Network Working Group M.
Westerlund
Internet-Draft
Ericsson
Obsoletes: 5117 (if approved) S.
Wenger
Intended status: Informational
Vidyo
Expires: April 25, 2014 October 22,
2013

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

Abstract

This document discusses point to point and 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 is intended to replace RFC 5117.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

Real-time Transport Protocol (RTP) [RFC3550] topologies describe methods for interconnecting RTP entities and their processing behavior of RTP and RTCP. This document tries to address past and existing confusion, especially with respect to terms not defined in RTP but in common use in the conversational communication industry, such as the Multipoint Control Unit or MCU.

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. In practice, however, most multipoint conferences operate utilizing centralized units referred to as MCUs. MCUs may implement Mixer or Translator functionality (in RTP [RFC3550] terminology), and signalling support. They may also contain additional application layer 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.

This document also codifies new multipoint architectures that have recently been introduced and which were not anticipated in RFC 5117. These architectures use scalable video coding and simulcasting, and their associated centralized units are referred to as Selective Forwarding Units (SFU). This codification provides a common information basis for future discussion and specification work.

The document's attempt to clarify and explain sections of the Real-time Transport Protocol (RTP) spec [RFC3550] is informal. It is not intended to update or change what is normatively specified within RFC 3550.

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

Middlebox: A device that is on the Path that media travel between two Endpoints

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

SFU: Selective Forwarding Unit

SSM: Source-Specific Multicast

SSRC: Synchronization Source

3. Topologies

This subsection defines several topologies that are relevant for codec control but also RTP usage in other contexts. The section starts with point to point cases, with or without middleboxes. Then follows a number of different methods for establishing point to multipoint communication. These are structured around the most fundamental enabler, i.e., multicast, a mesh of connections, translators, mixers and finally MCUs and SFUs. The section ends by discussing de-composited endpoints, asymmetric middlebox behaviors and combining topologies.

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 discuss the handling of RTCP feedback messages as defined in [RFC4585] and [RFC5104].

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 stemming 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, subject only to system level limitations like the number range of ports.

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. For RTP sessions which use multiple SSRC per endpoint it can be relevant to implement support for cross-reporting suppression as defined in "Sending Multiple Media Streams in a Single RTP Session" [I-D.ietf-avtcore-rtp-multi-stream].

3.2. Point to Point via Middlebox

This section discusses cases where two endpoints communicate but have one or more middleboxes involved in the RTP session.

3.2.1. Translators

Shortcut name: Topo-PtP-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 keeps the SSRC for a stream across the translation, whereas a Mixer can select a single media stream, or send out multiple mixed media streams, but always under its own SSRC, possibly using the CSRC field to indicate the source(s) of the content. Mixers are more common in point to multipoint cases than in PtP. The reason is that in PtP use cases the primary focus is interoperability, such as transcoding to a codec the receiver supports, which can be done by a media translator.

As specified in Section 7.1 of [RFC3550], the SSRC space is common for all participants in the RTP session, independent of on which side of the Translator the session resides. Therefore, it is the responsibility of the participants to run SSRC collision detection, and the SSRC is thus a field the Translator cannot change. Any SDES information associated with a SSRC or CSRC also needs to be forwarded between the domains for any SSRC/CSRC used in the different domains.

A Translator commonly does not use an SSRC of its own, and is not visible as an active participant in the session. One reason to have its own SSRC 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 in a translator configured to detect packet loss of important video packets and wants to trigger repair by the media sender, by sending feedback messages. While such feedback could use the SSRC of the target for the translator, this in turn would require translation of the targets RTCP reports to make them consistent. It may be simpler to expose an additional SSRC in the session. The only concern is endpoints failing to support the full RTP specification, thus having issues with multiple SSRCs reporting on the RTP streams sent by that endpoint.

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.

3.2.1.1. Transport Relay/Anchoring

There exist a number of different types of middleboxes that might be inserted between two RTP endpoints on the transport level, e.g., to perform changes on the IP/UDP headers, and are, therefore, basic transport translators. These middleboxes come in many variations including NAT [RFC3022] 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 for audit by employing relaying, or preserving privacy by hiding each peer's transport addresses to the other party. Other protocols or functionalities that provide this behavior are TURN [RFC5766] servers, Session Border Gateways and Media Processing Nodes with media anchoring functionalities.

