Network Working Group
Internet-Draft
Intended status: Informational
Expires: June 17, 2019

M. Westerlund
B. Burman
Ericsson
C. Perkins
University of Glasgow
H. Alvestrand
Google
R. Even
Huawei

December 14, 2018

Guidelines for using the Multiplexing Features of RTP to Support Multiple Media Streams
draft-ietf-avtcore-multiplex-guidelines-08

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.

Table of Contents

1. Introduction 3
2. Definitions 4
2.1. Terminology 4
2.2. Subjects Out of Scope 5
3. RTP Multiplexing Overview 5
3.1. Reasons for Multiplexing and Grouping RTP Streams 5
3.2. RTP Multiplexing Points 6
3.2.1. RTP Session 7
3.2.2. Synchronisation Source (SSRC) 8
3.2.3. Contributing Source (CSRC) 10
3.2.4. RTP Payload Type 10
3.3. Issues Related to RTP Topologies 11
3.4. Issues Related to RTP and RTCP Protocol 12
3.4.1. The RTP Specification 13
3.4.2. Multiple SSRCs in a Session 15
3.4.3. Binding Related Sources 15
3.4.4. Forward Error Correction 17
4. Considerations for RTP Multiplexing 17
4.1. Interworking Considerations 17
4.1.1. Application Interworking 17
4.1.2. RTP Translator Interworking 18
4.1.3. Gateway Interworking 18
4.1.4. Multiple SSRC Legacy Considerations 19
4.2. Network Considerations 20
4.2.1. Quality of Service 20
4.2.2. NAT and Firewall Traversal 21
4.2.3. Multicast 22
4.3. Security and Key Management Considerations 24
4.3.1. Security Context Scope 24
4.3.2. Key Management for Multi-party sessions 25
4.3.3. Complexity Implications 25

5. RTP Multiplexing Design Choices 26
5.1. Multiple Media Types in one Session 26
5.2. Multiple SSRCs of the Same Media Type 27
5.3. Multiple Sessions for one Media type 28
5.4. Single SSRC per Endpoint 29
5.5. Summary 31
6. Guidelines 31
7. IANA Considerations 32
8. Security Considerations 33
9. Contributors 33
10. References 33
10.1. Normative References 33
10.2. Informative References 33
Appendix A. Dismissing Payload Type Multiplexing 37
Appendix B. Signalling Considerations 39
B.1. Session Oriented Properties 40
B.2. SDP Prevents Multiple Media Types 40
B.3. Signalling RTP stream Usage 41
Authors' Addresses 41

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. RTP was from the beginning 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 enable support of multiple RTP streams and switching between different encoding or packetization of the media. By using multiple RTP sessions, sets of RTP streams can be structured for efficient processing or identification. Thus, the question for any RTP application designer is how to best use the RTP session, the RTP stream identifier (SSRC), and the RTP payload type to meet the application's needs.

There have 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) that need to be sent simultaneously. Consequently, questions are raised regarding the most appropriate RTP usage. The limitations in some implementations, RTP/RTCP extensions, and signalling has also been exposed. The authors also hope that clarification on the usefulness of some functionalities in RTP will result in more complete implementations in the future.

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 that come from a particular usage of the RTP multiplexing points. The document will recommend against some usages as being unsuitable, in general or for particular purposes.

The document starts with some definitions and then goes into the existing RTP functionalities around multiplexing. Both the desired behaviour and the implications of a particular behaviour depend on which topologies are used, which requires some consideration. This is followed by a discussion of some choices in multiplexing behaviour and their impacts. Some designs of RTP usage are 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:

Multiparty: A communication situation including multiple endpoints. In this document, it will be used to refer to situations where more than two endpoints communicate.

Multiplexing: The operation of taking 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 (de-multiplexed) 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 are 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 synchronisation contexts that can cross RTP sessions; i.e. SSRCs that map to a common CNAME can be assumed

to have RTCP SR timing information derived from a common clock such that they can be synchronised for playout.

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

Note: The above definitions of RTP Receiver and RTP sender are intended to be consistent with the usage in [RFC3550].

2.2. Subjects Out of Scope

This document is focused on issues that affect RTP. Thus, issues that involve signalling protocols, such as whether SIP, 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 signalling protocols, are mentioned only as examples in order to describe the RTP issues more precisely.

This document assumes 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 the RTP protocol.

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 below discussion, please keep in mind that the reasons for having multiple RTP streams vary and include but are not limited to the following:

- Multiple media sources
- Multiple RTP streams might be needed to represent one media source (for instance when using layered encodings)
- A retransmission stream might repeat some parts of the content of another RTP stream
- An FEC stream might provide material that can be used to repair another RTP stream
- Alternative encodings, for instance using different codecs for the same audio stream

- Alternative formats, for instance multiple resolutions of the same video stream

For each of these reasons, it is necessary to decide if each additional RTP stream is sent within the same RTP session as the other RTP streams, or if it is necessary to use additional RTP sessions to group the RTP streams. The choice suitable for one reason, might not be the choice suitable for another reason. The clearest understanding is associated with multiplexing multiple media sources of the same media type. However, all reasons warrant discussion and clarification on how to deal with them. As the discussion below will show, in reality we cannot choose a single one of SSRC or RTP session multiplexing solutions. To utilise RTP well and as efficiently as possible, both are needed. The real issue is finding the right guidance on when to create additional RTP sessions and when additional RTP streams in the same RTP session is the right choice.

