Network Working Group
Internet-Draft
Intended status: Informational
Expires: September 13, 2012

M. Westerlund
B. Burman
Ericsson
C. Perkins
University of Glasgow
March 12, 2012

RTP Multiplexing Architecture
draft-westerlund-avtcore-multiplex-architecture-01

Abstract

Real-time Transport Protocol is a flexible protocol possible to use in a wide range of applications and network and system topologies. This flexibility and the implications of different choices should be understood by any application developer using RTP. To facilitate that understanding, this document contains an in-depth discussion of the usage of RTP's multiplexing points; the RTP session, the Synchronization Source Identifier (SSRC), and the payload type. The focus is put on the first two, trying to give guidance and source material for an analysis on the most suitable choices for the application being designed.

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on September 13, 2012.

Table of Contents

1. Introduction

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 should understand the implications that come from a particular choice of 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 may 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 translation in addition to forwarding, or an RTP mixer that creates new conceptual sources from

the received streams. In addition, multiple streams may occur when a single end-point 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 behavior and the implications of a particular behavior depend on which topologies are used, which requires some consideration. This is followed by a discussion of some choices in multiplexing behavior and their impacts. Some arch-types of RTP usage are discussed.

Finally, some recommendations and examples are provided.

This document is currently an individual contribution, but it is the intention of the authors that this should become a WG document that objectively describes and provides suitable recommendations for which there is WG consensus. Currently this document only represents the views of the authors. The authors gladly accept any feedback on the document and will be happy to discuss suitable recommendations.

2. Definitions

2.1. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

2.2. Terminology

The following terms and abbreviations are used in this document:

End-point: A single entity sending or receiving RTP packets. It may be decomposed into several functional blocks, but as long as it behaves a single RTP stack entity it is classified as a single end-point.

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.

Media Source: The originator or source of a particular Media Stream. 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.

Media Aggregate: All Media Streams related to a particular Source.

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

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

Source: See Media Source.

3. RTP Multiplex Points

This section describes the existing RTP tools that enable multiplexing of different media streams.

3.1. Session

The RTP Session is the highest semantic level in RTP and contains all of the RTP functionality.

Identifier: RTP in itself does not contain any Session identifier, but relies either on the underlying transport or on the used signalling protocol, depending on in which context the identifier is used (e.g. transport or signalling). Due to this, a single RTP Session may have multiple associated identifiers belonging to different contexts.

Position: Depending on underlying transport and signalling protocol. For example, when running RTP on top of UDP, an RTP endpoint can identify and delimit an RTP Session from other RTP Sessions through the UDP source and destination transport address, consisting of network address and port number(s). Commonly, RTP and RTCP use separate ports and the destination transport address is in fact an address pair, but in the case of RTP/RTCP multiplex [RFC5761] there is only a single port. Another example is SDP signalling [RFC4566], where the grouping framework [RFC5888] uses an identifier per "m="-line. If there is a one-to-one mapping between "m="-line and RTP Session, that grouping framework identifier can identify a single RTP Session.

Usage: Identify separate RTP Sessions.

Uniqueness: Globally unique within the general communication context for the specific end-point.

Inter-relation: Depending on the underlying transport and signalling protocol.

Special Restrictions: None.

A source that changes its source transport address during a session must also choose a new SSRC identifier to avoid being interpreted as a looped source.

The set of participants considered part of the same RTP Session is defined by[RFC3550] as those that share a single SSRC space. That is, those participants that can see an SSRC identifier transmitted by any one of the other participants. A participant can receive an SSRC either as SSRC or CSRC in RTP and RTCP packets. Thus, the RTP Session scope is decided by the participants' network interconnection topology, in combination with RTP and RTCP forwarding strategies deployed by end-points and any interconnecting middle nodes.

3.2. SSRC

An RTP Session serves one or more Media Sources, each sending a Media Stream.

Identifier: Synchronization Source (SSRC), 32-bit unsigned number.

Position: In every RTP and RTCP packet header. May be present in RTCP payload. May be present in SDP signalling.

Usage: Identify individual Media Sources within an RTP Session. Refer to individual Media Sources in RTCP messages and SDP signalling.

Uniqueness: Randomly chosen, globally unique within an RTP Session and not dependent on network address.

Inter-relation: SSRC belonging to the same synchronization context (originating from the same end-point), within or between RTP Sessions, are indicated through use of identical SDES CNAME items in RTCP compound packets with those SSRC as originating source. SDP signalling can provide explicit SSRC grouping [RFC5576]. When CNAME is inappropriate or insufficient, there exist a few other methods to relate different SSRC. One such case is session-based RTP retransmission [RFC4588]. 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].

Special Restrictions: All RTP implementations must be prepared to use procedures for SSRC collision handling, which results in an SSRC number change. A Media Source that changes its RTP Session identifier (e.g. source transport address) during a session must also choose a new SSRC identifier to avoid being interpreted as looped source. Note that RTP sequence number and RTP timestamp are scoped by SSRC and thus independent between different SSRCs.

A media source having an SSRC identifier can be of different types:

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

Conceptual: 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.

Virtual: A source that does not generate any RTP media stream in itself (e.g. an end-point only receiving in an RTP session), but anyway need a sender SSRC for use as source in RTCP reports.

Note that a "multimedia source" that generates more than one media type, e.g. a conference participant sending both audio and video, need not (and commonly should not) use the same SSRC value across RTP sessions. RTCP Compound packets containing the CNAME SDES item is the designated method to bind an SSRC to a CNAME, effectively cross-correlating SSRCs within and between RTP Sessions as coming from the same end-point. The main property attributed to SSRCs associated with the same CNAME is that they are from a particular synchronization context and may be synchronized at playback.

Note also that RTP sequence number and RTP timestamp are scoped by SSRC and thus independent between different SSRCs.

An RTP receiver receiving a previously unseen SSRC value must interpret it as a new source. It may 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 should 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.

Some RTP extension mechanisms already require the RTP stacks to handle additional SSRCs, like SSRC multiplexed RTP retransmission

[RFC4588]. However, that still only requires handling a single media decoding chain per pair of SSRCs.

3.3. CSRC

The Contributing Source (CSRC) can arguably be seen as a sub-part of a specific SSRC and thus a multiplexing point. It is optionally included in the RTP header, shares the SSRC number space and specifies which set of SSRCs that has contributed to the RTP payload. However, even though each RTP packet and SSRC can be tagged with the contained CSRCs, the media representation of an individual CSRC is in general not possible to extract from the RTP payload 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. Due to these restrictions, CSRC will not be considered a fully qualified multiplex point and will be disregarded in the rest of this document.

