Network Working Group
Internet-Draft
Obsoletes: 5117 (if approved)
Intended status: Informational
Expires: April 18, 2013

M. Westerlund
Ericsson
S. Wenger
Vidyo

October 15, 2012

RTP Topologies
draft-westerlund-avtcore-rtp-topologies-update-01

Abstract

This document discusses multi-endpoint topologies used in Real-time Transport Protocol (RTP)-based environments. In particular, centralized topologies commonly employed in the video conferencing industry are mapped to the RTP terminology.

This document is updated with additional topologies and are intended to replace RFC 5117.

Status of this Memo

Copyright Notice

Table of Contents

1. Introduction

When working on the Codec Control Messages [RFC5104], considerable confusion was noticed in the community with respect to terms such as Multipoint Control Unit (MCU), Mixer, and Translator, and their usage in various topologies. This document tries to address this confusion by providing a common information basis for future discussion and specification work. It attempts to clarify and explain sections of the Real-time Transport Protocol (RTP) spec [RFC3550] in an informal way. It is not intended to update or change what is normatively specified within RFC 3550.

When the Audio-Visual Profile with Feedback (AVPF) [RFC4585] was developed the main emphasis lay in the efficient support of point to point and small multipoint scenarios without centralized multipoint control. However, in practice, many small multipoint conferences operate utilizing devices known as Multipoint Control Units (MCUs). MCUs may implement Mixer or Translator (in RTP [RFC3550] terminology) functionality and signalling support. They may also contain additional application functionality. This document focuses on the media transport aspects of the MCU that can be realized using RTP, as discussed below. Further considered are the properties of Mixers and Translators, and how some types of deployed MCUs deviate from these properties.

2. Definitions

2.1. Glossary

ASM:	Any Source Multicast
AVPF:	The Extended RTP Profile for RTCP-based Feedback
CSRC:	Contributing Source
Link:	The data transport to the next IP hop
MCU:	Multipoint Control Unit
Path:	The concatenation of multiple links, resulting in an end-to-end data transfer.
PtM:	Point to Multipoint

PtP: Point to Point

SSM: Source-Specific Multicast

SSRC: Synchronization Source

2.2. Indicating Requirement Levels

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

The RFC 2119 language is used in this document to highlight those important requirements and/or resulting solutions that are necessary to address the issues raised in this document.

3. Topologies

This subsection defines several topologies that are relevant for codec control but also RTP usage in other contexts. The first relate to the RTP system model utilizing multicast and/or unicast, as envisioned in RFC 3550. Two later topologies (MCU and RTCP terminating), in contrast, describe the deployed system models as used in many H.323 [H323] video conferences, where both the media streams and the RTP Control Protocol (RTCP) control traffic terminate at the MCU. In these two cases, the media sender does not receive the (unmodified or Translator-modified) Receiver Reports from all sources (which it needs to interpret based on Synchronization Source (SSRC) values) and therefore has no full information about all the endpoint's situation as reported in RTCP Receiver Reports (RRs). More topologies can be constructed by combining any of the models; see Section 3.11.

The topologies may be referenced in other documents by a shortcut name, indicated by the prefix "Topo-".

For each of the RTP-defined topologies, we discuss how RTP, RTCP, and the carried media are handled. With respect to RTCP, we also introduce the handling of RTCP feedback messages as defined in [RFC4585] and [RFC5104]. Any important differences between the two will be illuminated in the discussion.

3.1. Point to Point

Shortcut name: Topo-Point-to-Point

The Point to Point (PtP) topology (Figure 1) consists of two

endpoints, communicating using unicast. Both RTP and RTCP traffic are conveyed endpoint-to-endpoint, using unicast traffic only (even if, in exotic cases, this unicast traffic happens to be conveyed over an IP-multicast address).

```
                         +---+         +---+
                         | A |<------->| B |
                         +---+         +---+
```

Figure 1: Point to Point

The main property of this topology is that A sends to B, and only B, while B sends to A, and only A. This avoids all complexities of handling multiple endpoints and combining the requirements from them. Note that an endpoint can still use multiple RTP Synchronization Sources (SSRCs) in an RTP session. The number of RTP sessions in use between A and B can also be of any number.

RTCP feedback messages for the indicated SSRCs are communicated directly between the endpoints. Therefore, this topology poses minimal (if any) issues for any feedback messages.

3.2. Point to Multipoint Using Multicast

Multicast is a IP layer functionality that is available in some networks. It comes in two main flavors, Any Source Multicast (ASM) where any multicast group participant can send to the group address and expect the packet to reach all group participants. The other model is Source Specific Multicast (SSM) where only a particular IP host is allowed to send to the multicast group. Both these models are discussed below in their respective section.

3.2.1. Any Source Multicast (ASM)

Shortcut name: Topo-ASM (was Topo-Multicast)

```
                              +-----+
                    +---+    /       \    +---+
                    | A |----/         \---| B |
                    +---+   /   Multi-  \  +---+
                           +    Cast     +
                    +---+   \  Network  /  +---+
                    | C |----\         /---| D |
                    +---+    \       /    +---+
                              +-----+
```

Figure 2: Point to Multipoint Using Multicast

Point to Multipoint (PtM) is defined here as using a multicast topology as a transmission model, in which traffic from any participant reaches all the other participants, except for cases such as:

- o packet loss, or

- o when a participant does not wish to receive the traffic for a specific multicast group and therefore has not subscribed to the IP-multicast group in question. This is for the cases where a multi-media session is distributed using two or more multicast groups.

In the above context, "traffic" encompasses both RTP and RTCP traffic. The number of participants can vary between one and many, as RTP and RTCP scale to very large multicast groups (the theoretical limit of the number of participants in a single RTP session is approximately two billion). The above can be realized using Any Source Multicast (ASM).

For feedback usage it is relevant to make distinction of that subset of multicast sessions wherein the number of participants in the multicast group is so low that it allows the participants to use early or immediate feedback, as defined in AVPF [RFC4585]. This document refers to those groups as "small multicast groups". Some applications may still want to use larger multicast groups where the RTCP feedback possibilities are more limited.

RTCP feedback messages in multicast will, like media, reach everyone (subject to packet losses and multicast group subscription). Therefore, the feedback suppression mechanism discussed in [RFC4585] is required. Each individual node needs to process every feedback message it receives to determine if it is affected or if the feedback message applies only to some other participant.

