Multiplexing Multiple RTP Sessions onto a Single Lower-Layer Transport
draft-westerlund-avtcore-transport-multiplexing-07

Abstract

   This memo defines a mechanism to allow multiple RTP sessions to be
   multiplexed onto a single lower-layer transport flow (e.g., onto a
   single UDP 5-tuple). Requirements for multiplexing RTP sessions are
   discussed, along with the trade-off between the different options. A
   shim-based multiplexing layer is proposed, along with associated
   signalling.

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 April 24, 2014.

Copyright Notice

Table of Contents

1. Introduction

   With the ongoing development of the WebRTC conferencing and CLUE
   telepresence standards, there is renewed interest in defining a
   mechanism that allows multiple RTP sessions [RFC3550] to share a
   single lower layer transport, such as a bi-directional UDP flow. The
   main problem driving this is the cost of doing NAT/firewall traversal
   for each individual RTP flow. ICE and other NAT/firewall traversal
   solutions are clearly capable of attempting to open multiple flows.
   However, there is both increased risk for failure, and an increased
   cost in the creation of multiple flows. The increased cost comes as
   slightly higher delay in establishing the traversal, and the amount
   of consumed NAT/firewall resources. The latter might be an
   increasing problem in the IPv4 to IPv6 transition period.

   There is ongoing work on specifying how and when one RTP session can
   contain multiple media types
   [I-D.ietf-avtcore-multi-media-rtp-session]. That addresses certain
   use cases, while this proposal addresses a different set of use cases
   and motivations (discussed further in Section 3). The classical
   method of having each RTP session run over a specific transport flow
   is still motivated for a number of use cases, especially when flow
   based QoS is to be used for some media streams.

   This memo draws up some requirements for consideration on how to
   transport multiple RTP sessions over a single lower-layer transport.
   These requirements have to be weighted carefully, as no known
   solution exists that can fulfil the combined set of requirements
   completely. A number of possible solutions where considered and
   discussed with respect to their properties. Based on that, this memo
   defines a multiplexing shim, along with SDP signalling, and examples.
   The other considered proposals and the comparison is available as
   appendices.

2. Terminology

   Unless specifically noted, all mentioning of multiplexing in this
   memo refer to the multiplexing of multiple RTP Sessions onto the same
   lower layer transport. It is important to make this distinction as
   RTP contains a number of multiplexing points for various purposes,
   such as media formats (Payload Type), media sources (SSRC), and RTP
   sessions.

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].

3. Motivation

RTP has always allowed applications to use of multiple RTP sessions,
by using different transport-layer flows for each session [RFC3550].
The primary motivation was to support differential quality of service
per session, using flow-level differentiated services mechanisms, but
it also lets applications separate flows into several RTP sessions to
better reflect application-level semantics where appropriate.

More recently, there has been a desire to send multiple types of
media in a single RTP session. This uses one RTP session instead of
several RTP sessions, giving up flow-level quality of service, and
semantic separation of traffic, but reducing the number of transport
level flows to ease NAT and firewall traversal. Clarifications to
the RTP specification to support this can be found in
[I-D.ietf-avtcore-multi-media-rtp-session].

There is also a third option that can be useful in some cases. This
is to somehow multiplex several RTP sessions onto a single transport
layer flow. The motivations for why this alternative is needed are
as follows.

To Ease NAT and Firewall Traversal: The existence of network address
    translation (NAT/NAPT) and firewalls on almost all Internet access
    has implications for protocols, such as RTP, that were designed to
    use multiple transport-layer flows. Any NAT or firewall traversal
    solution has to to ensure that all the necessary transport-layer
    flows are established. This has three impacts:

    1. Increased delay to perform the transport flow establishment

    2. The more transport flows, the more state and the more resource
        consumption in the NAT and Firewalls. When the resource
        consumption in NAT/firewalls reaches their limits, unexpected
        behaviours usually occur. Commonly resulting in service
        disruptions.

    3. More transport flows means a higher risk that some transport
        flow fails to be established, thus preventing the application
        to communicate.

Using fewer transport-layer flows, by multiplexing several RTP
sessions onto a single transport-layer flow, reduces the risk of
communication failure, improves establishment behaviour, and
reduces the load on NATs and firewalls.

To Support Application-level Session-layer Semantics: Applications
can use multiple RTP sessions to separate media streams that have
different uses or purposes. For example, a group conferencing
application might use one RTP session to distribute high-quality
video of the active speaker, switching the source of that video as
the conversation progresses, coupled with a second RTP session to
send always-on low-quality views of the inactive speakers, making
it easier of the MCU to manage the traffic. Separation of flows
into different RTP sessions also allows different processing based
on the media type, such as audio and video, in end-points and
middleboxes. This can give middleboxes the knowledge that any
SSRC within the session is supposed to be processed in a similar
way, saving them the need to perform differential processing on a
per-SSRC basis.

