Internet Engineering Task Force (IETF)
Request for Comments: 7667
Obsoletes: 5117
Category: Informational
ISSN: 2070-1721
November 2015

M. Westerlund
Ericsson
S. Wenger
Vidyo

# RTP Topologies

## Abstract

This document discusses point-to-point and multi-endpoint topologies used in environments based on the Real-time Transport Protocol (RTP). 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 replaces RFC 5117.

## Status of This Memo

Table of Contents

# 1. Introduction

Real-time Transport Protocol (RTP) [RFC3550] topologies describe methods for interconnecting RTP entities and their processing behavior for RTP and the RTP Control Protocol (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 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 signaling 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 that were not anticipated in RFC 5117; thus, this document replaces [RFC5117]. These architectures use scalable video coding and simulcasting, and their associated centralized units are referred to as Selective Forwarding Middleboxes (SFMs). This codification provides a common information basis for future discussion and specification work.

The new topologies are Point to Point via Middlebox (Section 3.2), Source-Specific Multicast (Section 3.3.2), SSM with Local Unicast Resources (Section 3.3.3), Point to Multipoint Using Mesh (Section 3.4), Selective Forwarding Middlebox (Section 3.7), and Split Component Terminal (Section 3.10). The Point to Multipoint Using the RFC 3550 Mixer Model (Section 3.6) has been significantly expanded to cover two different versions, namely Media-Mixing Mixer (Section 3.6.1) and Media-Switching Mixer (Section 3.6.2).

The document's attempt to clarify and explain sections of the 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

SFM: Selective Forwarding Middlebox

SSM: Source-Specific Multicast

SSRC: Synchronization Source

## 2.2. Definitions Related to RTP Grouping Taxonomy

The following definitions have been taken from [RFC7656].

Communication Session: A Communication Session is an association among two or more Participants communicating with each other via one or more Multimedia Sessions.

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

Media Source: A Media Source is the logical source of a time progressing digital media stream synchronized to a reference clock. This stream is called a Source Stream.

Multimedia Session: A Multimedia Session is an association among a group of participants engaged in communication via one or more RTP sessions.

## 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 it 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 SFMs. The section ends by discussing decomposited terminals, 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 that use multiple SSRCs 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" [MULTI-STREAM-OPT].

### 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 RTP stream that the translator receives, it generates an individual RTP stream in the other domain. A translator keeps the SSRC for an RTP stream across the translation, whereas a mixer can select a single RTP stream from multiple received RTP streams (in cases like audio/video switching) or send out an RTP stream composed of multiple mixed media received in multiple RTP streams (in cases like audio mixing or video tiling), 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 of a middlebox is enabling interoperability, between otherwise non-interoperable endpoints, 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 endpoints (as the RTP session participants) to run SSRC collision detection, and the SSRC is thus a field the translator cannot change. Any Source Description (SDES) information associated with an 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 RTP 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 it wants to trigger repair by the media sending endpoint, by sending feedback messages. While such feedback could use the SSRC of the target for the translator (the receiving endpoint), this in turn would require translation of the target RTCP reports to make them consistent. It may be simpler to expose an additional SSRC in the session. The only concern is that endpoints failing to support the full RTP specification may have issues with multiple SSRCs reporting on the RTP streams sent by that endpoint, as this use case may be viewed as exotic by implementers.

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

Shortcut name: Topo-PtP-Relay

There exist a number of different types of middleboxes that might be inserted between two 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 and network topologies where the RTP stream 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 Traversal Using Relays around NAT (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 since the RTP and RTCP packets are routed between the endpoints in the RTP session, and thereby they 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 signaling solution. Both in signaling 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 cannot support this will also not work correctly when endpoints are subject to NAT.

The Transport Relay implementations also have to take into account security considerations. In particular, source address filtering of incoming packets is usually important in relays, to prevent attackers from injecting traffic into a session, which one peer may, in the absence of adequate security in the relay, think it comes from the other peer.

#### 3.2.1.2. Transport Translator

Shortcut name: Topo-Trn-Translator

Transport Translators (Topo-Trn-Translator) do not modify the RTP 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 as unicast to multicast) and the media packetization to allow other transport protocols to be interconnected to a session (in gateways).

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.

Of the Transport Translators, this memo is primarily interested in those that use RTP on both sides, and this is assumed 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

Shortcut name: Topo-Media-Translator

Media Translators (Topo-Media-Translator) modify the media inside the RTP stream. This process is commonly known as transcoding. The modification of the media 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 endpoints 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 and the transport aspects of the RTP stream carrying the media between two transport domains.

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 endpoint B's RTCP receiver report before forwarding them to endpoint A. The rewriting is needed as the RTP stream received by B is not the same RTP 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 was substantially behind in reception, while it most likely would not be. Therefore, the translator must translate that number to a corresponding sequence number for the stream the translator received. Similar requirements exist for most other fields in the RTCP receiver reports.