3.4. Payload Type

Each Media Stream can be represented in various encoding formats.

Identifier: Payload Type number.

Position: In every RTP header and in SDP signalling.

Usage: Identify a specific Media Stream encoding format. The format definition may be taken from [RFC3551] for statically allocated Payload Types, but should 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 signaling means. In SDP, the term "format" includes media type, RTP timestamp sampling rate, codec, codec configuration, payload format configurations, and various robustness mechanisms such as redundant encodings [RFC2198].

Uniqueness: Scoped by sending end-point within an RTP Session. To avoid any potential for ambiguity, it is desirable that payload types are unique across all sending end-points within an RTP session, but this is often not true in practice. All SSRC in an RTP session sent from an single end-point 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 SSRC's RTP timestamp time line, effectively time-multiplexing different Payload Types if any change occurs.

Used Payload Type may change on a per-packet basis for an SSRC, for example a speech codec making use of generic Comfort Noise

[RFC3389].

Inter-relation: There are some uses where Payload Type numbers need be unique across RTP Sessions. This is for example the case in Media Decoding Dependency [RFC5583] where Payload Types are used to describe media dependency across RTP Sessions. Another example is session-based RTP retransmission [RFC4588].

Special Restrictions: Using different RTP timestamp clock rates for the RTP Payload Types in use in the same RTP Session have issues such as loss of synchronization. Payload Type clock rate switching requires some special consideration that is described in the multiple clock rates specification [I-D.ietf-avtext-multiple-clock-rates].

If there is a true need to send multiple Payload Types for the same SSRC that are valid for the same RTP Timestamps, 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 must not exceed the transport MTU.

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

4. Multiple Streams Alternatives

This section reviews the alternatives to enable multi-stream handling. Let's start with describing mechanisms that could enable multiple media streams, independent of the purpose for having multiple streams.

SSRC Multiplexing: Each additional Media Stream gets its own SSRC within a RTP Session.

Session Multiplexing: Using additional RTP Sessions to handle additional Media Streams

Payload Type Multiplexing: Using different RTP payload types for different additional streams.

Independent of the reason to use additional media streams, achieving it using payload type multiplexing is not a good choice as can be seen in the Appendix A. The RTP payload type alone is not suitable for cases where additional media streams are required. Streams need their own SSRCs, so that they get their own sequence number space. The SSRC itself is also important so that the media stream can be

referenced and reported on.

This leaves us with two main choices, either using SSRC multiplexing to have multiple SSRCs from one end-point in one RTP session, or create an additional RTP session to hold that additional SSRC. As the below discussion will show, in reality we cannot choose a single one of the two solutions. To utilize 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.

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
- Retransmission streams
- FEC stream
- Alternative Encodings
- Scalability layers

Thus the choice made due to one reason may 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.

5. RTP Topologies and Issues

The impact of how RTP Multiplex is performed will in general vary with how the RTP Session participants are interconnected; the RTP Topology [RFC5117]. This section describes the topologies and attempts to highlight the important behaviors concerning RTP multiplexing and multi-stream handling. It lists any identified issues regarding RTP and RTCP handling, and introduces additional topologies that are supported by RTP beyond those included in RTP Topologies [RFC5117]. The RTP Topologies that do not follow the RTP specification or do not attempt to utilize the facilities of RTP are ignored in this document.

5.1. Point to Point

This is the most basic use case with an RTP session containing two end-points. Each end-point has one or more SSRCs.

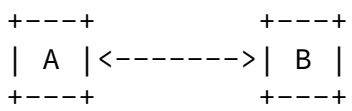


Figure 1: Point to Point

5.1.1. RTCP Reporting

In cases when an end-point uses multiple SSRCs, we have found two closely related issues. The first is if every SSRC shall report on all other SSRC, even the ones originating from the same end-point. The reason for this would be to ensure that no monitoring function should suspect a breakage in the RTP session.

The second issue around RTCP reporting arise when an end-point receives one or more media streams, and when the receiving end-point itself sends multiple SSRC in the same RTP session. As transport statistics are gathered per end-point and shared between the nodes, all the end-point's SSRC will report based on the same received data, the only difference will be which SSRCs sends the report. This could

be considered unnecessary overhead, but for consistency it might be simplest to always have all sending SSRCs send RTCP reports on all media streams the end-point receives.

The current RTP text is silent about sending RTCP Receiver Reports for an endpoint's own sources, but does not preclude either sending or omitting them. The uncertainty in the expected behavior in those cases has likely caused variations in the implementation strategy. This could cause an interoperability issue where it is not possible to determine if the lack of reports is a true transport issue, or simply a result of implementation.

Although this issue is valid already for the simple point to point case, it needs to be considered in all topologies. From the perspective of an end-point, any solution needs to take into account what a particular end-point can determine without explicit information of the topology. For example, a Transport Translator (Relay) topology will look quite similar to point to point on a transport level but is different on RTP level. Assume a first scenario with two SSRC being sent from an end-point to a Transport Translator, and a second scenario with two single SSRC remote end-points sending to the same Transport Translator. The main differences between those two scenarios are that in the second

scenario, the RTT may vary between the SSRCs (but it is not guaranteed), and the SSRCs may also have different CNAMEs.

5.1.2. Compound RTCP Packets

When an end-point has multiple SSRCs and it needs to send RTCP packets on behalf of these SSRCs, the question arises if and how RTCP packets with different source SSRCs can be sent in the same compound packet. If it is allowed, then some consideration of the transmission scheduling is needed.

5.2. Point to Multipoint Using Multicast

This section discusses the Point to Multi-point using Multicast to interconnect the session participants. This needs to consider both Any Source Multicast (ASM) and Source-Specific Multicast (SSM). There are large commercial deployments of multicast for applications like IPTV.

