Network Working Group   M. Westerlund
Internet-Draft   B. Burman
Intended status: Informational   Ericsson
Expires: May 3, 2018   C. Perkins
                                                   University of Glasgow
                                                          H. Alvestrand
                                                                 Google
                                                                R. Even
                                                               H. Zheng
                                                                 Huawei
                                                       October 30, 2017

           Guidelines for using the Multiplexing Features of RTP to Support
                          Multiple Media Streams
                draft-ietf-avtcore-multiplex-guidelines-04

Abstract

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

Copyright Notice

Table of Contents

1. Introduction

   The Real-time Transport Protocol (RTP) [RFC3550] is a commonly used
   protocol for real-time media transport. It is a protocol that
   provides great flexibility and can support a large set of different
   applications. RTP was from the beginning designed for multiple
   participants in a communication session. It supports many paradigms
   of topologies and usages, as defined in [RFC7667]. 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.

   There have been increased interest in more advanced usage of RTP, for
   example, multiple streams can occur when a single endpoint have
   multiple media sources, like multiple cameras or microphones that
   need to be sent simultaneously. Consequently, questions are raised
   regarding the most appropriate RTP usage. The limitations in some
   implementations, RTP/RTCP extensions, and signalling has also been
   exposed. The authors also hope that clarification on the usefulness
   of some functionalities in RTP will result in more complete
   implementations in the future.

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

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

2. Definitions

2.1. Terminology

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

The taxonomy defined in [RFC7656] is referenced normatively.

The following terms and abbreviations are used in this document:

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

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

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

Multiplexing: The operation of taking multiple entities as input,
   aggregating them onto some common resource while keeping the
   individual entities addressable such that they can later be fully
   and unambiguously separated (de-multiplexed) again.

RTP Session Group: One or more RTP sessions that are used together
   to perform some function. Examples are multiple RTP sessions used
   to carry different layers of a layered encoding. In an RTP
   Session Group, CNAMEs are assumed to be valid across all RTP

sessions, and designate synchronisation contexts that can cross
RTP sessions.

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

2.2. Subjects Out of Scope

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

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

3. RTP Multiplexing Overview

3.1. Reasons for Multiplexing and Grouping RTP Media Streams

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

o Multiple Media Sources

o Multiple Media Streams might be needed to represent one Media
   Source (for instance when using layered encodings)