A Media Translator may in some cases act on behalf of the "real" source (the endpoint originally sending the media to the translator) and respond to RTCP feedback messages. This may occur, for example, when a receiving endpoint requests a bandwidth reduction, and the Media Translator has not detected any congestion or other reasons for bandwidth reduction between the sending endpoint and itself. In that case, it is sensible that the Media Translator reacts to codec control messages itself, for example, by transcoding to a lower media rate.

A variant of translator behavior worth pointing out is the one depicted in Figure 3 of an endpoint A sending an RTP stream containing media (only) to B. On the path, there is a device T that manipulates the RTP streams on A's behalf. One common example is that T adds a second RTP stream containing Forward Error Correction (FEC) information in order to protect A's (non FEC-protected) RTP stream. In this case, T needs to semantically bind the new FEC RTP stream to A's media-carrying RTP stream, for example, by using the same CNAME as A.

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

Figure 3: Media Translator Adding FEC

There may also be cases where information is added into the original RTP stream, while leaving most or all of the original RTP packets intact (with the exception of certain RTP header fields, such as the sequence number). One example is the injection of metadata into the RTP stream, carried in their own RTP packets.

Similarly, a Media Translator can sometimes remove information from the RTP stream, while otherwise leaving the remaining RTP packets unchanged (again with the exception of certain RTP header fields).

Either type of functionality where T manipulates the RTP stream, or adds an accompanying RTP stream, on behalf of A is also covered under the Media Translator definition.

### 3.2.2. Back-to-Back RTP sessions

Shortcut name: Topo-Back-To-Back

There exist middleboxes that interconnect two endpoints (A and B) through themselves (MB), but not by being part of a common RTP session. Instead, they establish two different RTP sessions: one between A and the middlebox and another between the middlebox and B. This topology is called Topo-Back-To-Back.

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

Figure 4: Back-to-Back RTP Sessions through Middlebox

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 of this topology relative to 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 responsibility to maintain the correct relations.

The signaling or other above RTP-level functionalities referencing RTP 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 on each of the two segments or by bridging statistics from the receiving endpoint through the middlebox to the sending endpoint. From an implementation point, however, the latter requires dealing with a number of inconsistencies. First, packet loss must be detected for an RTP stream sent from A to the middlebox, and that loss must be reported through a skipped sequence number in the RTP stream from the middlebox to B. This coupling and the resulting inconsistencies are conceptually easier to handle when considering the two RTP streams 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. Each of 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 multicast group participant reaches all the other multicast group participants, except for cases such as:

- o packet loss, or
- o when a multicast group 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 multicast group participant is subscribed only to a subset of these sessions.

In the above context, "traffic" encompasses both RTP and RTCP traffic. The number of multicast group 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 ASM.

For feedback usage, it is useful to define a "small multicast group" as a group where the number of multicast group 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 multicast group 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 endpoint that is a multicast group participant needs to process every feedback message it receives, not only to determine if it is affected or if the feedback message applies only to some other endpoint but also to derive timing restrictions for the sending of its own feedback messages, if any.

### 3.3.2. Source-Specific Multicast (SSM)

Shortcut name: Topo-SSM

In Any-Source Multicast, any of the multicast group participants can send to all the other multicast group 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 sending endpoints (RTP sources henceforth) (1 to M) are allowed to send media to the SSM group. These sources send media to a dedicated Distribution Source, which forwards the RTP streams to the multicast group on behalf of the original RTP sources. The RTP streams reach the receiving endpoints (receivers henceforth) (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 IP sender. 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 distributed and each instance can have responsibility for subgroups of the receivers. For summary reports, however, there typically must be a single Feedback Target 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, need to be accounted for.

The 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 signaling functions to identify the relationships between RTP sessions.

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

All multicast configurations share a signaling requirement: all of the endpoints need to have the same RTP and payload type configuration. Otherwise, endpoint A could, for example, be using payload type 97 to identify the video codec H.264, while endpoint B would identify it as MPEG-2, with unpredictable but almost certainly not visually pleasing results.

Security solutions for this type of group communication are also challenging. First, the key management and the security protocol must support group communication. Source authentication becomes more difficult and requires specialized solutions. For more discussion on this, please review "Options for Securing RTP Sessions" [RFC7201].