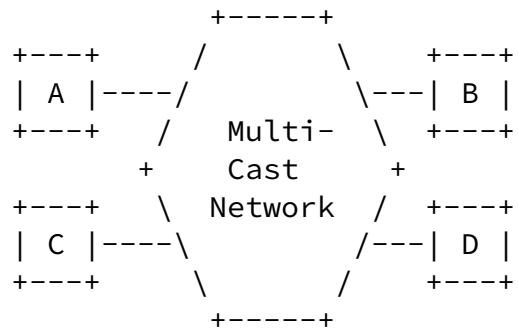
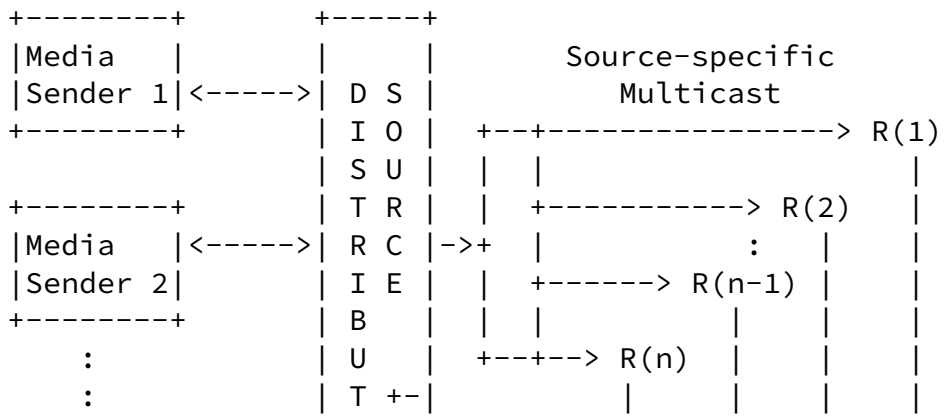
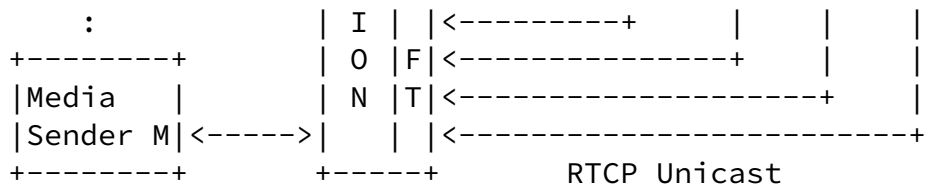


Figure 2: Point to Multipoint Using Any Source Multicast

In Any Source Multicast, any of the participants can send to all the other participants, simply by sending a packet to the multicast group. That is not possible in Source Specific Multicast [RFC4607] where only a single source (Distribution Source) can send to the multicast group, creating a topology that looks like the one below:





FT = Feedback Target
Transport from the Feedback Target to the Distribution Source is via unicast or multicast RTCP if they are not co-located.

Figure 3: Point to Multipoint using Source Specific Multicast

In this topology a number of Media Senders (1 to M) are allowed to send media to the SSM group, sends media to the distribution source which then forwards the media streams to the multicast group. The media streams reach the Receivers (R(1) to R(n)). The Receiver's RTCP cannot be sent to the multicast group. To support RTCP, an RTP extension for SSM [RFC5760] was defined to use unicast transmission to send RTCP from the receivers to one or more Feedback Targets (FT).

As multicast is a one to many distribution system, this must be taken into consideration. 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 is sent in its own multicast group to allow for interworking with legacy or to support different levels of protection.

The result of this is three common behaviors for RTP multicast:

1. Use of multiple RTP sessions for the same media type.

2. The need for identifying RTP sessions that are related in one of several possible ways.

3. The need for binding related SSRCs in different RTP sessions together.

This indicates that Multicast is an important consideration when working with the RTP multiplexing and multi stream architecture questions. It is also important to note that so far there is no special mode for basic behavior between multicast and unicast usages of RTP. Yes, there are extensions targeted to deal with multicast specific cases, but the general applicability does need to be considered.

5.3. Point to Multipoint Using an RTP Translator

Transport Translators (Relays) are a very important consideration for this document as they result in an RTP session situation that is very similar to how an ASM group RTP session would behave.

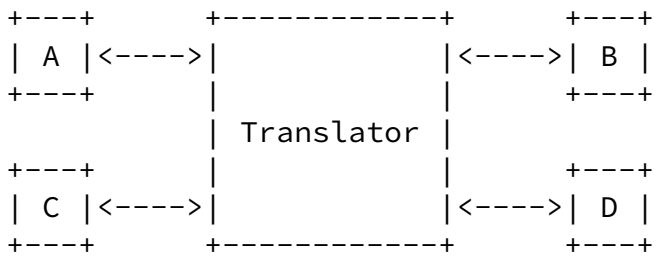


Figure 4: Transport Translator (Relay)

One of the most important aspects with the simple relay is that it is both easy to implement and require minimal amount of resources as only transport headers are rewritten, no RTP modifications nor media transcoding occur. Thus it is most likely the cheapest and most generally deployable method for multi-point sessions. The most obvious downside of this basic relaying is that the translator has no control over how many streams needs to be delivered to a receiver. Nor can it simply select to deliver only certain streams, as it creates session inconsistencies. If some middlebox temporarily stops a stream, this prevents some receivers from reporting on it. From the senders perspective it will look like a transport failure. Applications having needs to stop or switch streams in the central node should consider using an RTP mixer to avoid this issue.

The Transport Translator does not need to have an SSRC of itself, nor need it send any RTCP reports on the flows that pass it, but it may

choose to do that.

Use of a transport translator results in that any of the end-points will receive multiple SSRCs over a single unicast transport flow from the translator. That is independent of the other end-points having only a single or several SSRCs. End-points that have multiple SSRCs put further requirements on how SSRCs can be related or bound within and across RTP sessions and how they can be identified on an application level. The transport translator has a signalling requirement that also exist in any source multicast; 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 should be noted that SDP offer/answer [RFC3264] has issues with ensuring this property.

A Media Translator can perform a large variety of media functions affecting the media stream passing the translator, coming from one source and destined to a particular end-point. The translator can transcode to a different bit-rate, transcode to use another encoder, change the packetization of the media stream, add FEC streams, or terminate RTP retransmissions. The latter behaviors require the translator to use SSRCs that only exist in a particular sub-domain of the RTP session, and it may also create additional sessions when the mechanism applied on one side so requires.

5.4. Point to Multipoint Using an RTP Mixer

The most commonly used topology in centralized conferencing is based on the RTP Mixer. The main reason for this is that it provides a very consistent view of the RTP session towards each participant. That is accomplished through the mixer having its own SSRCs and any media sent to the participants will be sent using those SSRCs. If the mixer wants to identify the underlying media sources for its conceptual streams, it can identify them using CSRC. The media stream the mixer provides can be an actual media mixing of multiple media sources. It might also be as simple as selecting one of the underlying sources based on some mixer policy or control signalling.