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

Figure 2: Point to Point with Translator

A common element in these functions is that they are normally transparent at the RTP level, i.e., they perform no changes on any RTP or RTCP packet fields and only affect the lower layers. They may affect, however, the path the RTP and RTCP packets are routed between the endpoints in the RTP session, and thereby only indirectly affect the RTP session. For this reason, one could believe that transport translator-type middleboxes do not need to be included in this document. This topology, however, can raise additional requirements in the RTP implementation and its interactions with the signalling solution. Both in signalling and in certain RTCP fields, network addresses other than those of the relay can occur since B has a different network address than the relay (T). Implementations that can not support this will also not work correctly when endpoints are subject to NAT.

The transport relay implementation also have some considerations, where security considerations are an important aspect. Source address filtering of incoming packets are usually important in relays, to prevent attackers to inject traffic into a session, which one peer will think comes from the other peer.

3.2.1.2. 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 such unicast to multicast) 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, an important consideration is how to handle different sources and their identities. This problem space is not discussed henceforth.

The most basic transport translators that operate below the RTP level were already discussed in Section 3.2.1.1.

3.2.1.3. Media Translator

Media Translators (Topo-Media-Translator) 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 decoding and re-encoding (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 in the media encoding.

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

When media translation occurs, the Translator's task regarding handling of RTCP traffic becomes substantially more complex. In this case, the Translator needs to rewrite B's RTCP Receiver Report before forwarding them to A. 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 A sends, 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 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.

A variant of translator behaviour worth pointing out is the one depicted in Figure 3 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. In this case, T needs 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 3: When De-composition is a Translator

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

3.2.2. Back to Back RTP sessions

There exist middleboxes that interconnect two endpoints through themselves, but not by being part of a common RTP session. They establish instead two different RTP sessions, one between A and the middlebox and another between the middlebox and B.

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

Figure 4: When De-composition is a Translator

The middlebox acts as an application-level gateway and bridges the two RTP sessions. This bridging can be as basic as forwarding the RTP payloads between the sessions, or more complex including media transcoding. The difference with the single RTP session context is the handling of the SSRCs and the other session-related identifiers, such as CNAMEs. With two different RTP sessions these can be freely changed and it becomes the middlebox's task to maintain the correct relations.

The signalling or other above-RTP level functionalities referencing RTP media streams may be what is most impacted by using two RTP sessions and changing identifiers. The structure with two RTP sessions also puts a congestion control requirement on the middlebox, because it becomes fully responsible for the media stream it sources into each of the sessions.

Adherence to congestion control can be solved locally or by bridging also statistics from the receiving endpoint. From an implementation point, however, this requires dealing with a number of inconsistencies. First, packet loss must be detected for an RTP flow sent from A to the middlebox, and that loss must be reported through a skipped sequence number in the flow from the middlebox to B. This coupling and the resulting inconsistencies is conceptually easier to handle when considering the two flows as belonging to a single RTP session.

3.3. Point to Multipoint Using Multicast

Multicast is an IP layer functionality that is available in some networks. Two main flavors can be distinguished: Any Source Multicast (ASM) [RFC1112] where any multicast group participant can send to the group address and expect the packet to reach all group participants; and Source Specific Multicast (SSM) [RFC3569], where only a particular IP host sends to the multicast group. Both these models are discussed below in their respective sections.

3.3.1. Any Source Multicast (ASM)

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

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

Figure 5: 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 scenario can occur, for example, where a multimedia session is distributed using two or more multicast groups and a participant is subscribed only to a subset of these sessions.

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 in the range of billions). The above can be realized using Any Source Multicast (ASM).

For feedback usage, it is useful to define a "small multicast group" as a group where the number of participants is so low (and other factors such as the connectivity is so good) that it allows the

participants to use early or immediate feedback, as defined in AVPF [RFC4585]. Even when the environment would allow for the use of a small multicast group, some applications may still want to use the more limited options for RTCP feedback available to large multicast groups, for example when there is a likelihood that the threshold of the small multicast group (in terms of participants) may be exceeded during the lifetime of a session.

RTCP feedback messages in multicast reach, like media data, every subscriber (subject to packet losses and multicast group subscription). Therefore, the feedback suppression mechanism discussed in [RFC4585] is typically required. Each individual node needs to process every feedback message it receives, not to determine if it is affected or if the feedback message applies only to some other participant, but also to derive timing restrictions for the sending of its own feedback messages, if any.