3.2. RTP Multiplexing Points

This section describes the multiplexing points present in the RTP protocol that can be used to distinguish RTP streams and groups of RTP streams. Figure 1 outlines the process of demultiplexing incoming RTP streams:

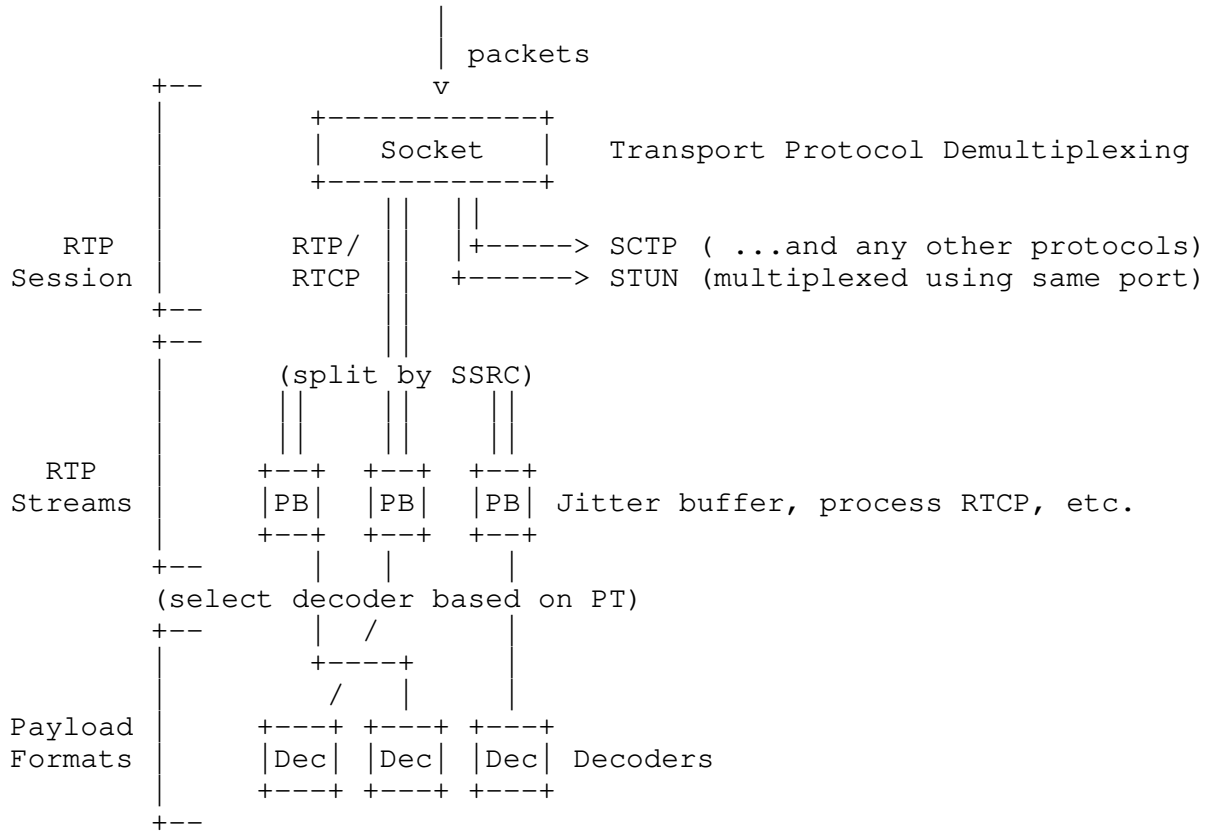

Figure 1: RTP Demultiplexing Process

3.2.1. RTP Session

An RTP Session is the highest semantic layer in the RTP protocol, and represents an association between a group of communicating endpoints. RTP does not contain a session identifier, yet RTP sessions must be possible to separate both across different endpoints and within 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 synchronisation source space [RFC3550]. That is, if a group of participants are each aware of the synchronisation source identifiers belonging to the other participants, then those participants are in a single RTP session. A participant can become aware of a synchronisation source identifier by receiving an RTP packet containing it in the SSRC field or CSRC list, by receiving an RTCP packet mentioning it in an SSRC field, or through signalling (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 signalling.

For RTP session separation within a single endpoint, RTP relies on the underlying transport layer, and on the signalling 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. The signalling 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 receiving the multiple UDP flows used as identified based on their UDP source and destination IP addresses and UDP port numbers. Another example is SDP media descriptions (the "m=" line and the following associated lines) signals the transport flow and RTP session configuration for the endpoints part of the RTP session. SDP grouping framework [RFC5888] allows labeling of the media descriptions, for example used so that RTP Session Groups can be created. With Negotiating Media Multiplexing Using the Session Description Protocol (SDP)[I-D.ietf-mmusic-sdp-bundle-negotiation], multiple media descriptions where each represents the RTP streams sent or received for a media source are part of a common RTP session.