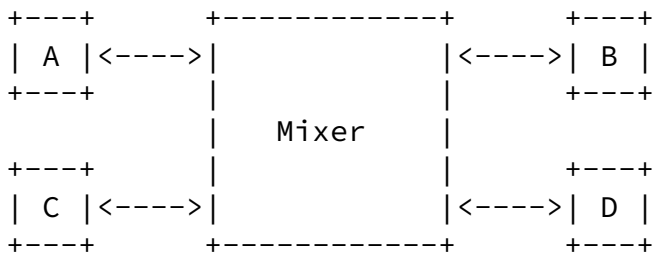


Figure 5: RTP Mixer

In the case where the mixer does stream selection, an application may in fact desire multiple simultaneous streams but only as many as the mixer can handle. As long as the mixer and an end-point can agree on the maximum number of streams and how the streams that are delivered are selected, this provides very good functionality. As these streams are forwarded using the mixer's SSRCs, there are no inconsistencies within the session.

5.5. Point to Multipoint using Multiple Unicast flows

Based on the RTP session definition, it is clearly possible to have a joint RTP session over multiple transport flows like the below three end-point joint session. In this case, A needs to send its' media streams and RTCP packets to both B and C over their respective transport flows. As long as all participants do the same, everyone will have a joint view of the RTP session.

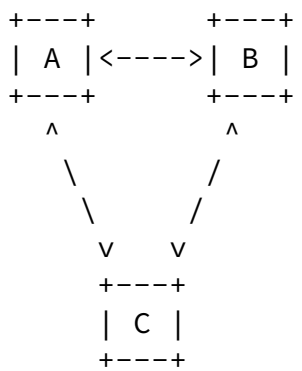


Figure 6: Point to Multi-Point using Multiple Unicast Transports

This doesn't create any additional requirements beyond the need to have multiple transport flows associated with a single RTP session. Note that an end-point may use a single local port to receive all these transport flows, or it might have separate local reception ports for each of the end-points.

There exists an alternative structure for establishing the above communication scenario (Figure 6) which uses independent RTP sessions between each pair of peers, i.e. three different RTP sessions. Unless independently adapted the same RTP media stream could be sent

in both of the RTP sessions an end-point has. The difference exists in the behaviors around RTCP, for example common RTCP bandwidth for one joint session, rather than three independent pools, and the awareness based on RTCP reports between the peers of how that third leg is doing.

5.6. De-composite End-Point

There is some possibility that an RTP end-point implementation in fact reside on multiple devices, each with their own network address. A very basic use case for this would be to separate audio and video processing for a particular end-point, like a conference room, into one device handling the audio and another handling the video, being interconnected by some control functions allowing them to behave as a single end-point.

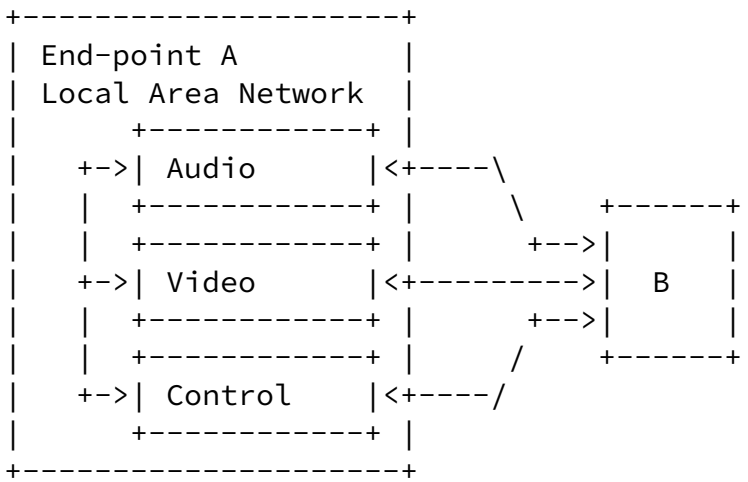


Figure 7: De-composite End-Point

In the above usage, let us assume that the RTP sessions are different for audio and video. The audio and video parts will use a common CNAME and also have a common clock to ensure that synchronization and clock drift handling works despite the decomposition.

However, if the audio and video were in a single RTP session then this use case becomes problematic. This as all transport flow

receivers will need to receive all the other media streams that are part of the session. Thus the audio component will receive also all the video media streams, while the video component will receive all the audio ones, doubling the site's bandwidth requirements from all other session participants. With a joint RTP session it also becomes evident that a given end-point, as interpreted from a CNAME perspective, has two sets of transport flows for receiving the streams and the decomposition is not hidden.

The requirements that can derived from the above usage is that the transport flows for each RTP session might be under common control but still go to what looks like different end-points based on addresses and ports. A conclusion can also be reached that decomposition without using separate RTP sessions has downsides and potential for RTP/RTCP issues.

There exist another use case which might be considered as a de-composite end-point. However, as will be shown this should be considered a translator instead. An example of this is when an end-point A sends a media flow to B. On the path there is a device C that on A's behalf does something with the media streams, for example adds an RTP session with FEC information for A's media streams. C will in this case need to bind the new FEC streams to A's media stream by using the same CNAME as A.

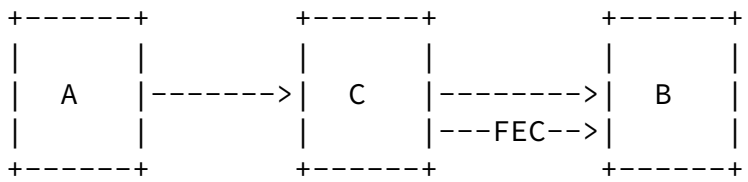


Figure 8: When De-composition is a Translator

This type of functionality where C does something with the media stream on behalf of A is clearly covered under the media translator definition (Section 5.3).

6. Multiple Streams Discussion

6.1. Introduction

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.

6.2. RTP/RTCP Aspects

This section discusses RTP and RTCP aspects worth considering when selecting between SSRC multiplexing and Session multiplexing.

6.2.1. 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 minimized, 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 may 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 still 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. This is to simplify the processing at any receiver of the media stream. If all media streams that exist in an RTP session are of one media type and one particular purpose, there is no need for deeper inspection of the packets before processing them in both end-points and RTP aware middle nodes.

The fifth argument discusses network aspects that we will discuss more below in Section 6.5. It also goes into aspects of implementation, like decomposed end-points 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 secondly use different RTP sessions for media streams that don't share media type and purpose, to maximize flexibility when it comes to processing and handling of the media streams.

