Network Working Group
Internet-Draft
Intended status: Informational
Expires: April 11, 2015

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

October 08, 2014

Guidelines for using the Multiplexing Features of RTP to Support Multiple Media Streams
draft-ietf-avtcore-multiplex-guidelines-03

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

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 has several multiplexing points designed for different purposes. These enable support of multiple media streams and switching between different encoding or packetization of the media. By using multiple RTP sessions, sets of media 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 SSRC and the payload type to meet the application's needs.

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.

RTP was from the beginning designed for multiple participants in a communication session. This is not restricted to multicast, as some believe, but also provides functionality over unicast, using either multiple transport flows below RTP or a network node that re-distributes the RTP packets. The re-distributing node can for example be a transport translator (relay) that forwards the packets unchanged, a translator performing media or protocol translation in addition to forwarding, or an RTP mixer that creates new sources from the received streams. In addition, multiple streams can occur when a single endpoint have multiple media sources, like multiple cameras or microphones that need to be sent simultaneously.

This document has been written due to increased interest in more advanced usage of RTP, resulting in questions regarding the most appropriate RTP usage. The limitations in some implementations, RTP/RTCP extensions, and signalling has also been exposed. It is expected that some limitations will be addressed by updates or new extensions resolving the shortcomings. The authors also hope that clarification on the usefulness of some functionalities in RTP will result in more complete implementations in the future.

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 archetypes of RTP usage are discussed. Finally, some recommendations and examples are provided.

2. Definitions

2.1. Terminology

The following terms and abbreviations are used in this document:

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

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

Media Source: The source of a stream of data of one Media Type, It can either be a single media capturing device such as a video camera, a microphone, or a specific output of a media production function, such as an audio mixer, or some video editing function.

Sending data from a Media Source can cause multiple RTP sources to send multiple Media Streams.

Media Stream: A sequence of RTP packets using a single SSRC that together carries part or all of the content of a specific Media Type from a specific sender source within a given RTP session.

RTP Source: The originator or source of a particular Media Stream. Identified using an SSRC in a particular RTP session. An RTP source is the source of a single media stream, and is associated with a single endpoint and a single Media Source. An RTP Source is just called a Source in RFC 3550.

RTP Sink: A recipient of a Media Stream. The Media Sink is identified using one or more SSRCs. There can be more than one RTP Sink for one RTP source.

CNAME: "Canonical name" - identifier associated with one or more RTP sources from a single endpoint. Defined in the RTP specification [RFC3550]. A CNAME identifies a synchronisation context. A CNAME is associated with a single endpoint, although some RTP nodes will use an endpoint's CNAME on that endpoints behalf. An endpoint can use multiple CNAMEs. A CNAME is intended to be globally unique and stable for the full duration of a communication session. [RFC6222][I-D.ietf-avtcore-6222bis] gives updated guidelines for choosing CNAMEs.

Media Type: Audio, video, text or data whose form and meaning are defined by a specific real-time application.

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 Session: As defined by [RFC3550], the endpoints belonging to the same RTP Session are those that share a single SSRC space. That is, those endpoints can see an SSRC identifier transmitted by any one of the other endpoints. An endpoint can receive an SSRC either as SSRC or as CSRC in RTP and RTCP packets. Thus, the RTP Session scope is decided by the endpoints' network interconnection topology, in combination with RTP and RTCP forwarding strategies deployed by endpoints and any interconnecting middle nodes.

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.

Source: Term that ought not be used alone. An RTP Source, as identified by its SSRC, is the source of a single Media Stream; a Media Source can be the source of mutiple Media Streams.

SSRC: A 32-bit unsigned integer used as identifier for a RTP Source.

CSRC: Contributing Source, A SSRC identifier used in a context, like the RTP headers CSRC list, where it is clear that the Media Source is not the source of the media stream, instead only a contributor to the Media Stream.

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

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 such 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. Reasons for Multiplexing and Grouping RTP Media Streams

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

- Multiple Media Sources
- Multiple Media Streams might be needed to represent one Media Source (for instance when using layered encodings)
- A Retransmission stream might repeat the content of another Media Stream

- An FEC stream might provide material that can be used to repair another Media Stream

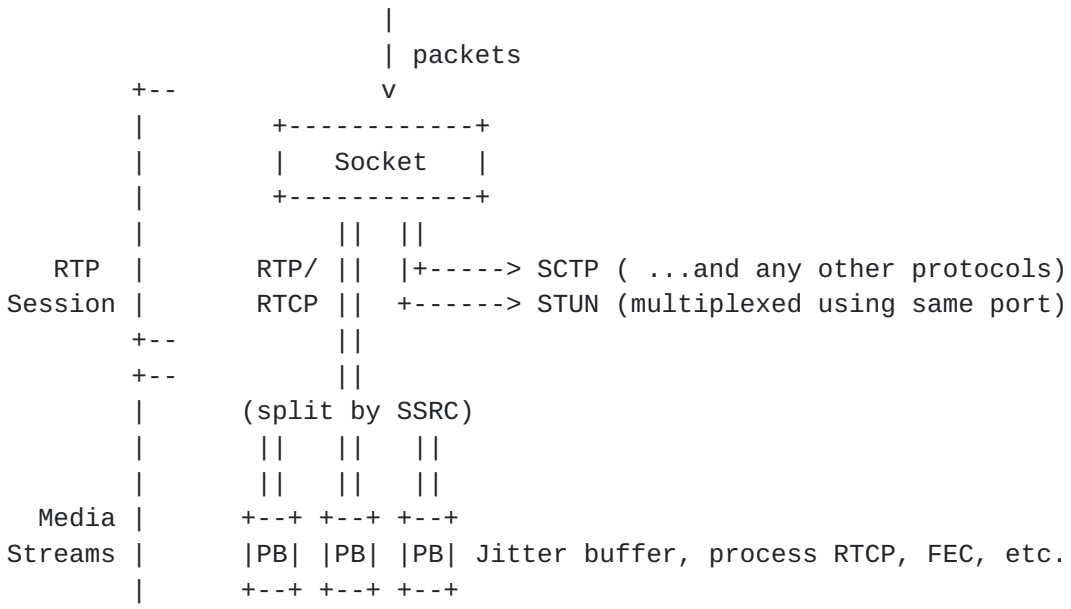
- Alternative Encodings, for instance different codecs for the same audio stream