o A Retransmission stream might repeat the content of another Media
   Stream

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

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

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

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

3.2. RTP Multiplexing Points

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

```
                      |

                      | packets

         +--               v

         |          +------------+

         |          |   Socket   |

         |          +------------+

         |              ||  ||

    RTP  |         RTP/ ||  |+-----> SCTP ( ...and any other protocols)

  Session |        RTCP ||  +------> STUN (multiplexed using same port)

         +--           ||

         +--           ||

         |         (split by SSRC)

         |          ||  ||   ||

         |          ||  ||   ||
```

```
   Media  |        +--+ +--+ +--+

 Streams  |        |PB| |PB| |PB| Jitter buffer, process RTCP, FEC, etc.

          |        +--+ +--+ +--+

      +--      |    |      |

          (pick rendering context based on PT)

      +--      |  /       |

          |       +---+      |

          |       /  |       |

 Payload  |        +--+ +--+ +--+

 Formats  |        |CR| |CR| |CR| Codecs and rendering

          |        +--+ +--+ +--+

      +--
```
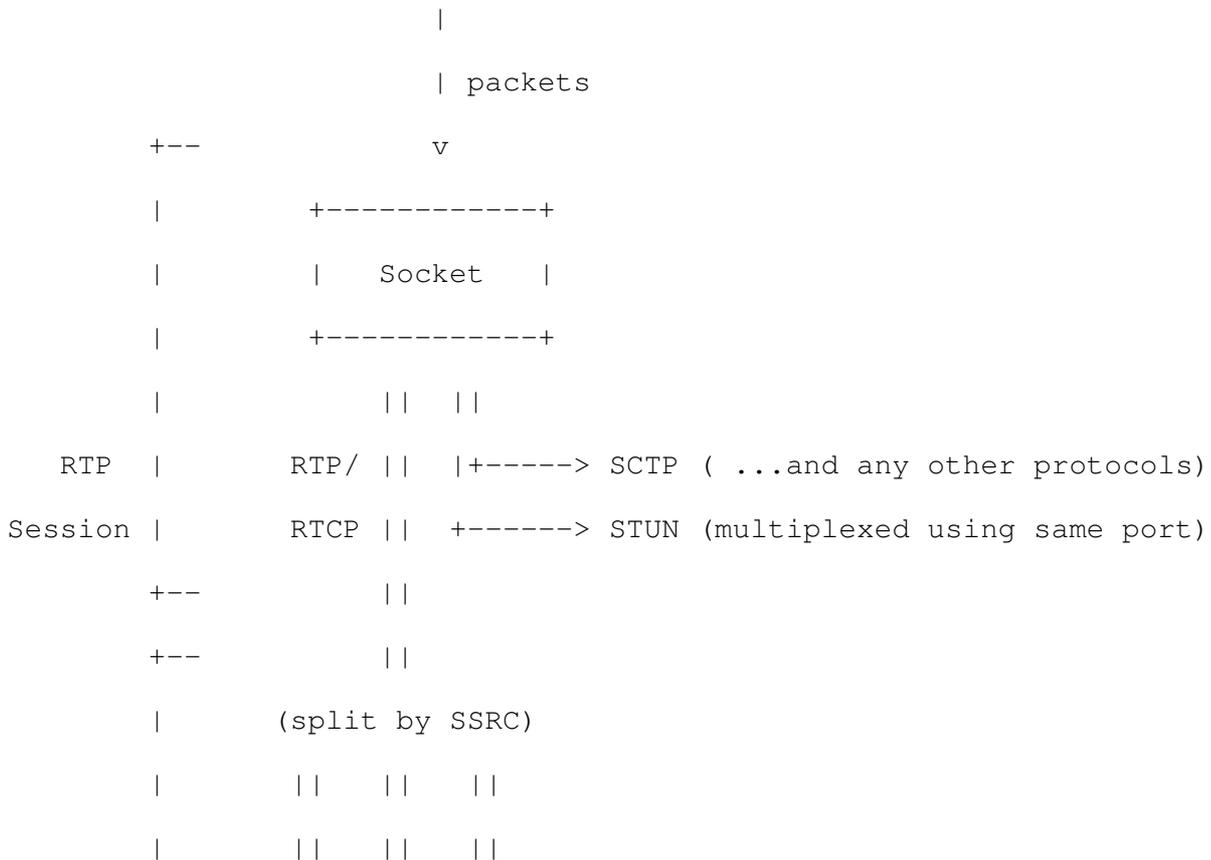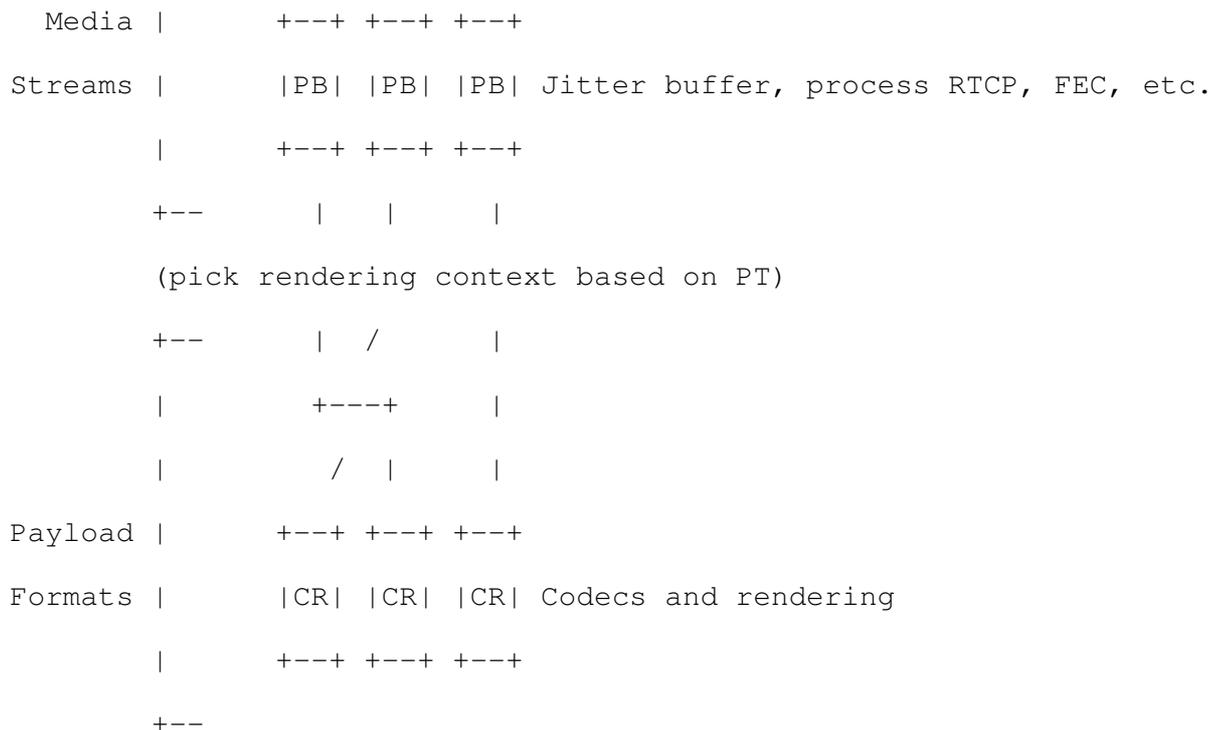
Figure 1: RTP Demultiplexing Process

3.2.1. RTP Session

   An RTP Session is the highest semantic layer in the RTP protocol, and
   represents an association between a group of communicating endpoints.
   The set of participants that form an RTP session is defined as those
   that share a single synchronisation source space [RFC3550]. That is,
   if a group of participants are each aware of the synchronisation
   source identifiers belonging to the other participants, then those
   participants are in a single RTP session. A participant can become
   aware of a synchronisation source identifier by receiving an RTP
   packet containing it in the SSRC field or CSRC list, by receiving an
   RTCP packet mentioning it in an SSRC field, or through signalling
   (e.g., the SDP OCGBPa=ssrc:OCOe attribute). Thus, the scope of an
   RTP session is determined by the participants' network
   interconnection topology, in combination with RTP and RTCP forwarding
   strategies deployed by the endpoints and any middleboxes, and by the
   signalling.

