Stream: Internet Engineering Task Force (IETF)
RFC: 8872
Category: Informational
Published: January 2021
ISSN: 2070-1721
Authors:
M. Westerlund, *Ericsson*
B. Burman, *Ericsson*
C. Perkins, *University of Glasgow*
H. Alvestrand, *Google*
R. Even

RFC 8872
Guidelines for Using the Multiplexing Features of RTP to Support Multiple Media Streams

Abstract

The Real-time Transport Protocol (RTP) is a flexible protocol that can be used in a wide range of applications, networks, and system topologies. That flexibility makes for wide applicability but can complicate the application design process. One particular design question that has received much attention is how to support multiple media streams in RTP. This memo discusses the available options and design trade-offs, and provides guidelines on how to use the multiplexing features of RTP to support multiple media streams.

Status of This Memo

This document is not an Internet Standards Track specification; it is published for informational purposes.

This document is a product of the Internet Engineering Task Force (IETF). It represents the consensus of the IETF community. It has received public review and has been approved for publication by the Internet Engineering Steering Group (IESG). Not all documents approved by the IESG are candidates for any level of Internet Standard; see Section 2 of RFC 7841.

Information about the current status of this document, any errata, and how to provide feedback on it may be obtained at https://www.rfc-editor.org/info/rfc8872.

Copyright Notice

Table of Contents

1. Introduction

The Real-time Transport Protocol (RTP) [RFC3550] is a commonly used protocol for real-time media transport. It is a protocol that provides great flexibility and can support a large set of different applications. From the beginning, RTP was designed for multiple participants in a communication session. It supports many topology paradigms and usages, as defined in [RFC7667]. RTP has several multiplexing points designed for different purposes; these points enable support of multiple RTP streams and switching between different encoding or packetization techniques for the media. By using multiple RTP sessions, sets of RTP streams can be structured for efficient processing or identification. Thus, to meet an application's needs, an RTP application designer needs to understand how best to use the RTP session, the RTP stream identifier (synchronization source (SSRC)), and the RTP payload type.

There has been increased interest in more-advanced usage of RTP. For example, multiple RTP streams can be used when a single endpoint has multiple media sources (like multiple cameras or microphones) from which streams of media need to be sent simultaneously. Consequently, questions are raised regarding the most appropriate RTP usage. The limitations in some implementations, RTP/RTCP extensions, and signaling have also been exposed. This document aims to clarify the usefulness of some functionalities in RTP that, hopefully, will result in future implementations that are more complete.

The purpose of this document is to provide clear information about the possibilities of RTP when it comes to multiplexing. The RTP application designer needs to understand the implications arising from a particular usage of the RTP multiplexing points. This document provides some guidelines and recommends against some usages as being unsuitable, in general or for particular purposes.

This document starts with some definitions and then goes into existing RTP functionalities around multiplexing. Both the desired behavior and the implications of a particular behavior depend on which topologies are used; therefore, this topic requires some consideration. We then discuss some choices regarding multiplexing behavior and the impacts of those choices. Some designs of RTP usage are also discussed. Finally, some guidelines and examples are provided.

2. Definitions

2.1. Terminology

The definitions in Section 3 of [RFC3550] are referenced normatively.

The taxonomy defined in [RFC7656] is referenced normatively.

The following terms and abbreviations are used in this document:

Multi-party:
: Communication that includes multiple endpoints. In this document, "multi-party" will be used to refer to scenarios where more than two endpoints communicate.

Multiplexing:
: An operation that takes multiple entities as input, aggregating them onto some common resource while keeping the individual entities addressable such that they can later be fully and unambiguously separated (demultiplexed) again.

RTP Receiver:
: An endpoint or middlebox receiving RTP streams and RTCP messages. It uses at least one SSRC to send RTCP messages. An RTP receiver may also be an RTP sender.

RTP Sender:
: An endpoint sending one or more RTP streams but also sending RTCP messages.

RTP Session Group:
: One or more RTP sessions that are used together to perform some function. Examples include multiple RTP sessions used to carry different layers of a layered encoding. In an RTP Session Group, CNAMEs are assumed to be valid across all RTP sessions and designate synchronization contexts that can cross RTP sessions; i.e., SSRCs that map to a common CNAME can be assumed to have RTCP Sender Report (SR) timing information derived from a common clock such that they can be synchronized for playout.

Signaling:
: The process of configuring endpoints to participate in one or more RTP sessions.

> Note: The above definitions of "RTP receiver" and "RTP sender" are consistent with the usage in [RFC3550].

2.2. Focus of This Document

This document is focused on issues that affect RTP. Thus, issues that involve signaling protocols -- such as whether SIP [RFC3261], Jingle [JINGLE], or some other protocol is in use for session configuration; the particular syntaxes used to define RTP session properties; or the constraints imposed by particular choices in the signaling protocols -- are mentioned only as examples in order to describe the RTP issues more precisely.

This document assumes that the applications will use RTCP. While there are applications that don't send RTCP, they do not conform to the RTP specification and thus can be regarded as reusing the RTP packet format but not implementing RTP.

3. RTP Multiplexing Overview

3.1. Reasons for Multiplexing and Grouping RTP Streams

There are several reasons why an endpoint might choose to send multiple media streams. In the discussion below, please keep in mind that the reasons for having multiple RTP streams vary and include, but are not limited to, the following:

- There might be multiple media sources.
- Multiple RTP streams might be needed to represent one media source, for example:
  - To carry different layers of a scalable encoding of a media source
  - Alternative encodings during simulcast, using different codecs for the same audio stream
  - Alternative formats during simulcast, multiple resolutions of the same video stream
- A retransmission stream might repeat some parts of the content of another RTP stream.
- A Forward Error Correction (FEC) stream might provide material that can be used to repair another RTP stream.

For each of these reasons, it is necessary to decide whether each additional RTP stream is sent within the same RTP session as the other RTP streams or it is necessary to use additional RTP sessions to group the RTP streams. For a combination of reasons, the suitable choice for one situation might not be the suitable choice for another situation. The choice is easiest when multiplexing multiple media sources of the same media type. However, all reasons warrant discussion and clarification regarding how to deal with them. As the discussion below will show, a single solution does not suit all purposes. To utilize RTP well and as efficiently as possible, both are needed. The real issue is knowing when to create multiple RTP sessions versus when to send multiple RTP streams in a single RTP session.

3.2. RTP Multiplexing Points

This section describes the multiplexing points present in RTP that can be used to distinguish RTP streams and groups of RTP streams. Figure 1 outlines the process of demultiplexing incoming RTP streams, starting with one or more sockets representing the reception of one or more transport flows, e.g., based on the UDP destination port. It also demultiplexes RTP/RTCP from any other protocols, such as Session Traversal Utilities for NAT (STUN) [RFC5389] and DTLS-SRTP [RFC5764] on the same transport as described in [RFC7983]. The Processing and Buffering (PB) step in Figure 1 terminates RTP/RTCP and prepares the RTP payload for input to the decoder.

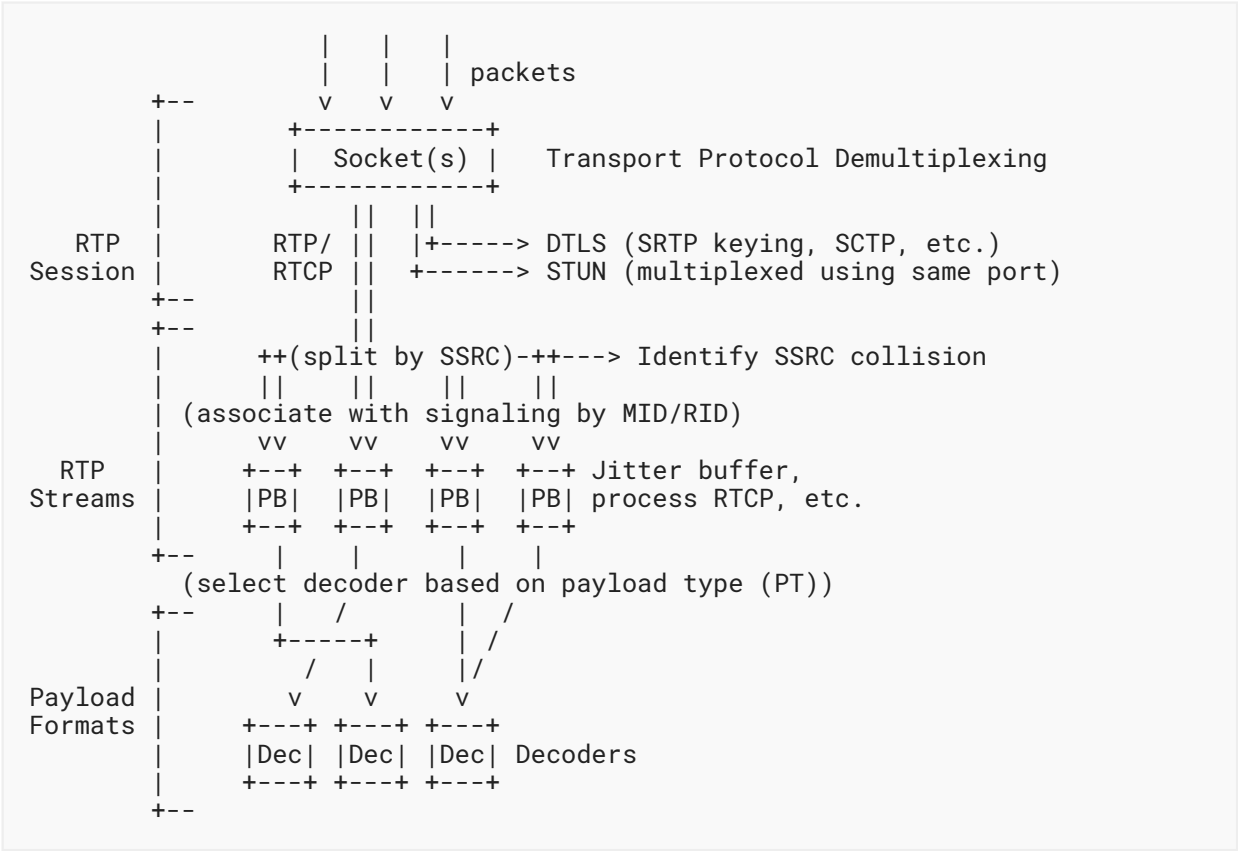

Figure 1: RTP Demultiplexing Process

3.2.1. RTP Session

An RTP session is the highest semantic layer in RTP and represents an association between a group of communicating endpoints. RTP does not contain a session identifier, yet different RTP sessions must be possible to identify both across a set of different endpoints and from the perspective of a single endpoint.