Not all applications need to separate their traffic into different
semantic classes. And, for those that do, it is clearly possible
to find other multiplexing solutions for many simpler cases, for
example based on signalled semantics for SSRC, or looking at the
payload type and differences in encoding. This lack of semantic
separation for some flows becomes more critical as the application
semantics get more complex. For example, an application that has
one set of video streams showing session participants, and another
set that shares an application or presentation slides, would
likely want to separate those streams for reasons such as control,
prioritization, QoS, methods for robustness, etc. In those cases,
using the RTP session for separation of flows with different
semantics is a powerful tool that can ease the application design,
and something that we would like to preserve when providing a
solution for how to use only a single lower-layer transport.

Multiplexing and the use of different RTP session is discussed
further in [I-D.ietf-avtcore-multiplex-guidelines].

To Allow Use of Certain RTP Extensions: Different applications use
different sets of RTP extensions. Several of these extensions are
known to have limitations that prevent them from being used in RTP
sessions that carry different types of media. This is discussed
more in [I-D.ietf-avtcore-multi-media-rtp-session]. The
extensions that are known to be problematic include parity FEC
[RFC5109], RTP Retransmission in session mode [RFC4588], and some
forms of layered coding. This prevents some applications from
sending multiple types of media in a single RTP session, forcing

them to use multiple RTP sessions. To prevent those applications
from having to use several transport-layer flows for the different
RTP sessions, it is desirable to have a way of multiplexing
several RTP sessions on a single transport-layer flow.

The centre of the motivation is to ensure that the use of multiple
RTP sessions is available, and usable, for applications that have no
need for transport-layer separation of their media streams and want
to reduce their exposure to any NAT or Firewall inconsistencies and
minimize the resource consumption. As a benefit, a well designed
solution will remove the limitations on what existing RTP mechanisms
or extensions that can be used by the application, when compared to
sending multiple media types in a single RTP session.

4. Requirements

This section lists and discusses a number of potential requirements.
However, it is not difficult to realize that it is in fact possible
to put requirements that makes the set of feasible solutions an empty
set. It is thus necessary to consider which requirements that are
essential to fulfil and which can be compromised on to arrive at a
solution.

Support Use of Multiple RTP Sessions: As stated in the RTP
   specification [RFC3550], "The distinguishing feature of an RTP
   session is that each maintains a full, separate space of SSRC
   identifiers [...]. The set of participants included in one RTP
   session consists of those that can receive an SSRC identifier
   transmitted by any one of the participants either in RTP as the
   SSRC or a CSRC [...] or in RTCP". Accordingly, any mechanism to
   multiplex several RTP sessions onto a single transport-layer flow
   needs to allow each RTP session to use the complete SSRC space,
   independent of any other RTP sessions multiplexed onto that
   transport-layer flow.

   As a corollary of the above, two different RTP sessions that are
   being multiplexed onto the same transport-layer flow need to be
   able to use the same SSRC value. This is a absolute requirement,
   for two reasons. Firstly, to avoid mandating SSRC assignment
   rules that are coordinated between the sessions. If the RTP
   sessions multiplexed together need to have unique SSRC values,
   then additional code that works between RTP Sessions is needed in
   the implementations. Thus raising the bar for implementing this
   solution. In addition, if one gateways between parts of a system
   using this multiplexing and parts that aren't multiplexing, the
   part that isn't multiplexing also needs to fulfil the requirements
   on how SSRC is assigned or force the gateway to translate SSRCs.
   Translating SSRC is actually hard as it requires one to understand

the semantics of all current and future RTP and RTCP extensions.
Otherwise a barrier for deploying new extensions is created.
Second, there are some few RTP extensions that currently rely on
being able to use the same SSRC in different RTP sessions,
including parity FEC [RFC5109], RTP Retransmission in session mode
[RFC4588], and some forms of layered coding.

Support the Secure RTP (SRTP) Profile: SRTP [RFC3711] is one of the
most commonly used security solutions for RTP. In addition, it is
the only one defined by IETF that is integrated into RTP. This
integration has several aspects that needs to be considered when
designing a solution for multiplexing RTP sessions on the same
lower layer transport.

Determining Crypto Context: SRTP first of all needs to know which
session context a received or to-be-sent packet relates to.
It also normally relies on the lower layer transport to
identify the session. It uses the Master Key Indicator
(MKI), if present, to determine which key set is to be used.
Then the SSRC and sequence number are used by most crypto
suites, including the most common use of AES Counter Mode,
to actually generate the correct cipher stream.