3.3.2. Source Specific Multicast (SSM)

In Any Source Multicast, any of the participants can send to all the other participants, by sending a packet to the multicast group. In contrast, Source Specific Multicast [RFC3569][RFC4607] refers to scenarios 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 6: Point to Multipoint using Source Specific Multicast

In the SSM topology (Figure 6) a number of RTP sources (1 to M) are allowed to send media to the SSM group. These sources send media to a dedicated distribution source, which forwards the media streams to the multicast group on behalf of the original senders. The media streams reach the Receivers (R(1) to R(n)). The Receivers' RTCP messages cannot be sent to the multicast group, as the SSM multicast group by definition has only a single source. To support RTCP, an RTP extension for SSM [RFC5760] was defined. It uses unicast transmission to send RTCP from each of the receivers to one or more Feedback Targets (FT). The feedback targets relay the RTCP unmodified, or provide a summary of the participants RTCP reports towards the whole group by forwarding the RTCP traffic to the distribution source. Figure 6 only shows a single feedback target integrated in the distribution source, but for scalability the FT can be many and have responsibility for sub-groups of the receivers. For summary reports, however, there must be a single feedback aggregating all the summaries to a common message to the whole receiver group.

The RTP extension for SSM specifies how feedback (both reception information and specific feedback events) are handled. The more general problems associated with the use of multicast, where everyone receives what the distribution source sends needs to be accounted for.

Aforementioned situation results in common behavior for RTP multicast:

1. Multicast applications often use a group of RTP sessions, not one. Each endpoint needs to be a member of most or all of these 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 need to have the same RTP and payload type configuration. Otherwise, A could, for example, be using payload type 97 to identify the video codec H.264, while B would identify it as MPEG-2.

Security solutions for this type of group communications are also challenging. First, 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].

3.3.3. SSM with Local Unicast Resources

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

```
 -----------                                         --------------
|           |------------------------------------>|               |
|           |.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.->|               |
|           |                                     |               |
| Multicast |          ----------------           |               |
|  Source   |         | Retransmission |          |               |
|           |-------->|  Server  (RS)  |          |               |
|           |.-.-.-.->|                |          |               |
|           |         |  ------------  |          |               |
 -----------          | |  Feedback  | |<.=.=.=.=.|               |
                      | | Target (FT)| |<~~~~~~~~~| RTP Receiver  |
PRIMARY MULTICAST     |  ------------  |          |   (RTP_Rx)    |
RTP SESSION with      |                |          |               |
UNICAST FEEDBACK      |                |          |               |
```

```
                         |                |          |              |
   - - - - - - - - - - - |- - - - - - - - |- - - - - |- - - - - - - |-
-
                         |                |          |              |
   UNICAST BURST         |  ------------  |          |              |
   (or RETRANSMISSION)   | |   Burst/   | |<~~~~~~~~>|              |
   RTP SESSION           | |  Retrans.  | |.........>|              |
                         | |Source (BRS)| |<.=.=.=.=>|              |
                         |  ------------  |          |              |
                         |                |          |              |
                          ---------------             -------------

      -------> Multicast RTP Flow
      .-.-.-.> Multicast RTCP Flow
      .=.=.=.> Unicast RTCP Reports
      ~~~~~~~> Unicast RTCP Feedback Messages
      .......> Unicast RTP Flow
```

Figure 7

The Rapid acquisition extension allows an endpoint joining an SSM multicast session to request media starting with the last sync-point (from where media can be decoded without requiring context established by the decoding of prior packets) to be sent at high speed until such time where, after decoding of these burst-delivered media packets, the correct media timing is established, i.e. media packets are received within adequate buffer intervals for this application. This is accomplished by first establishing a unicast PtP RTP session between the Burst/Retransmission Source (BRS, Figure 7) and the RTP Receiver. The unicast session is used to transmit cached packets from the multicast group at higher then normal speed in order to synchronize the receiver to the ongoing multicast packet flow. Once the RTP receiver and its decoder have caught up with the multicast session's current delivery, the receiver switches over to receiving directly from the multicast group. The (still existing) PtP RTP session is, in many deployed applications, be used as a repair channel, i.e., for RTP Retransmission traffic of those packets that were not received from the multicast group.

3.4. Point to Multipoint Using Mesh

Shortcut name: Topo-Mesh

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

```
                                  v   v
                                  +---+
                                  | C |
                                  +---+
```

Figure 8: 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 joint three endpoint 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 does not 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.

```
+-A----------------------+                +-B-------------+
|+---+                   |                |               |
||CAM|                   |                |               |
|+---+     +-UDP1------|                |-UDP1------+ |
|  |       | +-RTP1----|                |-RTP1----+ | |
|  V       | | +-Video-|                |-Video-+ | | |
|+----+    | | |       |<-----------------|BV1    | | | |
||ENC |----+-+-+--->AV1|----------------->|       | | | |
|+----+    | | +-------|                |-------+ | | |
|  |       | +---------|                |---------+ | |
|  |       +-----------|                |-----------+ |
|  |       ------------|                |------------ |
|  |                   |                |-------------+
|  |                   |
|  |                   |                +-C-------------+
|  |                   |                |               |
|  |       +-UDP2------|                |-UDP2------+ |
|  |       | +-RTP1----|                |-RTP1----+ | |
|  |       | | +-Video-|                |-Video-+ | | |
|  +-------+-+-+--->AV1|----------------->|       | | | |
|          | | |       |<-----------------|CV1    | | | |
|          | | +-------|                |-------+ | | |
|          | +---------|                |---------+ | |
|          +-----------|                |-----------+ |
|          ------------|                |------------ |
+----------------------+                +-------------+
```