3.2.2. Source Specific Multicast (SSM)

In Any Source Multicast, any of the participants can send to all the other participants, simply by sending a packet to the multicast group. That is not possible in Source Specific Multicast [RFC4607] where only a single source (Distribution Source) can send to the multicast group, creating a topology that looks like the one below:

```
       +--------+       +-----+
       |Media   |       |     |       Source-specific
       |Sender 1|<----->| D S |          Multicast
       +--------+       | I O |  +--+----------------> R(1)
                        | S U |  |  |                    |
       +--------+       | T R |  |  +-----------> R(2)   |
       |Media   |<----->| R C |->+  |           :   |    |
       |Sender 2|       | I E |  |  +------> R(n-1) |    |
       +--------+       | B   |  |  |          |    |    |
           :            | U   |  +--+--> R(n)  |    |    |
           :            | T +-|          |     |    |    |
           :            | I | |<---------+     |    |    |
       +--------+       | O |F|<---------------+    |    |
       |Media   |       | N |T|<--------------------+    |
       |Sender M|<----->|   | |<-------------------------+
       +--------+       +-----+       RTCP Unicast

       FT = Feedback Target
       Transport from the Feedback Target to the Distribution
       Source is via unicast or multicast RTCP if they are not
       co-located.
```

Figure 3: Point to Multipoint using Source Specific Multicast

In the SSM topology (Figure 3) a number of RTP sources (1 to M) are allowed to send media to the SSM group. These send media to the distribution source which then forwards the media streams to the multicast group. The media streams reach the Receivers (R(1) to R(n)). The Receivers' RTCP cannot be sent to the multicast group. To support RTCP, an RTP extension for SSM [RFC5760] was defined to use unicast transmission to send RTCP from the receivers to one or more Feedback Targets (FT).

The RTP extension for SSM deals with how feedback both general reception information and specific feedback events are generally handled. The general problems of multicast that everyone will receive what the distribution source sends needs to be accounted for.

The result of this is some common behaviours for RTP multicast:

1. Multicast applications often use a group of RTP sessions, not one. Each endpoint will need to be a member of a number of RTP sessions in order to perform well.

2. Within each RTP session, the number of media sinks is likely to be much larger than the number of RTP sources.

3. Multicast applications need signalling functions to identify the relationships between RTP sessions.

4. Multicast applications need signalling functions to identify the relationships between SSRCs in different RTP sessions.

All multicast configurations share a signalling requirement; all of the participants will need to have the same RTP and payload type configuration. Otherwise, A could for example be using payload type 97 as the video codec H.264 while B thinks it is MPEG-2.

Security solutions for this type of group communications are also challenging. First of all the key-management and the security protocol must support group communication. Source authentication becomes more difficult and requires special solutions. For more discussion on this please review Options for Securing RTP Sessions [I-D.ietf-avtcore-rtp-security-options].

3.2.3. SSM with Local Unicast Resources

[RFC6285] "Unicast-Based Rapid Acquisition of Multicast RTP Sessions" results in additional extensions to SSM Topology. Should be described.

3.3. Point to Multipoint Using Mesh

Shortcut name: Topo-Mesh

```
                   +---+      +---+
                   | A |<---->| B |
                   +---+      +---+
                     ^         ^
                      \       /
                       \     /
                        v   v
                        +---+
                        | C |
                        +---+
```

Figure 4: Point to Multi-Point using Mesh

Based on the RTP session definition, it is clearly possible to have a joint RTP session over multiple unicast transport flows like the above three endpoint joint session. In this case, A needs to send its' media streams and RTCP packets to both B and C over their respective transport flows. As long as all participants do the same, everyone will have a joint view of the RTP session.

This doesn't create any additional requirements beyond the need to have multiple transport flows associated with a single RTP session. Note that an endpoint may use a single local port to receive all these transport flows, or it might have separate local reception ports for each of the endpoints.

There exists an alternative structure for establishing the above topology which uses independent RTP sessions between each pair of peers, i.e. three different RTP sessions. Unless independently adapted the same RTP media stream could be sent in both of the RTP sessions an endpoint has. The difference exists in the behaviours around RTCP, for example common RTCP bandwidth for one joint session, rather than three independent pools, and the awareness based on RTCP reports between the peers of how that third leg is doing.

3.4. Point to Multipoint Using the RFC 3550 Translator

Shortcut name: Topo-Translator

Two main categories of Translators can be distinguished; Transport Translators and Media translators. Both Translator types share common attributes that separate them from Mixers. For each media stream that the Translator receives, it generates an individual stream in the other domain. A Translator always keeps the SSRC for a stream across the translation, where a Mixer can select a media stream, or send them out mixed, always under its own SSRC, using the CSRC field to indicate the source(s) of the content.

As specified in Section 7.1 of [RFC3550], the SSRC space is common for all participants in the session, independent of on which side they are of the Translator. Therefore, it is the responsibility of the participants to run SSRC collision detection, and the SSRC is a field the Translator should not change.

A Translator commonly does not use an SSRC of its own, and is not visible as an active participant in the session. One reason is when a Translator acts as a quality monitor that sends RTCP reports and therefore is required to have an SSRC. Another example is the case when a Translator is prepared to use RTCP feedback messages. This may, for example, occur when it suffers packet loss of important video packets and wants to trigger repair by the media sender, by sending feedback messages. This can be done using the SSRC of the target for the translator, but this requires translation of the targets RTCP reports to make them consistent, so it is likely simpler to expose an additional SSRC in the session.

In general, a Translator implementation should consider which RTCP feedback messages or codec-control messages it needs to understand in

relation to the functionality of the Translator itself. This is completely in line with the requirement to also translate RTCP messages between the domains.

The RTCP translation process can be trivial, for example, when Transport Translators just need to adjust IP addresses and transport protocol ports, or they can be quite complex as in the case of media Translators. See Section 7.2 of [RFC3550].

3.4.1. Relay - Transport Translator

Transport Translators (Topo-Trn-Translator) do not modify the media stream itself, but are concerned with transport parameters. Transport parameters, in the sense of this section, comprise the transport addresses (to bridge different domains) and the media packetization to allow other transport protocols to be interconnected to a session (in gateways). Of the transport Translators, this memo is primarily interested in those that use RTP on both sides, and this is assumed henceforth. Translators that bridge between different protocol worlds need to be concerned about the mapping of the SSRC/ CSRC (Contributing Source) concept to the non-RTP protocol. When designing a Translator to a non-RTP-based media transport, one crucial factor lies in how to handle different sources and their identities. This problem space is not discussed henceforth.

```
                     +------+
          +---+     /        \     +------------+      +---+
          | A |<---/          \    |            |<---->| B |
          +---+   /   Multi-  \   |            |      +---+
                 +    Cast     +->| Translator |
          +---+   \  Network  /   |            |      +---+
          | C |<---\         /    |            |<---->| D |
          +---+     \       /     +------------+      +---+
                     +------+
```

Figure 5: Point to Multipoint Using Multicast

Figure 5 depicts an example of a Transport Translator performing at least IP address translation. It allows the (non-multicast-capable) participants B and D to take part in an any source multicast session by having the Translator forward their unicast traffic to the multicast addresses in use, and vice versa. It must also forward B's traffic to D, and vice versa, to provide each of B and D with a complete view of the session.