   RTP does not contain a session identifier. Rather, it relies on the
   underlying transport layer to separate different sessions, and on the
   signalling to identify sessions in a manner that is meaningful to the

application. The signalling layer might give sessions an explicit
identifier, or their identification might be implicit based on the
addresses and ports used. Accordingly, a single RTP Session can have
multiple associated identifiers, explicit and implicit, belonging to
different contexts. For example, when running RTP on top of UDP/IP,
an RTP endpoint can identify and delimit an RTP Session from other
RTP Sessions using the UDP source and destination IP addresses and
UDP port numbers. Another example is when using SDP grouping
framework [RFC5888] which uses an identifier per OCGBPm=OCOe-line; if
there is a one-to-one mapping between OCGBPm=OCOe-lines and RTP
sessions, that grouping framework identifier will identify an RTP
Session. [I-D.ietf-mmusic-sdp-bundle-negotiation] extends the
OCGBPm-OCGBP-line for bundled media, which adds complexity to
demultiplexing media stream. Section 10.2 of
[I-D.ietf-mmusic-sdp-bundle-negotiation] provides information about
how RTP/RTCP streams are associated with SDP media description.

RTP sessions are globally unique, but their identity can only be
determined by the communication context at an endpoint of the
session, or by a middlebox that is aware of the session context. The
relationship between RTP sessions depending on the underlying
application, transport, and signalling protocol. The RTP protocol
makes no normative statements about the relationship between
different RTP sessions, however the applications that use more than
one RTP session will have some higher layer understanding of the
relationship between the sessions they create.

3.2.2. Synchronisation Source (SSRC)

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

The synchronisation Source identifier is a 32-bit unsigned integer.
It is present in every RTP and RTCP packet header, and in the payload
of some RTCP packet types. It can also be present in SDP signalling.
Unless pre-signalled using the SDP OCGBPa=ssrc:OCOe attribute
[RFC5576], the synchronisation source identifier is chosen at random.
It is not dependent on the network address of the endpoint, and is
intended to be unique within an RTP session. Synchronisation source

identifier collisions can occur, and are handled as specified in
[RFC3550] and [RFC5576], resulting in the synchronisation source
identifier of the affecting RTP sources and/or sinks changing. An
RTP source that changes its RTP Session identifier (e.g. source
transport address) during a session has to choose a new SSRC
identifier to avoid being interpreted as looped source.

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

   In some cases, the same SSRC Identifier value is used to relate
   streams in two different RTP Sessions, such as in Multi-Session
   Transmission of scalable video [RFC6190]. This is to be avoided
   since there is no guarantee of uniqueness in SSRC values across
   RTP sessions.

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

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

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

Processed Media Source: A source with some attributed property
   generated by some network node, for example a filtering function
   in an RTP mixer that provides the most active speaker based on
   some criteria, or a mix representing a set of other sources.

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

Note that an endpoint that generates more than one media type, e.g.
a conference participant sending both audio and video, need not (and
commonly does not) use the same SSRC value across RTP sessions. RTCP
Compound packets containing the CNAME SDES item is the designated
method to bind an SSRC to a CNAME, effectively cross-correlating
SSRCs within and between RTP Sessions as coming from the same
endpoint. The main property attributed to SSRCs associated with the

same CNAME is that they are from a particular synchronisation context
and can be synchronised at playback.

An RTP receiver receiving a previously unseen SSRC value will
interpret it as a new source. It might in fact be a previously
existing source that had to change SSRC number due to an SSRC
conflict. However, the originator of the previous SSRC ought to have
ended the conflicting source by sending an RTCP BYE for it prior to
starting to send with the new SSRC, so the new SSRC is anyway
effectively a new source.

3.2.3. Contributing Source (CSRC)

The Contributing Source (CSRC) is not a separate identifier. Rather
a synchronisation source identifier is listed as a CSRC in the RTP
header of a packet generated by an RTP mixer if the corresponding
SSRC was in the header of one of the packets that contributed to the
mix.

It is not possible, in general, to extract media represented by an
individual CSRC since it is typically the result of a media mixing
(merge) operation by an RTP mixer on the individual media streams
corresponding to the CSRC identifiers. The exception is the case
when only a single CSRC is indicated as this represent forwarding of
a media stream, possibly modified. The RTP header extension for
Mixer-to-Client Audio Level Indication [RFC6465] expands on the
receivers information about a packet with a CSRC list. Due to these
restrictions, CSRC will not be considered a fully qualified
multiplexing point and will be disregarded in the rest of this
document.

3.2.4. RTP Payload Type

Each Media Stream utilises one or more RTP payload formats. An RTP
payload format describes how the output of a particular media codec
is framed and encoded into RTP packets. The payload format used is
identified by the payload type field in the RTP data packet header.
The combination therefore identifies a specific Media Stream encoding
format. The format definition can be taken from [RFC3551] for
statically allocated payload types, but ought to be explicitly
defined in signalling, such as SDP, both for static and dynamic
Payload Types. The term OCGBPformatOCOe here includes whatever can
be described by out-of-band signalling means. In SDP, the term
OCGBPformatOCOe includes media type, RTP timestamp sampling rate,
codec, codec configuration, payload format configurations, and
various robustness mechanisms such as redundant encodings [RFC2198].

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

Other aspects of RTP payload format use are described in RTP Payload
HowTo [RFC8088].

The payload type is not a multiplexing point at the RTP layer (see
Appendix A for a detailed discussion of why using the payload type as
an RTP multiplexing point does not work). The RTP payload type is,
however, used to determine how to render a media stream, and so can
be viewed as selecting a rendering context. The rendering context
can be defined by the signalling, and the RTP payload type number is
sometimes used to associate an RTP media stream with the signalling.
This association is possible provided unique RTP payload type numbers
are used in each context. For example, an RTP media stream can be
associated with an SDP OCGBPm=OCOe line by comparing the RTP payload
type numbers used by the media stream with payload types signalled in
the OCGBPa=rtpmap:OCOe lines in the media sections of the SDP. If
RTP media streams are being associated with signalling contexts based
on the RTP payload type, then the assignment of RTP payload type
numbers needs to be unique across signalling contexts; if the same
RTP payload format configuration is used in multiple contexts, then a
different RTP payload type number has to be assigned in each context
to ensure uniqueness. If the RTP payload type number is not being
used to associated RTP media streams with a signalling context, then
the same RTP payload type number can be used to indicate the exact
same RTP payload format configuration in multiple contexts. In case
of bundled media, Section 10.2 of
[I-D.ietf-mmusic-sdp-bundle-negotiation] provides more information on
SDP signalling.