Figure 9: An Multi-unicast Mesh with a joint RTP session

A joint RTP session from A's perspective for the Mesh depicted in Figure 8 with a joint RTP session have multiple transport flows, here enumerated as UDP1 and UDP2. However, there is only one RTP session (RTP1). The media source (CAM) is encoded and transmitted over the SSRC (AV1) across both transport layers. However, as this is a joint RTP session, the two streams must be the same. Thus, an congestion control adaptation needed for the paths A to B and A to C needs to use the most restricting path's properties.

An alternative structure for establishing the above topology is to use independent RTP sessions between each pair of peers, i.e., three different RTP sessions. In some scenarios, the same RTP media stream may be sent from transmitting endpoint, however it also supports local adaptation taking place in one or more of the RTP media streams, rendering them non-identical.

```
   +-A----------------------+              +-B-----------+
   |+---+                   |              |             |
   ||MIC|       +-UDP1------|              |-UDP1------+ |
   |+---+       | +-RTP1----|              |-RTP1----+ | |
   | |  +----+  | | +-Audio-|              |-Audio-+ | | |
   | +->|ENC1|--+-+-+--->AA1|------------->|       | | | |
   | |  +----+  | | |       |<-------------|BA1    | | | |
   | |          | | +-------|              |-------+ | | |
   | |          | +---------|              |---------+ | |
   | |          +-----------|              |-----------+ |
   | |          ------------|              |-------------|
   | |                      |              |-------------+
   | |                      |
   | |                      |              +-C-----------+
   | |                      |              |             |
   | |          +-UDP2------|              |-UDP2------+ |
   | |          | +-RTP2----|              |-RTP2----+ | |
   | |  +----+  | | +-Audio-|              |-Audio-+ | | |
   | +->|ENC2|--+-+-+--->AA2|------------->|       | | | |
   |    +----+  | | |       |<-------------|CA1    | | | |
   |            | | +-------|              |-------+ | | |
   |            | +---------|              |---------+ | |
   |            +-----------|              |-----------+ |
   +------------------------+              +-------------+
```

Figure 10: An Multi-unicast Mesh with independent RTP session

Lets review the topology when independent RTP sessions are used, from A's perspective in Figure 8 by considering both how the media is a handled and the RTP sessions that are set-up in Figure 10. A's microphone is captured and the digital audio can then be feed into two different encoder instances, as each beeing associated with two

independent RTP sessions (RTP1 and RTP2). The SSRCs (AA1 and AA2) in each RTP session will be completely independent and the media bit-rate produced by the encoders can also be tuned differently to address any congestion control requirements differing for the paths A to B compared to A to C.

From a topologies viewpoint, an important difference exists in the behavior around RTCP. First, when a single RTP session spans all three endpoints and their connecting flows, an common RTCP bandwidth is calculated and used for this single joint session. In contrast, when there are multiple independent RTP sessions, each RTP session has its local RTCP bandwidth allocation.

Further, when multiple sessions are used, endpoints not directly involved in a session, do not have any awareness of the conditions in those sessions. For example, in the case of the three endpoint configuration in Figure 8, endpoint A has no awareness of the conditions occurring in the session between endpoints B and C (whereas, if a single RTP session were used, it would have such awareness).

Loop detection is also affected. With independent RTP sessions, the SSRC/CSRC cannot be used to determine when an endpoint receives its own media stream, or a mixed media stream including its own media stream (a condition known as a loop). The identification of loops and, in most cases, their avoidance, has to be achieved by other means, for example through signaling or the use of an RTP external name space binding SSRC/CSRC among any communicating RTP sessions in the mesh.

3.5. Point to Multipoint Using the RFC 3550 Translator

This section discusses some additional usages related to point to multipoint of Translators compared to the point to point only cases in Section 3.2.1.