This mostly agrees with the discussion and recommendations in this document. However, there has been an evolution of RTP since that text was written which needs to be reflected in the discussion. Additional clarifications for specific cases are also needed.

6.2.1.1. 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 has been 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 document has a more detailed analysis of

potential issues in having multiple media types in the same RTP session in Section 6.7. An important influence for underlying thinking for the RTP design and likely this statement can be found in the academic paper by David Clark and David Tennenhouse "Architectural considerations for a new generation of protocols"

[ALF].

6.2.2. Handling Varying sets of Senders

A potential issue that some application designers may need to consider is the case where the set of simultaneously active sources varies within a larger set of session members. As each media decoding chain may contain state, it is important that this type of usage ensures that a receiver can flush a decoding state for an inactive source and if that source becomes active again, it does not assume that this previous state exists.

This behavior will cause similar issues independent of SSRC or Session multiplexing. It might be possible in certain applications to limit the changes to a subset of communication session participants by have the sub-set use particular RTP Sessions.

6.2.3. Cross Session RTCP Requests

There currently exists no functionality to make truly synchronized 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 SSRC multiplexed streams 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.

In Session multiplexed cases, 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.

6.2.4. Binding Related Sources

A common problem in a number of various RTP extensions has been how to bind related sources together. This issue is common to SSRC multiplexing and Session Multiplexing, and any solution and recommendation related to the problem should work equally well with both methods to avoid creating barriers between using session multiplexing and SSRC multiplexing.

The current solutions do not have these properties. There exists one

solution for grouping RTP session together in SDP [RFC5888] to know which RTP session contains for example the FEC data for the source data in another session. However, this mechanism does not work on individual media flows and is thus not directly applicable to the problem. The other solution is also SDP based and can group SSRCs within a single RTP session [RFC5576]. Thus this mechanism can bind media streams in SSRC multiplexed cases. Both solutions have the shortcoming of being restricted to SDP based signalling and also do not work 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.

One possible solution could be to mandate the same SSRC being used in all RTP session in case of session multiplexing. We do note that Section 8.3 of the RTP Specification [RFC3550] recommends using a single SSRC space across all RTP sessions for layered coding. However this recommendation has some downsides and is less applicable beyond the field of layered coding. To use the same sender SSRC in all RTP sessions from a particular end-point can cause issues if an SSRC collision occurs. If the same SSRC is used as the required binding between the streams, then all streams in the related RTP sessions must change their SSRC. This is extra likely to cause problems if the participant populations are different in the different sessions. For example, in case of large number of receivers having selected totally random SSRC values in each RTP session as RFC 3550 specifies, a change due to a SSRC collision in one session can then cause a new collision in another session. This cascading effect is not severe but there is an increased risk that this occurs for well populated sessions. In addition, being forced to change the SSRC affects all the related media streams; instead of having to re-synchronize only the originally conflicting stream, all streams will suddenly need to be re-synchronized with each other. This will prevent also the media streams not having an actual collision from being usable during the re-synchronization and also increases the time until synchronization is finalized. In addition, it requires exception handling in the SSRC generation.

The above collision issue does not occur in case of having only one SSRC space across all sessions and all participants will be part of at least one session, like the base layer in layered encoding. In that case the only downside is the special behavior that needs to be well defined by anyone using this. But, having an exception behavior where the SSRC space is common across all session is an issue as this behavior does not fit all the RTP extensions or payload formats. It is possible to create a situation where the different mechanisms cannot be combined due to the non standard SSRC allocation behavior.

Existing mechanisms with known issues:

RTP Retransmission (RFC4588): Has two modes, one for SSRC multiplexing and one for Session multiplexing. The session multiplexing requires the same CNAME and mandates that the same SSRC is used in both sessions. Using the same SSRC does work but will potentially have issues in certain cases. In SSRC multiplexed mode the CNAME is used to bind media and retransmission streams together. However, if multiple media streams are sent from the same end-point in the same session this does not provide non-ambiguous binding. Therefore when the first retransmission request for a media stream is sent, one must not have another retransmission request outstanding for an SSRC which don't have a binding between the original SSRC and the retransmission stream's SSRC. This works but creates some limitations that can be avoided by a more explicit mechanism. The SDP based ssrc-group mechanism is sufficient in this case as long as the application can rely on the signalling based solution.

Scalable Video Coding (RFC6190): As an example of scalable coding, SVC [RFC6190] has various modes. The Multi Session Transmission (MST) uses Session multiplexing to separate scalability layers. However, this specification has failed to be explicit on how these layers are bound together in cases where CNAME is not sufficient. CNAME is no longer sufficient when more than one media source occur within a session that has the same CNAME, for example due to multiple video cameras capturing the same lecture hall. This likely implies that a single SSRC space as recommend by Section 8.3 of RTP [RFC3550] is to be used.

Forward Error Correction: If some type of FEC or redundancy stream is being sent, it needs its own SSRC, with the exception of constructions like redundancy encoding [RFC2198]. Thus in case of transmitting the FEC in the same session as the source data, the inter SSRC relation within a session is needed. In case of sending the redundant data in a separate session from the source, the SSRC in each session needs to be related. This occurs for example in RFC5109 when using session separation of original and FEC data. SSRC multiplexing is not supported, only using redundant encoding is supported.

This issue appears to need action to harmonize and avoid future shortcomings in extension specifications. A proposed solution for handling this issue is [I-D.westerlund-avtext-rtcp-sdes-srcname].

6.2.5. 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 loss 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 the related flows as discussed above. They also create additional flows that need to be transported. Looking at the history of these schemes, there is both SSRC multiplexed and Session multiplexed solutions and some schemes that support both.

Using a Session multiplexed solution provides good support for legacy when deploying FEC or changing the scheme used, in the sense that it supports the case where some set of receivers may not be able to utilize 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 RTP session allows for flexibility, for example when using Rapid Acquisition of Multicast Groups (RAMS) [RFC6285]. During the RAMS burst where data is received over unicast and where it is possible to combine with unicast based retransmission [RFC4588], there is no need to burst the FEC data related to the burst of the source media streams needed to catch up with the multicast group. This saves bandwidth to the receiver during the burst, enabling quicker catch up. When the receiver has caught up and joins the multicast group(s) for the source, it can at the same time join the multicast group with the FEC information. Having the source stream and the FEC in separate groups allow for easy separation in the Burst/Retransmission Source (BRS) without having to individually classify packets.