3.3. Issues Related to RTP Topologies

The impact of how RTP multiplexing is performed will in general vary
with how the RTP Session participants are interconnected, described
by RTP Topology [RFC7667].

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

o Does my communication peer support RTP as defined with multiple
   SSRCs?

o Do I need network differentiation in form of QoS?

o Can the application more easily process and handle the media
   streams if they are in different RTP sessions?

o Do I need to use additional media streams for RTP retransmission
   or FEC.

o etc.

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

Sometimes an RTP communication can end up in a situation when the
peer it is communicating with is not compatible with the other peer
for various reasons:

o No common media codec for a media type thus requiring transcoding

o Different support for multiple RTP sources and RTP sessions

o Usage of different media transport protocols, i.e RTP or other.

o Usage of different transport protocols, e.g. UDP, DCCP, TCP

o Different security solutions, e.g. IPsec, TLS, DTLS, SRTP with
   different keying mechanisms.

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

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

3.4. Issues Related to RTP and RTCP Protocol

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.

3.4.1. The RTP Specification

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

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

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

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

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

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

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

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

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

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

Let's consider one argument at a time. The first is an argument for using different SSRC for each individual media stream, which is very applicable.

The second argument is advocating against using payload type multiplexing, which still stands as can been seen by the extensive list of issues found in Appendix A.

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

The fourth is an argument against multiplexing media streams that require different handling into the same session. As we saw in the discussion of RTP mixers, the RTP mixer has to embed application logic in order to handle streams anyway; the separation of streams according to stream type is just another piece of application logic, which might or might not be appropriate for a particular application. A type of application that can mix different media sources OCGBPblindlyOCOe is the audio only OCGBPtelephoneOCOe bridge; most other type of application needs application-specific logic to perform the mix correctly.

The fifth argument discusses network aspects that we will discuss more below in Section 4.2. It also goes into aspects of

implementation, like decomposed endpoints where different processes
or inter-connected devices handle different aspects of the whole
multi-media session.

A summary of RFC 3550's view on multiplexing is to use unique SSRCs
for anything that is its own media/packet stream, and to use
different RTP sessions for media streams that don't share a media
type. This document supports the first point; it is very valid. The
later is one thing which needs to be further discussed, as imposing a
single solution on all usages of RTP is inappropriate. Multiple
Media Types in an RTP Session specification
[I-D.ietf-avtcore-multi-media-rtp-session] provides a detailed
analysis of the potential issues in having multiple media types in
the same RTP session. This document tries to provide an wider scoped
consideration regarding the usage of RTP session and considers
multiple media types in one RTP session as possible choice for the
RTP application designer.

3.4.2. Multiple SSRCs in a Session

Using multiple SSRCs in an RTP session at one endpoint requires
resolving some unclear aspects of the RTP specification. These could
potentially lead to some interoperability issues as well as some
potential significant inefficiencies. These are further discussed in
OCGBPRTP Considerations for Endpoints Sending Multiple Media
StreamsOCOe [RFC8108]. A application designer needs to consider
these issues and the impact availability or lack of the optimization
in the endpoints has on their application.

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

3.4.3. Binding Related Sources

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

The solutions can be divided into some groups, RTP/RTCP based,
Signalling based (SDP), grouping related RTP sessions, and grouping
SSRCs within an RTP session. Most solutions are explicit, but some
implicit methods have also been applied to the problem.

The SDP-based signalling solutions are:

SDP Media Description Grouping: The SDP Grouping Framework [RFC5888]
   uses various semantics to group any number of media descriptions.

These has previously been considered primarily as grouping RTP
sessions, [I-D.ietf-mmusic-sdp-bundle-negotiation] groups multiple
media descriptors as a single RTP session.

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

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

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

Within RTP/RTCP based solutions when binding to an endpoint or
synchronization context, i.e. the CNAME has not been sufficient and
one way to bind related streams in multiple RTP sessions has been to
use the same SSRC value across all the RTP sessions. RTP
Retransmission [RFC4588] is multiple RTP session mode, Generic FEC
[RFC5109], as well as the RTP payload format for Scalable Video
Coding [RFC6190] in Multi Session Transmission (MST) mode uses this
method. This method clearly works but might have some downside in
RTP sessions with many participating SSRCs. The birthday paradox
ensures that if you populate a single session with 9292 SSRCs at
random, the chances are approximately 1% that at least one collision
will occur. When a collision occur this will force one to change
SSRC in all RTP sessions and thus resynchronizing all of them instead
of only the single media stream having the collision. Therefore it
is not recommended to use such method. Using [RFC7656] streams from
the same media source should use the same RTP session.

It can be noted that Section 8.3 of the RTP Specification [RFC3550]
recommends using a single SSRC space across all RTP sessions for
layered coding.

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

[I-D.ietf-mmusic-rid] provides an RTP/RTCP based mechanism capable of
supporting explicit association within an RTP session.

3.4.4. Forward Error Correction

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

   Using multiple RTP sessions supports the case where some set of
   receivers might not be able to utilise the FEC information. By
   placing it in a separate RTP session, it can easily be ignored.