3.5.1. Relay - Transport Translator

Shortcut name: Topo-PtM-Trn-Translator

This section discusses Transport Translator only usages to enable multipoint sessions.

```
                   +-----+
          +---+     /       \     +------------+      +---+
          | A |<---/         \    |            |<---->| B |
          +---+   /   Multi-  \   |            |      +---+
                 +    cast     +->| Translator |
```

```
             +---+   \  Network  /   |            |      +---+
             | C |<---\         /    |            |<---->| D |
             +---+     \       /     +------------+      +---+
                        +------+
```

Figure 11: Point to Multipoint Using Multicast

Figure 11 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.

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

Figure 12: RTP Translator (Relay) with Only Unicast Paths

Another Translator scenario is depicted in Figure 12. The Translator in this case 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 set of considerations to those described in the ASM RTP topology. It also puts some additional signalling requirements onto the session establishment; for example, a common configuration of RTP payload types is required.

Transport translators and relays should always consider doing source address filtering, to prevent attackers to inject traffic using the listening ports on the translator. The translator can however go one step further, and especially if explicit SSRC signalling is used, prevent other session participants to send SSRCs that are used by other participants in the session. This can improve the security properties of the session, despite the use of group keys that on cryptographic level allows anyone to impersonate another in the same RTP session.

A Translator that doesn't change the RTP/RTCP packets content can be operated without the requiring the translator to have access to the security contexts used to protect the RTP/RTCP traffic between the participants.

3.5.2. Media Translator

In the context of multipoint communications a Media Translator is not providing new mechanisms to establish a multipoint session. It is more of an enabler, or facilitator, that ensures one or some sub-set of session participants can participate in the session.

If B in Figure 11 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. This transcoding can help the other participants in the Multicast part of the session, by not requiring the quality transmitted by A to be lowered to the bitrates that B is actually capable of receiving.

3.6. Point to Multipoint Using the RFC 3550 Mixer Model

Shortcut name: Topo-Mixer

A Mixer is a middlebox that aggregates multiple RTP streams that are part of a session by generating a new RTP stream and, in most cases, by manipulating the media data. 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 13: 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 can be used to indicate the contributors 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 different conference participants 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. This also implies that any SDES information normally needs to be forwarded across the mixer.

The Mixer is responsible for generating RTCP packets in accordance with its role. It is a receiver and should therefore send receiver reports for the media streams it receives. In its role as a media sender, it should also generate sender reports for those media streams it sends. 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 13 is involved in three domains that need to be separated: the any source multicast network (including participants A and C), participant B, and participant D. Assuming all four participants in the conference are interested in receiving content from each other participant, the Mixer produces different mixed streams for 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 where it 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 3.11 for more discussion on this topic.
The mixing operation, however, 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 by the Mixer may result in the generation and transmission of RTCP feedback messages by the Mixer to the

participants in the other domain(s). 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 13 (to the left of the Mixer) with individual unicast paths as depicted in Figure 14, the Mixer model is very similar to the one discussed in Section 3.9 below. Please see the discussion in Section 3.9 about the differences between these two models.

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

Figure 14: RTP Mixer with Only Unicast Paths

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

3.6.1. Media Mixing

The media mixing mixer is likely the one that most think of when they hear the term "mixer". Its basic mode of operation is that it receives media streams from several participants and selects the stream(s) to be included in a media-domain mix. The selection can be through static configuration or by dynamic, content dependent means such as voice activation. The mixer then creates 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 a mixture of all the input streams; this needs minimal signalling to be successfully set up. Audio mixing is relatively straightforward and commonly possible for a reasonable number of participants. Assume, for example, that one wants to mix N streams from different participants. The mixer needs to decode those N streams, typically into the sample domain, and then produce N or N+1 mixes. Different mixes are needed so that each contributing source gets a mix of all other sources except its own, 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 packetized and sent out. In many cases, audio level normalization is also required before the actual mixing process.

In video, the term "mixing" has a different interpretation than audio. It is commonly used to refer to the process of spatially combining contributed video streams is known as "tiling". The reconstructed, appropriately scaled down videos can be spatially arranged in a set of tiles, each tile containing the video from a participant. Tiles can be of different sizes, so that, for example, a particularly important participant, or the loudest speaker, is being shown on in larger tile than other participants. A self-view picture can be included in the tiling, which can either be locally produced or be a feedback from a received and reconstructed video image. Such remote loopback allows for confidence monitoring, i.e., it enables the participant to see himself/herself just as other participants see him/her. The tiling normally operates on reconstructed video in the sample domain. The tiled image is encoded, packetized, and sent by the mixer. It is possible that a middlebox with media mixing duties contains only a single mixer of the aforementioned type, in which case all participants necessarily see the same tiled video, even if it is being sent over different RTP streams. More common, however, are mixing arrangement where an individual mixer is available for each outgoing port of the middlebox, allowing individual compositions for each participant (a feature referred to as personalized layout).