### 3.3.3. SSM with Local Unicast Resources

Shortcut name: Topo-SSM-RAMS

"Unicast-Based Rapid Acquisition of Multicast RTP Sessions" [RFC6285] 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 Stream
      .-.-.-.> Multicast RTCP Stream
      .=.=.=.> Unicast RTCP Reports
      ~~~~~~~> Unicast RTCP Feedback Messages
      .......> Unicast RTP Stream
```

Figure 7: SSM with Local Unicast Resources (RAMS)

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 the 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 RTP stream. 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. In many deployed applications, the (still existing) PtP RTP session is 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 Multipoint Using Mesh

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

This topology 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 (in which case the sending port, IP address, or SSRC can be used to demultiplex), or it might have separate local reception ports for each of the endpoints.

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

Figure 9: A Multi-Unicast Mesh with a Joint RTP Session

Figure 9 depicts endpoint A's view of using a common RTP session when establishing the mesh as shown in Figure 8. There is only one RTP session (RTP1) but two transport flows (UDP1 and UDP2). 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, a 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 stream may be sent from the transmitting endpoint; however, it also supports local adaptation taking place in one or more of the RTP 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: A Multi-Unicast Mesh with an Independent RTP Session

Let's review the topology when independent RTP sessions are used from A's perspective in Figure 10 by considering both how the media is handled and how the RTP sessions are set up in Figure 10. A's microphone is captured and the audio is fed into two different encoder instances, each with a different independent RTP session, i.e., RTP1 and RTP2, respectively. The SSRCs (AA1 and AA2) in each RTP session are completely independent, and the media bitrate 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 A, B, and C, and their connecting RTP streams, a 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 namespace 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 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 |
   +---+      /            \      |            |      +---+
             +  Multicast  +->| 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) endpoints B and D to take part in an Any-Source Multicast session involving endpoints A and C, 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 both 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 endpoints 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 endpoints. 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 signaling requirements onto the session establishment; for example, a common configuration of RTP payload types is required.

Transport Translators and relays should always consider implementing source address filtering, to prevent attackers from using the listening ports on the translator to inject traffic. The translator can, however, go one step further, especially if explicit SSRC signaling is used, to prevent endpoints from sending SSRCs other than its own (that are, for example, used by other participants in the session). This can improve the security properties of the session, despite the use of group keys that on a cryptographic level allows anyone to impersonate another in the same RTP session.

A translator that doesn't change the RTP/RTCP packet content can be operated without requiring it 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 a given endpoint or a defined subset of endpoints can participate in the session.

If endpoint B in Figure 11 were behind a limited network path, the translator may perform media transcoding to allow the traffic received from the other endpoints to reach B without overloading the path. This transcoding can help the other endpoints 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 (and vice versa).

### 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 one or more new RTP streams 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 RTP streams received from other endpoints in the same RTP session. Using the media data carried in the received RTP streams, a mixer generates derived RTP streams that are sent to the receiving endpoints.

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 Communication Session.

The mixer creates the Media Source and the source RTP stream just like an endpoint, as it mixes the content (often in the uncompressed domain) and then encodes and packetizes it for transmission to a receiving endpoint. The CSRC Count (CC) and CSRC fields in the RTP header can be used to indicate the contributors to the newly generated RTP 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 endpoints to allow for loop detection and identification of sources that are part of the Communication 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 an RTP receiver and should therefore send RTCP receiver reports for the RTP streams it receives and terminates. In its role as an RTP sender, it should also generate RTCP sender reports for those RTP 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 endpoints A and C), endpoint B, and endpoint D. Assuming all four endpoints in the conference are interested in receiving content from all other endpoints, the mixer produces different mixed RTP 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 endpoints in the other domain(s). In other cases, a message is handled by the mixer locally 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 Mixer

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 RTP streams from several endpoints 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 RTP stream from this mix.

The most commonly deployed Media-Mixing Mixer is probably the audio mixer, used in voice conferencing, where the output consists of a mixture of all the input audio signals; this needs minimal signaling to be successfully set up. From a signal processing viewpoint, audio mixing is relatively straightforward and commonly possible for a reasonable number of endpoints. Assume, for example, that one wants to mix N streams from N different endpoints. 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 endpoint 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 endpoints, 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, noise suppression, and similar signal processing steps are also required or desirable before the actual mixing process commences.

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, which is also known as "tiling". The reconstructed, appropriately scaled down videos can be spatially arranged in a set of tiles, with each tile containing the video from an endpoint (typically showing a human participant). Tiles can be of different sizes so that, for example, a particularly important participant, or the loudest speaker, is being shown in a larger tile than other participants. A self-view picture can be included in the tiling, which can be either locally produced or feedback from a mixer-received and reconstructed video image. Such remote loopback allows for confidence monitoring, i.e., it enables the participant to see himself/herself in the same quality 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 to the receiving endpoints. 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 arrangements where an individual mixer is available for each outgoing port of the middlebox, allowing individual compositions for each receiving endpoint (a feature commonly referred to as personalized layout).

One problem with media mixing is that it consumes both large amounts of media processing resources (for the decoding and 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, which is the result of the lossy nature of the 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-synchronized with high-latency video. The advantage of media mixing is that it is straightforward for the endpoints 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 (that, in some protocol environments, actually didn't need to be aware of the multipoint nature of the conference) 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 <----+ | +---+  |     | |
           | | +-------|          |(AA1+CA1)|\| +---+  |     | |
           | +---------|          |---------+ +-|ENC|<-| A+C | |
           +-----------+          |-----------+ +---+  |     | |
                                  |                    |  M  | |
           +-C---------+          |                    |  I  | |
           | +-RTP3----|          |-RTP3------+        |  X  | |
           | | +-Audio-|          |-Audio---+ | +---+  |  E  | |
           | | |    CA1|--------->|---------+-+-|DEC|->|  R  | |
           | | |       |<---------|MA3 <----+ | +---+  |     | |
           | | +-------|          |(AA1+BA1)|\| +---+  |     | |
           | +---------|          |---------+ +-|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 endpoint, e.g., MA1 to Peer A, where the associated stream is the media mix of the other endpoints. 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 Sources 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 RTP header extension for Mixer-to-Client Audio Level Indication [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 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 Mixer

Media-Switching Mixers are used in limited functionality scenarios where no, or only very limited, concurrent presentation of multiple sources is required by the application and also in 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 RTP streams that the mixer can produce. These RTP streams are created by selecting media from one of the RTP 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 in their bitstream.

To achieve a coherent RTP 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 the 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-to-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 participating endpoints (A-F) where the two most recent speakers in the conference are shown to each receiving endpoint. Thus, the mixer has 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---+ |        |     | |
      | | |    FV1|------------>|---------+-+------->|     | |
      | | |       |<------------|MV11 <---+-+-AV1----|     | |
      | | |       |<------------|MV12 <---+-+-EV1----|     | |
      | | +-------|             |---------+ |        |     | |
      | +---------|             |-----------+        +-----+ |
      +-----------+             +----------------------------+
```