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

4. Particular Considerations for RTP Multiplexing

4.1. Interworking Considerations

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

4.1.1. Types of Interworking

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

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

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

4.1.2. RTP Translator Interworking

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

When one is gatewaying via an RTP Translator, a natural requirement is that the two applications being interconnected need to use the same approach to multiplexing. Furthermore, if one of the applications is capable of working in several modes (such as being able to use Additional SSRCs or Multiple RTP sessions at will), and the other one is not, successful interconnection depends on locking the more flexible application into the operating mode where interconnection can be successful, even if no participants using the less flexible application are present when the RTP sessions are being created.

4.1.3. Gateway Interworking

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

o Generating appropriate RTCP reports for all media streams (possibly based on incoming RTCP reports), originating from SSRCs controlled by the gateway.

o Handling SSRC collision resolution in each application's RTP sessions.

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

For applications that uses any security mechanism, e.g. in the form of SRTP, then the gateway needs to be able to decrypt incoming packets and re-encrypt them in the other application's security context. This is necessary even if all that's needed is a simple remapping of SSRC numbers. If this is done, the gateway also needs to be a member of the security contexts of both sides, of course.

Other tasks a gateway might need to apply include transcoding (for incompatible codec types), rescaling (for incompatible video size requirements), suppression of content that is known not to be handled in the destination application, or the addition or removal of redundancy coding or scalability layers to fit the need of the destination domain.

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

This fact reveals the potential for these gateways to block evolution of the applications by blocking unknown RTP and RTCP extensions that the regular application has been extended with.

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

4.1.4. Multiple SSRC Legacy Considerations

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

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

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

2. Be capable of decoding multiple streams simultaneously.

3. Be capable of rendering multiple streams simultaneously.

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

4.2. Network Considerations

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

4.2.1. Quality of Service

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

However, for a flow based scheme there is a clear difference between the methods. Additional SSRC will result in all media streams being part of the same 5-tuple (protocol, source address, destination address, source port, destination port) which is the most common selector for flow based QoS.

It also needs to be noted that packet marking based QoS mechanisms can have limitations. A general observation is that different DSCP can be assigned to different packets within a flow as well as within an RTP Media Stream. However, care needs to be taken when considering which forwarding behaviours that are applied on path due to these DSCPs. In some cases the forwarding behaviour can result in packet reordering. For more discussion of this see [RFC7657].

More specific to the choice between using one or more RTP session can be the method for assigning marking to packets. If this is done using a network ingress function, it can have issues discriminating the different RTP media streams. The network API on the endpoint also needs to be capable of setting the marking on a per packet basis to reach the full functionality.

4.2.2. NAT and Firewall Traversal

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

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

End-Point Port Consumption: A given IP address only has 65536
    available local ports per transport protocol for all consumers of
    ports that exist on the machine. This is normally never an issue
    for an end-user machine. It can become an issue for servers that
    handle large number of simultaneous streams. However, if the
    application uses ICE to authenticate STUN requests, a server can
    serve multiple endpoints from the same local port, and use the
    whole 5-tuple (source and destination address, source and
    destination port, protocol) as identifier of flows after having
    securely bound them to the remote endpoint address using the STUN
    request. In theory the minimum number of media server ports
    needed are the maximum number of simultaneous RTP Sessions a
    single endpoint can use. In practice, implementation will
    probably benefit from using more server ports to simplify
    implementation or avoid performance bottlenecks.

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

NAT Traversal Excess Time: Performing the NAT/FW traversal takes a
    certain amount of time for each flow. It also takes time in a
    phase of communication between accepting to communicate and the
    media path being established which is fairly critical. The best
    case scenario for how much extra time it takes after finding the
    first valid candidate pair following the specified ICE procedures
    are: $1.5*RTT + Ta*(Additional\_Flows-1)$, where Ta is the pacing
    timer, which ICE specifies to be no smaller than 20 ms. That
    assumes a message in one direction, and then an immediate
    triggered check back. The reason it isn't more, is that ICE first
    finds one candidate pair that works prior to attempting to
    establish multiple flows. Thus, there is no extra time until one
    has found a working candidate pair. Based on that working pair
    the needed extra time is to in parallel establish the, in most

cases 2-3, additional flows. However, packet loss causes extra
delays, at least 100 ms, which is the minimal retransmission timer
for ICE.

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

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

Additional SSRC keeps the additional media streams within one RTP
Session and transport flow and does not introduce any additional NAT
traversal complexities per media stream. This can be compared with
normally one or two additional transport flows per RTP session when
using multiple RTP sessions. Additional lower layer transport flows
will be needed, unless an explicit de-multiplexing layer is added
between RTP and the transport protocol. At time of writing no such
mechanism was defined.

4.2.3. Multicast

Multicast groups provides a powerful semantics for a number of real-
time applications, especially the ones that desire broadcast-like
behaviours with one endpoint transmitting to a large number of
receivers, like in IPTV. But that same semantics do result in a
certain number of limitations.

One limitation is that for any group, sender side adaptation to the
actual receiver properties causes degradation for all participants to
what is supported by the receiver with the worst conditions among the
group participants. In most cases this is not acceptable. Instead
various receiver based solutions are employed to ensure that the
receivers achieve best possible performance. By using scalable
encoding and placing each scalability layer in a different multicast
group, the receiver can control the amount of traffic it receives.
To have each scalability layer on a different multicast group, one
RTP session per multicast group is used.

In addition, the transport flow considerations in multicast are a bit
different from unicast; NATs with port translation are not useful in

the multicast environment, meaning that the entire port range of each multicast address is available for distinguishing between RTP sessions.