For RTP session separation across endpoints, the set of participants that form an RTP session is defined as those that share a single SSRC space [RFC3550]. That is, if a group of participants are each aware of the SSRC identifiers belonging to the other participants, then those participants are in a single RTP session. A participant can become aware of an SSRC identifier by receiving an RTP packet containing the identifier in the SSRC field or contributing source (CSRC) list, by receiving an RTCP packet listing it in an SSRC field, or through signaling (e.g., the Session Description Protocol (SDP) [RFC4566] "a=ssrc:" attribute [RFC5576]). Thus, the scope of an RTP session is determined by the participants' network interconnection topology, in combination with RTP and RTCP forwarding strategies deployed by the endpoints and any middleboxes, and by the signaling.

For RTP session separation within a single endpoint, RTP relies on the underlying transport layer and the signaling to identify RTP sessions in a manner that is meaningful to the application. A single endpoint can have one or more transport flows for the same RTP session, and a single RTP

session can span multiple transport-layer flows even if all endpoints use a single transport-layer flow per endpoint for that RTP session. The signaling layer might give RTP sessions an explicit identifier, or the identification might be implicit based on the addresses and ports used. Accordingly, a single RTP session can have multiple associated identifiers, explicit and implicit, belonging to different contexts. For example, when running RTP on top of UDP/IP, an endpoint can identify and delimit an RTP session from other RTP sessions by their UDP source and destination IP addresses and their UDP port numbers. A single RTP session can be using multiple IP/UDP flows for receiving and/or sending RTP packets to other endpoints or middleboxes, even if the endpoint does not have multiple IP addresses. Using multiple IP addresses only makes it more likely that multiple IP/UDP flows will be required. Another example is SDP media descriptions (the "m=" line and the subsequent associated lines) that signal the transport flow and RTP session configuration for the endpoint's part of the RTP session. The SDP grouping framework [RFC5888] allows labeling of the media descriptions to be used so that RTP Session Groups can be created. Through the use of "Negotiating Media Multiplexing Using the Session Description Protocol (SDP)" [RFC8843], multiple media descriptions become part of a common RTP session where each media description represents the RTP streams sent or received for a media source.

RTP makes no normative statements about the relationship between different RTP sessions; however, applications that use more than one RTP session need to understand how the different RTP sessions that they create relate to one another.

3.2.2. Synchronization Source (SSRC)

An SSRC identifies a source of an RTP stream, or an RTP receiver when sending RTCP. Every endpoint has at least one SSRC identifier, even if it does not send RTP packets. RTP endpoints that are only RTP receivers still send RTCP and use their SSRC identifiers in the RTCP packets they send. An endpoint can have multiple SSRC identifiers if it sends multiple RTP streams. Endpoints that function as both RTP sender and RTP receiver use the same SSRC(s) in both roles.

The SSRC is a 32-bit identifier. It is present in every RTP and RTCP packet header and in the payload of some RTCP packet types. It can also be present in SDP signaling. Unless presignaled, e.g., using the SDP "a=ssrc:" attribute [RFC5576], the SSRC is chosen at random. It is not dependent on the network address of the endpoint and is intended to be unique within an RTP session. SSRC collisions can occur and are handled as specified in [RFC3550] and [RFC5576], resulting in the SSRC of the colliding RTP streams or receivers changing. An endpoint that changes its network transport address during a session has to choose a new SSRC identifier to avoid being interpreted as a looped source, unless a mechanism providing a virtual transport (such as Interactive Connectivity Establishment (ICE) [RFC8445]) abstracts the changes.

SSRC identifiers that belong to the same synchronization context (i.e., that represent RTP streams that can be synchronized using information in RTCP SR packets) use identical CNAME chunks in corresponding RTCP source description (SDES) packets. SDP signaling can also be used to provide explicit SSRC grouping [RFC5576].

In some cases, the same SSRC identifier value is used to relate streams in two different RTP sessions, such as in RTP retransmission [RFC4588]. This is to be avoided, since there is no guarantee that SSRC values are unique across RTP sessions. In the case of RTP retransmission [RFC4588], it is recommended to use explicit binding of the source RTP stream and the redundancy stream, e.g., using the RepairedRtpStreamId RTCP SDES item [RFC8852]. The RepairedRtpStreamId is a rather recent mechanism, so one cannot expect older applications to follow this recommendation.

Note that the RTP sequence number and RTP timestamp are scoped by the SSRC and are thus specific per RTP stream.

Different types of entities use an SSRC to identify themselves, as follows:

- A real media source uses the SSRC to identify a "physical" media source.
- A conceptual media source uses the SSRC to identify the result of applying some filtering function in a network node -- for example, a filtering function in an RTP mixer that provides the most active speaker based on some criteria, or a mix representing a set of other sources.
- An RTP receiver uses the SSRC to identify itself as the source of its RTCP reports.

An endpoint that generates more than one media type, e.g., a conference participant sending both audio and video, need not (and, indeed, should not) use the same SSRC value across RTP sessions. Using RTCP compound packets containing the CNAME SDES item is the designated method for binding an SSRC to a CNAME, effectively cross-correlating SSRCs within and between RTP sessions as coming from the same endpoint. The main property attributed to SSRCs associated with the same CNAME is that they are from a particular synchronization context and can be synchronized at playback.

An RTP receiver receiving a previously unseen SSRC value will interpret it as a new source. It might in fact be a previously existing source that had to change its SSRC number due to an SSRC conflict. Using the media identification (MID) extension [RFC8843] helps to identify which media source the new SSRC represents, and using the restriction identifier (RID) extension [RFC8851] helps to identify what encoding or redundancy stream it represents, even though the SSRC changed. However, the originator of the previous SSRC ought to have ended the conflicting source by sending an RTCP BYE for it prior to starting to send with the new SSRC, making the new SSRC a new source.

3.2.3. Contributing Source (CSRC)

The CSRC is not a separate identifier. Rather, an SSRC identifier is listed as a CSRC in the RTP header of a packet generated by an RTP mixer or video Multipoint Control Unit (MCU) / switch, if the corresponding SSRC was in the header of one of the packets that contributed to the output.

It is not possible, in general, to extract media represented by an individual CSRC, since it is typically the result of a media merge (e.g., mix) operation on the individual media streams corresponding to the CSRC identifiers. The exception is the case where only a single CSRC is indicated, as this represents the forwarding of an RTP stream that might have been modified. The RTP header extension ("A Real-time Transport Protocol (RTP) Header Extension for Mixer-to-

Client Audio Level Indication" [RFC6465]) expands on the receiver's information about a packet with a CSRC list. Due to these restrictions, a CSRC will not be considered a fully qualified multiplexing point and will be disregarded in the rest of this document.

3.2.4. RTP Payload Type

Each RTP stream utilizes one or more RTP payload formats. An RTP payload format describes how the output of a particular media codec is framed and encoded into RTP packets. The payload format is identified by the payload type (PT) field in the RTP packet header. The combination of SSRC and PT therefore identifies a specific RTP stream in a specific encoding format. The format definition can be taken from [RFC3551] for statically allocated payload types but ought to be explicitly defined in signaling, such as SDP, for both static and dynamic payload types. The term "format" here includes those aspects described by out-of-band signaling means; in SDP, the term "format" includes media type, RTP timestamp sampling rate, codec, codec configuration, payload format configurations, and various robustness mechanisms such as redundant encodings [RFC2198].

The RTP payload type is scoped by the sending endpoint within an RTP session. PT has the same meaning across all RTP streams in an RTP session. All SSRCs sent from a single endpoint share the same payload type definitions. The RTP payload type is designed such that only a single payload type is valid at any instant in time in the RTP stream's timestamp timeline, effectively time-multiplexing different payload types if any change occurs. The payload type can change on a per-packet basis for an SSRC -- for example, a speech codec making use of generic comfort noise [RFC3389]. If there is a true need to send multiple payload types for the same SSRC that are valid for the same instant, then redundant encodings [RFC2198] can be used. Several additional constraints, other than those mentioned above, need to be met to enable this usage, one of which is that the combined payload sizes of the different payload types ought not exceed the transport MTU.

Other aspects of using the RTP payload format are described in "How to Write an RTP Payload Format" [RFC8088].