Also a point to point communication can end up in a situation when the peer it is communicating with needs basic transport translators functions. This include NAT traversal by pinning the media path to a

public address domain relay, network topologies where the media flow is required to pass a particular point by employing relaying or preserving privacy by hiding each peers transport addresses to the other party.

```
+---+        +---+         +---+
| A |<------>| T |<------->| B |
+---+        +---+         +---+
```

Point to Point with Translator

This type of very basic relay service should in most case need to have no RTP functionality. Thus one can believe that they do not need to included in this document. However, due to that the network level addressing and the RTP identifier view of the RTP session and who the peer is doesn't match as in the PtP unicast scenario depicted above this topology can raise additional requirements.

```
+---+      +------------+      +---+
| A |<---->|            |<---->| B |
+---+      |            |      +---+
           | Translator |
+---+      |            |      +---+
| C |<---->|            |<---->| D |
+---+      +------------+      +---+
```

Figure 6: RTP Translator (Relay) with Only Unicast Paths

Another Translator scenario is depicted in Figure 6. Herein, the Translator connects multiple users of a conference through unicast. This can be implemented using a very simple transport Translator, which in this document is called a relay. The relay forwards all traffic it receives, both RTP and RTCP, to all other participants. In doing so, a multicast network is emulated without relying on a multicast-capable network infrastructure.

For RTCP feedback this results in a similar considerations that arise for the ASM RTP topology. It also puts some special signalling requirements where common configuration of RTP payload types for example are required.

3.4.2. **Media Translator**

Media Translators (Topo-Media-Translator), in contrast, modify the media stream itself. This process is commonly known as transcoding. The modification of the media stream can be as small as removing parts of the stream, and it can go all the way to a full transcoding

(down to the sample level or equivalent) utilizing a different media codec. Media Translators are commonly used to connect entities without a common interoperability point.

Stand-alone Media Translators are rare. Most commonly, a combination of Transport and Media Translators are used to translate both the media stream and the transport aspects of a stream between two transport domains (or clouds).

If B in Figure 5 were behind a limited network path, the Translator may perform media transcoding to allow the traffic received from the other participants to reach B without overloading the path.

When, in the example depicted in Figure 5, the Translator acts only as a Transport Translator, then the RTCP traffic can simply be forwarded, similar to the media traffic. However, when media translation occurs, the Translator's task becomes substantially more complex, even with respect to the RTCP traffic. In this case, the Translator needs to rewrite B's RTCP Receiver Report before forwarding them to D and the multicast network. The rewriting is needed as the stream received by B is not the same stream as the other participants receive. For example, the number of packets transmitted to B may be lower than what D receives, due to the different media format and data rate. Therefore, if the Receiver Reports were forwarded without changes, the extended highest sequence number would indicate that B were substantially behind in reception, while it most likely it would not be. Therefore, the Translator must translate that number to a corresponding sequence number for the stream the Translator received. Similar arguments can be made for most other fields in the RTCP Receiver Reports.

A media Translator may in some cases act on behalf of the "real" source and respond to RTCP feedback messages. This may occur, for example, when a receiver requests a bandwidth reduction, and the media Translator has not detected any congestion or other reasons for bandwidth reduction between the media source and itself. In that case, it is sensible that the media Translator reacts to the codec control messages itself, for example, by transcoding to a lower media rate. If it were not reacting, the media quality in the media sender's domain may suffer, as a result of the media sender adjusting its media rate (and quality) according to the needs of the slow past-Translator endpoint, at the expense of the rate and quality of all other session participants.

A variant of translator behaviour worth pointing out is the one depicted in Figure 7 of an endpoint A sends a media flow to B. On the path there is a device T that on A's behalf does something with the media streams, for example adds an RTP session with FEC information

for A's media streams. T will in this case need to bind the new FEC streams to A's media stream, for example by using the same CNAME as A.

```
+------+        +------+         +------+
|      |        |      |         |      |
|  A   |------->|  T   |-------->|  B   |
|      |        |      |---FEC-->|      |
+------+        +------+         +------+
```

Figure 7: When De-composition is a Translator

This type of functionality where T does something with the media stream on behalf of A is clearly covered under the media translator definition.

3.5. Point to Multipoint Using the RFC 3550 Mixer Model

Shortcut name: Topo-Mixer

A Mixer is a middlebox that aggregates multiple RTP streams, which are part of a session, by manipulation of the media data and generating a new RTP stream. One common application for a Mixer is to allow a participant to receive a session with a reduced amount of resources.

```
            +-----+
+---+      /       \     +-----------+      +---+
| A |<---/          \    |           |<---->| B |
+---+   /   Multi-   \   |           |      +---+
       +    Cast      +->|   Mixer   |
+---+   \  Network   /   |           |      +---+
| C |<---\          /    |           |<---->| D |
+---+      \       /     +-----------+      +---+
            +-----+
```

Figure 8: Point to Multipoint Using the RFC 3550 Mixer Model

A Mixer can be viewed as a device terminating the media streams received from other session participants. Using the media data from the received media streams, a Mixer generates a media stream that is sent to the session participant.

The content that the Mixer provides is the mixed aggregate of what the Mixer receives over the PtP or PtM paths, which are part of the same conference session.

The Mixer is the content source, as it mixes the content (often in

the uncompressed domain) and then encodes it for transmission to a participant. The CSRC Count (CC) and CSRC fields in the RTP header are used to indicate the contributors of to the newly generated stream. The SSRCs of the to-be-mixed streams on the Mixer input appear as the CSRCs at the Mixer output. That output stream uses a unique SSRC that identifies the Mixer's stream. The CSRC should be forwarded between the two domains to allow for loop detection and identification of sources that are part of the global session. Note that Section 7.1 of RFC 3550 requires the SSRC space to be shared between domains for these reasons.

The Mixer is responsible for generating RTCP packets in accordance with its role. It is a receiver and should therefore send reception reports for the media streams it receives. In its role as a media sender, it should also generate Sender Reports for those media streams sent. As specified in Section 7.3 of RFC 3550, a Mixer must not forward RTCP unaltered between the two domains.

The Mixer depicted in Figure 8 is involved in three domains that need to be separated: the any source multicast network, participant B, and participant D. The Mixer produces different mixed streams to B and D, as the one to B may contain content received from D, and vice versa. However, the Mixer may only need one SSRC per media type in each domain that is the receiving entity and transmitter of mixed content.