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

Here are some common behaviours for RTP multicast:

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

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

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

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

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

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

4.3. Security and Key Management Considerations

When dealing with point-to-point, 2-member RTP sessions only, there are few security issues that are relevant to the choice of having one RTP session or multiple RTP sessions. However, there are a few aspects of multiparty sessions that might warrant consideration. For general information of possible methods of securing RTP, please review RTP Security Options [RFC7201].

4.3.1. Security Context Scope

   When using SRTP [RFC3711] the security context scope is important and
   can be a necessary differentiation in some applications. As SRTP's
   crypto suites (so far) are built around symmetric keys, the receiver
   will need to have the same key as the sender. This results in that
   no one in a multi-party session can be certain that a received packet
   really was sent by the claimed sender or by another party having
   access to the key. In most cases this is a sufficient security
   property, but there are a few cases where this does create issues.

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

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

   SRTP [RFC3711] has no special functions for dealing with different
   sets of master keys for different SSRCs. The key-management
   functions have different capabilities to establish different set of
   keys, normally on a per endpoint basis. For example, DTLS-SRTP
   [RFC5764] and Security Descriptions [RFC4568] establish different
   keys for outgoing and incoming traffic from an endpoint. This key
   usage has to be written into the cryptographic context, possibly
   associated with different SSRCs.

4.3.2. Key Management for Multi-party session

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

   Multi-party sessions, such as transport translator based sessions and
   multicast sessions, cannot use Security Description [RFC4568] nor
   DTLS-SRTP [RFC5764] without an extension as each endpoint provides
   its set of keys. In centralised conferences, the signalling
   counterpart is a conference server and the media plane unicast
   counterpart (to which DTLS messages would be sent) is the transport
   translator. Thus an extension like Encrypted Key Transport
   [I-D.ietf-avt-srtp-ekt] is needed or a MIKEY [RFC3830] based solution
   that allows for keying all session participants with the same master
   key.

4.3.3. Complexity Implications

The usage of security functions can surface complexity implications
of the choice of multiplexing and topology. This becomes especially
evident in RTP topologies having any type of middlebox that processes
or modifies RTP/RTCP packets. Where there is very small overhead for
an RTP translator or mixer to rewrite an SSRC value in the RTP packet
of an unencrypted session, the cost of doing it when using
cryptographic security functions is higher. For example if using
SRTP [RFC3711], the actual security context and exact crypto key are
determined by the SSRC field value. If one changes it, the
encryption and authentication tag needs to be performed using another
key. Thus changing the SSRC value implies a decryption using the old
SSRC and its security context followed by an encryption using the new
one.

5. Archetypes

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

5.1. Single SSRC per Session

In this archetype each endpoint in a point-to-point session has only
a single SSRC, thus the RTP session contains only two SSRCs, one
local and one remote. This session can be used both unidirectional,
i.e. only a single media stream or bi-directional, i.e. both
endpoints have one media stream each. If the application needs
additional media flows between the endpoints, they will have to
establish additional RTP sessions.

The Pros:

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

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

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

4. It is possible to control security association per RTP media
    stream with current key-management, since each media stream is

directly related to an RTP session, and the keying operates on a per-session basis.

The Cons:

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

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

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

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

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

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

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

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

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

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

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

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

RTP applications that need to inter-work with legacy RTP
applications, like most deployed VoIP and video conferencing
solutions, can potentially benefit from this structure. However, a
large number of media descriptions in SDP can also run into issues
with existing implementations. For any application needing a larger
number of media flows, the overhead can become very significant.
This structure is also not suitable for multi-party sessions, as any
given media stream from each participant, although having same usage
in the application, needs its own RTP session. In addition, the
dynamic behaviour that can arise in multi-party applications can tax
the signalling system and make timely media establishment more
difficult.

5.2. Multiple SSRCs of the Same Media Type

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

The Pros:

1. Low number of RTP sessions needed compared to single SSRC case.
    This implies:

    * Reduced NAT/FW state

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

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

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

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

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

6. Low overhead for security association establishment.

The Cons:

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

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

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

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

5.3. Multiple Sessions for one Media type

In this archetype one goes one step further than in the above
(Section 5.2) by using multiple RTP sessions also for a single media
type, but still not as far as having a single SSRC per RTP session.
The main reason for going in this direction is that the RTP
application needs separation of the media streams due to their usage.
Some typical reasons for going to this archetype are scalability over
multicast, simulcast, need for extended QoS prioritisation of media
streams due to their usage in the application, or the need for fine-
grained signalling using today's tools.

The Pros:

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

2. Indication of the application's usage of the media stream, where
    multiple different usages exist.

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

4. Enables detailed QoS prioritisation for flow based mechanisms.

5. Works well with de-composite endpoints.

6. Handles dynamic usage of media streams well.

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

8. The scope for who is included in a security association can be
    structured around the different RTP sessions, thus enabling such
    functionality with existing key-management.

The Cons:

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

b. Increased amount of session configuration state.

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

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

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

f. Higher overhead for security association establishment.

g. If the applications need finer control than on media type level
    over which session participants that are included in different
    sets of security associations, most of today's key-management
    will have difficulties establishing such a session.

For more complex RTP applications that have several different usages
for media streams of the same media type and / or uses scalability or
simulcast, this solution can enable those functions at the cost of
increased overhead associated with the additional sessions. This
type of structure is suitable for more advanced applications as well

as multicast based applications requiring differentiation to
different participants.

5.4. Multiple Media Types in one Session

This archetype is to use a single RTP session for multiple different
media types, like audio and video, and possibly also transport
robustness mechanisms like FEC or Retransmission. Each media stream
will use its own SSRC and a given SSRC value from a particular
endpoint will never use the SSRC for more than a single media type.