The payload type is not a multiplexing point at the RTP layer (see Appendix A for a detailed discussion of why using the payload type as an RTP multiplexing point does not work). The RTP payload type is, however, used to determine how to consume and decode an RTP stream. The RTP payload type number is sometimes used to associate an RTP stream with the signaling, which in general requires that unique RTP payload type numbers be used in each context. Using MID, e.g., when bundling "m=" sections [RFC8843], can replace the payload type as a signaling association, and unique RTP payload types are then no longer required for that purpose.

3.3. Issues Related to RTP Topologies

The impact of how RTP multiplexing is performed will in general vary with how the RTP session participants are interconnected, as described in "RTP Topologies" [RFC7667].

Even the most basic use case -- "Topo-Point-to-Point" as described in [RFC7667] -- raises a number of considerations, which are discussed in detail in the following sections. They range over such aspects as the following:

- Does my communication peer support RTP as defined with multiple SSRCs per RTP session?
- Do I need network differentiation in the form of QoS (Section 4.2.1)?
- Can the application more easily process and handle the media streams if they are in different RTP sessions?
- Do I need to use additional RTP streams for RTP retransmission or FEC?

For some point-to-multipoint topologies (e.g., Topo-ASM and Topo-SSM [RFC7667]), multicast is used to interconnect the session participants. Special considerations (documented in Section 4.2.3) are then needed, as multicast is a one-to-many distribution system.

Sometimes, an RTP communication session can end up in a situation where the communicating peers are not compatible, for various reasons:

- No common media codec for a media type, thus requiring transcoding.
- Different support for multiple RTP streams and RTP sessions.
- Usage of different media transport protocols (i.e., one peer uses RTP, but the other peer uses a different transport protocol).
- Usage of different transport protocols, e.g., UDP, the Datagram Congestion Control Protocol (DCCP), or TCP.
- Different security solutions (e.g., IPsec, TLS, DTLS, or the Secure Real-time Transport Protocol (SRTP)) with different keying mechanisms.

These compatibility issues can often be resolved by the inclusion of a translator between the two peers -- the Topo-PtP-Translator, as described in [RFC7667]. The translator's main purpose is to make the peers look compatible to each other. There can also be reasons other than compatibility for inserting a translator in the form of a middlebox or gateway -- for example, a need to monitor the RTP streams. Beware that changing the stream transport characteristics in the translator can require a thorough understanding of aspects ranging from congestion control and media-level adaptations to application-layer semantics.

Within the uses enabled by the RTP standard, the point-to-point topology can contain one or more RTP sessions with one or more media sources per session, each having one or more RTP streams per media source.

3.4. Issues Related to RTP and RTCP

Using multiple RTP streams is a well-supported feature of RTP. However, for most implementers or people writing RTP/RTCP applications or extensions attempting to apply multiple streams, it can be unclear when it is most appropriate to add an additional RTP stream in an existing RTP session and when it is better to use multiple RTP sessions. This section discusses the various considerations that need to be taken into account.

3.4.1. The RTP Specification

RFC 3550 contains some recommendations and a numbered list (Section 5.2 of [RFC3550]) of five arguments regarding different aspects of RTP multiplexing. Please review Section 5.2 of [RFC3550]. Five important aspects are quoted below.

1. > If, say, two audio streams shared the same RTP session and the same SSRC value, and one were to change encodings and thus acquire a different RTP payload type, there would be no general way of identifying which stream had changed encodings.

   This argument advocates the use of different SSRCs for each individual RTP stream, as this is fundamental to RTP operation.

2. > An SSRC is defined to identify a single timing and sequence number space. Interleaving multiple payload types would require different timing spaces if the media clock rates differ and would require different sequence number spaces to tell which payload type suffered packet loss.

   This argument advocates against demultiplexing RTP streams within a session based only on their RTP payload type numbers; it still stands, as can be seen by the extensive list of issues discussed in Appendix A.

3. > The RTCP sender and receiver reports (see Section 6.4) can only describe one timing and sequence number space per SSRC and do not carry a payload type field.

   This argument is yet another argument against payload type multiplexing.

4. > An RTP mixer would not be able to combine interleaved streams of incompatible media into one stream.

   This argument advocates against multiplexing RTP packets that require different handling into the same session. In most cases, the RTP mixer must embed application logic to handle streams; the separation of streams according to stream type is just another piece of application logic, which might or might not be appropriate for a particular application. One type of application that can mix different media sources blindly is the audio-only telephone bridge, although the ability to do that comes from the well-defined scenario that is aided by the use of a single media type, even though individual streams may use incompatible codec types; most other types of applications need application-specific logic to perform the mix correctly.

5. Carrying multiple media in one RTP session precludes: the use of different network paths or network resource allocations if appropriate; reception of a subset of the media if desired, for example just audio if video would exceed the available bandwidth; and receiver implementations that use separate processes for the different media, whereas using separate RTP sessions permits either single- or multiple-process implementations.

   This argument discusses network aspects that are described in Section 4.2. It also goes into aspects of implementation, like split component terminals (see Section 3.10 of [RFC7667]) -- endpoints where different processes or interconnected devices handle different aspects of the whole multimedia session.

To summarize, RFC 3550's view on multiplexing is to use unique SSRCs for anything that is its own media/packet stream and use different RTP sessions for media streams that don't share a media type. This document supports the first point; it is very valid. The latter needs further discussion, as imposing a single solution on all usages of RTP is inappropriate. "Sending Multiple Types of Media in a Single RTP Session" [RFC8860] updates RFC 3550 to allow multiple media types in an RTP session and provides a detailed analysis of the potential benefits and issues related to having multiple media types in the same RTP session. Thus, [RFC8860] provides a wider scope for an RTP session and considers multiple media types in one RTP session as a possible choice for the RTP application designer.

3.4.2. Multiple SSRCs in a Session

Using multiple SSRCs at one endpoint in an RTP session requires that some unclear aspects of the RTP specification be resolved. These items could potentially lead to some interoperability issues as well as some potential significant inefficiencies, as further discussed in "Sending Multiple RTP Streams in a Single RTP Session" [RFC8108]. An RTP application designer should consider these issues and the application's possible impact caused by a lack of appropriate RTP handling or optimization in the peer endpoints.

Using multiple RTP sessions can potentially mitigate application issues caused by multiple SSRCs in an RTP session.

3.4.3. Binding Related Sources

A common problem in a number of various RTP extensions has been how to bind related RTP streams together. This issue is common to both using additional SSRCs and multiple RTP sessions.

The solutions can be divided into a few groups:

- RTP/RTCP based
- Signaling based, e.g., SDP
- Grouping related RTP sessions
- Grouping SSRCs within an RTP session

Most solutions are explicit, but some implicit methods have also been applied to the problem.

The SDP-based signaling solutions are:

SDP media description grouping:
: The SDP grouping framework [RFC5888] uses various semantics to group any number of media descriptions. SDP media description grouping has primarily been used to group RTP sessions, but in combination with [RFC8843], it can also group multiple media descriptions within a single RTP session.

SDP media multiplexing:
: "Negotiating Media Multiplexing Using the Session Description Protocol (SDP)" [RFC8843] uses information taken from both SDP and RTCP to associate RTP streams to SDP media descriptions. This allows both SDP and RTCP to group RTP streams belonging to an SDP media description and group multiple SDP media descriptions into a single RTP session.

SDP SSRC grouping:
: "Source-Specific Media Attributes in the Session Description Protocol (SDP)" [RFC5576] includes a solution for grouping SSRCs in the same way that the grouping framework groups media descriptions.

The above grouping constructs support many use cases. Those solutions have shortcomings in cases where the session's dynamic properties are such that it is difficult or a drain on resources to keep the list of related SSRCs up to date.

One RTP/RTCP-based grouping solution is to use the RTCP SDES CNAME to bind related RTP streams to an endpoint or a synchronization context. For applications with a single RTP stream per type (media, source, or redundancy stream), the CNAME is sufficient for that purpose, independent of whether one or more RTP sessions are used. However, some applications choose not to use a CNAME because of perceived complexity or a desire not to implement RTCP and instead use the same SSRC value to bind related RTP streams across multiple RTP sessions. RTP retransmission [RFC4588], when configured to use multiple RTP sessions, and generic FEC [RFC5109] both use the CNAME method to relate the RTP streams, which may work but might have some downsides in RTP sessions with many participating SSRCs. It is not recommended to use identical SSRC values across RTP sessions to relate RTP streams; when an SSRC collision occurs, this will force a change of that SSRC in all RTP sessions and will thus resynchronize all of the streams instead of only the single media stream experiencing the collision.

Another method for implicitly binding SSRCs is used by RTP retransmission [RFC4588] when using the same RTP session as the source RTP stream for retransmissions. A receiver that is missing a packet issues an RTP retransmission request and then awaits a new SSRC carrying the RTP retransmission payload, where that SSRC is from the same CNAME. This limits a requester to having only one outstanding retransmission request on any new SSRCs per endpoint.

"RTP Payload Format Restrictions" [RFC8851] provides an RTP/RTCP-based mechanism to unambiguously identify the RTP streams within an RTP session and restrict the streams' payload format parameters in a codec-agnostic way beyond what is provided with the regular payload

types. The mapping is done by specifying an "a=rid" value in the SDP offer/answer signaling and having the corresponding RtpStreamId value as an SDES item and an RTP header extension [RFC8852]. The RID solution also includes a solution for binding redundancy RTP streams to their original source RTP streams, given that those streams use RID identifiers. The redundancy stream uses the RepairedRtpStreamId SDES item and RTP header extension to declare the RtpStreamId value of the source stream to create the binding.