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

For each of these, it is necessary to decide if each additional media stream gets its own SSRC multiplexed within a RTP Session, or if it is necessary to use additional RTP sessions to group the media streams. The choice between these made due to one reason might not be the choice suitable for another reason. In the above list, the different items have different levels of maturity in the discussion on how to solve them. The clearest understanding is associated with multiple media sources of the same media type. However, all 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 the two 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 RTP sessions and when additional SSRCs in an RTP session is the right choice.

4. RTP Multiplexing Points

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



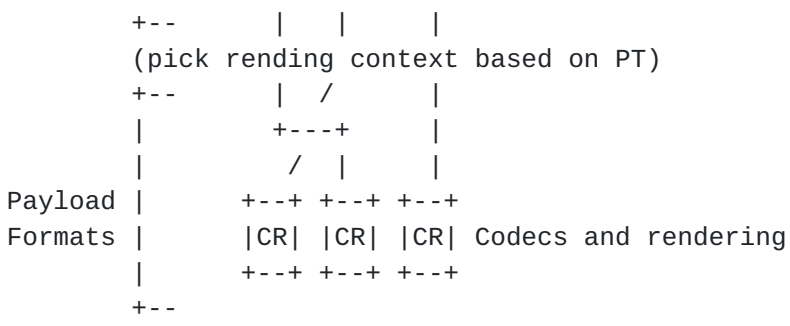


Figure 1: RTP Demultiplexing Process

4.1. RTP Session

An RTP Session is the highest semantic layer in the RTP protocol, and represents an association between a group of communicating endpoints. 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 SDP "a=ssrc:" attribute). 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.

RTP does not contain a session identifier. Rather, it relies on the underlying transport layer to separate different sessions, and on the signalling to identify sessions in a manner that is meaningful to the application. The signalling layer might give sessions an explicit identifier, or their 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 RTP endpoint can identify and delimit an RTP Session from other RTP Sessions using the UDP source and destination IP addresses and UDP port numbers. Another example is when using SDP grouping framework [RFC5888] which uses an identifier per "m="-line; if there is a one-to-one mapping between "m="-lines and RTP sessions, that grouping framework identifier will identify an RTP Session.

RTP sessions are globally unique, but their identity can only be determined by the communication context at an endpoint of the session, or by a middlebox that is aware of the session context. The relationship between RTP sessions depending on the underlying

application, transport, and signalling protocol. 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.

4.2. Synchronisation Source (SSRC)

A synchronisation source (SSRC) identifies an RTP source or an RTP sink. Every endpoint will have at least one synchronisation source identifier, even if it does not send media (endpoints that are only RTP sinks still send RTCP, and use their synchronisation source identifier in the RTCP packets they send). An endpoint can have multiple synchronisation sources identifiers if it contains multiple RTP sources (i.e., if it sends multiple media streams). Endpoints that are both RTP sources and RTP sinks use the same synchronisation sources in both roles. At any given time, a RTP source has one and only one SSRC - although that can change over the lifetime of the RTP source or sink.

The synchronisation Source identifier is a 32-bit unsigned integer. 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 using the SDP "a=ssrc:" attribute [RFC5576], the synchronisation source identifier 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. Synchronisation source identifier collisions can occur, and are handled as specified in [RFC3550] and [RFC5576], resulting in the synchronisation source identifier of the affecting RTP sources and/or sinks changing. An RTP source that changes its RTP Session identifier (e.g. source transport address) during a session has to choose a new SSRC identifier to avoid being interpreted as looped source.

Synchronisation source identifiers that belong to the same synchronisation context (i.e., that represent media streams that can be synchronised using information in RTCP SR packets) are indicated by use of identical CNAME chunks in corresponding RTCP SDES packets. SDP signalling can also be used to provide explicit grouping of synchronisation sources [RFC5576].

> In some cases, the same SSRC Identifier value is used to relate streams in two different RTP Sessions, such as in Multi-Session Transmission of scalable video [RFC6190]. This is NOT RECOMMENDED since there is no guarantee of uniqueness in SSRC values across RTP sessions.

Note that RTP sequence number and RTP timestamp are scoped by the synchronisation source. Each RTP source will have a different synchronisation source, and the corresponding media stream will have a separate RTP sequence number and timestamp space.

An SSRC identifier is used by different type of sources as well as sinks:

Real Media Source: Connected to a "physical" media source, for example a camera or microphone.

Processed Media Source: A source with some attributed property generated by some 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.

RTP Sink: A source that does not generate any RTP media stream in itself (e.g. an endpoint or middlebox only receiving in an RTP session). It still needs a sender SSRC for use as source in RTCP reports.