The Pros:

1. Single RTP session which implies:

    * Minimal NAT/FW state.

    * Minimal NAT/FW Traversal Cost.

    * Fate-sharing for all media flows.

2. Enables separation of the different media types based on the
    payload types so media type specific endpoint or central
    processing can still be supported despite single session.

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

4. Minimal overhead for security association establishment.

The Cons:

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

b. Negotiation of bandwidth for the different media types is
    currently not possible in SDP. This requires SDP extensions to
    enable payload or source specific bandwidth. Likely to be a
    problem due to media type asymmetry in needed bandwidth.

c. Not suitable for de-composite endpoints.

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

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

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

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

   h.   If the applications need finer control over which session
        participants that are included in different sets of security
        associations, most key-management will have difficulties
        establishing such a session.

5.5. Summary

   There are some clear relations between these archetypes. Both the
   OCGBPsingle SSRC per RTP sessionOCOe and the OCGBPmultiple media
   types in one sessionOCOe are cases which require full explicit
   signalling of the media stream relations. However, they operate on
   two different levels where the first primarily enables session level
   binding, and the second needs to do it all on SSRC level. From
   another perspective, the two solutions are the two extreme points
   when it comes to number of RTP sessions needed.

   The two other archetypes OCGBPMultiple SSRCs of the Same Media
   TypeOCOe and OCGBPMultiple Sessions for one Media TypeOCOe 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.

6. Summary considerations and guidelines

6.1. Guidelines

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

   Do not Require the same SSRC across Sessions: As discussed in
      Section 3.4.3 there exist drawbacks in using the same SSRC in
      multiple RTP sessions as a mechanism to bind related media streams
      together. It is instead suggested that a mechanism to explicitly
      signal the relation is used, either in RTP/RTCP or in the used
      signalling mechanism that establishes the RTP session(s).

   Use additional SSRCs for additional Media Sources: In the cases
      where an RTP endpoint needs to transmit additional media streams
      of the same media type in the application, with the same
      processing requirements at the network and RTP layers, it is
      suggested to send them as additional SSRCs in the same RTP
      session. For example a telepresence room where there are three
      cameras, and each camera captures 2 persons sitting at the table,
      sending each camera as its own SSRC within a single RTP session is
      suggested.

   Use additional RTP sessions for streams with different requirements:

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

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

   RTP/RTCP Extensions May Support Additional SSRCs as well as Multiple
   RTP sessions:
      When defining an RTP or RTCP extension, the creator needs to
      consider if this extension is applicable to usage with additional
      SSRCs and Multiple RTP sessions. Any extension intended to be
      generic is suggested to support both. Applications that are not
      as generally applicable will have to consider if interoperability
      is better served by defining a single solution or providing both
      options.

   Transport Support Extensions: When defining new RTP/RTCP extensions
      intended for transport support, like the retransmission or FEC

mechanisms, they are expected to include support for both
additional SSRCs and multiple RTP sessions so that application
developers can choose freely from the set of mechanisms without
concerning themselves with which of the multiplexing choices a
particular solution supports.

7. Open Issues

   There are currently some issues that needs to be resolved before this
   document is ready to be published:

   1. Use of RFC 2119 language is section on SSRC (3.2.2)

   2. Better align source and sink terminolgy with Taxonomy
       (Section 3.2.2)

   3. Section on Binding Related Sources (Section 3.4.3) needs more
       text on usage of the RID and other SDES based mechanisms created.

   4. Does the MSID text need to be updated and clarified based on the
       evoulsion of MSID since previous version. Section 3.4.3.

   5. Section 4.1.2 (RTP Translator Interworking) needs to be updated.
       It is not obvious that it is a natural requirement that the same
       multiplexing is used. This needs better discussion.

   6. Refernce to Ta for ICE being 20 ms will need to be updated due to
       ICE update.

   7. In Section 4.3.2 (Key Management for Multi-party session) the
       reference to EKT needs to be updated, question is if draft-ietf-
       perc-ekt-diet is appropriate here?

   8. Can we find a more approriate term than archetypes?

   9.

8. IANA Considerations

   This document makes no request of IANA.

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

9. Security Considerations

   There is discussion of the security implications of choosing SSRC vs
   Multiple RTP session in Section 4.3.

10. References

10.1. Normative References

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

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

10.2. Informative References

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

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

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

   [I-D.ietf-mmusic-msid]
              Alvestrand, H., "WebRTC MediaStream Identification in the
              Session Description Protocol", draft-ietf-mmusic-msid-16
              (work in progress), February 2017.

   [I-D.ietf-mmusic-rid]
             Thatcher, P., Zanaty, M., Nandakumar, S., Burman, B.,
             Roach, A., and B. Campen, "RTP Payload Format
             Restrictions", draft-ietf-mmusic-rid-11 (work in
             progress), July 2017.

   [I-D.ietf-mmusic-sdp-bundle-negotiation]
             Holmberg, C., Alvestrand, H., and C. Jennings,
             "Negotiating Media Multiplexing Using the Session
             Description Protocol (SDP)", draft-ietf-mmusic-sdp-bundle-
             negotiation-39 (work in progress), August 2017.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

   [RFC6190] Wenger, S., Wang, Y., Schierl, T., and A. Eleftheriadis,
              "RTP Payload Format for Scalable Video Coding", RFC 6190,
              DOI 10.17487/RFC6190, May 2011,
              <https://www.rfc-editor.org/info/rfc6190>.