Experience has shown that an explicit binding between the RTP streams, agnostic of SSRC values, behaves well. That way, solutions using multiple RTP streams in a single RTP session and in multiple RTP sessions will use the same type of binding.

3.4.4. Forward Error Correction

There exist a number of FEC-based schemes designed to mitigate packet loss in the original streams. Most of the FEC schemes protect a single source flow. This protection is achieved by transmitting a certain amount of redundant information that is encoded such that it can repair one or more instances of packet loss over the set of packets the redundant information protects. This sequence of redundant information needs to be transmitted as its own media stream or, in some cases, instead of the original media stream. Thus, many of these schemes create a need for binding related flows, as discussed above. Looking at the history of these schemes, there are schemes using multiple SSRCs and schemes using multiple RTP sessions, and some schemes that support both modes of operation.

Using multiple RTP sessions supports the case where some set of receivers might not be able to utilize the FEC information. By placing it in a separate RTP session and if separating RTP sessions at the transport level, FEC can easily be ignored at the transport level, without considering any RTP-layer information.

In usages involving multicast, sending FEC information in a separate multicast group allows for similar flexibility. This is especially useful when receivers see heterogeneous packet loss rates. A receiver can decide, based on measurement of experienced packet loss rates, whether to join a multicast group with suitable FEC data repair capabilities.

4. Considerations for RTP Multiplexing

4.1. Interworking Considerations

There are several different kinds of interworking, and this section discusses two: interworking directly between different applications and the interworking of applications through an RTP translator. The discussion includes the implications of potentially different RTP multiplexing point choices and limitations that have to be considered when working with some legacy applications.

4.1.1. Application Interworking

It is not uncommon that applications or services of similar but not identical usage, especially those intended for interactive communication, encounter a situation where one wants to interconnect two or more of these applications.

In these cases, one ends up in a situation where one might use a gateway to interconnect applications. This gateway must then either change the multiplexing structure or adhere to the respective limitations in each application.

There are two fundamental approaches to building a gateway: using RTP translator interworking (RTP bridging), where the gateway acts as an RTP translator with the two interconnected applications being members of the same RTP session; or using gateway interworking (Section 4.1.3) with RTP termination, where there are independent RTP sessions between each interconnected application and the gateway.

For interworking to be feasible, any security solution in use needs to be compatible and capable of exchanging keys with either the peer or the gateway under the trust model being used. Secondly, the applications need to use media streams in a way that makes sense in both applications.

4.1.2. RTP Translator Interworking

From an RTP perspective, the RTP translator approach could work if all the applications are using the same codecs with the same payload types, have made the same multiplexing choices, and have the same capabilities regarding the number of simultaneous RTP streams combined with the same set of RTP/RTCP extensions being supported. Unfortunately, this might not always be true.

When a gateway is implemented via an RTP translator, an important consideration is if the two applications being interconnected need to use the same approach to multiplexing. If one side is using RTP session multiplexing and the other is using SSRC multiplexing with BUNDLE [RFC8843], it may be possible for the RTP translator to map the RTP streams between both sides using some method, e.g., based on the number and order of SDP "m=" lines from each side. There are also challenges related to SSRC collision handling, since, unless SSRC translation is applied on the RTP translator, there may be a collision on the SSRC multiplexing side that the RTP session multiplexing side will not be aware of. Furthermore, if one of the applications is capable of working in several modes (such as being able to use additional RTP streams in one RTP session or multiple RTP sessions at will) and the other one is not, successful interconnection depends on locking the more flexible application into the operating mode where interconnection can be successful, even if none of the participants are using the less flexible application when the RTP sessions are being created.

4.1.3. Gateway Interworking

When one terminates RTP sessions at the gateway, there are certain tasks that the gateway has to carry out:

- Generating appropriate RTCP reports for all RTP streams (possibly based on incoming RTCP reports) originating from SSRCs controlled by the gateway.
- Handling SSRC collision resolution in each application's RTP sessions.
- Signaling, choosing, and policing appropriate bitrates for each session.

For applications that use any security mechanism, e.g., in the form of SRTP, the gateway needs to be able to decrypt and verify source integrity of the incoming packets and then re-encrypt, integrity protect, and sign the packets as the peer in the other application's security context. This is necessary even if all that's needed is a simple remapping of SSRC numbers. If this is done, the gateway also needs to be a member of the security contexts of both sides and thus a trusted entity.

The gateway might also need to apply transcoding (for incompatible codec types), media-level adaptations that cannot be solved through media negotiation (such as rescaling for incompatible video size requirements), suppression of content that is known not to be handled in the destination application, or the addition or removal of redundancy coding or scalability layers to fit the needs of the destination domain.

From the above, we can see that the gateway needs to have an intimate knowledge of the application requirements; a gateway is by its nature application specific and not a commodity product.

These gateways might therefore potentially block application evolution by blocking RTP and RTCP extensions that the applications have been extended with but that are unknown to the gateway.

If one uses a security mechanism like SRTP, the gateway and the necessary trust in it by the peers pose an additional risk to communication security. The gateway also incurs additional complexities in the form of the decrypt-encrypt cycles needed for each forwarded packet. SRTP, due to its keying structure, also requires that each RTP session need different master keys, as the use of the same key in two RTP sessions can, for some ciphers, result in a reuse of a one-time pad that completely breaks the confidentiality of the packets.

4.1.4. Legacy Considerations for Multiple SSRCs

Historically, the most common RTP use cases have been point-to-point Voice over IP (VoIP) or streaming applications, commonly with no more than one media source per endpoint and media type (typically audio or video). Even in conferencing applications, especially voice-only, the conference focus or bridge provides to each participant a single stream containing a mix of the other participants. It is also common to have individual RTP sessions between each endpoint and the RTP mixer, meaning that the mixer functions as an RTP-terminating gateway.

Applications and systems that aren't updated to handle multiple streams following these recommendations can have issues with participating in RTP sessions containing multiple SSRCs within a single session, such as:

1. The need to handle more than one stream simultaneously rather than replacing an already-existing stream with a new one.
2. Being capable of decoding multiple streams simultaneously.
3. Being capable of rendering multiple streams simultaneously.

This indicates that gateways attempting to interconnect to this class of devices have to make sure that only one RTP stream of each media type gets delivered to the endpoint if it's expecting only one and that the multiplexing format is what the device expects. It is highly unlikely that RTP translator-based interworking can be made to function successfully in such a context.

4.2. Network Considerations

The RTP implementer needs to consider that the RTP multiplexing choice also impacts network-level mechanisms.

4.2.1. Quality of Service

QoS mechanisms are either flow based or packet marking based. RSVP [RFC2205] is an example of a flow-based mechanism, while Diffserv [RFC2474] is an example of a packet-marking-based mechanism.

For a flow-based scheme, additional SSRCs will receive the same QoS as all other RTP streams being part of the same 5-tuple (protocol, source address, destination address, source port, destination port), which is the most common selector for flow-based QoS.

For a packet-marking-based scheme, the method of multiplexing will not affect the possibility of using QoS. Different Differentiated Services Code Points (DSCPs) can be assigned to different packets within a transport flow (5-tuple) as well as within an RTP stream, assuming the usage of UDP or other transport protocols that do not have issues with packet reordering within the transport flow (5-tuple). To avoid packet-reordering issues, packets belonging to the same RTP flow should limit their use of DSCPs to packets whose corresponding Per-Hop Behavior (PHB) do not enable reordering. If the transport protocol being used assumes in-order delivery of packets (e.g., TCP and the Stream Control Transmission Protocol (SCTP)), then a single DSCP should be used. For more discussion on this topic, see [RFC7657].

The method for assigning marking to packets can impact what number of RTP sessions to choose. If this marking is done using a network ingress function, it can have issues discriminating the different RTP streams. The network API on the endpoint also needs to be capable of setting the marking on a per-packet basis to reach full functionality.

4.2.2. NAT and Firewall Traversal

In today's networks, there exist a large number of middleboxes. Those that normally have the most impact on RTP are Network Address Translators (NATs) and Firewalls (FWs).

Below, we analyze and comment on the impact of requiring more underlying transport flows in the presence of NATs and FWs:

Endpoint Port Consumption:
: A given IP address only has 65536 available local ports per transport protocol for all consumers of ports that exist on the machine. This is normally never an issue for an end-user machine. It can become an issue for servers that handle a large number of simultaneous streams. However, if the application uses ICE to authenticate STUN requests, a server can

serve multiple endpoints from the same local port and use the whole 5-tuple (source and destination address, source and destination port, protocol) as the identifier of flows after having securely bound them to the remote endpoint address using the STUN request. In theory, the minimum number of media server ports needed is the maximum number of simultaneous RTP sessions a single endpoint can use. In practice, implementations will probably benefit from using more server ports to simplify implementation or avoid performance bottlenecks.

NAT State:
: If an endpoint sits behind a NAT, each flow it generates to an external address will result in a state that has to be kept in the NAT. That state is a limited resource. In home or Small Office/Home Office (SOHO) NATs, the most limited resource is memory or processing. For large-scale NATs serving many internal endpoints, available external ports are likely the scarce resource. Port limitations are primarily a problem for larger centralized NATs where endpoint-independent mapping requires each flow to use one port for the external IP address. This affects the maximum number of internal users per external IP address. However, as a comparison, a real-time video conference session with audio and video likely uses less than 10 UDP flows, compared to certain web applications that can use 100+ TCP flows to various servers from a single browser instance.