Note that a endpoint that generates more than one media type, e.g. a conference participant sending both audio and video, need not (and commonly does 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.

4.3. Contributing Source (CSRC)

The Contributing Source (CSRC) is not a separate identifier. Rather a synchronisation source 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 a media stream, possibly modified. The RTP header extension for Mixer-to-Client Audio Level Indication [RFC6465] expands on the receivers 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.

4.4. RTP Payload Type

Each Media 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 field in the RTP data packet header. The combination therefore identifies a specific Media 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 payload type is scoped by sending endpoint within an RTP Session. All synchronisation sources sent from an single endpoint share the same payload types definitions. The RTP Payload Type is designed such that only a single Payload Type is valid at any time instant in the RTP source's RTP 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.

Other aspects of RTP payload format use are described in RTP Payload HowTo [I-D.ietf-payload-rtp-howto].

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 render a media stream, and so can be viewed as selecting a rendering context. The rendering context can be defined by the signalling, and the RTP payload type number is sometimes used to associate an RTP media stream with the signalling. This association is possible provided unique RTP payload type numbers are used in each context. For example, an RTP media stream can be associated with an SDP "m=" line by comparing the RTP payload type numbers used by the media stream with payload types signalled in the "a=rtpmap:" lines in the media sections of the SDP. If RTP media streams are being associated with signalling contexts based on the RTP payload type, then the assignment of RTP payload type numbers MUST be unique across signalling contexts; if the same RTP payload format configuration is used in multiple contexts, then a different RTP payload type number has to be assigned in each context to ensure uniqueness. If the RTP payload type number is not being used to associated RTP media streams with a signalling context, then the same RTP payload type number can be used to indicate the exact same RTP payload format configuration in multiple contexts.

5. RTP Topologies and Issues

The impact of how RTP multiplexing is performed will in general vary with how the RTP Session participants are interconnected, described by RTP Topology [RFC5117] and its intended successor [I-D.westerlund-avtcore-rtp-topologies-update].

5.1. Point to Point

Even the most basic use case, denoted Topo-Point-to-Point in [I-D.westerlund-avtcore-rtp-topologies-update], raises a number of considerations that are discussed in detail below (Section 6). They range over such aspects as:

- Does my communication peer support RTP as defined with multiple SSRCs?
- 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 media streams for RTP retransmission or FEC.
- etc.

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 sources per media source.

5.2. Translators & Gateways

A point to point communication can end up in a situation when the peer it is communicating with is not compatible with the other peer for various reasons:

- No common media codec for a media type thus requiring transcoding
- Different support for multiple RTP sources and RTP sessions
- Usage of different media transport protocols, i.e RTP or other.
- Usage of different transport protocols, e.g. UDP, DCCP, TCP
- Different security solutions, e.g. IPsec, TLS, DTLS, 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 [I-D.westerlund-avtcore-rtp-topologies-update]. The translator's main purpose is to make the peer look to the other peer like something it is compatible with. 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 media 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.

5.3. Point to Multipoint Using Multicast

The Point to Multi-point topology is using Multicast to interconnect the session participants. This includes both Topo-ASM and Topo-SSM in [I-D.westerlund-avtcore-rtp-topologies-update].

Special considerations need to be made as multicast is a one to many distribution system. For example, the only practical method for adapting the bit-rate sent towards a given receiver for large groups is to use a set of multicast groups, where each multicast group represents a particular bit-rate. Otherwise the whole group gets media adapted to the participant with the worst conditions. The media encoding is either scalable, where multiple layers can be combined, or simulcast, where a single version is selected. By either selecting or combing multicast groups, the receiver can

control the bit-rate sent on the path to itself. 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.

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

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

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

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

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

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

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

5.4. Point to Multipoint Using an RTP Transport Translator

This mode is described as Topo-Translator in [I-D.westerlund-avtcore-rtp-topologies-update].

Transport Translators (Relays) result in an RTP session situation that is very similar to how an ASM group RTP session would behave.

One of the most important aspects with the simple relay is that it is only rewriting transport headers, no RTP modifications nor media transcoding occur. The most obvious downside of this basic relaying is that the translator has no control over how many streams need to be delivered to a receiver. Nor can it simply select to deliver only

certain streams, as this creates session inconsistencies: If the translator temporarily stops a stream, this prevents some receivers from reporting on it. From the sender's perspective it will look like a transport failure. Applications needing to stop or switch streams in the central node ought to consider using an RTP mixer to avoid this issue.

The Transport Translator has the same signalling requirement as multicast: All participants need to have the same payload type configuration. Most of the ASM security issues also arise here. Some alternatives when it comes to solution do exist, as there exists a central node to communicate with, one that also can enforce some security policies depending on the level of trust placed in the node.

5.5. Point to Multipoint Using an RTP Mixer

A mixer, described by Topo-Mixer in [I-D.westerlund-avtcore-rtp-topologies-update], is a centralised node that selects or mixes content in a conference to optimise the RTP session so that each endpoint only needs connect to one entity, the mixer. The media sent from the mixer to the endpoint can be optimised in different ways. These optimisations include methods like only choosing media from the currently most active speaker or mixing together audio so that only one audio stream is needed.

Mixers have some downsides, the first is that the mixer has to be a trusted node as they repacketize the media, and can perform media transformation operations. When using SRTP, both media operations and repacketization requires that the mixer verifies integrity, decrypts the content, performs the operation and forms new RTP packets, encrypts and integrity-protects them. This applies to all types of mixers. The second downside is that all these operations and optimisations of the session requires processing. How much depends on the implementation, as will become evident below.