6.2.6. Transport Translator Sessions

A basic Transport Translator relays any incoming RTP and RTCP packets

to the other participants. The main difference between SSRC multiplexing and Session multiplexing resulting from this use case is that for SSRC multiplexing 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.

6.3. Interworking

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 may have to be considered working with some legacy applications.

6.3.1. Interworking Applications

It is not uncommon that applications or services of similar usage, especially the ones intended for interactive communication, ends up in a situation where one want to interconnect two or more of these applications. From an RTP perspective this could be problem free if all the applications 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 may not always be true.

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. If one's goal is to make minimal amount of work in such a gateway, there are some multiplexing choices that one should avoid. The lowest amount of work represents solutions where one can take an SSRC from one RTP session in one application and forward it into another RTP session. For example if one has one application that has multiple SSRCs for one media type in one session and another application that instead has chosen to use multiple RTP sessions with only a single SSRC per end-point in each of these sessions. Then mapping an SSRC from the side with one session into an RTP session is possible. However mapping SSRC from different RTP sessions into a

single RTP session has the potential of creating SSRC collisions, especially if an end-point has not generated independent random SSRC values in each RTP session. This issue is even more likely in a case where one side uses a single RTP session with multiple media types and the other uses different RTP session for different media or robustness mechanism such as retransmission [RFC4588]. Then it is more likely or maybe even required to use the same SSRC in the different RTP sessions.

In cases where the used structure is incompatible, the gateway will need to make SSRC translation. Thus this incurs overhead and some potential loss of functionality. First of all, if one translates the SSRC in an RTP header then one will be forced to decrypt and re-encrypt if one uses SRTP and thus also needs to be part of the security association. Secondly, changing the SSRC also means that one needs to translate all RTCP messages. This can be more complex, but important so that the gateway does not end up having to terminate the end-to-end RTCP chain. In that case the gateway will need to be able to take the role of a true end-point in each session, which may include functions such as bit-rate adaptation and correctly respond

to whatever RTCP extensions are being used, and then translate them or locally respond to them. Thirdly, an SSRC translation may require that one changes RTP payloads; for example, an RTP retransmission packet contains an original sequence number that must match the sequence number used in for the corresponding packet with the new SSRC. And for FEC packets this is even worse, as the original SSRC is included as part of the data for which FEC redundant data is calculated. A fourth issue is 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 end-points, 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 makes it hard to move a flow from one RTP session to another as each RTP session will have one or more different master keys and these must not be the same in multiple RTP sessions as that can result in two-time pads that completely breaks the confidentiality of the packets.

An additional issue around interworking is that for multi-party applications it can be impossible to judge which different RTP multiplexing behaviors that will be used by end-points that attempt to join a session. Thus if one attempts to use a multiplexing choice that has poor interworking, one may have to switch at a later stage when someone wants to participate in a multi-party session using an RTP application supporting only another behavior. It is likely difficult to implement the switch without some media disruption.

To summarize, certain types of applications are likely to be inter-worked. Sets of applications of similar type should strive to use the same multiplexing structure to avoid the need to make an RTP session level gateway. This as it incurs complexity costs, can force the gateway to be part of security associations, force SSRC translation and even payload translation which is also a potential hinder to application evolution.

6.3.2. 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 end-point 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 end-point and the

RTP mixer.

When establishing RTP sessions that may contain end-points 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.

RTP Session multiplexing could potentially avoid these issues if there is only a single SSRC at each end-point, and in topologies which appears like point to point as seen the end-point. However, forcing the usage of session multiplexing due to this reason would be a great mistake, as it is likely that a significant set of applications will need a combination of SSRC multiplexing of several media sources and session multiplexing for other aspects such as encoding alternatives, adding robustness or simply to support legacy. However, this issue does need consideration when deploying multiple media streams within an RTP session where legacy end-points may occur.

6.4. 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.

6.4.1. 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
- Which SSRC that will use which RTP Payload Types

Some of these issues are clearly SDP's problem rather than RTP limitations. However, if the aim is to deploy an SSRC multiplexed

solution 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 SSRC multiplexing only, a number of signalling extensions are needed to clarify that there are multiple sets of media streams with different properties and that they shall in fact be kept different, since a single set will not satisfy the application's requirements.

This does in fact create a strong driver to use RTP session multiplexing for any case where different sets of media streams with different requirements exist.

6.4.2. SDP Prevents Multiple Media Types

SDP encoded in its structure prevention against using multiple media types in the same RTP session. A media description in SDP can only have a single 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. Thus a high fence against using multiple media types in the same session was created.

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

6.4.3. 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 by having all audio media streams arriving in the only audio RTP session they are to be decoded, mixed and played 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. Here the choice of SSRC multiplexing versus session multiplexing will have significant impact. If one uses SSRC multiplexing to its full extent one will have to explicitly indicate

for each SSRC what its' usage and purpose are using some signalling between the application instances.

This SSRC usage signalling will have some impact on the application and also on any central RTP nodes. It is important in the design to consider the implications of the need for additional signalling between the nodes. One consideration is if a receiver can utilize the media stream at all before it has received the signalling message describing the media stream and its usage. Another consideration is that any RTP central node, like an RTP mixer or translator that selects, mixes or processes streams, in most cases will need to receive the same signalling to know how to treat media streams with different usage in the right fashion.

Application designers should consider putting media streams of the same usage and/or receiving the same treatment in middleboxes in the same RTP sessions and use the RTP session as an explicit indication of how to deal with media streams. By having session level indication of usage and have different RTP sessions for different usages, the need for stream specific signalling can be reduced. Especially signalling of the type that is time critical and needs to be provided prior to the media stream being available.

6.5. Network Aspects

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

6.5.1. 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. SSRC multiplexing 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 for session based multiplexing, where each different version can be in a different RTP session that can be sent over different 5-tuples.

6.5.2. 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 end-points 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 end-point 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 end-point may use. In practice, implementation will probably benefit from using more server ports to simplify implementation or avoid performance bottlenecks.

NAT State: If an end-point 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 end-points, available external ports are typically the scarce resource. Port limitations is primarily a problem for larger centralized NATs where end-point 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 can take 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. This as ICE first finds one candidate pair that works prior 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.

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 it should 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, should 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.

SSRC multiplexing keeps additional media streams within one RTP Session and does not introduce any additional NAT traversal complexities per media stream. In contrast, the session multiplexing is using one RTP session per media stream. Thus additional lower layer transport flows will be required, 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-single-transport-multiplexing].