In the multicast domain, a Mixer still needs to provide a mixed view of the other domains. This makes the Mixer simpler to implement and avoids any issues with advanced RTCP handling or loop detection, which would be problematic if the Mixer were providing non-symmetric behavior. Please see Section 3.10 for more discussion on this topic. However, the mixing operation in each domain could potentially be different.

A Mixer is responsible for receiving RTCP feedback messages and handling them appropriately. The definition of "appropriate" depends on the message itself and the context. In some cases, the reception of a codec-control message may result in the generation and transmission of RTCP feedback messages by the Mixer to the participants in the other domain. In other cases, a message is handled by the Mixer itself and therefore not forwarded to any other domain.

When replacing the multicast network in Figure 8 (to the left of the Mixer) with individual unicast paths as depicted in Figure 9, the Mixer model is very similar to the one discussed in Section 3.8 below. Please see the discussion in Section 3.8 about the differences between these two models.

```
                    +---+      +------------+      +---+
                    | A |<---->|            |<---->| B |
                    +---+      |            |      +---+
                               |   Mixer    |
                    +---+      |            |      +---+
                    | C |<---->|            |<---->| D |
                    +---+      +------------+      +---+
```

Figure 9: RTP Mixer with Only Unicast Paths

Lets now discuss in more detail different mixing operations that a mixer can perform and how that can affect the RTP and RTCP.

3.5.1. Media Mixing

The media mixing mixer is likely the one that most thinks of when they hear the term mixer. Its basic pattern of operation is that it will receive the different participants RTP media stream. Select which that are to be included in a media domain mix of the incoming RTP media streams. Then create a single outgoing stream from this mix.

The most commonly deployed media mixer is probably the audio mixer, used in voice conferencing, where the output consists of some mixture of all the input streams; this needs minimal signalling to be successful. Audio mixing is straight forward and commonly possible to do for a number of participants. Lets assume that you want to mix N number of streams from different participants. Then the mixer need to perform N decodings. Then it needs to produce N or N+1 mixes, the reasons that different mixes are needed are so that each contributing source get a mix which don't contain themselves, as this would result in an echo. When N is lower than the number of all participants one may produce a Mix of all N streams for the group that are currently not included in the mix, thus N+1 mixes. These audio streams are then encoded again, RTP packetised and sent out.

Video can't really be "mixed" and produce something particular useful for the users, however creating an composition out of the contributed video streams can be done. In fact it can be done in a number of ways, tiling the different streams creating a chessboard, selecting someone as more important and showing them large and a number of other sources as smaller overlays is another. Also here one commonly need to produce a number of different compositions so that the contributing part doesn't need to see themselves. Then the mixer re-encodes the created video stream, RTP packetise it and send it out

The problem with media mixing is that it both consume large amount of media processing and encoding resources. The second is the quality

degradation created by decoding and re-encoding the RTP media stream. Its advantage is that it is quite simplistic for the clients to handle as they don't need to handle local mixing and composition.

```
           +-A---------+          +-MIXER----------------------+
           | +-RTP1----|          |-RTP1------+        +-----+ |
           | | +-Audio-|          |-Audio---+ | +---+  |     | |
           | | |    AA1|--------->|---------+-+-|DEC|->|     | |
           | | |       |<---------|MA1 <----+ | +---+  |     | |
           | | |       |          |(BA1+CA1)|\| +---+  |     | |
           | | +-------|          |---------+ +-|ENC|<-| B+C | |
           | +---------|          |-----------+ +---+  |     | |
           +-----------+          |                    |     | |
                                  |                    |  M  | |
           +-B---------+          |                    |  E  | |
           | +-RTP2----|          |-RTP2------+        |  D  | |
           | | +-Audio-|          |-Audio---+ | +---+  |  I  | |
           | | |    BA1|--------->|---------+-+-|DEC|->|  A  | |
           | | |       |<---------|MA2 <----+ | +---+  |     | |
           | | +-------|          |(BA1+CA1)|\| +---+  |     | |
           | +---------|          |---------+ +-|ENC|<-| A+C | |
           +-----------+          |-----------+ +---+  |     | |
                                  |                    |  M  | |
           +-C---------+          |                    |  I  | |
           | +-RTP3----|          |-RTP3------+        |  X  | |
           | | +-Audio-|          |-Audio---+ | +---+  |  E  | |
           | | |    CA1|--------->|---------+-+-|DEC|->|  R  | |
           | | |       |<---------|MA3 <----+ | +---+  |     | |
           | | +-------|          |(BA1+CA1)|\| +---+  |     | |
           | +---------|          |---------+ +-|ENC|<-| A+B | |
           +-----------+          |-----------+ +---+  +-----+ |
                                  +----------------------------+
```

Figure 10: Session and SSRC details for Media Mixer

From an RTP perspective media mixing can be very straight forward as can be seen in Figure 10. The mixer present one SSRC towards the peer client, e.g. MA1 to Peer A, which is the media mix of the other participants. As each peer receives a different version produced by the mixer there are no actual relation between the different RTP sessions in the actual media or the transport level information. There is however one connection between RTP1-RTP3 in this figure. It has to do with the SSRC space and the identity information. When A receives the MA1 stream which is a combination of BA1 and CA1 streams, the mixer may include CSRC information in the MA1 stream to identify the contributing source BA1 and CA1.

The CSRC has in its turn utility in RTP extensions, like the in Mixer to Client audio levels RTP header extension [RFC6465]. If the SSRC

from endpoint to mixer leg are used as CSRC in another RTP session then RTP1, RTP2 and RTP3 becomes one joint session as they have a common SSRC space. At this stage the mixer also need to consider which RTCP information it need to expose in the different legs. For the above situation commonly nothing more than the Source Description (SDES) information and RTCP BYE for CSRC need to be exposed. The main goal would be to enable the correct binding against the application logic and other information sources. This also enables loop detection in the RTP session.

3.5.2. Media Switching

An RTP Mixer based on media switching avoids the media decoding and encoding cycle in the mixer, but not the decryption and re-encryption cycle as it rewrites RTP headers. This both reduces the amount of computational resources needed in the mixer and increases the media quality per transmitted bit. This is achieve by letting the mixer have a number of SSRCs that represents conceptual or functional streams the mixer produces. These streams are created by selecting media from one of the by the mixer received RTP media streams and forward the media using the mixers own SSRCs. The mixer can then switch between available sources if that is required by the concept for the source, like currently active speaker.