One problem with media mixing is that it consumes both large amount of media processing (for the actual mixing process in the uncompressed domain) and encoding resources (for the encoding of the mixed signal). Another problem is the quality degradation created by decoding and re-encoding the media that is encapsulated in the RTP media stream, which is the result of the lossy nature of most commonly used media codecs. A third problem is the latency introduced by the media mixing, which can be substantial and annoyingly noticeable in case of video, or in case of audio if that mixed audio is lip-sychronized with high latency video. The advantage of media mixing is that it is straightforward for the clients to handle the single media stream (which includes the mixed aggregate of many sources), as they don't need to handle multiple decodings, local mixing and composition. In fact, mixers were introduced in pre-RTP times so that legacy, single stream receiving endpoints could successfully participate in what a user would recognize as a multiparty video conference.

```
           +-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 15: Session and SSRC details for Media Mixer

From an RTP perspective media mixing can be a very simple process, as can be seen in Figure 15. The mixer presents one SSRC towards the receiving client, e.g., MA1 to Peer A, where the associated stream is the media mix of the other participants. As each peer, in this example, receives a different version of a mix from the mixer, there is no actual relation between the different RTP sessions in terms of actual media or transport level information. There are, however, common relationships between RTP1-RTP3, namely SSRC space and 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, allowing the receiver to identify the contributing sources even if this were not possible through the media itself or through other signaling means.

The CSRC has, in turn, utility in RTP extensions, like the Mixer to Client audio levels RTP header extension [RFC6465]. If the SSRCs from the endpoint to mixer paths are used as CSRCs in another RTP session, then RTP1, RTP2 and RTP3 become one joint session as they have a common SSRC space. At this stage, the mixer also needs to consider which RTCP information it needs to expose in the different paths. In the above scenario, a mixer would normally expose nothing more than the Source Description (SDES) information and RTCP BYE for a CSRC leaving the session. 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.

3.6.2. Media Switching

Media switching mixers are used from limited functionality scenarios where no, or only very limited, concurrent presentation of multiple sources is required by the application to more complex multi-stream usages with receiver mixing or tiling, including combined with simulcast and/or scalability between source and mixer. An RTP Mixer based on media switching avoids the media decoding and encoding operations in the mixer, as it conceptually forwards the encoded media stream as it was being sent to the mixer. It does not avoid, however, the decryption and re-encryption cycle as it rewrites RTP headers. Forwarding media (in contrast to reconstructing-mixing-encoding media) reduces the amount of computational resources needed in the mixer and increases the media quality (both in terms of fidelity and reduced latency).

A media switching mixer maintains a pool of SSRCs representing conceptual or functional streams that the mixer can produce. These streams are created by selecting media from one of the RTP media streams received by the mixer and forwarded to the peer using the mixer's own SSRCs. The mixer can switch between available sources if that is required by the concept for the source, like the currently active speaker. Note that the mixer, in most cases, still needs to perform a certain amount of media processing, as many media formats do not allow to "tune into" the stream at arbitrary points of their bitstream.

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

We now consider the operation of a media switching mixer that supports a video conference with six participants (A-F) where the two most recent speakers in the conference are shown to each participant.

The mixer has thus two SSRCs sending video to each peer, and each peer is capable of locally handling two video streams simultaneously.

```
         +-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 16: Media Switching RTP Mixer

The Media Switching RTP mixer can, similarly to the Media Mixing Mixer, reduce the bit-rate required for media transmission towards the different peers by selecting and forwarding only a sub-set of RTP media streams it receives from the conference participants. In cases the mixer receives simulcast transmissions or a scalable encoding of the media source, the mixer has more degrees of freedom to select streams or sub-sets of stream to forward to a receiver, both based on transport or client restrictions as well as application logic.

To ensure that a media receiver can correctly decode the RTP media stream after a switch, a codec that uses temporal prediction needs to start its decoding from independent refresh points, or similar points in the bitstream. For some codecs, for example frame based speech and audio codecs, this is easily achieved by starting the decoding at

RTP packet boundaries, as each packet boundary provides a refresh point (assuming proper packetization on the encoder side). For other codecs, particularly in video, refresh points are less common in the bitstream or may not be present at all without an explicit request to the respective encoder. The Full Intra Request [RFC5104] RTCP codec control message has been defined for this purpose.

In this type of mixer one could consider to fully terminate the RTP sessions between the different endpoint and mixer paths. The same arguments and considerations as discussed in Section 3.9 need to be taken into consideration and apply here.