6.5.3. Multicast

Multicast groups provides a powerful semantics for a number of real-time applications, especially the ones that desire broadcast-like behaviors with one end-point 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.

If instead a single RTP session over multiple transports were to be deployed, i.e. multicast groups with each layer as it's own SSRC, then very different views of the RTP session would exist. That as one receiver may see only a single layer (SSRC), while another may see three SSRCs if it joined three multicast groups. This would cause disjoint RTCP reports where a management system would not be able to determine if a receiver isn't reporting on a particular SSRC due to that it is not a member of that multicast group, or because it doesn't receive it as a result of a transport failure.

Thus it appears easiest and most straightforward to use multiple RTP sessions. In addition, the transport flow considerations in multicast are a bit different from unicast. First of all there is no shortage of port space, as each multicast group has its own port space.

6.5.4. Multiplexing multiple RTP Session on a Single Transport

For applications that doesn'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-single-transport-multiplexing]. Using such a solution would allow session multiplexing without most of the perceived downsides of additional RTP sessions creating a need for additional transport flows.

6.6. Security Aspects

On the basic level there is no significant difference in security when having one RTP session and having multiple. However, there are a few more detailed considerations that might need to be considered in certain usages.

6.6.1. 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) is 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 situations.

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.

In the latter case it is likely easiest from signalling, transport (if done over multicast) and security to use a different RTP session. That way the user(s) not intended to receive a particular stream can easily be excluded. There is no need to have SSRC specific keys, which many of the key-management systems cannot handle.

6.6.2. Key-Management for Multi-party session

Performing key-management for Multi-party session can be a challenge.

This section considers some of the issues.

Transport translator based session cannot use Security Description [RFC4568] nor DTLS-SRTP [RFC5764] without an extension as each end-point provides its set of keys. In centralized conference, the signalling counterpart is a conference server and the media plane unicast counterpart (to which DTLS messages would be sent) is the 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.

Keying of multicast transported SRTP face similar challenges as the transport translator case.

6.6.3. 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 a not secured RTP translator or mixer to rewrite an SSRC value in the RTP packet, 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 must 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.

There exist many valid cases where a middlebox will be forced to perform such cryptographic operations due to the intended purpose of the middlebox, for example a media transcoding RTP translator cannot avoid performing these operations as they will produce a different payload compared to the input. However, there exist some cases where another topology and/or multiplexing choice could avoid the complexities.

6.7. Multiple Media Types in one RTP session

Having different media types, like audio and video, in the same RTP

sessions is not forbidden, only recommended against as earlier discussed in Section 6.2.1.1. When using multiple media types, there are a number of considerations:

Payload Type gives Media Type: This solution is dependent on getting the media type from the Payload Type. Thus overloading this de-multiplexing point in a receiver making it serve two purposes. First to provide the main media type and determining the processing chain, then later for the exact configuration of the encoder and packetization.

Payload Type field limitations: The total number of Payload Types available to use in an RTP session is fairly limited, especially if Multiplexing RTP Data and Control Packets on a Single Port [RFC5761] is used. For certain applications negotiating a large set of codes and configuration this may become an issue.

An SSRC cannot use two clock rates simultaneously: The used RTP clock rate for an SSRC is determined from the payload type. As discussed in Appendix A it is not possible to simultaneously use two different clock rates for the same SSRC. Even switching clock rate once has potential issues if packet loss occurs at the same time. Different media types commonly have different clock rates preventing or creating issues to use two different media types for the same SSRC.

Do not switch media types for an SSRC: The primary reasons to avoid switching from sending for example audio to sending video using the same SSRC is the implications on a receiver. When this happens, the processing chain in the receiver will have to switch from one media type to another. As the different media type's entire processing chains are different and are connected to different outputs it is difficult to reuse the decoding chain,

which a normal codec change likely can. Instead the entire processing chain has to be torn down and replaced. In addition, there is likely a clock rate switching problem, possibly resulting in synchronization loss at the point of switching media type if some packet loss occurs. So this is a behavior that shall be avoided.

RTCP Bit-rate Issues: If the media types are significantly different

in bit-rate, the RTCP bandwidth rates assigned to each source in a session can result in interesting effects, like that the RTCP bit-rate share for an audio stream is larger than the actual audio bit-rate. In itself this doesn't cause any conflicts, only potentially unnecessary overhead. It is possible to avoid this using AVPF or SAVPF and setting trr-int parameter, which can bring down unnecessary regular reporting while still allowing for rapid feedback.

De-composite end-points: De-composite nodes that rely on the regular network to separate audio and video to different devices do not work well with this session setup. If they are forced to work, all media receiver parts of a de-composite end-point will receive all media, thus doubling the bit-rate consumption for the end-point.

Flow based QoS Separation: Flow based QoS mechanisms will see all the media streams in the RTP session as part of a single flow. Therefore there is no possibility to provide separated QoS behavior for the different media types or flows.

RTP Mixers and Translators: An RTP mixer or Media Translator will also have to support this particular session setup, where it before could rely on the RTP session to determine what processing options should be applied to the incoming packets.

Legacy Implementations: The use of multiple media types has the potential for even larger issues with legacy implementations than single media type SSRC multiplexing due to the occurrence of multiple media types among the payload type configurations.

As can be seen, there is nothing in here that prevents using a single RTP session for multiple media types, however it does create a number of limitations and special case implementation requirements. So anyone considering using this setup should carefully review if the reasons for using a single RTP session are sufficient to motivate the needed special handling.

7. Arch-Types

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

7.1. Single SSRC per Session

In this arch-type each end-point 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 end-points have one media stream each. If the application needs additional media flows between the end-points, they will have to establish additional RTP sessions.

The Pros:

1. This arch-type 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 session with current key-management.

The Cons:

a. The number of required RTP sessions cannot really be higher, 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 end-points.

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 may become a problem for centralized 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 needs is likely to exist and may 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 have issues and require SSRC translation.

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

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

i. Does not support multiparty session within a session. Instead each multi-party participant will require an individual RTP session to a given end-point, even if a central node is used.

RTP applications that need to inter-work with legacy RTP applications, like VoIP and video conferencing, 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, must have its own RTP session. In addition, the dynamic behavior 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 arch-type, each RTP session serves only a single media type. The RTP session can contain multiple media streams, either from a single end-point or due to multiple end-points. This commonly creates a low number of RTP sessions, typically only two one for audio and one for video with a corresponding need for two listening ports when using RTP and 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 end-points.