Figure 16: Media-Switching RTP Mixer

The Media-Switching Mixer can, similarly to the Media-Mixing Mixer, reduce the bitrate required for media transmission towards the different peers by selecting and forwarding only a subset of RTP streams it receives from the sending endpoints. In case the mixer receives simulcast transmissions or a scalable encoding of the Media Source, the mixer has more degrees of freedom to select streams or subsets of streams to forward to a receiving endpoint, both based on transport or endpoint restrictions as well as application logic.

To ensure that a media receiver in an endpoint can correctly decode the media in the RTP stream after a switch, a codec that uses temporal prediction needs to start its decoding from independent refresh points, or points in the bitstream offering similar functionality (like "dirty refresh points"). 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 fully terminating 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 endpoint conference depicted above (in Figure 17), one can see that endpoint 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 endpoints with the two latest speaking endpoints, 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 endpoints, each RTP stream requires the rewriting of certain 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 an RTP session. The timestamp (possibly offset) stays the same.

The RTP sessions can be considered independent, resulting in that the SSRC numbers used can also be handled independently. This simplifies the SSRC collision detection and avoidance but requires tools such as remapping tables between the RTP sessions. Using independent RTP sessions is not required, as it is possible for the switching behavior to also perform with a common SSRC space. However, in this case, collision detection and handling becomes a different problem. It is up to the implementation to use a single common SSRC space or separate ones.

Using separate SSRC spaces has some implications. For example, the RTP stream that is being sent by endpoint B to the middlebox (BV1) may use an SSRC value of 12345678. When that RTP 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 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 the RTP level. Instead, this responsibility falls onto higher layers.

The middlebox is also responsible for receiving any RTCP codec control requests coming from an endpoint and deciding if it can act on the request locally or needs to translate the request into the RTP session/transport leg that contains the Media Source. Both endpoints 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 that switching points be provided between the sources and Temporary Maximum Media Bitrate Request (TMMBR) to enable the middlebox to aggregate congestion control responses towards the Media Source so to enable it to adjust its bitrate (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 bitrate, desired layout, etc. Contrary to transcoding MCUs, SFMs 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 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 that includes an on/off logic as part of its media translation. The topology has the property that all SSRCs present in the session are visible to an endpoint. It also has mixer aspects, as the streams it provides are not basically translated versions, but instead they have conceptual property assigned to them and can be both turned on/off as well as fully or partially delivered. Thus, this topology appears to be some hybrid between the translator and mixer model.

The differences between a Selective Forwarding Middlebox and a Switching-Media 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 information on what is included in a particular RTP stream that represents a particular concept. Selective forwarding gets the source information through the SSRC and instead uses other mechanisms to indicate the streams intended usage, if needed.

### 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 bitrate, 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 endpoints. 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 RTP stream results in most participants seeing only a subset of the sent RTP streams at any given time, often a single RTP stream per conference. Therefore, RTCP receiver reports only report on these RTP streams. Consequently, the endpoints emitting RTP streams that are not currently forwarded receive a view of the session that indicates their RTP 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 RTP 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 endpoint 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 endpoints or

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

In case (a), the MCU may modify the content in terms of bitrate, encoding format, or resolution. No explicit RTP mechanism is used to establish the relationship between the original RTP stream of the media being sent and the RTP stream the MCU sends. In other words, the outgoing RTP streams 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 RTP session for each endpoint.