A mixer, unlike a pure transport translator, is always application specific: the application logic for stream mixing or stream selection has to be embedded within the mixer, and controlled using application specific signalling. The implementation of a mixer can take several different forms, as discussed below.

A Mixer can also contain translator functionalities, like a media transcoder to adjust the media bit-rate or codec used for a particular RTP media stream.

6. RTP Multiplexing: When to Use Multiple RTP Sessions

Using multiple media streams is a well supported feature of RTP. However, it can be unclear for most implementers or people writing RTP/RTCP applications or extensions attempting to apply multiple streams when it is most appropriate to add an additional SSRC in an existing RTP session and when it is better to use multiple RTP sessions. This section tries to discuss the various considerations needed. The next section then concludes with some guidelines.

6.1. RTP and RTCP Protocol Considerations

This section discusses RTP and RTCP aspects worth considering when selecting between using an additional SSRC and Multiple RTP sessions.

6.1.1. 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 is an argument for using different SSRC for each individual media stream, which is very applicable.

The second argument is advocating against using payload type multiplexing, 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 is an argument against multiplexing media streams that require different handling into the same session. As we saw in the discussion of RTP mixers, the RTP mixer has to embed application logic in order 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. A type of application that can mix different media sources "blindly" is the audio only "telephone" bridge; most other type of application needs application-specific logic to perform the mix correctly.

The fifth argument discusses network aspects that we will discuss more below in Section 6.3. It also goes into aspects of implementation, like decomposed 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 later is one thing which is further discussed in this document as something the application developer needs to make a conscious choice for, but where imposing a single solution on all usages of RTP is inappropriate.

6.1.1.1. Different Media Types: Recommendations

The above quote from RTP [RFC3550] includes a strong recommendation:

> "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."

It was identified in "Why RTP Sessions Should Be Content Neutral" [I-D.alvestrand-rtp-sess-neutral] that the above statement is poorly supported by any of the motivations provided in the RTP specification. This has resulted in the creation of a specification Multiple Media Types in an RTP Session specification [I-D.ietf-avtcore-multi-media-rtp-session] which intends to update this recommendation. That document has a detailed analysis of the potential issues in having multiple media types in the same RTP session. This document tries to provide an moreover arching consideration regarding the usage of RTP session and considers multiple media types in one RTP session as possible choice for the RTP application designer.

6.1.2. Multiple SSRCs in a Session

Using multiple SSRCs in an RTP session at one endpoint requires resolving some unclear aspects of the RTP specification. These could potentially lead to some interoperability issues as well as some potential significant inefficencies. These are further discussed in "RTP Considerations for Endpoints Sending Multiple Media Streams" [I-D.lennox-avtcore-rtp-multi-stream]. A application designer needs to consider these issues and the impact availability or lack of the optimization in the endpoints has on their application.

If an application will become affected by the issues described, using Multiple RTP sessions can mitigate these issues.

6.1.3. Handling Varying Sets of Senders

In some applications, the set of simultaneously active sources varies within a larger set of session members. A receiver can then possibly try to use a set of decoding chains that is smaller than the number of senders, switching the decoding chains between different senders. As each media decoding chain can contain state, either the receiver needs to either be able to save the state of swapped-out senders, or the sender needs to be able to send data that permits the receiver to reinitialise when it resumes activity.

This behaviour will cause similar issues independent of Additional SSRC or Multiple RTP session.

6.1.4. Cross Session RTCP Requests

There currently exists no functionality to make truly synchronised and atomic RTCP messages with some type of request semantics across multiple RTP Sessions. Instead, separate RTCP messages will have to be sent in each session. This gives streams in the same RTP session a slight advantage as RTCP messages for different streams in the same session can be sent in a compound RTCP packet, thus providing an atomic operation if different modifications of different streams are requested at the same time.

When using multiple RTP sessions, the RTCP timing rules in the sessions and the transport aspects, such as packet loss and jitter, prevents a receiver from relying on atomic operations, forcing it to use more robust and forgiving mechanisms.

6.1.5. Binding Related Sources

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

The solutions can be divided into some groups, RTP/RTCP based, Signalling based (SDP), grouping related RTP sessions, and 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, but this might change.

SDP SSRC grouping: Source-Specific Media Attributes in SDP [RFC5576] includes a solution for grouping SSRCs the same way as the Grouping framework groupes Media Descriptions.

SDP MSID grouping: Media Stream Identifiers [I-D.ietf-mmusic-msid] includes a solution for grouping SSRCs that is independent of their allocation to RTP sessions.

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.

Within RTP/RTCP based solutions when binding to a endpoint or synchronization context, i.e. the CNAME has not be sufficient and one has multiple RTP sessions has been to using the same SSRC value across all the RTP sessions. RTP Retransmission [RFC4588] is multiple RTP session mode, Generic FEC [RFC5109], as well as the RTP payload format for Scalable Video Coding [RFC6190] in Multi Session Transmission (MST) mode uses this method. This method clearly works but might have some downside in RTP sessions with many participating SSRCs. The birthday paradox ensures that if you populate a single session with 9292 SSRCs at random, the chances are approximately 1% that at least one collision will occur. When a collision occur this will force one to change SSRC in all RTP sessions and thus resynchronizing all of them instead of only the single media stream having the collision.

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.

Another solution that has been applied to binding SSRCs has been an implicit method used by RTP Retransmission [RFC4588] when doing retransmissions in the same RTP session as the source RTP media stream. This issues an RTP retransmission request, and then await a new SSRC carrying the RTP retransmission payload and where that SSRC is from the same CNAME. This limits a requestor to having only one outstanding request on any new source SSRCs per endpoint.