3.7. Selective Forwarding Middlebox

Another method for handling media in the RTP mixer is to "project", or make available, all potential RTP sources (SSRCs) into a per-endpoint, independent RTP session. The middlebox can select which of the potential sources that are currently actively transmitting media will be sent to each of the endpoints. This is similar to the media switching Mixer but has some important differences in RTP details.

```
          +-A---------+             +-Middlebox-----------------+
          | +-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---+ |       |     | |
          | | |    FV1|------------>|---------+-+------>|     | |
          | | |       |<------------|AV1 <----+-+-------|     | |
          | | |       | :    :    : |: :  : : : : :  : :|     | |
          | | |       |<------------|EV1 <----+-+-------|     | |
          | | +-------|             |---------+ |       |     | |
          | +---------|             |-----------+       +-----+ |
          +-----------+             +---------------------------+
```

Figure 17: Selective Forwarding Middlebox

In the six participant conference depicted above in (Figure 17) one can see that end-point A is aware of five incoming SSRCs, BV1-FV1. If this middlebox intends to have a similar behavior as in Section 3.6.2 where the mixer provides the end-points with the two latest speaking end-points, then only two out of these five SSRCs need concurrently transmit media to A. As the middlebox selects the source in the different RTP sessions that transmit media to the end-points, each RTP media stream requires some rewriting of RTP header fields when being projected from one session into another. In particular, the sequence number needs to be consecutively incremented based on the packet actually being transmitted in each RTP session. Therefore, the RTP sequence number offset will change each time a source is turned on in a RTP session. The timestamp (possibly offset) stays the same.

As the RTP sessions are independent, the SSRC numbers used can also be handled independently, thereby bypassing the requirement for SSRC collision detection and avoidance. On the other hand, tools such as remapping tables between the RTP sessions are required. For example, the stream that is being sent by endpoint B to the middlebox (BV1) may use an SSRC value of 12345678. When that media stream is sent to endpoint F by the middlebox, it can use any SSRC value, e.g. 87654321. As a result, each endpoint may have a different view of the application usage of a particular SSRC. Any RTP level identity information, such as SDES items also needs to update the SSRC referenced, if the included SDES items are intended to be global. Thus the application must not use SSRC as references to RTP media streams when communicating with other peers directly. This also affects loop detection which will fail to work, as there is no common namespace and identities across the different legs in the communication session on RTP level. Instead this responsibility falls onto higher layers.

The middlebox is also responsible to receive any RTCP codec control requests coming from an end-point, and decide if it can act on the request locally or needs to translate the request into the RTP

session that contains the media source. Both end-points and the middlebox need to implement conference related codec control functionalities to provide a good experience. Commonly used are Full Intra Request to request from the media source to provide switching points between the sources, and Temporary Maximum Media Bit-rate Request (TMMBR) to enable the middlebox to aggregate congestion control responses towards the media source so to enable it to adjust its bit-rate (obviously only in case the limitation is not in the source to middlebox link).

The selective forwarding middlebox has been introduced in recently developed videoconferencing systems in conjunction with, and to capitalize on, scalable video coding as well as simulcasting. An example of scalable video coding is Annex G of H.264, but other codecs, including H.264 AVC and VP8 also exhibit scalability, albeit only in the temporal dimension. In both scalable coding and simulcast cases the video signal is represented by a set of two or more bitstreams, providing a corresponding number of distinct fidelity points. The middlebox selects which parts of a scalable bitstream (or which bitstream, in the case of simulcasting) to forward to each of the receiving endpoints. The decision may be driven by a number of factors, such as available bit rate, desired layout, etc. Contrary to transcoding MCUs, these "Selective Forwarding Units" (SFUs) have extremely low delay, and provide features that are typically associated with high-end systems (personalized layout, error localization) without any signal processing at the middlebox. They are also capable of scaling to a large number of concurrent users, and--due to their very low delay--can also be cascaded.

This version of the middlebox also puts different requirements on the endpoint 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 endpoint either needs to be able to handle as many decoder instances as the middlebox received, or have efficient switching of decoder contexts in a more limited set of actual decoder instances to cope with the switches. The application also gets more responsibility to update how the media provided is to be presented to the user.

Note that this topology 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. It also has mixer aspects, as the streams it provides are not basically translated version, but instead they have conceptual property assigned to them. Thus this topology appears to be some hybrid between the translator and mixer model.

The differences between selective forwarding middlebox and a switching mixer (Section 3.6.2) are minor, and they share most properties. The above requirement on having a large number of decoding instances or requiring efficient switching of decoder contexts, are one point of difference. The other is how the identification is performed, where the Mixer uses CSRC to provide info what is included in a particular RTP packet stream that represent a particular concept. Selective forwarding gets the source information through the SSRC, and instead have to use other mechanism to make clear the streams current purpose.