In case (b), the MCU is the Media Source and generates the Source RTP Stream as it mixes the received content and then encodes and packetizes it for transmission to an endpoint. According to RTP

[RFC3550], the SSRC of the contributors are to be signaled using the CSRC/CC mechanism. In practice, today, most deployed MCUs do not implement this feature. Instead, the identification of the endpoints whose content is included in the mixer's output is not indicated through any explicit RTP mechanism. That is, most deployed MCUs set the CC field in the RTP header to zero, thereby indicating no available CSRC information, even if they could identify the original sending endpoints 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 many/most deployed MCUs (and video conferencing endpoints) rely on signaling-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 Terminal

Shortcut name: Topo-Split-Terminal

In some applications, for example, in some telepresence systems, terminals may not be integrated into a single functional unit but composed of more than one subunits. For example, a telepresence room terminal employing multiple cameras and monitors may consist of multiple video conferencing subunits, each capable of handling a single camera and monitor. Another example would be a video conferencing terminal in which audio is handled by one subunit, and video by another. Each of these subunits uses its own physical network interface (for example: Ethernet jack) and network address.

The various (media processing) subunits need (logically and physically) to be interconnected by control functionality, but their media plane functionality may be split. These types of terminals are referred to as split component terminals. Historically, the earliest split component terminals were perhaps the independent audio and video conference software tools used over the MBONE in the late 1990s.

An example for such a split component terminal is depicted in Figure 20. Within split component terminal A, at least audio and video subunits are addressed by their own network addresses. In some of these systems, the control stack subunit may also have its own network address.

From an RTP viewpoint, each of the subunits terminates RTP and acts as an endpoint in the sense that each subunit includes its own, independent RTP stack. However, as the subunits are semantically part of the same terminal, it is appropriate that this semantic relationship is expressed in RTCP protocol elements, namely in the CNAME.

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

Figure 20: Split Component Terminal

It is further sensible that the subunits share a common clock from which RTP and RTCP clocks are derived, to facilitate synchronization and avoid clock drift.

To indicate that audio and video Source Streams generated by different subunits share a common clock, and can be synchronized, the RTP streams generated from those Source Streams need to include the same CNAME in their RTCP SDES packets. The use of a common CNAME for RTP flows carried in different transport-layer flows is entirely normal for RTP and RTCP senders, and fully compliant RTP endpoints, middleboxes, and other tools should have no problem with this.

However, outside of the split component terminal scenario (and perhaps a multihomed endpoint scenario, which is not further discussed herein), the use of a common CNAME in RTP streams sent from separate endpoints (as opposed to a common CNAME for RTP streams sent on different transport-layer flows between two endpoints) is rare. It has been reported that at least some third-party tools like some network monitors do not handle gracefully endpoints that use a common CNAME across multiple transport-layer flows: they report an error condition in which two separate endpoints are using the same CNAME. Depending on the sophistication of the support staff, such erroneous reports can lead to support issues.

The aforementioned support issue can sometimes be avoided if each of the subunits of a split component terminal is configured to use a different CNAME, with the synchronization between the RTP streams being indicated by some non-RTP signaling channel rather than using a common CNAME sent in RTCP. This complicates the signaling, especially in cases where there are multiple SSRCs in use with complex synchronization requirements, as is the same in many current telepresence systems. Unless one uses RTCP terminating topologies such as Topo-RTCP-terminating-MCU, sessions involving more than one video subunit with a common CNAME are close to unavoidable.

The different RTP streams comprising a split terminal system can form a single RTP session or they can form multiple RTP sessions, depending on the visibility of their SSRC values in RTCP reports. If the receiver of the RTP streams sent by the split terminal sends reports relating to all of the RTP flows (i.e., to each SSRC) in each RTCP report, then a single RTP session is formed. Alternatively, if the receiver of the RTP streams sent by the split terminal does not send cross-reports in RTCP, then the audio and video form separate RTP sessions.

For example, in Figure 20, B will send RTCP reports to each of the subunits of A. If the RTCP packets that B sends to the audio subunit of A include reports on the reception quality of the video as well as the audio, and similarly if the RTCP packets that B sends to the video subunit of A include reports on the reception quality of the audio as well as video, then a single RTP session is formed. However, if the RTCP packets B sends to the audio subunit of A only report on the received audio, and the RTCP packets B sends to the video subunit of A only report on the received video, then there are two separate RTP sessions.

Forming a single RTP session across the RTP streams sent by the different subunits of a split terminal gives each subunit visibility into reception quality of RTP streams sent by the other subunits.

This information can help diagnose reception quality problems, but at the cost of increased RTCP bandwidth use.

RTP streams sent by the subunits of a split terminal need to use the same CNAME in their RTCP packets if they are to be synchronized, irrespective of whether a single RTP session is formed or not.

## 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 RTP 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 media 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's and C's RTP streams 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's and D's sender reports. In the opposite direction, the receiver reports from A and C about B's and D's streams 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 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. Topology Properties

The topologies discussed in Section 3 have different properties. This section describes these properties. Note that, even if a certain property is supported within a particular topology concept, the necessary functionality may be optional to implement.