There exists no RTP/RTCP based mechanism capable of supporting explicit association accross multiple RTP sessions as well within an RTP session. A proposed solution for handling this issue is [I-D.westerlund-avtext-rtcp-sdes-srcname]. If accepted, this can potentially also be part of an SDP based solution also by reusing the same identifiers and name space.

6.1.6. 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 they protect. 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, it can easily be ignored.

In usages involving multicast, having the FEC information on its own multicast group, and therefore in its own RTP session, allows for 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.

6.1.7. Transport Translator Sessions

A basic Transport Translator relays any incoming RTP and RTCP packets to the other participants. The main difference between Additional SSRCs and Multiple RTP Sessions resulting from this use case is that with Additional SSRCs it is not possible for a particular session participant to decide to receive a subset of media streams. When using separate RTP sessions for the different sets of media streams, a single participant can choose to leave one of the sessions but not the other.

6.2. Interworking Considerations

There are several different kinds of interworking, and this section discusses two related ones. The interworking between different applications and the implications of potentially different choices of usage of RTP's multiplexing points. The second topic relates to what limitations have to be considered working with some legacy applications.

6.2.1. Types of Interworking

It is not uncommon that applications or services of similar 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 then needs to change the multiplexing structure or adhere to limitations in each application.

There are two fundamental approaches to gatewaying: RTP Translator interworking (RTP bridging), where the gateway acts as an RTP Translator, and the two applications are members of the same RTP session, and Gateway Interworking (with RTP termination), where there are independent RTP sessions running from each interconnected application to the gateway.

6.2.2. 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, have the same capabilities in number of simultaneous media streams combined with the same set of RTP/RTCP extensions being supported. Unfortunately this might not always be true.

When one is gatewaying via an RTP Translator, a natural requirement is that the two applications being interconnected need to use the same approach to multiplexing. Furthermore, if one of the applications is capable of working in several modes (such as being able to use Additional SSRCs 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 using the less flexible application are present when the RTP sessions are being created.

6.2.3. 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 media 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.

If either of the applications has any security applied, 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), 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 need 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 evolution of the applications by blocking unknown RTP and RTCP extensions that the regular application has been extended with.

If one uses security functions, like SRTP, they can as seen above incur both additional risk due to the gateway needing to be in 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 result in two-time pads that completely breaks the confidentiality of the packets.

6.2.4. 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 and 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 media 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.

6.3. Network Considerations

The multiplexing choice has impact on network level mechanisms that need to be considered by the implementor.

6.3.1. Quality of Service

When it comes to Quality of Service mechanisms, they are either flow based or marking based. RSVP [RFC2205] is an example of a flow based mechanism, while Diff-Serv [RFC2474] is an example of a Marking based one. For a 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 methods. Additional SSRC will result in all media 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. Thus, separation of the level of QoS between media streams is not possible. That is however possible when

using multiple RTP sessions, where each media stream for which a separate QoS handling is desired can be in a different RTP session that can be sent over different 5-tuples.

It also needs to be noted that packet marking based QoS mechanisms can have limitations. A general observation is that different DSCP can be assigned to different packets within in a flow as well as within an RTP Media Stream. However, care needs to 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 [I-D.ietf-dart-dscp-rtp].

More specific to the choice between using one or more RTP session can be the method for assigning marking to packets. If this is done using a network ingress function, it can have issues discriminating the different RTP media 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.

6.3.2. 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 analyze 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 Excess Time: Making 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, which ICE specifies to be no smaller than 20 ms. 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.

Additional SSRC keeps the additional media streams within one RTP Session and transport flow and does not introduce any additional NAT traversal complexities per media 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. A proposal for how to multiplex multiple RTP sessions over the same single lower layer transport exist in [I-D.westerlund-avtcore-transport-multiplexing].

6.3.3. Multicast

Multicast groups provides a powerful semantics 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. But that same 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. In most cases 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 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 it appears easiest and most straightforward to use multiple RTP sessions for sending different media flows used for adapting to network conditions.

6.3.4. Multiplexing multiple RTP Session on a Single Transport

For applications that don't need flow based QoS and like to save ports and NAT/FW traversal costs and where usage of multiple media types in one RTP session is not suitable, there is a proposal for how to achieve multiplexing of multiple RTP sessions over the same lower layer transport [I-D.westerlund-avtcore-transport-multiplexing]. Using such a solution would allow Multiple RTP session without most of the perceived downsides of Multiple RTP sessions creating a need for additional transport flows, but this solution would require support from all functions that handle RTP packets, including firewalls.

6.4. 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 [I-D.ietf-avtcore-rtp-security-options].

6.4.1. 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 (so far) are 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 or by another party having access to the key. In most cases this is a sufficient security property, 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 media 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 which 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 set 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.

6.4.2. Key Management for Multi-party session

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

Multi-party sessions, such as transport translator based sessions and multicast sessions, cannot 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-avt-srtp-ekt] is needed or a MIKEY [RFC3830] based solution that allows for keying all session participants with the same master key.

6.4.3. Complexity Implications

The usage of security functions can surface complexity implications of 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 of doing it when using cryptographic security functions is higher. For example if using SRTP [RFC3711], the actual security context and exact crypto key are determined by the SSRC field value. If one changes it, the encryption and authentication tag needs to be performed using 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.

7. Archetypes

This section discusses some archetypes of how RTP multiplexing can be used in applications to achieve certain goals and a summary of their implications. For each archetype there is discussion of benefits and downsides.