Unencrypted Headers: SRTP has chosen to leave the RTP headers and
the first two 32-bit words of the first RTCP header
unencrypted, to allow for both header compression and
monitoring to work also in the presence of encryption. As
these fields are in clear text they are used in most crypto
suites for SRTP to determine how to protect or recover the
plain text.

It is here important to contrast SRTP against a set of other
possible protection mechanisms. DTLS, TLS, and IPsec are all
protecting and encapsulating the entire RTP and RTCP packets.
They don't perform any partial operations on the RTP and RTCP
packets. Any change that is considered to be part of the RTP and
RTCP packet is transparent to them, but possibly not to SRTP.
Thus the impact on SRTP operations has to be considered when
defining a mechanism.

Support Legacy Implementations of RTP and RTCP: The core of RTP is
in use in many systems, and has an extremely large deployed base
with numerous implementations. Changing any of the RTP or RTCP
packet definitions, outside of defined extension points, is highly
problematic. First of all, the implementations need to change to
support this new semantics. Secondly, you get a large transition
period when you have some session participants that support the
new semantics and some that don't. Combing the two behaviours in

the same session can force the deployment of costly and less than
perfect translation devices.

Support NAT and Firewall Traversal: It is desirable that current NAT
   devices, firewalls, and application level gateways will accept
   multiplexed packets from several RTP sessions as they accept
   normal RTP packets. However, in the authors' opinion we can't let
   the firewall stifle invention and evolution of the protocol. It
   is also necessary to be aware that a change that will make most
   deep inspecting firewall consider the packet as not valid RTP/RTCP
   will have a more difficult deployment story.

Support Monitors and Reporting Tools: It is desirable that a third
   party monitor can still operate on the multiplexed RTP Sessions.
   It is however likely that they will require an update to correctly
   monitor and report on multiplexed RTP Sessions.

   Another type of function to consider is packet sniffers and their
   selector filters. These can be impacted by a change of the
   fields. An observation is that many such systems are usually
   quite rapidly updated to consider new types of standardized or
   simply common packet formats.

Support Use of IP Multicast: It is desirable that a solution can be
   used if RTP and RTCP packets are sent over multicast, both Any
   Source Multicast (ASM) and Single Source Multicast (SSM). The
   reason for this requirement is to allow a system using RTP to use
   the same configuration regardless of the transport being done over
   unicast or multicast. In addition, multicast can't be claimed to
   have an issue with using multiple ports, as each multicast group
   has a complete port space scoped by address.

Support Incremental Deployment: A good solution has the property
   that in topologies that contains RTP mixers or Translators, a
   single session participant can enable multiplexing without having
   any impact on any other session participants. Thus a node ought
   to be able to take a multiplexed packet and then easily send it
   out with minimal or no modification on another leg of the session,
   where each RTP session is transported over its own lower-layer
   transport. It also needs to be as easy to do the reverse
   forwarding operation.

5. Design Considerations

We propose a solution based around a shim layer, inserted between the
transport layer headers and the RTP layer headers, to demultiplex
separate RTP sessions. The design rationale for using a shim layer
header, as opposed to other demultiplexing points, is discussed in

Appendix A. In the following we discuss design considerations
regarding placement and use of the shim layer header.

5.1. Location of Multiplexing Shim Header

A major question affecting the SHIM is the location of the SHIM
header providing the Identifier of the session the packet relate to.
This section will discuss in detail about the impact of making the
different choices.

Identified aspects to consider are:

Possibility to Process: A prefixed shim header, i.e. between the
   transport protocol and the RTP/RTCP packet header has the
   advantage that any node on the network that likes to include the
   header in any per-packet processing can reach it. Reasons for
   per-packet processing are:

   a. Quality of Service classification

   b. SHIM ingress or egress

   c. Monitoring

   Many routers or similar devices can only read and process the
   first N bytes of the whole packet, where N is commonly on the
   order of 64-128 bytes. Any other type of processing means putting
   the packet on the slow path. Thus a prefixed solution enables
   this processing while a postfixed solution will most likely
   forever prevent this type of devices to process it.

Legacy Processing: RTP packets contain very few fixed bits and are
   difficult to distinguish using deep packet inspection without
   access to the signalling channel, or without keeping per-flow
   state to correlate changes in the (presumed) RTP headers across
   packets to gain confidence that the flow is of the expected type.
   Firewalls, application-level gateways, and other network entities
   that concern themselves with trying to track RTP flows will need
   to be updated. This can create a barrier to deployment. Using a
   postfix shim likely gives the least resistance for initial
   deployment. However, even with a postfix shim, deployment can be
   hindered when multiple RTP sessions using the same SSRC values,
   since this will appear to give irregular behaviour of the fields
   for what the third party believes is one media stream, when it is
   actually several multiple streams. The use of a prefixed shim
   will however maintain the long-term capabilities of such devices
   assuming they can be updated to include the SHIM header as part of
   the classification.