To achieve a coherent RTP media stream from the mixer's SSRC the mixer is forced to rewrite the incoming RTP packet's header. First the SSRC field must be set to the value of the Mixer's SSRC. Secondly, the sequence number must be the next in the sequence of outgoing packets it sent. Thirdly the RTP timestamp value needs to be adjusted using an offset that changes each time one switch media source. Finally depending on the negotiation the RTP payload type value representing this particular RTP payload configuration may have to be changed if the different endpoint mixer legs have not arrived on the same numbering for a given configuration. This also requires that the different end-points do support a common set of codecs, otherwise media transcoding for codec compatibility is still required.

Lets consider the operation of media switching mixer that supports a video conference with six participants (A-F) where the two latest speakers in the conference are shown to each participants. Thus the mixer has two SSRCs sending video to each peer.

```
          +-A---------+             +-MIXER----------------------+
          | +-RTP1----|             |-RTP1------+        +-----+ |
          | | +-Video-|             |-Video---+ |        |     | |
          | | |    AV1|------------>|---------+-+------->|  S  | |
          | | |       |<------------|MV1 <----+-+-BV1----|  W  | |
          | | |       |<------------|MV2 <----+-+-EV1----|  I  | |
          | | +-------|             |---------+ |        |  T  | |
          | +---------|             |-----------+        |  C  | |
          +-----------+             |                    |  H  | |
                                    |                    |     | |
          +-B---------+             |                    |  M  | |
          | +-RTP2----|             |-RTP2------+        |  A  | |
          | | +-Video-|             |-Video---+ |        |  T  | |
          | | |    BV1|------------>|---------+-+------->|  R  | |
          | | |       |<------------|MV3 <----+-+-AV1----|  I  | |
          | | |       |<------------|MV4 <----+-+-EV1----|  X  | |
          | | +-------|             |---------+ |        |     | |
          | +---------|             |-----------+        |     | |
          +-----------+             |                    |     | |
                                    :                    :     : :
                                    :                    :     : :
          +-F---------+             |                    |     | |
          | +-RTP6----|             |-RTP6------+        |     | |
          | | +-Video-|             |-Video---+ |        |     | |
          | | |    CV1|------------>|---------+-+------->|     | |
          | | |       |<------------|MV11 <---+-+-AV1----|     | |
          | | |       |<------------|MV12 <---+-+-EV1----|     | |
          | | +-------|             |---------+ |        |     | |
          | +---------|             |-----------+        +-----+ |
          +-----------+             +----------------------------+
```

Figure 11: Media Switching RTP Mixer

The Media Switching RTP mixer can similar to the Media Mixing one reduce the bit-rate needed towards the different peers by selecting and switching in a sub-set of RTP media streams out of the ones it receives from the conference participants.

To ensure that a media receiver can correctly decode the RTP media stream after a switch, it becomes necessary to ensure for state saving codecs that they start from default state at the point of switching. Thus one common tool for video is to request that the encoding creates an intra picture, something that isn't dependent on earlier state. This can be done using Full Intra Request [RFC5104] RTCP codec control message.

Also in this type of mixer one could consider to terminate the RTP

sessions fully between the different end-point and mixer legs. The same arguments and considerations as discussed in Section 3.8 applies here.

3.6. Source Projecting Middlebox

Another method for handling media in the RTP mixer is to project all potential RTP sources (SSRCs) into a per end-point independent RTP session. The mixer can then select which of the potential sources that are currently actively transmitting media, despite that the mixer in another RTP session receives media from that end-point. This is similar to the media switching Mixer but have some important differences in RTP details.

```
          +-A---------+             +-MIXER---------------------+
          | +-RTP1----|             |-RTP1------+       +-----+ |
          | | +-Video-|             |-Video---+ |       |     | |
          | | |    AV1|------------>|---------+-+------>|     | |
          | | |       |<------------|BV1 <----+-+-------|  S  | |
          | | |       |<------------|CV1 <----+-+-------|  W  | |
          | | |       |<------------|DV1 <----+-+-------|  I  | |
          | | |       |<------------|EV1 <----+-+-------|  T  | |
          | | |       |<------------|FV1 <----+-+-------|  C  | |
          | | +-------|             |---------+ |       |  H  | |
          | +---------|             |-----------+       |     | |
          +-----------+             |                   |  M  | |
                                    |                   |  A  | |
          +-B---------+             |                   |  T  | |
          | +-RTP2----|             |-RTP2------+       |  R  | |
          | | +-Video-|             |-Video---+ |       |  I  | |
          | | |    BV1|------------>|---------+-+------>|  X  | |
          | | |       |<------------|AV1 <----+-+-------|     | |
          | | |       |<------------|CV1 <----+-+-------|     | |
          | | |       | :    :    : |: :  : : : : :  : :|     | |
          | | |       |<------------|FV1 <----+-+-------|     | |
          | | +-------|             |---------+ |       |     | |
          | +---------|             |-----------+       |     | |
          +-----------+             |                   |     | |
                                    :                   :     : :
                                    :                   :     : :
          +-F---------+             |                   |     | |
          | +-RTP6----|             |-RTP6------+       |     | |
          | | +-Video-|             |-Video---+ |       |     | |
          | | |    CV1|------------>|---------+-+------>|     | |
          | | |       |<------------|AV1 <----+-+-------|     | |
          | | |       | :    :    : |: :  : : : : :  : :|     | |
          | | |       |<------------|EV1 <----+-+-------|     | |
          | | +-------|             |---------+ |       |     | |
          | +---------|             |-----------+       +-----+ |
          +-----------+             +---------------------------+
```

Figure 12: Media Projecting Mixer

So in this six participant conference depicted above in (Figure 12) one can see that end-point A will in this case be aware of 5 incoming SSRCs, BV1-FV1. If this mixer intend to have the same behaviour as in Section 3.5.2 where the mixer provides the end-points with the two latest speaking end-points, then only two out of these five SSRCs will concurrently transmit media to A. As the mixer selects which source in the different RTP sessions that transmit media to the end-points each RTP media stream will require some rewriting when being projected from one session into another. The main thing is that the

sequence number will need to be consecutively incremented based on the packet actually being transmitted in each RTP session. Thus the RTP sequence number offset will change each time a source is turned on in a RTP session.

As the RTP sessions are independent the SSRC numbers used can be handled independently also thus working around any SSRC collisions by having remapping tables between the RTP sessions. This will result that each endpoint may have a different view of the application usage of a particular SSRC. Thus the application must not use SSRC as references to RTP media streams when communicating with other peers directly.

The mixer will also be responsible to act on any RTCP codec control requests coming from an end-point and decide if it can act on it locally or needs to translate the request into the RTP session that contains the media source. Both end-points and the mixer will need to implement conference related codec control functionalities to provide a good experience. Full Intra Request to request from the media source to provide switching points between the sources, Temporary Maximum Media Bit-rate Request (TMMBR) to enable the mixer to aggregate congestion control response towards the media source and have it adjust its bit-rate in case the limitation is not in the source to mixer link.