7.1. Single SSRC per Session

In this archetype 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 media stream or bi-directional, i.e. both endpoints have one media stream each. If the application needs additional media flows between the endpoints, they will have to establish additional RTP sessions.

The Pros:

1. This archetype 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 media stream, especially using today's tools in SDP.

3. It does not matter if usage or purpose of the media stream is signalled on media stream level or session level as there is no difference.

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

The Cons:

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

   * Linear growth of the amount of NAT/FW state with number of media 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 media stream to fail during transport between the endpoints.

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

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

d. For applications where the media streams are highly dynamic in their usage, i.e. entering and leaving, the amount of signalling can grow high. Issues arising from the timely establishment of additional RTP sessions can also arise.

e. Cross session RTCP requests might be needed, and the fact that they're impossible can cause issues.

f. If the same SSRC value is reused in multiple RTP sessions rather than being randomly chosen, interworking with applications that uses another multiplexing structure than this application will require SSRC translation.

g. Cannot be used with Any Source Multicast (ASM) as one cannot guarantee that only two endpoints participate as packet senders. Using SSM, it is possible to restrict to these requirements if no RTCP feedback is injected back into the SSM group.

h. For most security mechanisms, each RTP session or transport flow requires individual key-management and security association establishment thus increasing the overhead.

RTP applications that need to inter-work with legacy RTP applications, like most deployed VoIP and video conferencing solutions, 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 media 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.

7.2. Multiple SSRCs of the Same Media Type

In this archetype, each RTP session serves only a single media type. The RTP session can contain multiple media 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. Low number of RTP sessions needed compared to single SSRC case. This implies:

   * Reduced NAT/FW state

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

2. Allows for early de-multiplexing in the processing chain in RTP applications where all media streams of the same type have the same usage in the application.

3. Works well with media type de-composite endpoints.

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

5. For applications with dynamic usage of media streams, i.e. they come and go frequently, having much of the state associated with the RTP session rather than an individual SSRC can avoid the need for in-session signalling of meta-information about each SSRC.

6. Low overhead for security association establishment.

The Cons:

a. May have some need for cross session RTCP requests for things that affect both media types in an asynchronous way.

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

c. Will not be able to control security association for sets of media 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 media 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 enabling 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.

7.3. Multiple Sessions for one Media type

In this archetype one goes one step further than in the above (Section 7.2) by using multiple RTP sessions also for a single media type, but still not as far as having a single SSRC per RTP session. The main reason for going in this direction is that the RTP application needs separation of the media streams due to their usage. Some typical reasons for going to this archetype are scalability over multicast, simulcast, need for extended QoS prioritisation of media 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. Indication of the application's usage of the media stream, where 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 de-composite endpoints.

6. Handles dynamic usage of media streams well.

7. For transport translator based multi-party sessions, this structure allows for improved control of which type of media streams an endpoint receives.

8. 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. May need synchronised cross-session RTCP requests and require some consideration due to this.

d. For media streams that are part of scalability, simulcast or transport robustness it will be needed to bind sources, which need to support multiple RTP sessions.

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

f. Higher overhead for security association establishment.

g. If the applications need finer control than on media type level over which session 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 media streams of the same media type and / 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.

7.4. Multiple Media Types in one Session

This archetype is to use a single RTP session for multiple different media types, like audio and video, and possibly also transport robustness mechanisms like FEC or Retransmission. Each media stream will use its own SSRC and a given SSRC value from a particular endpoint will never use the SSRC for more than a single media 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. Enables separation of the different media types based on the payload types so media type specific endpoint or central processing can still be supported despite single session.

3. Can handle dynamic allocations of media 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.

4. 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 need of SSRC translation.

b. Negotiation of bandwidth for the different media types is currently not possible in SDP. This requires SDP extensions to

enable payload or source specific bandwidth. Likely to be a problem due to media type asymmetry in needed bandwidth.

c. Not suitable for de-composite endpoints.

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

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

f. Not suitable for applications where some receivers like to receive only a subset of the media 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 needs 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.

7.5. Summary

There are some clear relations between these archetypes. Both the "single SSRC per RTP session" and the "multiple media types in one session" are cases which 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 to do it all on SSRC level. From another perspective, the two solutions are the two extreme points when it comes to number of RTP sessions needed.

The two other archetypes "Multiple SSRCs of the Same Media Type" and "Multiple Sessions for one Media Type" are examples of two other cases that first of all allows for some implicit mapping of the role or usage of the media streams based on which RTP session they appear in. It thus potentially allows for less signalling and in particular reduced need for real-time signalling in dynamic sessions. They also represent points in between the first two when it comes to amount of RTP sessions established, i.e. representing an attempt to reduce the amount of sessions as much as possible without compromising the functionality the session provides both on network level and on signalling level.

8. Summary considerations and guidelines

8.1. Guidelines

This section contains a number of recommendations for implementors or specification writers when it comes to handling multi-stream.

Do not Require the same SSRC across Sessions: As discussed in Section 6.1.5 there exist drawbacks in using the same SSRC in multiple RTP sessions as a mechanism to bind related media streams together. It is instead suggested that a mechanism to explicitly signal the relation is used, either in RTP/RTCP or in the used signalling mechanism that establishes the RTP session(s).

Use additional SSRCs additional Media Sources: In the cases where an RTP endpoint needs to transmit additional media 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 as additional SSRCs 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 SSRC within a single RTP session is suggested.