Header Compression: The different header compression techniques that
   has been developed compresses IP/UDP/RTP as complete combination.
   If one instead have a IP/UDP/SHIM/RTP then the compression for the
   full set might not work or poorly. Instead only IP/UDP header
   compression is likely to be applied. Thus a prefix will loose
   some compression efficiency until compression profiles for IP/UDP/
   SHIM/RTP has been developed, implemented and deployed. Postfix
   don't have that issue, but nor can it ever gain anything from
   header compression which an prefixed solution could once an
   updated profile is deployed. Postfix also will have reduced
   efficiency compressing sessions when the same SSRC is used in two
   different RTP sessions as the RTP header fields like sequence
   number, etc., will not behave as expected and need frequent
   explicit updates.

The question of a prefixed or a postfixed shim header comes down to a
trade-off between long term usability and deployment issues. A
prefixed shim offers a good long term possibility to adapt any
network function that needs to take the shim header into account, but
at the same time any function that tries to analyse packets might
block the packets and hinder deployment. A postfixed shim will
likely have the best short-term deployment possibilities, but long
term this choice will likely prevent many network nodes that like to
be capable of separating the RTP sessions being multiplexed together
from successfully doing that. After discussion in the working group
it has been determined that a prefixed shim is the preferred
solution.

5.2. ICE and DTLS-SRTP Integration

When using ICE [RFC5245] or DTLS-SRTP [RFC5764] or both with RTP
there exist the issue that RTP, STUN [RFC5389] and DTLS-SRTP are
simultaneously in use over the same lower layer transport flow, like
UDP. This multiplexing is based on the value of the first byte of
the lower layer transport payload as discussed in Section 5.1.2 of
DTLS-SRTP [RFC5764].

The replacement of a single RTP session with the multiple RTP
sessions identified by a SHIM ought not be misidentified to be either
STUN or DTLS-SRTP or any other protocol intending to take the
available free code-points in the range 193-255 (Decimal). Thus a
prefixed SHIM needs to have its first byte have the two first bits
set to 10 (Binary). Having the SHIM share the identity of RTP is not
an issue as there has to be mutual agreement that the SHIM is used
instead of RTP.

5.3. Signalling Fall Back

Both SIP and WebRTC applications use SDP signalling to describe the
RTP sessions and transport layer connections used in a call. It is
therefore necessary to consider how to signal multiple RTP sessions
multiplexed onto a single lower layer transport within SDP. It is
also important to consider backwards compatibility with any legacy
applications that do not understand any proposed SDP extension.

An SDP session description is built up using media ("m=") lines
describing media flows, with associated connection ("c=") lines
describing the transport layer flows. In the usual offer/answer use
of SDP the communicating parties use a single c= line to represent
the IP-layer path, with one m= line per type of media, running each
type of media on a separate transport layer port, and hence a
separate RTP session. This gives a clean separation of RTP sessions,
but requires multiple transport layer flows to be used, complicating
NAT/firewall traversal.

The SDP bundle extension [I-D.ietf-mmusic-sdp-bundle-negotiation]
provides a way to signal that several m= lines are to be bundled
together into a single RTP session running on a single transport
layer port. This is essentially the opposite semantic to the one we
want: it combines seemingly disparate RTP sessions into one using a
single transport layer flow, while we seek to use a single transport
layer flow, but keep the sessions separate. Accordingly, we do not
re-use the bundle mechanism.

We do, however, want to allow the case where an application would
prefer to use separate RTP sessions multiplexed over a single lower
layer transport, because that simplifies processing, but fall back to
using the bundle mechanism if necessary. Similarly, fall back to
using separate RTP sessions on separate transport layer flows needs
to be supported.

6. Specification

This section contains the specification of the RTP session
multiplexing SHIM, using an explicit session identifier of the
encapsulated payload.

6.1. Shim Layer

This solution is based on a shim layer that is inserted in the stack
between the RTP and RTCP packets and the transport layer being used
by the RTP sessions. Thus the layering is as shown in Figure 1.

```
             +-------------------------+
             |    RTP / RTCP Packet    |
             +-------------------------+
```

```
                         |      Session ID Layer     |
                         +-------------------------+
                         | Transport Layer Header  |
                         +-------------------------+
                         |  Network Layer Header   |
                         +-------------------------+
```
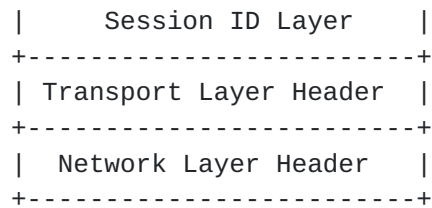
                Figure 1: Stack view with session ID layer shim