Extra Delay Added by NAT Traversal:
: Performing the NAT/FW traversal takes a certain amount of time for each flow. The best-case scenario for additional NAT/FW traversal time after finding the first valid candidate pair following the specified ICE procedures is 1.5*RTT + Ta*(Additional_Flows-1), where Ta is the pacing timer. That assumes a message in one direction, immediately followed by a return message in the opposite direction to confirm reachability. It isn't more, because ICE first finds one candidate pair that works, prior to attempting to establish multiple flows. Thus, there is no extra time until one has found a working candidate pair. Based on that working pair, the extra time is needed to establish the additional flows (two or three, in most cases) in parallel. However, packet loss causes extra delays of at least 500 ms (the minimal retransmission timer for ICE).

NAT Traversal Failure Rate:
: Due to the need to establish more than a single flow through the NAT, there is some risk that establishing the first flow will succeed but one or more of the additional flows will fail. The risk of this happening is hard to quantify but should be fairly low, as one flow from the same interfaces has just been successfully established. Thus, only such rare events as NAT resource overload, selecting particular port numbers that are filtered, etc., ought to be reasons for failure.

Deep Packet Inspection and Multiple Streams:
: FWs differ in how deeply they inspect packets. Previous experience using FWs and Session Border Gateways (SBGs) with RTP shows that there is a significant risk that the FWs and SBGs will reject RTP sessions that use multiple SSRCs.

Using additional RTP streams in the same RTP session and transport flow does not introduce any additional NAT traversal complexities per RTP stream. This can be compared with (normally) one or two additional transport flows per RTP session when using multiple RTP sessions. Additional lower-layer transport flows will be needed, unless an explicit demultiplexing layer is added between RTP and the transport protocol. At the time of this writing, no such mechanism was defined.

4.2.3. Multicast

Multicast groups provide a powerful tool for a number of real-time applications, especially those that desire broadcast-like behaviors with one endpoint transmitting to a large number of receivers, like in IPTV. An RTP/RTCP extension to better support Source-Specific Multicast (SSM) [RFC5760] is also available. Many-to-many communication, which RTP [RFC3550] was originally built to support, has several limitations in common with multicast.

One limitation is that, for any group, sender-side adaptations with the intent to suit all receivers would have to adapt to the most limited receiver experiencing the worst conditions among the group participants, which imposes degradation for all participants. For broadcast-type applications with a large number of receivers, this is not acceptable. Instead, various receiver-based solutions are employed to ensure that the receivers achieve the best possible performance. By using scalable encoding and placing each scalability layer in a different multicast group, the receiver can control the amount of traffic it receives. To have each scalability layer in a different multicast group, one RTP session per multicast group is used.

In addition, the transport flow considerations in multicast are a bit different from unicast; NATs with port translation are not useful in the multicast environment, meaning that the entire port range of each multicast address is available for distinguishing between RTP sessions.

Thus, when using broadcast applications it appears easiest and most straightforward to use multiple RTP sessions for sending different media flows used for adapting to network conditions. It is also common that streams improving transport robustness are sent in their own multicast group to allow for interworking with legacy applications or to support different levels of protection.

Many-to-many applications have different needs, and the most appropriate multiplexing choice will depend on how the actual application is realized. Multicast applications that are capable of using sender-side congestion control can avoid the use of multiple multicast sessions and RTP sessions that result from the use of receiver-side congestion control.

The properties of a broadcast application using RTP multicast are as follows:

1. The application uses a group of RTP sessions -- not just one. Each endpoint will need to be a member of a number of RTP sessions in order to perform well.
2. Within each RTP session, the number of RTP receivers is likely to be much larger than the number of RTP senders.
3. The application needs signaling functions to identify the relationships between RTP sessions.

4. The application needs signaling or RTP/RTCP functions to identify the relationships between SSRCs in different RTP sessions when more complex relations than those that can be expressed by the CNAME exist.

Both broadcast and many-to-many multicast applications share a signaling requirement; all of the participants need the same RTP and payload type configuration. Otherwise, A could, for example, be using payload type 97 as the video codec H.264 while B thinks it is MPEG-2. SDP offer/answer [RFC3264] is not appropriate for ensuring this property in a broadcast/multicast context. The signaling aspects of broadcast/multicast are not explored further in this memo.

Security solutions for this type of group communication are also challenging. First, the key-management mechanism and the security protocol need to support group communication. Second, source authentication requires special solutions. For more discussion on this topic, please review "Options for Securing RTP Sessions" [RFC7201].

4.3. Security and Key-Management Considerations

When dealing with point-to-point two-member RTP sessions only, there are few security issues that are relevant to the choice of having one RTP session or multiple RTP sessions. However, there are a few aspects of multi-party sessions that might warrant consideration. For general information regarding possible methods of securing RTP, please review [RFC7201].

4.3.1. Security Context Scope

When using SRTP [RFC3711], the security context scope is important and can be a necessary differentiation in some applications. As SRTP's crypto suites are (so far) built around symmetric keys, the receiver will need to have the same key as the sender. As a result, no one in a multi-party session can be certain that a received packet was really sent by the claimed sender and not by another party having access to the key. The single SRTP algorithm not having this property is Timed Efficient Stream Loss-Tolerant Authentication (TESLA) source authentication [RFC4383]. However, TESLA adds delay to achieve source authentication. In most cases, symmetric ciphers provide sufficient security properties, but in a few cases they can create issues.

The first case is when someone leaves a multi-party session and one wants to ensure that the party that left can no longer access the RTP streams. This requires that everyone rekey without disclosing the new keys to the excluded party.

A second case is when security is used as an enforcing mechanism for stream access differentiation between different receivers. Take, for example, a scalable layer or a high-quality simulcast version that only users paying a premium are allowed to access. The mechanism preventing a receiver from getting the high-quality stream can be based on the stream being encrypted with a key that users can't access without paying a premium, using the key-management mechanism to limit access to the key.

As specified in [RFC3711], SRTP uses unique keys per SSRC; however, the original assumption was a single-session master key from which SSRC-specific RTP and RTCP keys were derived. However, that assumption was proven incorrect, as the application usage and the developed key-management mechanisms have chosen many different methods for ensuring unique keys per

SSRC. The key-management functions have different abilities to establish different sets of keys, normally on a per-endpoint basis. For example, DTLS-SRTP [RFC5764] and Security Descriptions [RFC4568] establish different keys for outgoing and incoming traffic from an endpoint. This key usage has to be written into the cryptographic context, possibly associated with different SSRCs. Thus, limitations do exist, depending on the chosen key-management method and due to the integration of particular implementations of the key-management method and SRTP.

4.3.2. Key Management for Multi-party Sessions

The capabilities of the key-management method combined with the RTP multiplexing choices affect the resulting security properties, control over the secured media, and who has access to it.

Multi-party sessions contain at least one RTP stream from each active participant. Depending on the multi-party topology [RFC7667], each participant can both send and receive multiple RTP streams. Transport translator-based sessions (Topo-Trn-Translator) and multicast sessions (Topo-ASM) can use neither Security Descriptions [RFC4568] nor DTLS-SRTP [RFC5764] without an extension, because each endpoint provides its own set of keys. In centralized conferences, the signaling counterpart is a conference server, and the transport translator is the media-plane unicast counterpart (to which DTLS messages would be sent). Thus, an extension like Encrypted Key Transport [RFC8870] or a solution based on Multimedia Internet KEYing (MIKEY) [RFC3830] that allows for keying all session participants with the same master key is needed.

Privacy-Enhanced RTP Conferencing (PERC) also enables a different trust model with semi-trusted media-switching RTP middleboxes [RFC8871].

4.3.3. Complexity Implications

There can be complex interactions between the choice of multiplexing and topology and the security functions. This becomes especially evident in RTP topologies having any type of middlebox that processes or modifies RTP/RTCP packets. While the overhead of an RTP translator or mixer rewriting an SSRC value in the RTP packet of an unencrypted session is low, the cost is higher when using cryptographic security functions. For example, if using SRTP [RFC3711], the actual security context and exact crypto key are determined by the SSRC field value. If one changes the SSRC value, the encryption and authentication must use another key. Thus, changing the SSRC value implies a decryption using the old SSRC and its security context, followed by an encryption using the new one.

5. RTP Multiplexing Design Choices

This section discusses how some RTP multiplexing design choices can be used in applications to achieve certain goals and summarizes the implications of such choices. The benefits and downsides of each design are also discussed.

5.1. Multiple Media Types in One Session

This design uses a single RTP session for multiple different media types, like audio and video, and possibly also transport robustness mechanisms like FEC or retransmission. An endpoint can send zero, one, or multiple media sources per media type, resulting in a number of RTP streams of various media types for both source and redundancy streams.

Advantages:

1. Only a single RTP session is used, which implies:
   - Minimal need to keep NAT/FW state.
   - Minimal NAT/FW traversal cost.
   - Fate-sharing for all media flows.
   - Minimal overhead for security association establishment.
2. Dynamic allocation of RTP streams can be handled almost entirely at the RTP level. The extent to which this allocation can be kept at the RTP level depends on the application's needs for an explicit indication of stream usage and in how timely a fashion that information can be signaled.