The RTP protocol makes no normative statements about the relationship between different RTP sessions, however the applications that use more than one RTP session will have some higher layer understanding of the relationship between the sessions they create.

3.2.2. Synchronisation Source (SSRC)

A synchronisation source (SSRC) identifies an 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 are both RTP sender and RTP receiver use the same SSRC 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 signalling. Unless pre-signalled, 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 have to choose a new SSRC identifier to avoid being interpreted as looped source, unless the transport layer mechanism, e.g ICE [RFC8445], handles such changes.

SSRC identifiers that belong to the same synchronisation context (i.e., that represent RTP streams that can be synchronised using information in RTCP SR packets) use identical CNAME chunks in corresponding RTCP SDES packets. SDP signalling 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. For the RTP retransmission [RFC4588] case it is recommended to use explicit binding of the source RTP stream and the redundancy stream, e.g. using the RepairedRtpStreamId RTCP SDES item [I-D.ietf-avtext-rid].

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

Different types of entities use a 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.

Note that an endpoint that generates more than one media type, e.g. a conference participant sending both audio and video, need not (and should not) use the same SSRC value across RTP sessions. RTCP compound packets containing the CNAME SDES item is the designated method to bind 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 synchronisation context and can be synchronised 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 SSRC number due to an SSRC conflict. 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, so the new SSRC is anyway effectively a new source.

3.2.3. Contributing Source (CSRC)

The Contributing Source (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, if the corresponding SSRC was in the header of one of the packets that contributed to the mix.

It is not possible, in general, to extract media represented by an individual CSRC since it is typically the result of a media mixing (merge) operation by an RTP mixer on the individual media streams corresponding to the CSRC identifiers. The exception is the case when only a single CSRC is indicated as this represent forwarding of an RTP stream, possibly modified. The 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, 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 utilises 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 used 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 encoding format. The format definition can be taken from [RFC3551] for statically allocated payload types, but ought to be explicitly defined in signalling, such as SDP, both for static and dynamic payload types. The term "format" here includes whatever can be described by out-of-band signalling 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 time instant in the RTP stream's timestamp time line, effectively time-multiplexing different payload types if any change occurs. The payload type used 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 than the ones mentioned above need to be met to enable this use, one of which is that the combined payload sizes of the different payload types ought not exceed the transport MTU. If it is acceptable to send multiple formats of the same media source as separate RTP streams (with separate SSRC), simulcast [I-D.ietf-mmusic-sdp-simulcast] can be used.

Other aspects of RTP payload format use 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 signalling; this is not recommended since a specific payload type value can be used in multiple bundled "m=" sections [I-D.ietf-mmusic-sdp-bundle-negotiation]. This association is only possible if unique RTP payload type numbers are used in each context.

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, described by RTP Topology [RFC7667].

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

- Does my communication peer support RTP as defined with multiple SSRCs per RTP session?
- Do I need network differentiation in form of QoS?
- 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 multi-point topologies (e.g. Topo-ASM and Topo-SSM in [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 can end up in a situation when 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., RTP or other.
- Usage of different transport protocols, e.g., UDP, DCCP, or TCP.
- Different security solutions, e.g., IPsec, TLS, DTLS, or SRTP with different keying mechanisms.

In many situations this is resolved by the inclusion of a translator between the two peers, as described by Topo-PtP-Translator in [RFC7667]. The translator's main purpose is to make the peers look compatible to each other. There can also be other reasons than compatibility to insert a translator in the form of a middlebox or gateway, for example a need to monitor the RTP streams. If the stream transport characteristics are changed by the translator, appropriate media handling can require thorough understanding of the application logic, specifically any congestion control or media adaptation.

The point to point topology can contain one to many RTP sessions with one to many media sources per session, each having one or more RTP streams per media source.

3.4. Issues Related to RTP and RTCP Protocol

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 needed.

3.4.1. The RTP Specification

RFC 3550 contains some recommendations and a bullet list with 5 arguments for different aspects of RTP multiplexing. Let's review Section 5.2 of [RFC3550], reproduced below:

"For efficient protocol processing, the number of multiplexing points should be minimised, as described in the integrated layer processing design principle [ALF]. In RTP, multiplexing is provided by the destination transport address (network address and port number) which is different for each RTP session. For example, in a teleconference composed of audio and video media encoded separately, each medium SHOULD be carried in a separate RTP session with its own destination transport address.

Separate audio and video streams SHOULD NOT be carried in a single RTP session and demultiplexed based on the payload type or SSRC fields. Interleaving packets with different RTP media types but using the same SSRC would introduce several problems:

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.

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.

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.

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

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.

Using a different SSRC for each medium but sending them in the same RTP session would avoid the first three problems but not the last two.

On the other hand, multiplexing multiple related sources of the same medium in one RTP session using different SSRC values is the norm for multicast sessions. The problems listed above don't apply: an RTP mixer can combine multiple audio sources, for example, and the same treatment is applicable for all of them. It might also be appropriate to multiplex streams of the same medium using different SSRC values in other scenarios where the last two problems do not apply."

Let's consider one argument at a time. The first argument is for using different SSRC for each individual RTP stream, which is fundamental to RTP operation.