   The above stack is in fact a layered one as it does allow multiple
   RTP Sessions to be multiplexed on top of the Session ID shim layer.
   This enables the example presented in Figure 2 where four sessions,
   S1-S4, are sent over the same Transport layer, and where the Session
   ID layer will combine and encapsulate them with the session ID on
   transmission and separate and decapsulate them on reception.

```
                   +-------------------------+
                   | S1  |  S2  |  S3  |  S4 |
                   +-------------------------+
                   |      Session ID Layer     |
                   +-------------------------+
                   | Transport Layer Header  |
                   +-------------------------+
                   |  Network Layer Header   |
                   +-------------------------+
```
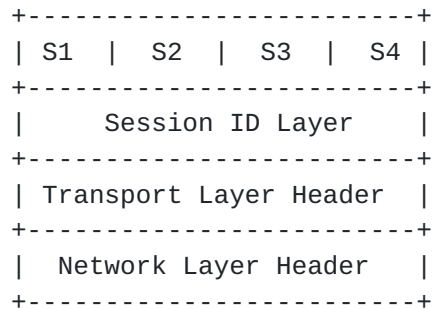
    Figure 2: Example with four RTP sessions on top of session ID layer

   The Session ID layer encapsulates one RTP or RTCP packet from a given
   RTP session and prefixes a 4-octet Session ID layer shim header to
   the packet. The Session ID layer shim header is depicted in Figure 3
   and comprises a 2 bit fixed header (10b), 14 reserved bits, and a 16
   bits unsigned integer field with the Session ID (SID) value.

```
    0                   1                   2                   3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |1 0|        reserved         |        Session ID (SID)        |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```
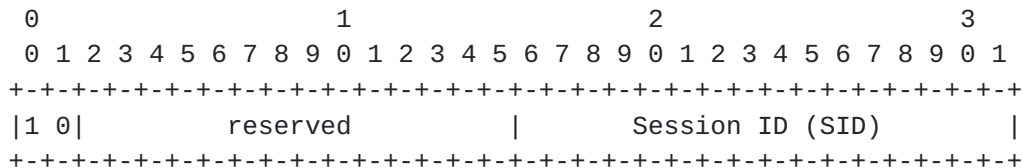
                   Figure 3: Session ID layer shim header

   Each RTP session being multiplexed on top of a given transport layer
   is assigned either a single or a pair of unique SID in the range
   0-65535. The reason for assigning a pair of SIDs to a given RTP
   session are for RTP Sessions that doesn't support "Multiplexing RTP
   Data and Control Packets on a Single Port" [RFC5761] to still be able
   to use a single 5-tuple. The reasons for supporting this extra

functionality is that RTP and RTCP multiplexing based on the payload
type/packet type fields enforces certain restrictions on the RTP
sessions. These restrictions might not be acceptable. As this
solution does not have these restrictions, performing RTP and RTCP
multiplexing in this way has benefits.

Each Session ID value space is scoped by the underlying transport
protocol. Common transport protocols like UDP [RFC0768], DCCP
[RFC4340], TCP [RFC0793], and SCTP [RFC4960] can all be scoped by one
or more 5-tuple (Transport protocol, source address and port,
destination address and port). The case of multiple 5-tuples occur
in the case of multi-unicast topologies, also called meshed
multiparty RTP sessions or in case any application would need more
than 32768 RTP sessions.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|1 0|          reserved           |        Session ID (SID)      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+<+
|V=2|P|X|  CC   |M|     PT        |        sequence number       | |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ |
|                           timestamp                           | |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ |
|           synchronization source (SSRC) identifier           | |
+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+ |
|            contributing source (CSRC) identifiers            | |
|                             ....                             | |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ |
|                  RTP extension (OPTIONAL)                     | |
+>+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ |
| |                        payload  ...                        | |
| |                             +-------------------------------+ |
| |                             | RTP padding   | RTP pad count | |
+>+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+<+
| ~                  SRTP MKI (OPTIONAL)                       ~ |
| +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ |
| :             authentication tag (RECOMMENDED)             : |
| +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ |
+- Encrypted Portion*                     Authenticated Portion ---+
```

        Figure 4: SRTP Packet encapsulated by Session ID Layer

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|1 0|          reserved           |        Session ID (SID)      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+<+
```