## 4.1. All-to-All Media Transmission

To recapitulate, multicast, and in particular ASM, provides the functionality that everyone may send to, or receive from, everyone else within the session. SSM can provide a similar functionality by having anyone intending to participate as a sender to send its media to the SSM Distribution Source. The SSM Distribution Source forwards the media to all receivers subscribed to the multicast group. Mesh, MCUs, mixers, Selective Forwarding Middleboxes (SFMs), 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 topologies that come closest to emulating Any-Source IP Multicast, with all-to-all transmission capabilities, are the Transport Translator function called "relay" in Section 3.5, as well as the Mesh with joint RTP sessions (Section 3.4). Media Translators, Mesh with independent RTP Sessions, mixers, SFUs, 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 endpoint considers the path to the least-capable receiving endpoint. Otherwise, the media transmissions may overload that path. Therefore, a sending endpoint needs to monitor the path from itself to any of the receiving endpoints, to detect the currently least-capable receiver and adapt its sending rate accordingly. As multiple endpoints may send simultaneously, the available resources may vary. RTCP's receiver reports help perform this monitoring, at least on a medium time scale.

The resource consumption for performing all-to-all transmission varies depending on the topology. Both ASM and SSM have the benefit that only one copy of each packet traverses a particular link. Using a relay causes the transmission of one copy of a packet per endpoint-to-relay path and packet transmitted. However, in most cases, the links carrying the multiple copies will be the ones close to the relay (which can be assumed to be part of the network infrastructure with good connectivity to the backbone) rather than the endpoints (which may be behind slower access links). The Mesh topologies causes N-1 streams of transmitted packets to traverse the first-hop link from the endpoint, in a mesh with N endpoints. How long the different paths are common is highly situation dependent.

The transmission of RTCP by design 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.

## 4.2. Transport or Media Interoperability

All translators, mixers, RTCP-terminating MCUs, and Mesh with individual RTP sessions allow changing the media encoding or the transport to other properties of the other domain, thereby providing extended interoperability in cases where the endpoints lack a common set of media codecs and/or transport protocols. Selective Forwarding Middleboxes can adopt the transport and (at least) selectively forward the encoded streams that match a receiving endpoint's capability. It requires an additional translator to change the media encoding if the encoded streams do not match the receiving endpoint's capabilities.

## 4.3. Per-Domain Bitrate Adaptation

Endpoints are often connected to each other with a heterogeneous set of paths. This makes congestion control in a Point-to-Multipoint set problematic. In the ASM, SSM, Mesh with common RTP session, and Transport Relay scenarios, each individual sending endpoint has to adapt to the receiving endpoint behind the least-capable path, yielding suboptimal quality for the endpoints behind the more capable paths. This is no longer an issue when Media Translators, mixers, SFMs, or MCUs are involved, as each endpoint only needs to adapt to the slowest path within its own domain. The translator, mixer, SFM, or MCU topologies all require their respective outgoing RTP streams to adjust the bitrate, 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 endpoint in all the domains at the cost (complexity, delay, equipment) of the mixer, SFM, or translator, and potentially the media sender (multicast/layered encoding and sending the different representations).

### 4.4. Aggregation of Media

In the all-to-all media property mentioned above and provided by ASM, SSM, Mesh with common RTP session, and relay, all simultaneous media transmissions share the available bitrate. For endpoints with limited reception capabilities, this may result in a situation where even a minimal, acceptable media quality cannot be accomplished, because multiple RTP streams need to share the same resources. One solution to this problem is to use a mixer, or MCU, to aggregate the multiple RTP streams into a single one, where the single RTP stream takes up less resources in terms of bitrate. This aggregation can be performed according to different methods. Mixing or selection are two common methods. Selection is almost always possible and easy to implement. Mixing requires resources in the mixer and may be relatively easy and not impair the quality too badly (audio) or quite difficult (video tiling, which is not only computationally complex but also reduces the pixel count per stream, with corresponding loss in perceptual quality).

### 4.5. 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 SDES. In topologies that provide a full all-to-all functionality, i.e., ASM, Mesh with common RTP session, and relay, a compliant RTP implementation offers the functionality directly as specified in RTP. In topologies that do not offer all-to-all communication, it is necessary that RTCP is handled correctly in domain bridging functions. RTP includes explicit specification text for translators and mixers, and for SFMs the required functionality can be derived from that text. However, the MCU described in Section 3.8 cannot offer the full functionality for session participant identification through RTP means. The topologies that create independent RTP sessions per endpoint or pair of endpoints, like a Back-to-Back RTP session, MESH with independent RTP sessions, and the RTCP terminating MCU (Section 3.9), with an exception of SFM, do not support RTP-based identification of session participants. In all those cases, other non-RTP-based mechanisms need to be implemented if such knowledge is required or desirable. When it comes to SFM, the SSRC namespace is not necessarily joint. Instead, identification will require knowledge of SSRC/CSRC mappings that the SFM performed; see Section 3.7.