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

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

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

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

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

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

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

Appendix A. Dismissing Payload Type Multiplexing

   This section documents a number of reasons why using the payload type
   as a multiplexing point for most things related to multiple streams
   is unsuitable. If one attempts to use Payload type multiplexing
   beyond it's defined usage, that has well known negative effects on
   RTP. To use Payload type as the single discriminator for multiple
   streams implies that all the different media streams are being sent
   with the same SSRC, thus using the same timestamp and sequence number
   space. This has many effects:

   1.   Putting restraint on RTP timestamp rate for the multiplexed
        media. For example, media streams that use different RTP
        timestamp rates cannot be combined, as the timestamp values need
        to be consistent across all multiplexed media frames. Thus
        streams are forced to use the same rate. When this is not
        possible, Payload Type multiplexing cannot be used.

   2.   Many RTP payload formats can fragment a media object over
        multiple packets, like parts of a video frame. These payload
        formats need to determine the order of the fragments to
        correctly decode them. Thus it is important to ensure that all
        fragments related to a frame or a similar media object are
        transmitted in sequence and without interruptions within the
        object. This can relatively simple be solved on the sender side
        by ensuring that the fragments of each media stream are sent in
        sequence.

   3.   Some media formats require uninterrupted sequence number space
        between media parts. These are media formats where any missing
        RTP sequence number will result in decoding failure or invoking
        of a repair mechanism within a single media context. The text/
        T140 payload format [RFC4103] is an example of such a format.
        These formats will need a sequence numbering abstraction

function between RTP and the individual media stream before
being used with Payload Type multiplexing.

4.   Sending multiple streams in the same sequence number space makes
     it impossible to determine which Payload Type and thus which
     stream a packet loss relates to.

5.   If RTP Retransmission [RFC4588] is used and there is a loss, it
     is possible to ask for the missing packet(s) by SSRC and
     sequence number, not by Payload Type. If only some of the
     Payload Type multiplexed streams are of interest, there is no
     way of telling which missing packet(s) belong to the interesting
     stream(s) and all lost packets need be requested, wasting
     bandwidth.

6.   The current RTCP feedback mechanisms are built around providing
     feedback on media streams based on stream ID (SSRC), packet
     (sequence numbers) and time interval (RTP Timestamps). There is
     almost never a field to indicate which Payload Type is reported,
     so sending feedback for a specific media stream is difficult
     without extending existing RTCP reporting.

7.   The current RTCP media control messages [RFC5104] specification
     is oriented around controlling particular media flows, i.e.
     requests are done addressing a particular SSRC. Such mechanisms
     would need to be redefined to support Payload Type multiplexing.

8.   The number of payload types are inherently limited.
     Accordingly, using Payload Type multiplexing limits the number
     of streams that can be multiplexed and does not scale. This
     limitation is exacerbated if one uses solutions like RTP and
     RTCP multiplexing [RFC5761] where a number of payload types are
     blocked due to the overlap between RTP and RTCP.

9.   At times, there is a need to group multiplexed streams and this
     is currently possible for RTP Sessions and for SSRC, but there
     is no defined way to group Payload Types.

10. It is currently not possible to signal bandwidth requirements
     per media stream when using Payload Type Multiplexing.

11. Most existing SDP media level attributes cannot be applied on a
     per Payload Type level and would require re-definition in that
     context.

12. A legacy endpoint that does not understand the indication that
     different RTP payload types are different media streams might be

slightly confused by the large amount of possibly overlapping or
identically defined RTP Payload Types.

Appendix B. Signalling considerations

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

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

B.1. Signalling Aspects

There exist various signalling solutions for establishing RTP
sessions. Many are SDP [RFC4566] based, however SDP functionality is
also dependent on the signalling protocols carrying the SDP. Where
RTSP [RFC7826] and SAP [RFC2974] both use SDP in a declarative
fashion, while SIP [RFC3261] uses SDP with the additional definition
of Offer/Answer [RFC3264]. The impact on signalling and especially
SDP needs to be considered as it can greatly affect how to deploy a
certain multiplexing point choice.

B.1.1. 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:

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

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

Some of these issues are clearly SDP's problem rather than RTP
limitations. However, if the aim is to deploy an solution using
additional SSRCs that contains several sets of media streams with
different properties (encoding/packetization parameter, bit-rate,

etc.), putting each set in a different RTP session would directly
enable negotiation of the parameters for each set. If insisting on
additional SSRC only, a number of signalling extensions are needed to
clarify that there are multiple sets of media streams with different
properties and that they need in fact be kept different, since a
single set will not satisfy the application's requirements.

For some parameters, such as resolution and framerate, a SSRC-linked
mechanism has been proposed:
[I-D.lennox-mmusic-sdp-source-selection].

B.1.2. SDP Prevents Multiple Media Types

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

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

B.1.3. Signalling Media Stream Usage

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

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

If all media streams in an RTP session are to be treated in the same
way, identifying the session is enough. If SSRCs in a session are to

be treated differently, signalling needs to identify both the session
and the SSRC.

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

Authors' Addresses

Magnus Westerlund
Ericsson
Torshamsgatan 23
SE-164 80 Kista
Sweden

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

Bo Burman
Ericsson
Farogatan 6
SE-164 80 Kista
Sweden

Phone: +46 10 714 13 11
Email: bo.burman@ericsson.com

Colin Perkins
University of Glasgow
School of Computing Science
Glasgow G12 8QQ
United Kingdom

Email: csp@csperkins.org

Harald Tveit Alvestrand
Google
Kungsbron 2
Stockholm 11122
Sweden

Email: harald@alvestrand.no

   Roni Even
   Huawei

   Email: roni.even@huawei.com

   Hui Zheng
   Huawei

   Email: marvin.zhenghui@huawei.com