```
      |V=2|P|    RC   |  PT=SR or RR |              length         | |
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ |
      |                         SSRC of sender                      | |
   +>+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+ |
   | ~                          sender info                       ~ |
   | +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ |
   | ~                         report block 1                     ~ |
   | +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ |
   | ~                         report block 2                     ~ |
   | +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ |
   | ~                             ...                            ~ |
   | +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ |
   | |V=2|P|    SC   |  PT=SDES=202  |              length         | |
   | +=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+ |
   | |                         SSRC/CSRC_1                         | |
   | +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ |
   | ~                          SDES items                        ~ |
   | +=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+ |
   | ~                             ...                            ~ |
   +>+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+ |
   | |E|                        SRTCP index                       | |
   | +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+<+
   | ~                     SRTCP MKI (OPTIONAL)                   ~ |
   | +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ |
   | :                       authentication tag                  : |
   | +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ |
   +-- Encrypted Portion                    Authenticated Portion -----+
```

           Figure 5: SRTCP packet encapsulated by Session ID layer

   The processing in a receiver when the Session ID layer is present
   will be to

   1. Pick up the packet from the lower layer transport

   2. Inspect the SID field value

   3. Strip the SID field from the packet

   4. Forward it to the (S)RTP Session context identified by the SID
       value

6.2. **Signalling**

   There are several aspects to negotiating the use of multiple RTP
   sessions multiplexing onto a single transport layer flow within SDP.
   Firstly, the SDP offer needs to indicate the desire the use the shim-
   based multiplexing scheme and suggest a transport layer port for the
   multiplex. Then, if the answering party agrees to use the shim, they
   need to agree on the transport layer port to use, and assign session
   ID values for the individual RTP sessions. This all needs to be done
   in a manner that allows graceful fall back to separate RTP sessions,
   or a single bundled RTP session.

   This section defines how to negotiate the use of the Session ID shim
   layer, using the SDP [RFC4566] offer/answer model [RFC3264]. A new
   SDP grouping semantics is defined, "SHIM", along with a new media
   type to represent the shim layer. The grouping semantics allow each
   media description ("m=" line) associated with a 'SHIM' group to be
   identified, and associated with the multiplexed transport flow.

   When it is desired to use multiple RTP sessions multiplexed over a
   single lower layer transport flow, the SDP offer will contain one
   "m=" line for each RTP session, plus one additional "m=" line
   representing the transport layer flow to be used for the multiplex.
   The "m=" lines that represent the media will flows be created as-if
   the multiplex was not present, with transport layer ports assigned in
   the usual manner. The "m=" line representing the multiplex will also
   have a transport layer port assigned, and will use the "application/
   rtp-shim" media type running over UDP (i.e., it will be signalled as
   "m=application <port> udp rtp-shim" in the SDP). All the "m=" lines
   representing the media flows and the multiplexing shim will be part
   of an SDP group, with "SHIM" semantics.

   There MUST be exactly one "m=" line representing an RTP multiplex in
   each "SHIM" group in the SDP offer. If the offer contains more than
   one "m=" line representing an RTP multiplex in a single "SHIM" group,
   then the answering party MUST reject all the RTP multiplexes in that
   "SHIM" group. A "SHIM" group that does not include any "m=" line
   representing an RTP multiplex is malformed; the answering party MUST
   reject all "m=" lines in that "SHIM" group.

   If the answering party does not understand, or does not want to use,
   the RTP multiplexing shim, it will reject the "m=" line for the flow
   representing the multiplex. This is be done by setting the port for
   that "m=" line to zero in the answer. The endpoints will then fall
   back to using separate RTP sessions for each "m=" line, with separate
   transport layer flows for each on the assigned ports.

If the answering party chooses to use the multiplexing shim, it will return an answer that includes a valid port for the multiplex. The ports for the other media lines in the SHIM group that the answering party wants to accept MUST be set to port 9 (the discard port) to indicate that the media for those ports is to be sent as part of the multiplex (the intuition is that the separate port is discarded, and only the multiplex remains). Ports for some "m=" lines in the SHIM group MAY be set to zero to reject some or all of the flows in the group.

> (tbd: it is an open issue whether the answering party is allowed to accept some "m=" lines from the SHIM group into the multiplex while sending others as separate flows on their own ports)

If the multiplex was accepted, multiplexed media corresponding to the "m=" lines whose port was set to 9 in the answer will start to flow. This multiplexed media MUST use the shim on the transport layer ports corresponding to the "m=" line of the multiplexing shim. The session identifiers used in the shim MUST match the ports that were included in the "m=" lines in the offer. The transport layer ports included in those "m=" lines MUST NOT be used for media, and the offering party SHOULD issue a follow-up offer closing down the "m=" lines used for those ports (i.e., setting the ports in their "m=" line to 9) and keeping just the multiplex.

> (tbd: an alternative would be for the answer to reject all except the multiplex stream by setting their ports to zero, but include an attribute for each rejected "m=" line to indicate that if it is to form part of the multiplex. This can perhaps be expected to work better with middleboxes, but is a more significant change to offer/answer processing at the endpoints.)

6.3. SRTP Key Management

Key management for SRTP do needs discussion as we do cause multiple SRTP sessions to exist on the same underlying transport flow. Thus we need to ensure that the key management mechanism still are properly associated with the SRTP session context it intends to key. To ensure that we do look at the three SRTP key management mechanism that IETF has specified, one after another.

6.3.1. Security Description

Session Description Protocol (SDP) Security Descriptions for Media Streams [RFC4568] as being based on SDP has no issue with the RTP session multiplexing on lower layer specified here. The reason is that the actual keying is done using a media level SDP attribute. Thus the attribute is already associated with a particular media

description. A media description that also will have an instance of
the "a=session-mux-id" attribute carrying the SID value/pair used
with this particular crypto parameters.

6.3.2. DTLS-SRTP

Datagram Transport Layer Security (DTLS) Extension to Establish Keys
for the Secure Real-time Transport Protocol (SRTP) [RFC5764] is a
keying mechanism that works on the media plane on the same lower
layer transport that SRTP/SRTCP will be transported over.

The most direct solution would be to use the SHIM and the SID context
identifier to be applied also on DTLS packets. Thus using the same
SID that is used with RTP and/or RTCP also for the DTLS message
intended to key that particular SRTP and/or SRTCP flow(s). This of
course requires independent usage of DTLS-SRTP for each RTP session.
In addition it requires changing the layering for DTLS-SRTP as well
as RTP. Thus this behaviour doesn't gain you anything in regards to
key-management when using SHIM and have some costs.

Instead we propose that an DTLS-SRTP key-derivation change is
introduced. By including the Session ID value in the derivation of
the keying material a single DTLS-SRTP key-management operation could
apply keys and parameters for all the RTP sessions in the same
transport flow. Thus the keying cost is significantly reduced,
especially in regards to network communication and delay impact and
vulnerability to packet loss.

Details to be written up.

6.3.3. MIKEY

MIKEY: Multimedia Internet KEYing [RFC3830] is a key management
protocol that has several transports. In some cases it is used
directly on a transport protocol such as UDP, but there is also a
specification for how MIKEY is used with SDP "Key Management
Extensions for Session Description Protocol (SDP) and Real Time
Streaming Protocol (RTSP)" [RFC4567].

Lets start with the later, i.e. the SDP transport, which shares the
properties with Security Description in that is can be associated
with a particular media description in a SDP. As long as one avoids
using the session level attribute one can be certain to correctly
associate the key exchange with a given SRTP/SRTCP context.

It does appear that MIKEY directly over a lower layer transport
protocol will have similar issues as DTLS.

6.4. Examples

6.4.1. Secure RTP Packet with Multiplexing Shim

The figure below contains an example Secure RTP packet with the RTP
multiplexing shim header, encapsulated by a UDP packet. The RTP
multiplexing shim immediately follows the UDP header, and is followed
by the encapsulated secure RTP packet. The Secure RTP authentication
tag protects the RTP packet only; it does not authenticate the RTP
multiplexing shim or the UDP headers.