The second argument is advocating against demultiplexing RTP streams within a session based on their RTP payload type numbers, which still stands as can been seen by the extensive list of issues found in Appendix A.

The third argument is yet another argument against payload type multiplexing.

The fourth argument is against multiplexing RTP packets that require different handling into the same session. As we saw in the discussion of RTP mixers, the RTP mixer must embed application logic to handle streams anyway; 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; most other types of applications need application-specific logic to perform the mix correctly.

The fifth argument discusses network aspects that we will discuss more below in Section 4.2. It also goes into aspects of implementation, like Split Component Terminal (see Section 3.10 of [RFC7667]) endpoints where different processes or inter-connected devices handle different aspects of the whole multi-media session.

A summary of RFC 3550's view on multiplexing is to use unique SSRCs for anything that is its own media/packet stream, and to 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. Multiple Media Types in an RTP Session specification [I-D.ietf-avtcore-multi-media-rtp-session]

provides a detailed analysis of the potential issues in having multiple media types in the same RTP session. This document 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 resolving some unclear aspects of the RTP specification. These could potentially lead to some interoperability issues as well as some potential significant inefficiencies, as further discussed in "RTP Considerations for Endpoints Sending Multiple Media Streams" [RFC8108]. An RTP application designer should consider these issues and the possible application impact from 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
- Signalling based (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 signalling solutions are:

SDP Media Description Grouping: The SDP Grouping Framework [RFC5888] uses various semantics to group any number of media descriptions. These has previously been considered primarily as grouping RTP sessions, [I-D.ietf-mmusic-sdp-bundle-negotiation] groups multiple media descriptions as a single RTP session.

SDP SSRC grouping: Source-Specific Media Attributes in SDP [RFC5576]

includes a solution for grouping SSRCs the same way as the Grouping framework groups Media Descriptions.

This supports a lot of use cases. All these solutions have shortcomings in cases where the session's dynamic properties are such that it is difficult or resource consuming to keep the list of related SSRCs up to date.

An RTP/RTCP-based solution is to use the RTCP SDES CNAME to bind the RTP streams to an endpoint or synchronization context. For applications with a single RTP stream per type (Media, Source or Redundancy) this is sufficient independent if one or more RTP sessions are used. However, some applications choose not to use it 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] in multiple RTP session mode and Generic FEC [RFC5109] both use this method. This method may work but might have some downsides in RTP sessions with many participating SSRCs. When an SSRC collision occurs, this will force one to change SSRC in all RTP sessions and thus resynchronize all of them instead of only the single media stream having the collision. Therefore, it is not recommended to use identical SSRC values to relate RTP streams.

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

RTP Payload Format Restrictions [I-D.ietf-mmusic-rid] 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 signalling and having the corresponding "rtp-stream-id" value as an SDES item and an RTP header extension. The RID solution also includes a solution for binding redundancy RTP streams to their original source RTP streams, given that those use RID identifiers.

It can be noted that Section 8.3 of the RTP Specification [RFC3550] recommends using a single SSRC space across all RTP sessions for layered coding. Based on the experience so far however, we recommend to use a solution doing explicit binding between the RTP streams so what the used SSRC values are do not matter. That way solutions

using multiple RTP streams in a single RTP session and multiple RTP sessions uses the same solution.

3.4.4. Forward Error Correction

There exist a number of Forward Error Correction (FEC) based schemes for how to reduce the packet loss of the original streams. Most of the FEC schemes will protect a single source flow. The protection is achieved by transmitting a certain amount of redundant information that is encoded such that it can repair one or more packet losses over the set of packets the redundant information protects. This sequence of redundant information also 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 utilise the FEC information. By placing it in a separate RTP session and if separating RTP sessions on transport level, FEC can easily be ignored already on transport level.

In usages involving multicast, having the FEC information on its own multicast group allows for similar flexibility. This is especially useful when receivers see very heterogeneous packet loss rates. Those receivers that are not seeing packet loss don’t need to join the multicast group with the FEC data, and so avoid the overhead of receiving unnecessary FEC packets, for example.

4. Considerations for RTP Multiplexing

4.1. Interworking Considerations

There are several different kinds of interworking, and this section discusses two; interworking between different applications including 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 the ones intended for interactive communication, encounter a situation where one want 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 an RTP Translator interworking (RTP bridging), where the gateway acts as an RTP Translator, with the two applications being members of the same RTP session; or Gateway Interworking with RTP termination, where there are independent RTP sessions running from each interconnected application to the gateway.

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 in 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, it is possible for the RTP translator to map the RTP streams between both sides if the order of SDP "m=" lines between both sides are the same. There are also challenges with SSRC collision handling since there may be a collision on the SSRC multiplexing side but the RTP session multiplexing side will not be aware of any collision unless SSRC translation is applied on the RTP translator. 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 no 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.

- Signalling, choosing and policing appropriate bit-rates for each session.

For applications that uses any security mechanism, e.g., in the form of SRTP, the gateway needs to be able to decrypt incoming packets and re-encrypt them 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, of course.

Other tasks a gateway might need to apply include 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, not a commodity product.