Use additional RTP sessions for streams with different requirements: When media 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 the signalling mechanism, for example The Session Description Protocol (SDP) Grouping Framework. [RFC5888], using some appropriate grouping semantics.

RTP/RTCP Extensions May Support Additional SSRCs as well as Multiple RTP sessions:

When defining an RTP or RTCP extension, the creator needs to consider if this extension is applicable to usage with additional SSRCs and Multiple RTP sessions. Any extension intended to be generic is suggested to support both. Applications 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 are expected to include support for both additional SSRCs and multiple RTP sessions so that application developers can choose freely from the set of mechanisms without concerning themselves with which of the multiplexing choices a particular solution supports.

9. IANA Considerations

This document makes no request of IANA.

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

10. Security Considerations

There is discussion of the security implications of choosing SSRC vs Multiple RTP session in Section 6.4.

11. References

11.1. Normative References

[RFC3550] Schulzrinne, H., Casner, S., Frederick, R., and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications", STD 64, RFC 3550, July 2003.

11.2. 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.alvestrand-rtp-sess-neutral]

Alvestrand, H., "Why RTP Sessions Should Be Content Neutral", draft-alvestrand-rtp-sess-neutral-01 (work in progress), June 2012.

[I-D.ietf-avt-srtp-ekt]
Wing, D., McGrew, D., and K. Fischer, "Encrypted Key Transport for Secure RTP", draft-ietf-avt-srtp-ekt-03 (work in progress), October 2011.

[I-D.ietf-avtcore-6222bis]
Begen, A., Perkins, C., Wing, D., and E. Rescorla, "Guidelines for Choosing RTP Control Protocol (RTCP) Canonical Names (CNAMEs)", draft-ietf-avtcore-6222bis-06 (work in progress), July 2013.

[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-05 (work in progress), February 2014.

[I-D.ietf-avtcore-rtp-security-options]
Westerlund, M. and C. Perkins, "Options for Securing RTP Sessions", draft-ietf-avtcore-rtp-security-options-10 (work in progress), January 2014.

[I-D.ietf-avtext-multiple-clock-rates]
Petit-Huguenin, M. and G. Zorn, "Support for Multiple Clock Rates in an RTP Session", draft-ietf-avtext-multiple-clock-rates-11 (work in progress), November 2013.

[I-D.ietf-dart-dscp-rtp]
Black, D. and P. Jones, "Differentiated Services (DiffServ) and Real-time Communication", draft-ietf-dart-dscp-rtp-07 (work in progress), September 2014.

[I-D.ietf-mmusic-msid]
Alvestrand, H., "WebRTC MediaStream Identification in the Session Description Protocol", draft-ietf-mmusic-msid-06 (work in progress), June 2014.

[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-11 (work in progress), September 2014.

[I-D.ietf-payload-rtp-howto]

Westerlund, M., "How to Write an RTP Payload Format", draft-ietf-payload-rtp-howto-13 (work in progress), January 2014.

[I-D.lennox-avtcore-rtp-multi-stream]
Lennox, J., Westerlund, M., Wu, W., and C. Perkins, "RTP Considerations for Endpoints Sending Multiple Media Streams", draft-lennox-avtcore-rtp-multi-stream-02 (work in progress), February 2013.

[I-D.lennox-mmusic-sdp-source-selection]
Lennox, J. and H. Schulzrinne, "Mechanisms for Media Source Selection in the Session Description Protocol (SDP)", draft-lennox-mmusic-sdp-source-selection-05 (work in progress), October 2012.

[I-D.westerlund-avtcore-max-ssrc]
Westerlund, M., Burman, B., and F. Jansson, "Multiple Synchronization sources (SSRC) in RTP Session Signaling", draft-westerlund-avtcore-max-ssrc-02 (work in progress), July 2012.

[I-D.westerlund-avtcore-rtp-topologies-update]
Westerlund, M. and S. Wenger, "RTP Topologies", draft-westerlund-avtcore-rtp-topologies-update-02 (work in progress), February 2013.

[I-D.westerlund-avtcore-transport-multiplexing]
Westerlund, M. and C. Perkins, "Multiplexing Multiple RTP Sessions onto a Single Lower-Layer Transport", draft-westerlund-avtcore-transport-multiplexing-07 (work in progress), October 2013.

[I-D.westerlund-avtext-rtcp-sdes-srcname]
Westerlund, M., "RTCP Source Description Item SRCNAME to Label Individual Media Sources", draft-westerlund-avtext-rtcp-sdes-srcname-03 (work in progress), October 2013.

[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, September 1997.

[RFC2205] Braden, B., Zhang, L., Berson, S., Herzog, S., and S. Jamin, "Resource ReSerVation Protocol (RSVP) -- Version 1 Functional Specification", RFC 2205, September 1997.

[RFC2326] Schulzrinne, H., Rao, A., and R. Lanphier, "Real Time Streaming Protocol (RTSP)", RFC 2326, April 1998.

[RFC2474] Nichols, K., Blake, S., Baker, F., and D.L. Black, "Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers", RFC 2474, December 1998.

[RFC2974] Handley, M., Perkins, C., and E. Whelan, "Session Announcement Protocol", RFC 2974, October 2000.

[RFC3261] Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston, A., Peterson, J., Sparks, R., Handley, M., and E. Schooler, "SIP: Session Initiation Protocol", RFC 3261, June 2002.

[RFC3264] Rosenberg, J. and H. Schulzrinne, "An Offer/Answer Model with Session Description Protocol (SDP)", RFC 3264, June 2002.