```
    0                   1                   2                   3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   | Source Port                   | Destination Port              | U
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ D
   | Length                        | Checksum                      | P
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |1 0|        reserved           |       Session ID (SID)        |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+<+
   |V=2|P|X|  CC   |M|     PT      |       sequence number         | |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ |
   |                           timestamp                           | |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ |
   |           synchronization source (SSRC) identifier           | |
   +=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+ |
   |           contributing source (CSRC) identifiers             | |
   |                             ....                             | |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ |
   |                   RTP extension (OPTIONAL)                    | |
 +>+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ |
 | |                          payload  ...                        | |
 | |                               +-------------------------------+ |
 | |                               | RTP padding   | RTP pad count | |
 +>+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+<+
 | ~                     SRTP MKI (OPTIONAL)                     ~ |
 | +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ |
 | :                 authentication tag (RECOMMENDED)            : |
 | +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ |
 +- Encrypted Portion*                  Authenticated Portion ---+

           SRTP Packet Encapsulated by Session ID Layer
```

6.4.2. **Basic RTP Multiplex Negotiation in SDP**

   This section contains SDP offer/answer examples. In the below SDP
   offer, one audio and one video is being offered. The audio is using
   session identifier 10000, and the video is using session identifier
   10002. If the answer were to reject the "m=application...rtp-shim"
   line, then separate RTP sessions would be set up for the audio and
   video on ports 10000 and 10002 respectively.

   v=0
   o=alice 2890844526 2890844526 IN IP4 atlanta.example.com
   s=
   c=IN IP4 atlanta.example.com
   t=0 0
   a=group:SHIM foo bar baz
   m=audio 10000 RTP/AVP 0 8 97
   b=AS:200
   a=mid:foo
   a=rtpmap:0 PCMU/8000
   a=rtpmap:8 PCMA/8000
   a=rtpmap:97 iLBC/8000
   m=video 10002 RTP/AVP 31 32
   b=AS:1000
   a=mid:bar
   a=rtpmap:31 H261/90000
   a=rtpmap:32 MPV/90000
   m=application 10004 udp rtp-shim
   a=mid:baz

   The SDP answer from an end-point that supports the RTP multiplexing
   shim follows. Note that the ports on the audio and video lines are
   set to 9, to indicate that these flows are included in the multiplex.
   The port of the m= line corresponding to the multiplex is set to the
   transport port used for the multiplex.

   v=0
   o=bob 2808844564 2808844564 IN IP4 biloxi.example.com
   s=
   c=IN IP4 biloxi.example.com
   t=0 0
   a=group:SHIM foo bar baz
   m=audio 9 RTP/AVP 0
   b=AS:200
   a=mid:foo
   a=rtpmap:0 PCMU/8000
   m=video 9 RTP/AVP 32
   b=AS:1000