## 4.6. Loop Detection

In complex topologies with multiple interconnected domains, it is possible to unintentionally form media loops. RTP and RTCP support detecting such loops, as long as the SSRC and CSRC identities are maintained and correctly set in forwarded packets. Loop detection will work in ASM, SSM, Mesh with joint RTP session, and relay. It is likely that loop detection works for the video-switching MCU, Section 3.8, at least as long as it forwards the RTCP between the endpoints. However, the Back-to-Back RTP sessions, Mesh with independent RTP sessions, and SFMs will definitely break the loop detection mechanism.

## 4.7. Consistency between Header Extensions and RTCP

Some RTP header extensions have relevance not only end to end but also hop to hop, meaning at least some of the middleboxes in the path are aware of their potential presence through signaling, intercept and interpret such header extensions, and potentially also rewrite or generate them. Modern header extensions generally follow "A General Mechanism for RTP Header Extensions" [RFC5285], which allows for all of the above. Examples for such header extensions include the Media ID (MID) in [SDP-BUNDLE]. At the time of writing, there was also a proposal for how to include some SDES into an RTP header extension [RTCP-SDES].

When such header extensions are in use, any middlebox that understands it must ensure consistency between the extensions it sees and/or generates and the RTCP it receives and generates. For example, the MID of the bundle is sent in an RTP header extension and also in an RTCP SDES message. This apparent redundancy was introduced as unaware middleboxes may choose to discard RTP header extensions. Obviously, inconsistency between the MID sent in the RTP header extension and in the RTCP SDES message could lead to undesirable results, and, therefore, consistency is needed. Middleboxes unaware of the nature of a header extension, as specified in [RFC5285], are free to forward or discard header extensions.

# 5. 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-ASM (ASM), Topo-SSM (SSM), Topo-Trn-Translator (TT), Topo-Media-Translator (including Transport Translator) (MT), Topo-Mesh with joint session (MJS), Topo-Mesh with individual sessions (MIS), Topo-Mixer (Mix), Topo-Asymmetric (ASY), Topo-Video-switch-MCU (VSM), Topo-RTCP-terminating-MCU (RTM), and Selective Forwarding Middlebox (SFM). 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 | ASM | SSM | TT | MT | MJS | MIS | Mix | ASY | VSM | RTM | SFM |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| All-to-All Media | N | Y | (Y) | Y | Y | Y | (Y) | (Y) | (Y) | (Y) | (Y) | (Y) |
| Interoperability | N/A | N | N | Y | Y | Y | Y | Y | Y | N | Y | Y |
| Per-Domain Adaptation | N/A | N | N | N | Y | N | Y | Y | Y | N | Y | Y |
| Aggregation of Media | N | N | N | N | N | N | N | Y | (Y) | Y | Y | N |
| Full Session View | Y | Y | Y | Y | Y | Y | N | Y | Y | (Y) | N | Y |
| Loop Detection | Y | Y | Y | Y | Y | Y | N | Y | Y | (Y) | N | N |

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

## 6. Security Considerations

The use of mixers, SFMs, and translators has impact on security and the security functions used. The primary issue is that mixers, SFMs, 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 Real-time Transport Protocol (SRTP) and Secure Real-time Transport Control Protocol (SRTCP) [RFC3711] packets to endpoints in the Communication Session. If encryption is employed, the Media Translator, SFM, 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, SFM, or mixer needs to be part of the signaling context and get the necessary security associations (e.g., SRTP crypto contexts) established with its RTP session participants.

Including the mixer, SFM, 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 human session participants that they are protected.

Transport Translators have no interactions with cryptography that work 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, SFMs, and mixers, it is possible that the translator, SFM, 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 rather 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, SFMs, 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.

The ASM, SSM, Relay, and Mesh (with common RTP session) topologies each have multiple endpoints that require shared knowledge about the different crypto contexts for the endpoints. These multiparty topologies have special requirements on the key management as well as the security functions. Specifically, source authentication in these environments has special requirements.

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 [RFC7201] to get a better understanding of most of the available options.

# 7. References

## 7.1. Normative References

[RFC3550] Schulzrinne, H., Casner, S., Frederick, R., and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications", STD 64, RFC 3550, DOI 10.17487/RFC3550, July 2003, <http://www.rfc-editor.org/info/rfc3550>.

[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, DOI 10.17487/RFC4585, July 2006, <http://www.rfc-editor.org/info/rfc4585>.

[RFC7656] Lennox, J., Gross, K., Nandakumar, S., Salgueiro, G., and B. Burman, Ed., "A Taxonomy of Grouping Semantics and Mechanisms for Real-Time Transport Protocol (RTP) Sources", RFC 7656, November 2015, <http://www.rfc-editor.org/info/rfc7656>.