This version of the mixer also puts different requirements on the end-point when it comes to decoder instances and handling of the RTP media streams providing media. As each projected SSRC can at any time provide media the end-point either needs to handle having thus many allocated decoder instances or have efficient switching of decoder contexts in a more limited set of actual decoder instances to cope with the switches. The WebRTC application also gets more responsibility to update how the media provides is to be presented to the user.

Note, this could potentially be seen as a media translator which include an on/off logic as part of its media translation. The main difference would be a common global SSRC space in the case of the Media Translator and the mapped one used in the above.

3.7. Point to Multipoint Using Video Switching MCUs

Shortcut name: Topo-Video-switch-MCU

```
                    +---+      +------------+      +---+
                    | A |------| Multipoint |------| B |
                    +---+      |  Control   |      +---+
                               |   Unit     |
                    +---+      |   (MCU)    |      +---+
                    | C |------|            |------| D |
                    +---+      +------------+      +---+
```

Figure 13: Point to Multipoint Using a Video Switching MCU

This PtM topology is still deployed today, although the RTCP-terminating MCUs, as discussed in the next section, are perhaps more common. This topology, as well as the following one, reflect today's lack of wide availability of IP multicast technologies, as well as the simplicity of content switching when compared to content mixing. The technology is commonly implemented in what is known as "Video Switching MCUs".

A video switching MCU forwards to a participant a single media stream, selected from the available streams. The criteria for selection are often based on voice activity in the audio-visual conference, but other conference management mechanisms (like presentation mode or explicit floor control) are known to exist as well.

The video switching MCU may also perform media translation to modify the content in bit-rate, encoding, or resolution. However, it still may indicate the original sender of the content through the SSRC. In this case, the values of the CC and CSRC fields are retained.

If not terminating RTP, the RTCP Sender Reports are forwarded for the currently selected sender. All RTCP Receiver Reports are freely forwarded between the participants. In addition, the MCU may also originate RTCP control traffic in order to control the session and/or report on status from its viewpoint.

The video switching MCU has most of the attributes of a Translator. However, its stream selection is a mixing behavior. This behavior has some RTP and RTCP issues associated with it. The suppression of all but one media stream results in most participants seeing only a subset of the sent media streams at any given time, often a single stream per conference. Therefore, RTCP Receiver Reports only report on these streams. Consequently, the media senders that are not currently forwarded receive a view of the session that indicates their media streams disappear somewhere en route. This makes the use of RTCP for congestion control, or any type of quality reporting, very problematic.

To avoid the aforementioned issues, the MCU needs to implement two features. First, it needs to act as a Mixer (see Section 3.5) and forward the selected media stream under its own SSRC and with the appropriate CSRC values. Second, the MCU needs to modify the RTCP RRs it forwards between the domains. As a result, it is RECOMMENDED that one implement a centralized video switching conference using a Mixer according to RFC 3550, instead of the shortcut implementation described here.

3.8. Point to Multipoint Using RTCP-Terminating MCU

Shortcut name: Topo-RTCP-terminating-MCU

```
                 +---+      +------------+      +---+
                 | A |<---->| Multipoint |<---->| B |
                 +---+      |  Control   |      +---+
                            |   Unit     |
                 +---+      |   (MCU)    |      +---+
                 | C |<---->|            |<---->| D |
                 +---+      +------------+      +---+
```

Figure 14: Point to Multipoint Using Content Modifying MCUs

In this PtM scenario, each participant runs an RTP point-to-point session between itself and the MCU. This is a very commonly deployed topology in multipoint video conferencing. The content that the MCU provides to each participant is either:

a. a selection of the content received from the other participants, or

b. the mixed aggregate of what the MCU receives from the other PtP paths, which are part of the same conference session.

In case a), the MCU may modify the content in bit-rate, encoding, or resolution. No explicit RTP mechanism is used to establish the relationship between the original media sender and the version the MCU sends. In other words, the outgoing sessions typically use a different SSRC, and may well use a different payload type (PT), even if this different PT happens to be mapped to the same media type. This is a result of the individually negotiated session for each participant.

In case b), the MCU is the content source as it mixes the content and then encodes it for transmission to a participant. According to RTP [RFC3550], the SSRC of the contributors are to be signalled using the CSRC/CC mechanism. In practice, today, most deployed MCUs do not implement this feature. Instead, the identification of the

participants whose content is included in the Mixer's output is not indicated through any explicit RTP mechanism. That is, most deployed MCUs set the CSRC Count (CC) field in the RTP header to zero, thereby indicating no available CSRC information, even if they could identify the content sources as suggested in RTP.

The main feature that sets this topology apart from what RFC 3550 describes is the breaking of the common RTP session across the centralized device, such as the MCU. This results in the loss of explicit RTP-level indication of all participants. If one were using the mechanisms available in RTP and RTCP to signal this explicitly, the topology would follow the approach of an RTP Mixer. The lack of explicit indication has at least the following potential problems:

1. Loop detection cannot be performed on the RTP level. When carelessly connecting two misconfigured MCUs, a loop could be generated.

2. There is no information about active media senders available in the RTP packet. As this information is missing, receivers cannot use it. It also deprives the client of information related to currently active senders in a machine-usable way, thus preventing clients from indicating currently active speakers in user interfaces, etc.

Note that deployed MCUs (and endpoints) rely on signalling layer mechanisms for the identification of the contributing sources, for example, a SIP conferencing package [RFC4575]. This alleviates, to some extent, the aforementioned issues resulting from ignoring RTP's CSRC mechanism.

As a result of the shortcomings of this topology, it is RECOMMENDED to instead implement the Mixer concept as specified by RFC 3550.

3.9. De-composite Endpoint

The implementation of an application may desire to send a subset of the application's data to each of multiple devices, each with their own network address. A very basic use case for this would be to separate audio and video processing for a particular endpoint, like a conference room, into one device handling the audio and another handling the video, being interconnected by some control functions allowing them to behave as a single endpoint in all aspects except for transport Figure 15.

Which decomposition that is possible is highly dependent on the RTP session usage. It is not really feasible to decomposed one logical end-point into two different transport node in one RTP session. From

a third party monitor of such an attempt the two entities would look like two different end-points with a CNAME collision. This put a requirement on that the only type of de-composited endpoint that RTP really supports is one where the different parts have separate RTP sessions to send and/or receive media streams intended for them.

```
             +---------------------+
             | Endpoint A          |
             | Local Area Network  |
             |      +------------+ |
             |   +->| Audio      |<+-RTP---\
             |   |  +------------+ |        \    +------+
             |   |  +------------+ |         +-->|      |
             |   +->| Video      |<+-RTP-------->|  B   |
             |   |  +------------+ |         +-->|      |
             |   |  +------------+ |        /    +------+
             |   +->| Control    |<+-SIP---/
             |      +------------+ |
             +---------------------+
```

Figure 15: De-composite End-Point

In the above usage, let us assume that the RTP sessions are different for audio and video. The audio and video parts will use a common CNAME and also have a common clock to ensure that synchronisation and clock drift handling works despite the decomposition. Also the RTCP handling works correctly as long as only one part of the de-composite is part of each RTP session. That way any differences in the path between A's audio entity and B and A's video and B are related to different SSRCs in different RTP sessions.