This fact reveals the potential for these gateways to 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 security functions, like SRTP, and as can be seen from above, they incur both additional risk due to the requirement to have the gateway in the security association between the endpoints (unless the gateway is on the transport level), and additional complexities in form of the decrypt-encrypt cycles needed for each forwarded packet. SRTP, due to its keying structure, also requires that each RTP session needs different master keys, as use of the same key in two RTP sessions can for some ciphers result in two-time pads that completely breaks the confidentiality of the packets.

4.1.4. Multiple SSRC Legacy Considerations

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 has provided a single stream with a mix of the other participants to each participant. 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.

When establishing RTP sessions that can contain endpoints that aren't updated to handle multiple streams following these recommendations, a particular application can have issues with multiple SSRCs within a single session. These issues include:

1. Need to handle more than one stream simultaneously rather than replacing an already existing stream with a new one.

2. Be capable of decoding multiple streams simultaneously.

3. Be capable of rendering multiple streams simultaneously.

This indicates that gateways attempting to interconnect to this class of devices has to make sure that only one RTP stream of each 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 multiplexing choice has impact on network level mechanisms that need to be considered by the implementer.

4.2.1. Quality of Service

When it comes to Quality of Service mechanisms, they are either flow based or packet marking based. RSVP [RFC2205] is an example of a flow based mechanism, while Diff-Serv [RFC2474] is an example of a packet marking based one. For a packet marking based scheme, the method of multiplexing will not affect the possibility to use QoS.

However, for a flow based scheme there is a clear difference between the multiplexing methods. Additional SSRC will result in all 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.

It must also be noted that packet marking based QoS mechanisms can have limitations. A general observation is that different Differentiated Services Code Points (DSCP) can be assigned to different packets within a flow as well as within an RTP stream. However, care must be taken when considering which forwarding behaviours that are applied on path due to these DSCPs. In some

cases the forwarding behaviour can result in packet reordering. For more discussion of this 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 the full functionality.

4.2.2. NAT and Firewall Traversal

In today's network there exist a large number of middleboxes. The ones that normally have most impact on RTP are Network Address Translators (NAT) and Firewalls (FW).

Below we analyse and comment on the impact of requiring more underlying transport flows in the presence of NATs and Firewalls:

End-Point Port Consumption: A given IP address only has 65536 available local ports per transport protocol for 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 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 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 are the maximum number of simultaneous RTP Sessions a single endpoint can use. In practice, implementation 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, memory or processing are usually the most limited resources. For large scale NATs serving many internal endpoints, available external ports are likely the scarce resource. Port limitations is primarily a problem for larger centralised 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, it is worth pointing out that a real-time video conference session with audio and video is likely using less than

10 UDP flows, compared to certain web applications that can use 100+ TCP flows to various servers from a single browser instance.

NAT Traversal Extra Delay: Performing the NAT/FW traversal takes a certain amount of time for each flow. It also takes time in a phase of communication between accepting to communicate and the media path being established which is fairly critical. The best case scenario for how much extra time it takes after finding the first valid candidate pair following the specified ICE procedures are: 1.5*RTT + Ta*(Additional_Flows-1), where Ta is the pacing timer. That assumes a message in one direction, and then an immediate triggered check back. The reason it isn't more, is that 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 needed extra time is to in parallel establish the, in most cases 2-3, additional flows. However, packet loss causes extra delays, at least 100 ms, which is 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 succeeds but that one or more of the additional flows fail. The risk that this happens is hard to quantify, but ought to be fairly low as one flow from the same interfaces has just been successfully established. Thus only rare events such as NAT resource overload, or selecting particular port numbers that are filtered etc., ought to be reasons for failure.

Deep Packet Inspection and Multiple Streams: Firewalls differ in how deeply they inspect packets. There exist some potential that deeply inspecting firewalls will have similar legacy issues with multiple SSRCs as some stack implementations.

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 de-multiplexing layer is added between RTP and the transport protocol. At time of writing no such mechanism was defined.

4.2.3. Multicast

Multicast groups provides a powerful tool for a number of real-time applications, especially the ones that desire broadcast-like behaviours with one endpoint transmitting to a large number of receivers, like in IPTV. There are also the RTP/RTCP extension to

better support Source Specific Multicast (SSM) [RFC5760]. Another application is the Many to Many communication, which RTP [RFC3550] was originally built to support, but the multicast semantics do result in a certain number of limitations.

One limitation is that for any group, sender side adaptation to the actual receiver properties causes degradation for all participants to what is supported by the receiver with the worst conditions among the group participants. For broadcast type of applications this is not acceptable. Instead, various receiver-based solutions are employed to ensure that the receivers achieve 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 on 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 that improve transport robustness are sent in their own multicast group to allow for interworking with legacy or to support different levels of protection.

For many to many applications there are different needs. Here, the most appropriate choice will depend on how the actual application is realized. With sender side congestion control there might not exist any benefit with using multiple RTP sessions.

The properties of a broadcast application using RTP multicast:

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

2. Within each RTP session, the number of RTP receivers is likely to be much larger than the number of RTP senders.

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

4. The applications need signalling or RTP/RTCP functions to identify the relationships between SSRCs in different RTP sessions when needs beyond CNAME exist.