Disadvantages:

1. It is less suitable for interworking with other applications that use individual RTP sessions per media type or multiple sessions for a single media type, due to the risk of SSRC collisions and thus a potential need for SSRC translation.
2. Negotiation of individual bandwidths for the different media types is currently only possible in SDP when using RID [RFC8851].
3. It is not suitable for split component terminals (see Section 3.10 of [RFC7667]).
4. Flow-based QoS cannot be used to provide separate treatment of RTP streams compared to others in the single RTP session.
5. If there is significant asymmetry between the RTP streams' RTCP reporting needs, there are some challenges related to configuration and usage to avoid wasting RTCP reporting on the RTP stream that does not need such frequent reporting.
6. It is not suitable for applications where some receivers like to receive only a subset of the RTP streams, especially if multicast or a transport translator is being used.
7. There are some additional concerns regarding legacy implementations that do not support the RTP specification fully when it comes to handling multiple SSRCs per endpoint, as multiple simultaneous media types are sent as separate SSRCs in the same RTP session.
8. If the applications need finer control over which session participants are included in different sets of security associations, most key-management mechanisms will have difficulties establishing such a session.

5.2. Multiple SSRCs of the Same Media Type

In this design, each RTP session serves only a single media type. The RTP session can contain multiple RTP streams, from either a single endpoint or multiple endpoints. This commonly creates a low number of RTP sessions, typically only one for audio and one for video, with a corresponding need for two listening ports when using RTP/RTCP multiplexing [RFC5761].

Advantages:

1. It works well with split component terminals (see Section 3.10 of [RFC7667]) where the split is per media type.
2. It enables flow-based QoS with different prioritization levels between media types.
3. For applications with dynamic usage of RTP streams (i.e., streams are frequently added and removed), having much of the state associated with the RTP session rather than per individual SSRC can avoid the need for in-session signaling of meta-information about each SSRC. In simple cases, this allows for unsignaled RTP streams where session-level information and an RTCP SDES item (e.g., CNAME) are sufficient. In the more complex cases where more source-specific metadata needs to be signaled, the SSRC can be associated with an intermediate identifier, e.g., the MID conveyed as an SDES item as defined in Section 15 of [RFC8843].
4. The overhead of security association establishment is low.

Disadvantages:

1. A slightly higher number of RTP sessions are needed, compared to multiple media types in one session (Section 5.1). This implies the following:
   - More NAT/FW state is needed.
   - The cost of NAT/FW traversal is increased in terms of both processing and delay.
2. There is some potential for concern regarding legacy implementations that don't support the RTP specification fully when it comes to handling multiple SSRCs per endpoint.
3. It is not possible to control security associations for sets of RTP streams within the same media type with today's key-management mechanisms, unless these are split into different RTP sessions (Section 5.3).

For RTP applications where all RTP streams of the same media type share the same usage, this structure provides efficiency gains in the amount of network state used and provides more fate-sharing with other media flows of the same type. At the same time, it still maintains almost all functionalities for the negotiation signaling of properties per individual media type and also enables flow-based QoS prioritization between media types. It handles multi-party sessions well, independently of multicast or centralized transport distribution, as additional sources can dynamically enter and leave the session.

5.3. Multiple Sessions for One Media Type

This design goes one step further than the design discussed in Section 5.2 by also using multiple RTP sessions for a single media type. The main reason for going in this direction is that the RTP application needs separation of the RTP streams according to their usage, such as, for example, scalability over multicast, simulcast, the need for extended QoS prioritization, or the need for fine-grained signaling using RTP session-focused signaling tools.

Advantages:

1. This design is more suitable for multicast usage where receivers can individually select which RTP sessions they want to participate in, assuming that each RTP session has its own multicast group.
2. When multiple different usages exist, the application can indicate its usage of the RTP streams at the RTP session level.
3. There is less need for SSRC-specific explicit signaling for each media stream and thus a reduced need for explicit and timely signaling when RTP streams are added or removed.
4. It enables detailed QoS prioritization for flow-based mechanisms.
5. It works well with split component terminals (see Section 3.10 of [RFC7667]).
6. The scope for who is included in a security association can be structured around the different RTP sessions, thus enabling such functionality with existing key-management mechanisms.

Disadvantages:

1. There is an increased amount of session configuration state compared to multiple SSRCs of the same media type (Section 5.2), due to the increased amount of RTP sessions.
2. For RTP streams that are part of scalability, simulcast, or transport robustness, a method for binding sources across multiple RTP sessions is needed.
3. There is some potential for concern regarding legacy implementations that don't support the RTP specification fully when it comes to handling multiple SSRCs per endpoint.
4. The overhead of security association establishment is higher, due to the increased number of RTP sessions.
5. If the applications need finer control over which participants in a given RTP session are included in different sets of security associations, most of today's key-management mechanisms will have difficulties establishing such a session.

For more-complex RTP applications that have several different usages for RTP streams of the same media type or that use scalability or simulcast, this solution can enable those functions, at the cost of increased overhead associated with the additional sessions. This type of structure is suitable for more-advanced applications as well as multicast-based applications requiring differentiation to different participants.

5.4. Single SSRC per Endpoint

In this design, each endpoint in a point-to-point session has only a single SSRC; thus, the RTP session contains only two SSRCs -- one local and one remote. This session can be used either unidirectionally (i.e., one SSRC sends an RTP stream that is received by the other SSRC) or bidirectionally (i.e., the two SSRCs both send an RTP stream and receive the RTP stream sent by the other endpoint). If the application needs additional media flows between the endpoints, it will have to establish additional RTP sessions.

Advantages:

1. This design has great potential for interoperability with legacy applications, as it will not tax any RTP stack implementations.
2. The signaling system makes it possible to negotiate and describe the exact formats and bitrates for each RTP stream, especially using today's tools in SDP.
3. It is possible to control security associations per RTP stream with current key-management functions, since each RTP stream is directly related to an RTP session and the most commonly used keying mechanisms operate on a per-session basis.

Disadvantages:

1. The amount of NAT/FW state grows linearly with the number of RTP streams.
2. NAT/FW traversal increases delay and resource consumption.
3. There are likely more signaling message and signaling processing requirements due to the increased amount of session-related information.
4. There is higher potential for a single RTP stream to fail during transport between the endpoints, due to the need for a separate NAT/FW traversal for every RTP stream, since there is only one stream per session.
5. The amount of explicit state for relating RTP streams grows, depending on how the application relates RTP streams.
6. Port consumption might become a problem for centralized services, where the central node's port or 5-tuple filter consumption grows rapidly with the number of sessions.
7. For applications where RTP stream usage is highly dynamic, i.e., entities frequently enter and leave sessions, the amount of signaling can become high. Issues can also arise from the need for timely establishment of additional RTP sessions.
8. If, against the recommendation in [RFC3550], the same SSRC value is reused in multiple RTP sessions rather than being randomly chosen, interworking with applications that use a different multiplexing structure will require SSRC translation.

RTP applications with a strong need to interwork with legacy RTP applications can potentially benefit from this structure. However, a large number of media descriptions in SDP can also run into issues with existing implementations. For any application needing a larger number of media flows, the overhead can become very significant. This structure is also not suitable for non-mixed multi-party sessions, as any given RTP stream from each participant, although having the same

usage in the application, needs its own RTP session. In addition, the dynamic behavior that can arise in multi-party applications can tax the signaling system and make timely media establishment more difficult.

5.5. Summary

Both the "single SSRC per endpoint" (Section 5.4) and "multiple media types in one session" (Section 5.1) cases require full explicit signaling of the media stream relationships. However, they operate on two different levels, where the first primarily enables session-level binding and the second needs SSRC-level binding. From another perspective, the two solutions are the two extremes when it comes to the number of RTP sessions needed.

The two other designs -- multiple SSRCs of the same media type (Section 5.2) and multiple sessions for one media type (Section 5.3) -- are two examples that primarily allow for some implicit mapping of the role or usage of the RTP streams based on which RTP session they appear in. Thus, they potentially allow for less signaling and, in particular, reduce the need for real-time signaling in sessions with a dynamically changing number of RTP streams. They also represent points between the first two designs when it comes to the amount of RTP sessions established, i.e., they represent an attempt to balance the amount of RTP sessions with the functionality the communication session provides at both the network level and the signaling level.

6. Guidelines

This section contains a number of multi-stream guidelines for implementers, system designers, and specification writers.

Do not require the use of the same SSRC value across RTP sessions:
: As discussed in Section 3.4.3, there are downsides to using the same SSRC in multiple RTP sessions as a mechanism to bind related RTP streams together. It is instead recommended to use a mechanism to explicitly signal the relationship, in either RTP/RTCP or the signaling mechanism used to establish the RTP session(s).

Use additional RTP streams for additional media sources:
: In the cases where an RTP endpoint needs to transmit additional RTP streams of the same media type in the application, with the same processing requirements at the network and RTP layers, it is suggested to send them in the same RTP session. For example, in the case of a telepresence room where there are three cameras and each camera captures two persons sitting at the table, we suggest that each camera send its own RTP stream within a single RTP session.

Use additional RTP sessions for streams with different requirements:
: When RTP streams have different processing requirements from the network or the RTP layer at the endpoints, it is suggested that the different types of streams be put in different RTP sessions. This includes the case where different participants want different subsets of the set of RTP streams.

Use grouping when using multiple RTP sessions:
: When using multiple RTP session solutions, it is suggested to explicitly group the involved RTP sessions when needed using a signaling mechanism -- for example, see "The Session Description Protocol (SDP) Grouping Framework" [RFC5888] -- using some appropriate grouping semantics.

Ensure that RTP/RTCP extensions support multiple RTP streams as well as multiple RTP sessions:
: When defining an RTP or RTCP extension, the creator needs to consider if this extension is applicable for use with additional SSRCs and multiple RTP sessions. Any extension intended to be generic must support both. Extensions that are not as generally applicable will have to consider whether interoperability is better served by defining a single solution or providing both options.