4. Enables Flow-based QoS with different prioritization 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 end-point.

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, only between SDP media descriptions.

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 faith 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 prioritization between media types. It handles multi-party session well, independently of multicast or centralized transport distribution, as additional sources can dynamically enter and leave the session.

7.3. Multiple Sessions for one Media type

In this arch-type one goes one step further than in the above (Section 7.2) by using multiple RTP sessions also for a single media type. The main reason for going in this direction is that the RTP application needs separation of the media streams due to their usage. Some typical reasons for going to this arch-type are scalability over multicast, simulcast, need for extended QoS prioritization of media streams due to their usage in the application, or the need for fine granular 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. Detailed 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 prioritization for flow based mechanisms.

5. Works well with de-composite end-points.

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 end-point 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 synchronized 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 must 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 end-point.

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 arch-type 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 end-point 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 end-point 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. Not suitable for interworking with other applications that uses

individual RTP sessions per media type or multiple sessions for a single media type, due to high risk of forced 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 required bandwidth.

c. Does enforce higher bandwidth and processing on de-composite end-points.

d. Flow based QoS cannot provide separate treatment to some media streams compared to other 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 does not support the RTP specification fully when it comes to handling multiple SSRC per end-point, 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.

The analysis in this document and considerations in Section 6.7 implies that this is suitable only in a set of restricted use cases. The aspect in the above list that can be most difficult to judge long term is likely the potential need for interworking with other applications and services.

7.5. Summary

There are some clear relations between these arch-types. 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 required.

The two other arch-types "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. 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 6.2.4 there exist drawbacks in using the same SSRC in multiple RTP sessions as a mechanism to bind related media streams together. It is instead recommended 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 SSRC multiplexing for additional Media Sources: In the cases an RTP end-point needs to transmit additional media source(s) of the same media type and purpose in the application, it is recommended to send them as additional SSRCs in the same RTP session. For

example a tele-presence 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 recommended.

Use additional RTP sessions for streams with different purposes: When media streams have different purpose or processing requirements it is recommended that the different types of streams are put in different RTP sessions.

When using Session Multiplexing use grouping: When using Session Multiplexing solutions, it is recommended to be explicitly group the involved RTP sessions using the signalling mechanism, for example The Session Description Protocol (SDP) Grouping Framework. [RFC5888], using some appropriate grouping semantics.

RTP/RTCP Extensions May Support SSRC and Session Multiplexing: When defining an RTP or RTCP extension, the creator needs to consider if this extension is applicable in both SSRC multiplexed and Session multiplexed usages. Any extension intended to be generic is recommended 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 recommended to include support for both SSRC and Session multiplexing 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. Proposal 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 Session multiplexing or SSRC multiplexing. 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.

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 end-point can handle within a given RTP Session. That ensures that usage of SSRC multiplexing occurs when supported and without overloading an end-point. This extension is proposed in [I-D.westerlund-avtcore-max-ssrc].

10. RTP Specification Clarifications

This section describes a number of clarifications to the RTP specifications that are likely necessary for aligned behavior when RTP sessions contain more SSRCs than one local and one remote.

10.1. RTCP Reporting from all SSRCs

When one have multiple SSRC in an RTP node, all these SSRC must send RTCP SR or RR as long as the SSRC exist. It is not sufficient that only one SSRC in the node sends report blocks on the incoming RTP streams. The reason for this is that a third party monitor may not necessarily be able to determine that all these SSRC are in fact co-located and originate from the same stack instance that gather report data.

10.2. RTCP Self-reporting

For any RTP node that sends more than one SSRC, there is the question if SSRC1 needs to report its reception of SSRC2 and vice versa. The reason that they in fact need to report on all other local streams as being received is report consistency. A third party monitor that considers the full matrix of media streams and all known SSRC reports on these media streams would detect a gap in the reports which could be a transport issue unless identified as in fact being sources from same node.

10.3. Combined RTCP Packets

When a node contains multiple SSRCs, it is questionable if an RTCP compound packet can only contain RTCP packets from a single SSRC or if multiple SSRCs can include their packets in a joint compound packet. The high level question is a matter for any receiver processing on what to expect. In addition to that question there is the issue of how to use the RTCP timer rules in these cases, as the

existing rules are focused on determining when a single SSRC can

send.

11. IANA Considerations

This document makes no request of IANA.

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

12. Security Considerations

There is discussion of the security implications of choosing SSRC vs Session multiplexing in Section 6.6.

13. Acknowledgements

The authors would like to thanks Harald Alvestrand for providing input into the discussion regarding multiple media types in a single RTP session.

14. References

14.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

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

14.2. 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-00 (work in progress), December 2011.

[I-D.holmberg-mmusic-sdp-bundle-negotiation]
Holmberg, C. and H. Alvestrand, "Multiplexing Negotiation Using Session Description Protocol (SDP) Port Numbers", draft-holmberg-mmusic-sdp-bundle-negotiation-00 (work in progress), October 2011.

[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-avtext-multiple-clock-rates]
Petit-Huguenin, M., "Support for multiple clock rates in an RTP session", draft-ietf-avtext-multiple-clock-rates-02 (work in progress), January 2012.

[I-D.ietf-payload-rtp-howto]
Westerlund, M., "How to Write an RTP Payload Format", draft-ietf-payload-rtp-howto-01 (work in progress), July 2011.

[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 (work in progress), October 2011.

[I-D.westerlund-avtcore-single-transport-multiplexing]
Westerlund, M., "Multiple RTP Session on a Single Lower-Layer Transport", draft-westerlund-avtcore-transport-multiplexing (work in progress), October 2011.

[I-D.westerlund-avtext-rtcp-sdes-srcname]
Westerlund, M., Burman, B., and P. Sandgren, "RTCP SDES Item SRCNAME to Label Individual Sources", draft-westerlund-avtext-rtcp-sdes-srcname (work in progress), October 2011.

[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. 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.

[RFC4607] Holbrook, H. and B. Cain, "Source-Specific Multicast for IP", RFC 4607, August 2006.

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

[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.

[RFC5760] Ott, J., Chesterfield, J., and E. Schooler, "RTP Control

Protocol (RTCP) Extensions for Single-Source Multicast Sessions with Unicast Feedback", RFC 5760, February 2010.

[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., Schierl, T., and A. Eleftheriadis, "RTP Payload Format for Scalable Video Coding", RFC 6190, May 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.

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

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