3.8. Point to Multipoint Using Video Switching MCUs

Shortcut name: Topo-Video-switch-MCU

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

Figure 18: Point to Multipoint Using a Video Switching MCU

This PtM topology was popular in early implementations of multipoint videoconferencing systems due to its simplicity, and the corresponding middlebox design has been known as a "video switching MCU". The more complex RTCP-terminating MCUs, discussed in the next section, became the norm, however, when technology allowed implementations at acceptable costs.

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 3.6) 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.

3.9. Point to Multipoint Using RTCP-Terminating MCU

Shortcut name: Topo-RTCP-terminating-MCU

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

Figure 19: 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 terms of bit-rate, encoding format, 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.

3.10. Split Component Endpoint

Shortcut name: Topo-Split-Endpoint

The implementation of an application may desire to send a subset of the application's data to each of multiple devices, each with its own network address. A very basic use case for this would be to separate audio and video processing for a particular endpoint into different components. For example, in a video conference room system the endpoint could be considered as being composed of one device handling the audio and another handling the video, interconnected by some control functions allowing them to behave as a single endpoint in all aspects except for transport as depicted in Figure 20.

Which decomposition scheme is possible is highly dependent on the RTP session usage. It is not really feasible to decompose one logical end-point into two different transport nodes in one RTP session. A third party monitor would report such an attempt as two entities being two different end-points with a CNAME collision. As a result, a fully RTP conformant de-composited endpoint is one where the different decomposed parts use 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 20: Split Component Endpoint

In the above usage, let us assume that the different RTP sessions are used for audio and video. The audio and video parts, however, use a common CNAME and also have a common clock to ensure that synchronization and clock drift handling works, despite the fact that the components are separated. Also, RTCP handling works correctly as long as only one part of the split endpoint 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 requirement that can be derived from the above usage is that the transport flows for each RTP session might be under common control, but still are addressed to what looks like different endpoints (based on addresses and ports). This connection diagram cannot be accomplished using one RTP session and thus multiple RTP sessions are needed.

3.11. 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 13, 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. Therefore, 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 locally generated reports reflecting the situation 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 for implementation.

3.12. 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 does not forward RTCP from sources in other domains, but instead generates 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. It 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 describes these properties and then analyzes how these properties are supported by the different topologies. Note that, even if a certain property is supported within a particular topology concept, the necessary functionality may 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. Mesh, 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 3.5, as well as the Mesh is the ones 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 resource consumption for performing all to all transmission varies, where the benefit of ASM is that only one copy of each packet traverse a particular link. Using a relay, causes one copy per client to relay path and packet transmitted, however, in most cases the links with the multiple copies will be the ones close to the relay, rather than the clients unless they share LAN segment. The Mesh causes N-1 copies of of each transmitted packet to traverse the first hop link from the client, in a N client mesh. How long the different paths are common, is highly situation dependent.

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, and Mesh with individual RTP sessions, 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, Mesh with common RTP session, 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 3.8 does not entirely fulfill this. The one described in Section 3.9 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 3.8, at least as long as it forwards the RTCP between the participants. However, the MCU in Section 3.9 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 11, 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, Mark Baugher, and Alex Eleftheriadis for their help in reviewing this document.

8. References

8.1. 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.

[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

[I-D.ietf-avtcore-rtp-multi-stream]

Lennox, J., Westerlund, M., Wu, W., and C. Perkins,
"Sending Multiple Media Streams in a Single RTP Session",
draft-ietf-avtcore-rtp-multi-stream-01 (work in
progress),
July 2013.

[I-D.ietf-avtcore-rtp-security-options]
Westerlund, M. and C. Perkins, "Options for Securing RTP
Sessions", draft-ietf-avtcore-rtp-security-options-08
(work in progress), October 2013.

[RFC1112] Deering, S., "Host extensions for IP multicasting", STD
5,
RFC 1112, August 1989.

[RFC3022] Srisuresh, P. and K. Egevang, "Traditional IP Network
Address Translator (Traditional NAT)", RFC 3022, January
2001.

[RFC3569] Bhattacharyya, S., "An Overview of Source-Specific
Multicast (SSM)", RFC 3569, July 2003.

[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.

[RFC5766] Mahy, R., Matthews, P., and J. Rosenberg, "Traversal
Using
Relays around NAT (TURN): Relay Extensions to Session
Traversal Utilities for NAT (STUN)", RFC 5766, April
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