Provide adequate extensions for transport support:
: When defining new RTP/RTCP extensions intended for transport support, like the retransmission or FEC mechanisms, they must include support for both multiple RTP streams in the same RTP session and multiple RTP sessions, such that application developers can choose freely from the set of mechanisms without concerning themselves with which of the multiplexing choices a particular solution supports.

7. IANA Considerations

This document has no IANA actions.

8. Security Considerations

The security considerations discussed in the RTP specification [RFC3550]; any applicable RTP profile [RFC3551] [RFC4585] [RFC3711]; and the extensions for sending multiple media types in a single RTP session [RFC8860], RID [RFC8851], BUNDLE [RFC8843], [RFC5760], and [RFC5761] apply if selected and thus need to be considered in the evaluation.

Section 4.3 discusses the security implications of choosing multiple SSRCs vs. multiple RTP sessions.

9. References

9.1. Normative References

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

[RFC3551] Schulzrinne, H. and S. Casner, "RTP Profile for Audio and Video Conferences with Minimal Control", STD 65, RFC 3551, DOI 10.17487/RFC3551, July 2003, <https://www.rfc-editor.org/info/rfc3551>.

[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, <https://www.rfc-editor.org/info/rfc3711>.

[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, <https://www.rfc-editor.org/info/rfc4585>.

[RFC5576] Lennox, J., Ott, J., and T. Schierl, "Source-Specific Media Attributes in the Session Description Protocol (SDP)", RFC 5576, DOI 10.17487/RFC5576, June 2009, <https://www.rfc-editor.org/info/rfc5576>.

[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, <https://www.rfc-editor.org/info/rfc5760>.

[RFC5761] Perkins, C. and M. Westerlund, "Multiplexing RTP Data and Control Packets on a Single Port", RFC 5761, DOI 10.17487/RFC5761, April 2010, <https://www.rfc-editor.org/info/rfc5761>.

[RFC7656] Lennox, J., Gross, K., Nandakumar, S., Salgueiro, G., and B. Burman, Ed., "A Taxonomy of Semantics and Mechanisms for Real-Time Transport Protocol (RTP) Sources", RFC 7656, DOI 10.17487/RFC7656, November 2015, <https://www.rfc-editor.org/info/rfc7656>.

[RFC7667] Westerlund, M. and S. Wenger, "RTP Topologies", RFC 7667, DOI 10.17487/RFC7667, November 2015, <https://www.rfc-editor.org/info/rfc7667>.

[RFC8843] Holmberg, C., Alvestrand, H., and C. Jennings, "Negotiating Media Multiplexing Using the Session Description Protocol (SDP)", RFC 8843, DOI 10.17487/RFC8843, January 2021, <https://www.rfc-editor.org/info/rfc8843>.

[RFC8851] Roach, A.B., Ed., "RTP Payload Format Restrictions", RFC 8851, DOI 10.17487/RFC8851, January 2021, <https://www.rfc-editor.org/info/rfc8851>.

[RFC8852] Roach, A.B., Nandakumar, S., and P. Thatcher, "RTP Stream Identifier Source Description (SDES)", RFC 8852, DOI 10.17487/RFC8852, January 2021, <https://www.rfc-editor.org/info/rfc8852>.

[RFC8860] Westerlund, M., Perkins, C., and J. Lennox, "Sending Multiple Types of Media in a Single RTP Session", RFC 8860, DOI 10.17487/RFC8860, January 2021, <https://www.rfc-editor.org/info/rfc8860>.

[RFC8870] Jennings, C., Mattsson, J., McGrew, D., Wing, D., and F. Andreasen, "Encrypted Key Transport for DTLS and Secure RTP", RFC 8870, DOI 10.17487/RFC8870, January 2021, <https://www.rfc-editor.org/info/rfc8870>.

9.2. Informative References

[JINGLE] Ludwig, S., Beda, J., Saint-Andre, P., McQueen, R., Egan, S., and J. Hildebrand, "XEP-0166: Jingle", September 2018, <https://xmpp.org/extensions/xep-0166.html>.

[RFC2198] Perkins, C., Kouvelas, I., Hodson, O., Hardman, V., Handley, M., Bolot, J.C., Vega-Garcia, A., and S. Fosse-Parisis, "RTP Payload for Redundant Audio Data", RFC 2198, DOI 10.17487/RFC2198, September 1997, <https://www.rfc-editor.org/info/rfc2198>.

[RFC2205] Braden, R., Ed., Zhang, L., Berson, S., Herzog, S., and S. Jamin, "Resource ReSerVation Protocol (RSVP) -- Version 1 Functional Specification", RFC 2205, DOI 10.17487/RFC2205, September 1997, <https://www.rfc-editor.org/info/rfc2205>.

[RFC2474] Nichols, K., Blake, S., Baker, F., and D. Black, "Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers", RFC 2474, DOI 10.17487/RFC2474, December 1998, <https://www.rfc-editor.org/info/rfc2474>.

[RFC2974] Handley, M., Perkins, C., and E. Whelan, "Session Announcement Protocol", RFC 2974, DOI 10.17487/RFC2974, October 2000, <https://www.rfc-editor.org/info/rfc2974>.

[RFC3261] Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston, A., Peterson, J., Sparks, R., Handley, M., and E. Schooler, "SIP: Session Initiation Protocol", RFC 3261, DOI 10.17487/RFC3261, June 2002, <https://www.rfc-editor.org/info/rfc3261>.

[RFC3264] Rosenberg, J. and H. Schulzrinne, "An Offer/Answer Model with Session Description Protocol (SDP)", RFC 3264, DOI 10.17487/RFC3264, June 2002, <https://www.rfc-editor.org/info/rfc3264>.

[RFC3389] Zopf, R., "Real-time Transport Protocol (RTP) Payload for Comfort Noise (CN)", RFC 3389, DOI 10.17487/RFC3389, September 2002, <https://www.rfc-editor.org/info/rfc3389>.

[RFC3830] Arkko, J., Carrara, E., Lindholm, F., Naslund, M., and K. Norrman, "MIKEY: Multimedia Internet KEYing", RFC 3830, DOI 10.17487/RFC3830, August 2004, <https://www.rfc-editor.org/info/rfc3830>.

[RFC4103] Hellstrom, G. and P. Jones, "RTP Payload for Text Conversation", RFC 4103, DOI 10.17487/RFC4103, June 2005, <https://www.rfc-editor.org/info/rfc4103>.

[RFC4383] Baugher, M. and E. Carrara, "The Use of Timed Efficient Stream Loss-Tolerant Authentication (TESLA) in the Secure Real-time Transport Protocol (SRTP)", RFC 4383, DOI 10.17487/RFC4383, February 2006, <https://www.rfc-editor.org/info/rfc4383>.

[RFC4566] Handley, M., Jacobson, V., and C. Perkins, "SDP: Session Description Protocol", RFC 4566, DOI 10.17487/RFC4566, July 2006, <https://www.rfc-editor.org/info/rfc4566>.

[RFC4568] Andreasen, F., Baugher, M., and D. Wing, "Session Description Protocol (SDP) Security Descriptions for Media Streams", RFC 4568, DOI 10.17487/RFC4568, July 2006, <https://www.rfc-editor.org/info/rfc4568>.

[RFC4588] Rey, J., Leon, D., Miyazaki, A., Varsa, V., and R. Hakenberg, "RTP Retransmission Payload Format", RFC 4588, DOI 10.17487/RFC4588, July 2006, <https://www.rfc-editor.org/info/rfc4588>.

[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, <https://www.rfc-editor.org/info/rfc5104>.

[RFC5109] Li, A., Ed., "RTP Payload Format for Generic Forward Error Correction", RFC 5109, DOI 10.17487/RFC5109, December 2007, <https://www.rfc-editor.org/info/rfc5109>.

[RFC5389] Rosenberg, J., Mahy, R., Matthews, P., and D. Wing, "Session Traversal Utilities for NAT (STUN)", RFC 5389, DOI 10.17487/RFC5389, October 2008, <https://www.rfc-editor.org/info/rfc5389>.

[RFC5764] McGrew, D. and E. Rescorla, "Datagram Transport Layer Security (DTLS) Extension to Establish Keys for the Secure Real-time Transport Protocol (SRTP)", RFC 5764, DOI 10.17487/RFC5764, May 2010, <https://www.rfc-editor.org/info/rfc5764>.

[RFC5888] Camarillo, G. and H. Schulzrinne, "The Session Description Protocol (SDP) Grouping Framework", RFC 5888, DOI 10.17487/RFC5888, June 2010, <https://www.rfc-editor.org/info/rfc5888>.

[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, <https://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, <https://www.rfc-editor.org/info/rfc7201>.

[RFC7657] Black, D., Ed. and P. Jones, "Differentiated Services (Diffserv) and Real-Time Communication", RFC 7657, DOI 10.17487/RFC7657, November 2015, <https://www.rfc-editor.org/info/rfc7657>.

[RFC7826] Schulzrinne, H., Rao, A., Lanphier, R., Westerlund, M., and M. Stiemerling, Ed., "Real-Time Streaming Protocol Version 2.0", RFC 7826, DOI 10.17487/RFC7826, December 2016, <https://www.rfc-editor.org/info/rfc7826>.