[RFC3389] Zopf, R., "Real-time Transport Protocol (RTP) Payload for Comfort Noise (CN)", RFC 3389, September 2002.

[RFC3551] Schulzrinne, H. and S. Casner, "RTP Profile for Audio and Video Conferences with Minimal Control", STD 65, RFC 3551, July 2003.

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

[RFC3830] Arkko, J., Carrara, E., Lindholm, F., Naslund, M., and K. Norrman, "MIKEY: Multimedia Internet KEYing", RFC 3830, August 2004.

[RFC4103] Hellstrom, G. and P. Jones, "RTP Payload for Text Conversation", RFC 4103, June 2005.

[RFC4566] Handley, M., Jacobson, V., and C. Perkins, "SDP: Session Description Protocol", RFC 4566, July 2006.

[RFC4568] Andreasen, F., Baugher, M., and D. Wing, "Session Description Protocol (SDP) Security Descriptions for Media Streams", RFC 4568, July 2006.

[RFC4588] Rey, J., Leon, D., Miyazaki, A., Varsa, V., and R. Hakenberg, "RTP Retransmission Payload Format", RFC 4588, July 2006.

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

[RFC5109] Li, A., "RTP Payload Format for Generic Forward Error Correction", RFC 5109, December 2007.

[RFC5117] Westerlund, M. and S. Wenger, "RTP Topologies", RFC 5117, January 2008.

[RFC5576] Lennox, J., Ott, J., and T. Schierl, "Source-Specific Media Attributes in the Session Description Protocol (SDP)", RFC 5576, June 2009.

[RFC5583] Schierl, T. and S. Wenger, "Signaling Media Decoding Dependency in the Session Description Protocol (SDP)", RFC 5583, July 2009.

[RFC5761] Perkins, C. and M. Westerlund, "Multiplexing RTP Data and Control Packets on a Single Port", RFC 5761, April 2010.

[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, May 2010.

[RFC5888] Camarillo, G. and H. Schulzrinne, "The Session Description Protocol (SDP) Grouping Framework", RFC 5888, June 2010.

[RFC6190] Wenger, S., Wang, Y.-K., Schierl, T., and A. Eleftheriadis, "RTP Payload Format for Scalable Video Coding", RFC 6190, May 2011.

[RFC6222] Begen, A., Perkins, C., and D. Wing, "Guidelines for Choosing RTP Control Protocol (RTCP) Canonical Names (CNAMEs)", RFC 6222, April 2011.

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

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

Appendix A. Dismissing Payload Type Multiplexing

This section documents a number of reasons why using the payload type as a multiplexing point for most things related to multiple streams

is unsuitable. If one attempts to use Payload type multiplexing beyond it's 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 media 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, media 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 rate. When this is not possible, Payload Type multiplexing cannot be used.

2. Many RTP payload formats can fragment a media object over multiple 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 media 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 of 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 media 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 and thus which stream a packet loss relates to.

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 media 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 media stream 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 media 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 doesn't understand the indication that different RTP payload types are different media streams might be slightly confused by the large amount of possibly overlapping or identically defined RTP Payload Types.

Appendix B. Proposals for Future Work

The above discussion and guidelines indicates that a small set of extension mechanisms could greatly improve the situation when it comes to using multiple streams independently of Multiple RTP session or Additional SSRC. These extensions are:

Media Source Identification: A Media source identification that can be used to bind together media streams that are related to the same media source. A proposal [I-D.westerlund-avtext-rtcp-sdes-srcname] exist for a new SDES item SRCNAME that also can be used with the a=ssrc SDP attribute to provide signalling layer binding information.

MSID: A Media Stream identification scheme that can be used to signal relationships between SSRCs that can be in the same or in different RTP sessions. Described in [I-D.ietf-mmusic-msid]

SSRC limitations within RTP sessions: By providing a signalling solution that allows the signalling peers to explicitly express both support and limitations on how many simultaneous media streams an endpoint can handle within a given RTP Session. That ensures that usage of Additional SSRC occurs when supported and without overloading an endpoint. This extension is proposed in [I-D.westerlund-avtcore-max-ssrc].

Appendix C. 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.

C.1. Signalling Aspects

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. Where RTSP [RFC2326] 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.

C.1.1. Session Oriented Properties

One aspect of the existing signalling is that it is focused around sessions, or at least in the case of SDP the media description. There are a number of things that are signalled on a session level/ media description but those are not necessarily strictly bound to an RTP session and could be of interest to signal specifically for a particular media 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 a common any media 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 media 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 media 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 resolution and framerate, a SSRC-linked mechanism has been proposed: [I-D.lennox-mmusic-sdp-source-selection].

C.1.2. 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 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.

There is an accepted WG item in the MMUSIC WG to define how multiple media lines describe a single underlying transport [I-D.ietf-mmusic-sdp-bundle-negotiation] and thus it becomes possible in SDP to define one RTP session with media types having different MIME top level types.

C.1.3. Signalling Media Stream Usage

Media streams being transported in RTP has some particular usage in an RTP application. This usage of the media 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 media streams there will be more than a single usage or purpose among the set of media streams being sent or received. RTP applications will need to signal this usage somehow. The signalling used will have to identify the media 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 media stream at all before it has received the signalling message describing the media stream and its usage. In other applications, there exists a default handling that is appropriate.

If all media 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 media streams with different usage in the right fashion.

Authors' Addresses

Magnus Westerlund
Ericsson
Farogatan 6
SE-164 80 Kista
Sweden

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

Bo Burman
Ericsson
Farogatan 6
SE-164 80 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