## 7.2. Informative References

[MULTI-STREAM-OPT] Lennox, J., Westerlund, M., Wu, W., and C. Perkins, "Sending Multiple Media Streams in a Single RTP Session: Grouping RTCP Reception Statistics and Other Feedback", Work in Progress, draft-ietf-avtcore-rtp-multi-stream-optimisation-08, October 2015.

[RFC1112] Deering, S., "Host extensions for IP multicasting", STD 5, RFC 1112, DOI 10.17487/RFC1112, August 1989, <http://www.rfc-editor.org/info/rfc1112>.

[RFC3022] Srisuresh, P. and K. Egevang, "Traditional IP Network Address Translator (Traditional NAT)", RFC 3022, DOI 10.17487/RFC3022, January 2001, <http://www.rfc-editor.org/info/rfc3022>.

[RFC3569] Bhattacharyya, S., Ed., "An Overview of Source-Specific Multicast (SSM)", RFC 3569, DOI 10.17487/RFC3569, July 2003, <http://www.rfc-editor.org/info/rfc3569>.

[RFC3711] Baugher, M., McGrew, D., Naslund, M., Carrara, E., and K. Norrman, "The Secure Real-time Transport Protocol (SRTP)", RFC 3711, DOI 10.17487/RFC3711, March 2004, <http://www.rfc-editor.org/info/rfc3711>.

[RFC4575] Rosenberg, J., Schulzrinne, H., and O. Levin, Ed., "A Session Initiation Protocol (SIP) Event Package for Conference State", RFC 4575, DOI 10.17487/RFC4575, August 2006, <http://www.rfc-editor.org/info/rfc4575>.

[RFC4607] Holbrook, H. and B. Cain, "Source-Specific Multicast for IP", RFC 4607, DOI 10.17487/RFC4607, August 2006, <http://www.rfc-editor.org/info/rfc4607>.

[RFC5104] Wenger, S., Chandra, U., Westerlund, M., and B. Burman, "Codec Control Messages in the RTP Audio-Visual Profile with Feedback (AVPF)", RFC 5104, DOI 10.17487/RFC5104, February 2008, <http://www.rfc-editor.org/info/rfc5104>.

[RFC5117] Westerlund, M. and S. Wenger, "RTP Topologies", RFC 5117, DOI 10.17487/RFC5117, January 2008, <http://www.rfc-editor.org/info/rfc5117>.

[RFC5285] Singer, D. and H. Desineni, "A General Mechanism for RTP Header Extensions", RFC 5285, DOI 10.17487/RFC5285, July 2008, <http://www.rfc-editor.org/info/rfc5285>.

[RFC5760] Ott, J., Chesterfield, J., and E. Schooler, "RTP Control Protocol (RTCP) Extensions for Single-Source Multicast Sessions with Unicast Feedback", RFC 5760, DOI 10.17487/RFC5760, February 2010, <http://www.rfc-editor.org/info/rfc5760>.

[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, DOI 10.17487/RFC5766, April 2010, <http://www.rfc-editor.org/info/rfc5766>.

[RFC6285] Ver Steeg, B., Begen, A., Van Caenegem, T., and Z. Vax, "Unicast-Based Rapid Acquisition of Multicast RTP Sessions", RFC 6285, DOI 10.17487/RFC6285, June 2011, <http://www.rfc-editor.org/info/rfc6285>.

[RFC6465] Ivov, E., Ed., Marocco, E., Ed., and J. Lennox, "A Real-time Transport Protocol (RTP) Header Extension for Mixer-to-Client Audio Level Indication", RFC 6465, DOI 10.17487/RFC6465, December 2011, <http://www.rfc-editor.org/info/rfc6465>.

[RFC7201] Westerlund, M. and C. Perkins, "Options for Securing RTP Sessions", RFC 7201, DOI 10.17487/RFC7201, April 2014, <http://www.rfc-editor.org/info/rfc7201>.

[RTCP-SDES] Westerlund, M., Burman, B., Even, R., and M. Zanaty, "RTP Header Extension for RTCP Source Description Items", Work in Progress, draft-ietf-avtext-sdes-hdr-ext-02, July 2015.

[SDP-BUNDLE] Holmberg, C., Alvestrand, H., and C. Jennings, "Negotiating Media Multiplexing Using the Session Description Protocol (SDP)", Work in Progress, draft-ietf-mmusic-sdp-bundle-negotiation-23, July 2015.

## Acknowledgements

The authors would like to thank Mark Baugher, Bo Burman, Ben Campbell, Umesh Chandra, Alex Eleftheriadis, Roni Even, Ladan Gharai, Geoff Hunt, Suresh Krishnan, Keith Lantz, Jonathan Lennox, Scarlet Liuyan, Suhas Nandakumar, Colin Perkins, and Dan Wing for their help in reviewing and improving this document.

## Authors' Addresses

Magnus Westerlund
Ericsson
Farogatan 2
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
United States

Email: stewe@stewe.org