The requirements that can derived from the above usage is that the transport flows for each RTP session might be under common control but still go to what looks like different endpoints based on addresses and ports. This geometry cannot be accomplished using one RTP session, so in this case, multiple RTP sessions are needed.

3.10. Non-Symmetric Mixer/Translators

Shortcut name: Topo-Asymmetric

It is theoretically possible to construct an MCU that is a Mixer in one direction and a Translator in another. The main reason to consider this would be to allow topologies similar to Figure 8, where the Mixer does not need to mix in the direction from B or D towards the multicast domains with A and C. Instead, the media streams from B and D are forwarded without changes. Avoiding this mixing would save media processing resources that perform the mixing in cases where it

isn't needed. However, there would still be a need to mix B's stream towards D. Only in the direction B -> multicast domain or D -> multicast domain would it be possible to work as a Translator. In all other directions, it would function as a Mixer.

The Mixer/Translator would still need to process and change the RTCP before forwarding it in the directions of B or D to the multicast domain. One issue is that A and C do not know about the mixed-media stream the Mixer sends to either B or D. Thus, any reports related to these streams must be removed. Also, receiver reports related to A and C's media stream would be missing. To avoid A and C thinking that B and D aren't receiving A and C at all, the Mixer needs to insert its Receiver Reports for the streams from A and C into B and D's Sender Reports. In the opposite direction, the Receiver Reports from A and C about B's and D's stream also need to be aggregated into the Mixer's Receiver Reports sent to B and D. Since B and D only have the Mixer as source for the stream, all RTCP from A and C must be suppressed by the Mixer.

This topology is so problematic and it is so easy to get the RTCP processing wrong, that it is NOT RECOMMENDED to implement this topology.

3.11. Combining Topologies

Topologies can be combined and linked to each other using Mixers or Translators. However, care must be taken in handling the SSRC/CSRC space. A Mixer will not forward RTCP from sources in other domains, but will instead generate its own RTCP packets for each domain it mixes into, including the necessary Source Description (SDES) information for both the CSRCs and the SSRCs. Thus, in a mixed domain, the only SSRCs seen will be the ones present in the domain, while there can be CSRCs from all the domains connected together with a combination of Mixers and Translators. The combined SSRC and CSRC space is common over any Translator or Mixer. This is important to facilitate loop detection, something that is likely to be even more important in combined topologies due to the mixed behavior between the domains. Any hybrid, like the Topo-Video-switch-MCU or Topo-Asymmetric, requires considerable thought on how RTCP is dealt with.

4. Comparing Topologies

The topologies discussed in Section 3 have different properties. This section first lists these properties and then maps the different topologies to them. Please note that even if a certain property is supported within a particular topology concept, the necessary functionality may, in many cases, be optional to implement.

Note: This section has not yet been updated with the new additions of topologies.

4.1. Topology Properties

4.1.1. All to All Media Transmission

Multicast, at least Any Source Multicast (ASM), provides the functionality that everyone may send to, or receive from, everyone else within the session. MCUs, Mixers, and Translators may all provide that functionality at least on some basic level. However, there are some differences in which type of reachability they provide.

The transport Translator function called "relay", in Section 3.4, is the one that provides the emulation of ASM that is closest to true IP-multicast-based, all to all transmission. Media Translators, Mixers, and the MCU variants do not provide a fully meshed forwarding on the transport level; instead, they only allow limited forwarding of content from the other session participants.

The "all to all media transmission" requires that any media transmitting entity considers the path to the least capable receiver. Otherwise, the media transmissions may overload that path. Therefore, a media sender needs to monitor the path from itself to any of the participants, to detect the currently least capable receiver, and adapt its sending rate accordingly. As multiple participants may send simultaneously, the available resources may vary. RTCP's Receiver Reports help performing this monitoring, at least on a medium time scale.

The transmission of RTCP automatically adapts to any changes in the number of participants due to the transmission algorithm, defined in the RTP specification [RFC3550], and the extensions in AVPF [RFC4585] (when applicable). That way, the resources utilized for RTCP stay within the bounds configured for the session.

4.1.2. Transport or Media Interoperability

Translators, Mixers, and RTCP-terminating MCU all allow changing the media encoding or the transport to other properties of the other domain, thereby providing extended interoperability in cases where the participants lack a common set of media codecs and/or transport protocols.

4.1.3. Per Domain Bit-Rate Adaptation

Participants are most likely to be connected to each other with a heterogeneous set of paths. This makes congestion control in a Point to Multipoint set problematic. For the ASM and "relay" scenario, each individual sender has to adapt to the receiver with the least capable path. This is no longer necessary when Media Translators, Mixers, or MCUs are involved, as each participant only needs to adapt to the slowest path within its own domain. The Translator, Mixer, or MCU topologies all require their respective outgoing streams to adjust the bit-rate, packet-rate, etc., to adapt to the least capable path in each of the other domains. That way one can avoid lowering the quality to the least-capable participant in all the domains at the cost (complexity, delay, equipment) of the Mixer or Translator.

4.1.4. Aggregation of Media

In the all to all media property mentioned above and provided by ASM, all simultaneous media transmissions share the available bit-rate. For participants with limited reception capabilities, this may result in a situation where even a minimal acceptable media quality cannot be accomplished. This is the result of multiple media streams needing to share the available resources. The solution to this problem is to provide for a Mixer or MCU to aggregate the multiple streams into a single one. This aggregation can be performed according to different methods. Mixing or selection are two common methods.

4.1.5. View of All Session Participants

The RTP protocol includes functionality to identify the session participants through the use of the SSRC and CSRC fields. In addition, it is capable of carrying some further identity information about these participants using the RTCP Source Descriptors (SDES). To maintain this functionality, it is necessary that RTCP is handled correctly in domain bridging function. This is specified for Translators and Mixers. The MCU described in Section 3.7 does not entirely fulfill this. The one described in Section 3.8 does not support this at all.