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

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

4.3. Security and Key Management Considerations

When dealing with point-to-point, 2-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 multiparty sessions that might warrant consideration. For general information of possible methods of securing RTP, please review RTP Security Options [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. This results in that no one in a multi-party session can be certain that a received packet really was sent by the claimed sender and not by another party having access to the key. At least unless TESLA source authentication [RFC4383], which adds delay to achieve source authentication. In most cases symmetric ciphers provide sufficient security properties, but there are a few cases where this does 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 re-keys without disclosing the keys to the excluded party.

A second case is when using security as an enforcing mechanism for differentiation. Take for example a scalable layer or a high quality

simulcast version that only premium users 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 user can't access without paying premium, having the key-management limit access to the key.

SRTP [RFC3711] has no special functions for dealing with different sets of master keys for different SSRCs. The key-management functions have different capabilities 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.

4.3.2. Key Management for Multi-party sessions

Performing key-management for multi-party sessions can be a challenge. This section considers some of the issues.

Multi-party sessions, such as transport translator based sessions and multicast sessions, can neither use Security Description [RFC4568] nor DTLS-SRTP [RFC5764] without an extension as each endpoint provides its set of keys. In centralised conferences, the signalling counterpart is a conference server and the media plane unicast counterpart (to which DTLS messages would be sent) is the transport translator. Thus, an extension like Encrypted Key Transport [I-D.ietf-perc-srtp-ekt-diet] or a MIKEY [RFC3830] based solution that allows for keying all session participants with the same master key is needed.

4.3.3. Complexity Implications

The usage of security functions can surface complexity implications from the choice of multiplexing and topology. This becomes especially evident in RTP topologies having any type of middlebox that processes or modifies RTP/RTCP packets. Where there is very small overhead for an RTP translator or mixer to rewrite an SSRC value in the RTP packet of an unencrypted session, 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 SSRC, 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 a summary of the implications of such choices. For each design there is discussion of benefits and downsides.

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 have zero, one or more media sources per media type, resulting in a number of RTP streams of various media types and both source and redundancy type.

The Pros:

1. Single RTP session which implies:

   * Minimal NAT/FW state.

   * Minimal NAT/FW Traversal Cost.

   * Fate-sharing for all media flows.

2. Can handle dynamic allocations of RTP streams well on an RTP level. Depends on the application's needs for explicit indication of the stream usage and how timely that can be signalled.

3. Minimal overhead for security association establishment.

The Cons:

a. Less suitable for interworking with other applications that uses individual RTP sessions per media type or multiple sessions for a single media type, due to the potential need of SSRC translation.

b. Negotiation of bandwidth for the different media types is currently only possible using RID [I-D.ietf-mmusic-rid] in SDP.

c. Not suitable for Split Component Terminal (see Section 3.10 of [RFC7667]).

d. Flow-based QoS cannot provide separate treatment of RTP streams compared to others in the single RTP session.

e. If there is significant asymmetry between the RTP streams' RTCP reporting needs, there are some challenges in configuration and usage to avoid wasting RTCP reporting on the RTP stream that does not need that frequent reporting.

f. Not suitable for applications where some receivers like to receive only a subset of the RTP streams, especially if multicast or transport translator is being used.

g. Additional concern with legacy implementations that do not support the RTP specification fully when it comes to handling multiple SSRC per endpoint, as also multiple simultaneous media types need to be handled.

h. If the applications need finer control over which session participants that are included in different sets of security associations, most key-management 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, either from a single endpoint or from 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.

The Pros:

1. Works well with Split Component Terminal (see Section 3.10 of [RFC7667]) where the split is per media type.

2. Enables Flow-based QoS with different prioritisation between media types.

3. For applications with dynamic usage of RTP streams, i.e. 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 signalling of meta-information about each SSRC.

4. Low overhead for security association establishment.

The Cons:

a. Slightly higher number of RTP sessions needed compared to Multiple Media Types in one Session Section 5.1. This implies:

* More NAT/FW state

* Increased NAT/FW Traversal Cost in both processing and delay.

b. Some potential for concern with legacy implementations that don't support the RTP specification fully when it comes to handling multiple SSRC per endpoint.

c. Not possible to control security association for sets of RTP streams within the same media type with today's key- management mechanisms, unless these are split into different RTP sessions.

For RTP applications where all RTP streams of the same media type share same usage, this structure provides efficiency gains in amount of network state used and provides more fate sharing with other media flows of the same type. At the same time, it is still maintaining almost all functionalities when it comes to negotiation in the signalling of the properties for the individual media type, and also enables flow based QoS prioritisation between media types. It handles multi-party session well, independently of multicast or centralised 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 above (Section 5.2) by using multiple RTP sessions also for a single media type. The main reason for going in this direction is that the RTP application needs separation of the RTP streams due to their usage. Some typical reasons for going to this design are scalability over multicast, simulcast, need for extended QoS prioritisation of RTP streams due to their usage in the application, or the need for fine- grained signalling using today's tools.

The Pros:

1. More suitable for multicast usage where receivers can individually select which RTP sessions they want to participate in, assuming each RTP session has its own multicast group.

