recv pt=96 SMT:tb=128000:4096
```

```
a=bw:recv pt=96 AMT:tb=256000:4096
a=mid:4
a=recvonly
a=content:alt
```

Desktop Client Hypothetical Offer

Joe would send two versions of simulcast, 360p and 180p, from a single camera and can receive three sources of multi-stream, one 360p and two 180p streams.

Again, the same conference server is calling out to Joe and the offer SDP from the server would be almost identical to the one in the previous example. It is therefore not included here. The response from Joe would look like:

```
v=0
o=joe 239482639 4702341992 IN IP4 192.0.2.213
s=Answer from Desktop Client to Server Dial-out
t=0 0
c=IN IP4 192.0.2.213
b=AS:985
a=group:SCS 2 3
m=audio 49200 RTP/AVP 96
b=AS:145
a=rtpmap:96 G719/48000/2
a=bw:send pt=96 SMT:tb=128800:1500
a=bw:recv pt=* AMT:tb=128800:1500
a=ssrc:1223883729 cname:joe@foo.example.com
a=ssrc:1223883729 srcname:12:88:07:cf:81:65
a=mid:1
m=video 49300 RTP/AVP 96
b=AS:520
a=rtpmap:96 H264/90000
a=fmtp:96 profile-level-id=42c01e
a=imageattr:96 send [x=640,y=360] recv [x=640,y=360] [x=320,y=180]
a=bw:send pt=96 SMT:tb=500000:8192
a=bw:recv pt=96 SMT:tb=500000:8192
a=ssrc:3842394823 cname:joe@foo.example.com
a=ssrc:3842394823 srcname:12:88:07:cf:81:65
a=mid:2
a=content:main
m=video 0 RTP/AVP 96
a=mid:3
m=video 49400 RTP/AVP 96
b=AS:160
a=rtpmap:96 H264/90000
a=fmtp:96 profile-level-id=42c00d
```

```
a=imageattr:96 send [x=320,y=180]
a=bw:send pt=96 SMT:tb=128000:4096
a=bw:recv pt=96 SMT:tb=128000:4096
a=ssrc:1214232284 cname:joe@foo.example.com
a=ssrc:1214232284 srcname:12:88:07:cf:81:65
a=mid:4
a=sendonly
m=video 49300 RTP/AVP 96
b=AS:320
a=rtpmap:96 H264/90000
a=fmtp:96 profile-level-id=42c00c
a=imageattr:96 recv [x=320,y=180]
a=max-recv-ssrc:* 2
a=bw:recv pt=96 SMT:tb=128000:4096
a=bw:recv pt=96 AMT:tb=256000:4096
a=mid:5
a=recvonly
a=content:alt
```

Desktop Client Answer to Server Dial-out

Since the RTP mixer support all of the features that Joe does and more, the SDP does not differ much from what it should have been in an offer. It can be noted that as stated in [RFC5888], all media lines need mid attributes, even the rejected ones, which is why mid:3 is present even though the mid quality simulcast version is rejected by Joe.

9. IANA Considerations

Following the guidelines in [RFC4566], in [RFC5888], and in [RFC3550], the IANA is requested to register:

1. The SID grouping tag to be used with the grouping framework, as defined in Section 8.1.1

2. A new SDES Item named SRCNAME, as defined in Section 5.3.1

3. The max-send-ssrc and max-recv-ssrc SDP attributes as defined in Section 5.1

4. The bw attribute as defined in Section 5.2

5. The bw attribute scope registry rules

6. The bw attribute semantics registry rules

10. Security Considerations

There is minimal difference in security between the simulcast solutions. Session multiplexing may have some additional overhead in the key-management, but that is minor as most key management schemes can be performed in parallel.

The multi-stream signalling has as other SDP based signalling issues with man in the middles that may modify the SDP as an attack on either the service in general or a particular end-point. This can as usual be resolved by a security mechanism that provides integrity and source authentication between the signalling peers.

The SDES SRCNAME being 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 RFC6222 for more discussion.

11. Acknowledgements

12. References

12.1. 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.

[RFC5234] Crocker, D. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", STD 68, RFC 5234, January 2008.

[RFC6222] Begen, A., Perkins, C., and D. Wing, "Guidelines for Choosing RTP Control Protocol (RTCP) Canonical Names (CNAMEs)", RFC 6222, April 2011.

12.2. Informative References

[RFC2205] Braden, B., Zhang, L., Berson, S., Herzog, S., and S. Jamin, "Resource ReSerVation Protocol (RSVP) -- Version 1 Functional Specification", RFC 2205, September 1997.

[RFC2474] Nichols, K., Blake, S., Baker, F., and D. Black,

"Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers", RFC 2474, December 1998.

[RFC3264] Rosenberg, J. and H. Schulzrinne, "An Offer/Answer Model with Session Description Protocol (SDP)", RFC 3264, June 2002.

[RFC4103] Hellstrom, G. and P. Jones, "RTP Payload for Text Conversation", RFC 4103, June 2005.

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

[RFC4585] Ott, J., Wenger, S., Sato, N., Burmeister, C., and J. Rey, "Extended RTP Profile for Real-time Transport Control Protocol (RTCP)-Based Feedback (RTP/AVPF)", RFC 4585, July 2006.

[RFC4588] Rey, J., Leon, D., Miyazaki, A., Varsa, V., and R. Hakenberg, "RTP Retransmission Payload Format", RFC 4588, July 2006.

[RFC4796] Hautakorpi, J. and G. Camarillo, "The Session Description Protocol (SDP) Content Attribute", RFC 4796, February 2007.

[RFC5104] Wenger, S., Chandra, U., Westerlund, M., and B. Burman, "Codec Control Messages in the RTP Audio-Visual Profile with Feedback (AVPF)", RFC 5104, February 2008.

[RFC5117] Westerlund, M. and S. Wenger, "RTP Topologies", RFC 5117, January 2008.

[RFC5576] Lennox, J., Ott, J., and T. Schierl, "Source-Specific Media Attributes in the Session Description Protocol (SDP)", RFC 5576, June 2009.

[RFC5761] Perkins, C. and M. Westerlund, "Multiplexing RTP Data and Control Packets on a Single Port", RFC 5761, April 2010.

[RFC5888] Camarillo, G. and H. Schulzrinne, "The Session Description Protocol (SDP) Grouping Framework", RFC 5888, June 2010.

[RFC6236] Johansson, I. and K. Jung, "Negotiation of Generic Image Attributes in the Session Description Protocol (SDP)", RFC 6236, May 2011.

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 7141311
Email: bo.burman@ericsson.com