4.1.6. Loop Detection

In complex topologies with multiple interconnected domains, it is possible to form media loops. RTP and RTCP support detecting such loops, as long as the SSRC and CSRC identities are correctly set in forwarded packets. It is likely that loop detection works for the MCU, described in Section 3.7, at least as long as it forwards the RTCP between the participants. However, the MCU in Section 3.8 will

definitely break the loop detection mechanism.

4.2. Comparison of Topologies

The table below attempts to summarize the properties of the different topologies. The legend to the topology abbreviations are: Topo-Point-to-Point (PtP), Topo-Multicast (Multic), Topo-Trns-Translator (TTrn), Topo-Media-Translator (including Transport Translator) (MTrn), Topo-Mixer (Mixer), Topo-Asymmetric (ASY), Topo-Video-switch-MCU (MCUs), and Topo-RTCP-terminating-MCU (MCUt). In the table below, Y indicates Yes or full support, N indicates No support, (Y) indicates partial support, and N/A indicates not applicable.

Property	PtP	Multic	TTrn	MTrn	Mixer	ASY	MCUs	MCUt
All to All media	N	Y	Y	Y	(Y)	(Y)	(Y)	(Y)
Interoperability	N/A	N	Y	Y	Y	Y	N	Y
Per Domain Adaptation	N/A	N	N	Y	Y	Y	N	Y
Aggregation of media	N	N	N	N	Y	(Y)	Y	Y
Full Session View	Y	Y	Y	Y	Y	Y	(Y)	N
Loop Detection	Y	Y	Y	Y	Y	Y	(Y)	N

Please note that the Media Translator also includes the transport Translator functionality.

5. Security Considerations

The use of Mixers and Translators has impact on security and the security functions used. The primary issue is that both Mixers and Translators modify packets, thus preventing the use of integrity and source authentication, unless they are trusted devices that take part in the security context, e.g., the device can send Secure Realtime Transport Protocol (SRTP) and Secure Realtime Transport Control Protocol (SRTCP) [RFC3711] packets to session endpoints. If encryption is employed, the media Translator and Mixer need to be able to decrypt the media to perform its function. A transport Translator may be used without access to the encrypted payload in cases where it translates parts that are not included in the encryption and integrity protection, for example, IP address and UDP port numbers in a media stream using SRTP [RFC3711]. However, in general, the Translator or Mixer needs to be part of the signalling context and get the necessary security associations (e.g., SRTP crypto contexts) established with its RTP session participants.

Including the Mixer and Translator in the security context allows the entity, if subverted or misbehaving, to perform a number of very serious attacks as it has full access. It can perform all the

attacks possible (see RFC 3550 and any applicable profiles) as if the media session were not protected at all, while giving the impression to the session participants that they are protected.

Transport Translators have no interactions with cryptography that works above the transport layer, such as SRTP, since that sort of Translator leaves the RTP header and payload unaltered. Media Translators, on the other hand, have strong interactions with cryptography, since they alter the RTP payload. A media Translator in a session that uses cryptographic protection needs to perform cryptographic processing to both inbound and outbound packets.

A media Translator may need to use different cryptographic keys for the inbound and outbound processing. For SRTP, different keys are required, because an RFC 3550 media Translator leaves the SSRC unchanged during its packet processing, and SRTP key sharing is only allowed when distinct SSRCs can be used to protect distinct packet streams.

When the media Translator uses different keys to process inbound and outbound packets, each session participant needs to be provided with the appropriate key, depending on whether they are listening to the Translator or the original source. (Note that there is an architectural difference between RTP media translation, in which participants can rely on the RTP Payload Type field of a packet to determine appropriate processing, and cryptographically protected media translation, in which participants must use information that is not carried in the packet.)

When using security mechanisms with Translators and Mixers, it is possible that the Translator or Mixer could create different security associations for the different domains they are working in. Doing so has some implications:

First, it might weaken security if the Mixer/Translator accepts a weaker algorithm or key in one domain than in another. Therefore, care should be taken that appropriately strong security parameters are negotiated in all domains. In many cases, "appropriate" translates to "similar" strength. If a key management system does allow the negotiation of security parameters resulting in a different strength of the security, then this system SHOULD notify the participants in the other domains about this.

Second, the number of crypto contexts (keys and security related state) needed (for example, in SRTP [RFC3711]) may vary between Mixers and Translators. A Mixer normally needs to represent only a single SSRC per domain and therefore needs to create only one security association (SRTP crypto context) per domain. In contrast,

a Translator needs one security association per participant it translates towards, in the opposite domain. Considering Figure 5, the Translator needs two security associations towards the multicast domain, one for B and one for D. It may be forced to maintain a set of totally independent security associations between itself and B and D respectively, so as to avoid two-time pad occurrences. These contexts must also be capable of handling all the sources present in the other domains. Hence, using completely independent security associations (for certain keying mechanisms) may force a Translator to handle N*DM keys and related state; where N is the total number of SSRCs used over all domains and DM is the total number of domains.

There exist a number of different mechanisms to provide keys to the different participants. One example is the choice between group keys and unique keys per SSRC. The appropriate keying model is impacted by the topologies one intends to use. The final security properties are dependent on both the topologies in use and the keying mechanisms' properties, and need to be considered by the application. Exactly which mechanisms are used is outside of the scope of this document. Please review RTP Security Options [I-D.ietf-avtcore-rtp-security-options] to get a better understanding of most of the available options.

6. IANA Considerations

This document makes no request of IANA.

Note to RFC Editor: this section may be removed on publication as an RFC.

7. Acknowledgements

The authors would like to thank Bo Burman, Umesh Chandra, Roni Even, Keith Lantz, Ladan Gharai, Geoff Hunt, and Mark Baugher for their help in reviewing this document.

8. References

8.1. Normative References

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

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

Applications", STD 64, RFC 3550, July 2003.

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

[RFC4575] Rosenberg, J., Schulzrinne, H., and O. Levin, "A Session Initiation Protocol (SIP) Event Package for Conference State", RFC 4575, August 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.

8.2. Informative References

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

[I-D.ietf-avtcore-rtp-security-options]
Westerlund, M. and C. Perkins, "Options for Securing RTP Sessions", draft-ietf-avtcore-rtp-security-options-00 (work in progress), July 2012.

[RFC4607] Holbrook, H. and B. Cain, "Source-Specific Multicast for IP", RFC 4607, August 2006.

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

[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.

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

[RFC6465] Ivov, E., Marocco, E., and J. Lennox, "A Real-time Transport Protocol (RTP) Header Extension for Mixer-to-Client Audio Level Indication", RFC 6465, December 2011.

Authors' Addresses

Magnus Westerlund
Ericsson
Farogatan 6
SE-164 80 Kista
Sweden

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

Stephan Wenger
Vidyo
433 Hackensack Ave
Hackensack, NJ 07601
USA

Email: stewe@stewe.org