2. The application can indicate its usage of the RTP streams on RTP session level, in case multiple different usages exist.

3. Less need for SSRC specific explicit signalling for each media stream and thus reduced need for explicit and timely signalling.

4. Enables detailed QoS prioritisation for flow-based mechanisms.

5. Works well with Split Component Terminal (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.

The Cons:

a. Increases the amount of RTP sessions compared to Multiple SSRCs of the Same Media Type.

b. Increased amount of session configuration state.

c. For RTP streams that are part of scalability, simulcast or transport robustness, a method to bind sources across multiple RTP sessions is needed.

d. Some potential for concern with legacy implementations that does not support the RTP specification fully when it comes to handling multiple SSRC per endpoint.

e. Higher overhead for security association establishment due to the increased number of RTP sessions.

f. If the applications need finer control than on RTP session level over which participants that are included in different sets of security associations, most of today's key-management 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 uses 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 both unidirectional, i.e. only a single RTP stream or bi-directional, i.e. both endpoints have one RTP stream each. If the application needs additional media flows between the endpoints, they will have to establish additional RTP sessions.

The Pros:

1. This design has great legacy interoperability potential as it will not tax any RTP stack implementations.

2. The signalling has good possibilities to negotiate and describe the exact formats and bit-rates for each RTP stream, especially using today's tools in SDP.

3. It is possible to control security association per RTP stream with current key-management, since each RTP stream is directly related to an RTP session, and the most used keying mechanisms operates on a per-session basis.

The Cons:

a. The number of RTP sessions grows directly in proportion with the number of RTP streams, which has the implications:

   * Linear growth of the amount of NAT/FW state with number of RTP streams.

   * Increased delay and resource consumption from NAT/FW traversal.

   * Likely larger signalling message and signalling processing requirement due to the amount of session related information.

   * Higher potential for a single RTP stream to fail during transport between the endpoints.

b. When the number of RTP sessions grows, the amount of explicit state for relating RTP streams also grows, depending on how the application needs to relate RTP streams.

c. The port consumption might become a problem for centralised services, where the central node's port or 5-tuple filter consumption grows rapidly with the number of sessions.

d. For applications where the RTP stream usage is highly dynamic, i.e. entering and leaving, the amount of signalling can grow high. Issues can also arise from the timely establishment of additional RTP sessions.

e. If, against the recommendation, 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 that 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 multi-party sessions, as any given RTP stream from each participant, although having same usage in the application, needs its own RTP session. In addition, the dynamic behaviour that can arise in multi-party applications can tax the signalling system and make timely media establishment more difficult.

5.5. Summary

There are some clear similarities between these designs. Both the "Single SSRC per Endpoint" and the "Multiple Media Types in one Session" are cases that require full explicit signalling of the media stream relations. 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 extreme points when it comes to number of RTP sessions needed.

The two other designs "Multiple SSRCs of the Same Media Type" and "Multiple Sessions for one Media Type" are two examples that primarily allows for some implicit mapping of the role or usage of the RTP streams based on which RTP session they appear in. It thus potentially allows for less signalling and in particular reduces the need for real-time signalling in dynamic sessions. They also represent points in between the first two designs when it comes to amount of RTP sessions established, i.e. representing an attempt to balance the amount of RTP sessions with the functionality the communication session provides both on network level and on signalling level.

6. Guidelines

This section contains a number of multi-stream guidelines for implementers or specification writers.

Do not require use of the same SSRC value across RTP sessions:
: As discussed in Section 3.4.3 there exist drawbacks in 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 relation, either in RTP/RTCP or in the signalling 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 a telepresence room where there are three cameras, and each camera captures 2 persons sitting at the table, sending each camera as its own RTP stream within a single RTP session is suggested.

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 are put in different RTP sessions. This includes the case where different participants want different subsets of the set of RTP streams.

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

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 to 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 if interoperability is better served by defining a single solution or providing both options.

Transport Support Extensions: 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 sessions 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 makes no request of IANA.

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

8. Security Considerations

The security considerations of the RTP specification [RFC3550] and any applicable RTP profile [RFC3551],[RFC4585],[RFC3711], the extensions for sending multiple media types in a single RTP session [I-D.ietf-avtcore-multi-media-rtp-session], RID [I-D.ietf-mmusic-rid], BUNDLE [I-D.ietf-mmusic-sdp-bundle-negotiation], [RFC5760], [RFC5761], apply if selected and thus needs to be considered in the evaluation.

There is discussion of the security implications of choosing multiple SSRC vs multiple RTP sessions in Section 4.3.

9. Contributors

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

10. References

10.1. 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>.

[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>.

10.2. Informative References

[ALF] Clark, D. and D. Tennenhouse, "Architectural Considerations for a New Generation of Protocols", SIGCOMM Symposium on Communications Architectures and Protocols (Philadelphia, Pennsylvania), pp. 200--208, IEEE Computer Communications Review, Vol. 20(4), September 1990.

[I-D.ietf-avtcore-multi-media-rtp-session] Westerlund, M., Perkins, C., and J. Lennox, "Sending Multiple Types of Media in a Single RTP Session", draft-ietf-avtcore-multi-media-rtp-session-13 (work in progress), December 2015.