[RFC7983] Petit-Huguenin, M. and G. Salgueiro, "Multiplexing Scheme Updates for Secure Real-time Transport Protocol (SRTP) Extension for Datagram Transport Layer Security (DTLS)", RFC 7983, DOI 10.17487/RFC7983, September 2016, <https://www.rfc-editor.org/info/rfc7983>.

[RFC8088] Westerlund, M., "How to Write an RTP Payload Format", RFC 8088, DOI 10.17487/RFC8088, May 2017, <https://www.rfc-editor.org/info/rfc8088>.

[RFC8108] Lennox, J., Westerlund, M., Wu, Q., and C. Perkins, "Sending Multiple RTP Streams in a Single RTP Session", RFC 8108, DOI 10.17487/RFC8108, March 2017, <https://www.rfc-editor.org/info/rfc8108>.

[RFC8445] Keranen, A., Holmberg, C., and J. Rosenberg, "Interactive Connectivity Establishment (ICE): A Protocol for Network Address Translator (NAT) Traversal", RFC 8445, DOI 10.17487/RFC8445, July 2018, <https://www.rfc-editor.org/info/rfc8445>.

[RFC8871] Jones, P., Benham, D., and C. Groves, "A Solution Framework for Private Media in Privacy-Enhanced RTP Conferencing (PERC)", RFC 8871, DOI 10.17487/RFC8871, January 2021, <https://www.rfc-editor.org/info/rfc8871>.

Appendix A. Dismissing Payload Type Multiplexing

This section documents a number of reasons why using the payload type as a multiplexing point is unsuitable for most issues related to multiple RTP streams. Attempting to use payload type multiplexing beyond its defined usage has well-known negative effects on RTP, as discussed below. To use the payload type as the single discriminator for multiple streams implies that all the different RTP streams are being sent with the same SSRC, thus using the same timestamp and sequence number space. The many effects of using payload type multiplexing are as follows:

1. Constraints are placed on the RTP timestamp rate for the multiplexed media. For example, RTP streams that use different RTP timestamp rates cannot be combined, as the timestamp values need to be consistent across all multiplexed media frames. Thus, streams are forced to use the same RTP timestamp rate. When this is not possible, payload type multiplexing cannot be used.
2. Many RTP payload formats can fragment a media object over multiple RTP packets, like parts of a video frame. These payload formats need to determine the order of the fragments to correctly decode them. Thus, it is important to ensure that all fragments related to a frame or a similar media object are transmitted in sequence and without interruptions within the object. This can be done relatively easily on the sender side by ensuring that the fragments of each RTP stream are sent in sequence.
3. Some media formats require uninterrupted sequence number space between media parts. These are media formats where any missing RTP sequence number will result in decoding failure or invoking a repair mechanism within a single media context. The text/t140 payload format [RFC4103] is an example of such a format. These formats will need a sequence numbering abstraction function between RTP and the individual RTP stream before being used with payload type multiplexing.
4. Sending multiple media streams in the same sequence number space makes it impossible to determine which media stream lost a packet. Such a scenario causes difficulties, since the receiver cannot determine to which stream it should apply packet-loss concealment or other stream-specific loss-mitigation mechanisms.

5. If RTP retransmission [RFC4588] is used and packet loss occurs, it is possible to ask for the missing packet(s) by SSRC and sequence number -- not by payload type. If only some of the payload type multiplexed streams are of interest, there is no way to tell which missing packet or packets belong to the stream or streams of interest, and all lost packets need to be requested, wasting bandwidth.
6. The current RTCP feedback mechanisms are built around providing feedback on RTP streams based on stream ID (SSRC), packet (sequence numbers), and time interval (RTP timestamps). There is almost never a field to indicate which payload type is reported, so sending feedback for a specific RTP payload type is difficult without extending existing RTCP reporting.
7. The current RTCP media control messages specification [RFC5104] is oriented around controlling particular media flows, i.e., requests are done by addressing a particular SSRC. Such mechanisms would need to be redefined to support payload type multiplexing.
8. The number of payload types is inherently limited. Accordingly, using payload type multiplexing limits the number of streams that can be multiplexed and does not scale. This limitation is exacerbated if one uses solutions like RTP and RTCP multiplexing [RFC5761] where a number of payload types are blocked due to the overlap between RTP and RTCP.
9. At times, there is a need to group multiplexed streams. This is currently possible for RTP sessions and SSRCs, but there is no defined way to group payload types.
10. It is currently not possible to signal bandwidth requirements per RTP stream when using payload type multiplexing.
11. Most existing SDP media-level attributes cannot be applied on a per-payload-type basis and would require redefinition in that context.
12. A legacy endpoint that does not understand the indication that different RTP payload types are different RTP streams might be slightly confused by the large amount of possibly overlapping or identically defined RTP payload types.

Appendix B. Signaling Considerations

Signaling is not an architectural consideration for RTP itself, so this discussion has been moved to an appendix. However, it is extremely important for anyone building complete applications, so it is deserving of discussion.

We document some issues here that need to be addressed when using some form of signaling to establish RTP sessions. These issues cannot be addressed by simply tweaking, extending, or profiling RTP; rather, they require a dedicated and in-depth look at the signaling primitives that set up the RTP sessions.

There exist various signaling solutions for establishing RTP sessions. Many are based on SDP [RFC4566]; however, SDP functionality is also dependent on the signaling protocols carrying the SDP. The Real-Time Streaming Protocol (RTSP) [RFC7826] and the Session Announcement Protocol (SAP) [RFC2974] both use SDP in a declarative fashion, while SIP [RFC3261] uses SDP

with the additional definition of offer/answer [RFC3264]. The impact on signaling, and especially on SDP, needs to be considered, as it can greatly affect how to deploy a certain multiplexing point choice.

B.1. Session-Oriented Properties

One aspect of existing signaling protocols is that they are focused on RTP sessions or, in the case of SDP, the concept of media descriptions. A number of things are signaled at the media description level, but those are not necessarily strictly bound to an RTP session and could be of interest for signaling, especially for a particular RTP stream (SSRC) within the session. The following properties have been identified as being potentially useful for signaling, and not only at the RTP session level:

- Bitrate and/or bandwidth can be specified today only as an aggregate limit, or as a common "any RTP stream" limit, unless either codec-specific bandwidth limiting or RTCP signaling using Temporary Maximum Media Stream Bit Rate Request (TMMBR) messages [RFC5104] is used.
- Which SSRC will use which RTP payload type (this information will be visible in the first media packet but is sometimes useful to have before the packet arrives).

Some of these issues are clearly SDP's problem rather than RTP limitations. However, if the aim is to deploy a solution that uses several SSRCs and contains several sets of RTP streams with different properties (encoding/packetization parameters, bitrate, etc.), putting each set in a different RTP session would directly enable negotiation of the parameters for each set. If insisting on additional SSRCs only, a number of signaling extensions are needed to clarify that there are multiple sets of RTP streams with different properties and that they in fact need to be kept different, since a single set will not satisfy the application's requirements.

For some parameters, such as RTP payload type, resolution, and frame rate, an SSRC-linked mechanism has been proposed in [RFC8851].

B.2. SDP Prevents Multiple Media Types

SDP uses the "m=" line to both delineate an RTP session and specify the top-level media type: audio, video, text, image, application. This media type is used as the top-level media type for identifying the actual payload format and is bound to a particular payload type using the "a=rtpmap:" attribute. This binding has to be loosened in order to use SDP to describe RTP sessions containing multiple top-level media types.

[RFC8843] describes how to let multiple SDP media descriptions use a single underlying transport in SDP, which allows the definition of one RTP session with different top-level media types.

B.3. Signaling RTP Stream Usage

RTP streams being transported in RTP have a particular usage in an RTP application. In many applications to date, this usage of the RTP stream is implicitly signaled. For example, an application might choose to take all incoming audio RTP streams, mix them, and play them out. However, in more-advanced applications that use multiple RTP streams, there will be more than a single usage or purpose among the set of RTP streams being sent or received. RTP applications will need to somehow signal this usage. The signaling that is used will have to identify the RTP streams affected by their RTP-level identifiers, which means that they have to be identified by either their session or their SSRC + session.

In some applications, the receiver cannot utilize the RTP stream at all before it has received the signaling message describing the RTP stream and its usage. In other applications, there exists a default handling method that is appropriate.

If all RTP streams in an RTP session are to be treated in the same way, identifying the session is enough. If SSRCs in a session are to be treated differently, signaling needs to identify both the session and the SSRC.

If this signaling affects how any RTP central node, like an RTP mixer or translator that selects, mixes, or processes streams, treats the streams, the node will also need to receive the same signaling to know how to treat RTP streams with different usages in the right fashion.

Acknowledgments

The authors would like to acknowledge and thank Cullen Jennings, Dale R. Worley, Huang Yihong (Rachel), Benjamin Kaduk, Mirja Kühlewind, and Vijay Gurbani for review and comments.

Contributors

Hui Zheng (Marvin) contributed to WG draft versions -04 and -05 of the document.

Authors' Addresses

Magnus Westerlund
Ericsson
Torshamnsgatan 23
SE-164 80 Kista
Sweden
Email: magnus.westerlund@ericsson.com

Bo Burman
Ericsson
Gronlandsgatan 31
SE-164 60 Kista
Sweden
Email: bo.burman@ericsson.com

Colin Perkins
University of Glasgow
School of Computing Science
Glasgow
G12 8QQ
United Kingdom
Email: csp@csperkins.org

Harald Tveit Alvestrand
Google
Kungsbron 2
SE-11122 Stockholm
Sweden
Email: harald@alvestrand.no

Roni Even
Email: ron.even.tlv@gmail.com