[I-D.ietf-avtext-rid]
Roach, A., Nandakumar, S., and P. Thatcher, "RTP Stream Identifier Source Description (SDES)", draft-ietf-avtext-rid-09 (work in progress), October 2016.

[I-D.ietf-mmusic-rid]
Roach, A., "RTP Payload Format Restrictions", draft-ietf-mmusic-rid-15 (work in progress), May 2018.

[I-D.ietf-mmusic-sdp-bundle-negotiation]
Holmberg, C., Alvestrand, H., and C. Jennings, "Negotiating Media Multiplexing Using the Session Description Protocol (SDP)", draft-ietf-mmusic-sdp-bundle-negotiation-53 (work in progress), September 2018.

[I-D.ietf-mmusic-sdp-simulcast]
Burman, B., Westerlund, M., Nandakumar, S., and M. Zanaty, "Using Simulcast in SDP and RTP Sessions", draft-ietf-mmusic-sdp-simulcast-13 (work in progress), June 2018.

[I-D.ietf-perc-srtp-ekt-diet]
Jennings, C., Mattsson, J., McGrew, D., Wing, D., and F. Andreasen, "Encrypted Key Transport for DTLS and Secure RTP", draft-ietf-perc-srtp-ekt-diet-09 (work in progress), October 2018.

[RFC2198] Perkins, C., Kouvelas, I., Hodson, O., Hardman, V., Handley, M., Bolot, J., Vega-Garcia, A., and S. Fosse-Parisis, "RTP Payload for Redundant Audio Data", RFC 2198, 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>.

[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>.

[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>.

[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>.

[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>.

[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>.

[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>.

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

[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>.

[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>.

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 things related to multiple RTP streams. If one attempts to use Payload type multiplexing beyond its defined usage, that has well known negative effects on RTP. To use 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. This has many effects:

1. Putting restraint on 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 relatively simple be solved 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 streams in the same sequence number space makes it impossible to determine which payload type, which stream a packet loss relates to, and thus to which stream to potentially apply packet loss concealment or other stream-specific loss mitigation mechanisms.

5. If RTP Retransmission [RFC4588] is used and there is a loss, 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 of telling which missing packet(s) belong to the interesting stream(s) and all lost packets need 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 [RFC5104] specification is oriented around controlling particular media flows, i.e. requests are done addressing a particular SSRC. Such mechanisms would need to be redefined to support payload type multiplexing.

8. The number of payload types are 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 and this is currently possible for RTP sessions and for SSRC, 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 level and would require re-definition 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. Signalling Considerations

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

The issues raised here need to be addressed in the WGs that deal with signalling; they cannot be addressed by tweaking, extending or profiling RTP.

There exist various signalling solutions for establishing RTP sessions. Many are SDP [RFC4566] based, however SDP functionality is also dependent on the signalling protocols carrying the SDP. RTSP [RFC7826] and 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 signalling and especially 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 the existing signalling is that it is focused around RTP sessions, or at least in the case of SDP the media description. There are a number of things that are signalled on media description level but those are not necessarily strictly bound to an RTP session and could be of interest to signal specifically for a particular RTP stream (SSRC) within the session. The following properties have been identified as being potentially useful to signal not only on RTP session level:

- Bitrate/Bandwidth exist today only at aggregate or as a common "any RTP stream" limit, unless either codec-specific bandwidth limiting or RTCP signalling using TMMBR is used.

- Which SSRC that will use which RTP payload types (this will be visible from the first media packet, but is sometimes useful to know before packet arrival).

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

For some parameters, such as RTP payload type, resolution and framerate, a SSRC-linked mechanism has been proposed in [I-D.ietf-mmusic-rid]

B.2. SDP Prevents Multiple Media Types

SDP chose to use the m= line both to delineate an RTP session and to specify the top level of the MIME 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 rtpmap attribute. This binding has to be loosened in order to use SDP to describe RTP sessions containing multiple MIME top level types.

[I-D.ietf-mmusic-sdp-bundle-negotiation] describes how to let multiple SDP media descriptions use a single underlying transport in SDP, which allows to define one RTP session with media types having different MIME top level types.

B.3. Signalling RTP stream Usage

RTP streams being transported in RTP has some particular usage in an RTP application. This usage of the RTP stream is in many applications so far implicitly signalled. 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 signal this usage somehow. The signalling used will have to identify the RTP streams affected by their RTP-level identifiers, which means that they have to be identified either by their session or by their SSRC + session.

In some applications, the receiver cannot utilise the RTP stream at all before it has received the signalling message describing the RTP stream and its usage. In other applications, there exists a default handling 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, signalling needs to identify both the session and the SSRC.

If this signalling 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 signalling to know how to treat RTP streams with different usage in the right fashion.

Authors' Addresses

Magnus Westerlund
Ericsson
Torshamsgatan 23
SE-164 80 Kista
Sweden

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

Bo Burman
Ericsson
Gronlandsgatan 31
SE-164 60 Kista
Sweden

Phone: +46 10 714 13 11
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
Stockholm 11122
Sweden

Email: harald@alvestrand.no

Roni Even
Huawei

Email: roni.even@huawei.com