```
a=mid:bar
a=rtpmap:32 MPV/90000
m=application 10004 udp rtp-shim
a=mid:baz
```

The SDP answer from an end-point that does not support this SHIM.
The ports for the audio and video lines are kept, and the port is set
to 0 in the "m=" line corresponding to the multiplex.

```
v=0
o=bob 2808844564 2808844564 IN IP4 biloxi.example.com
s=
c=IN IP4 biloxi.example.com
t=0 0
a=group:SHIM foo bar baz
m=audio 10000 RTP/AVP 0
b=AS:200
a=mid:foo
a=rtpmap:0 PCMU/8000
m=video 10002 RTP/AVP 32
b=AS:1000
a=mid:bar
a=rtpmap:32 MPV/90000
m=application 0 udp rtp-shim
a=mid:baz
```

6.4.3. Advanced RTP Multiplex Negotiation in SDP

(tbd: add more examples)

7. Open Issues

This work is still at a relatively early phase. This section
contains a list of open issues where the author desires some input.

1. In Section 6.2 there is a discussion of which parameters that
    need to be configured. The scope of these rules and if they do
    make sense needs additional discussion.

2. Can we provide better control so that applications that doesn't
    desire fall back to single RTP session when Multiplexing shim
    fails to be supported but Bundle is supported ends up with a
    better alternative?

3. The details for how to do key-derivation, preferably in such a
    way that it can be reused by multiple key-management solutions
    like MIKEY and DTLS-SRTP

4. The signalling solution will be revisited when the BUNDLE
    solution discussion has yield some result.

8. IANA Considerations

(tbd: register the application/rtp-shim media type)

(tbd: register the "SHIM" semantics for the RTP grouping framework

9. Security Considerations

The security properties of the Session ID layer is depending on what
mechanism is used to protect the RTP and RTCP packets of a given RTP
session. If IPsec or transport layer security solutions such as DTLS
or TLS are being used then both the encapsulated RTP/RTCP packets and
the session ID layer will be protected by that security mechanism.
Thus potentially providing both confidentiality, integrity and source
authentication. If SRTP is used, the session ID layer will not be
directly protected by SRTP. However, it will be implicitly integrity
protected (assuming the RTP/RTCP packet is integrity protected) as
the only function of the field is to identify the session context.
Thus any modification of the SID field will attempt to retrieve the
wrong SRTP crypto context. If that retrieval fails, the packet will
be anyway be discarded. If it is successful, the context will not
lead to successful verification of the packet.

10. Acknowledgements

This memo is based on the input from various people, especially in
the context of the RTCWEB discussion of how to use only a single
lower layer transport. The RTP and RTCP packet figures are borrowed
from RFC3711. The SDP example is extended from the one present in
[I-D.ietf-mmusic-sdp-bundle-negotiation]. Eric Rescorla contributed
the basic idea of optimizing the DTLS-SRTP key-management by
modifying the key derivation process.

The proposal in Appendix A.5 is original suggested by Colin Perkins.
The idea in Appendix A.6 is from an Internet Draft
[I-D.rosenberg-rtcweb-rtpmux] written by Jonathan Rosenberg et. al.
The proposal in Appendix A.3 is a result of discussion by a group of
people at IETF meeting #81 in Quebec.

11. References

11.1. Normative References

[I-D.ietf-mmusic-sdp-bundle-negotiation]
           Holmberg, C., Alvestrand, H., and C. Jennings,
           "Multiplexing Negotiation Using Session Description
           Protocol (SDP) Port Numbers", draft-ietf-mmusic-sdp-
           bundle-negotiation-05 (work in progress), October 2013.

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

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

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

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

[RFC5234] Crocker, D. and P. Overell, "Augmented BNF for Syntax
           Specifications: ABNF", STD 68, RFC 5234, January 2008.

11.2. Informational References

[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-03 (work in
           progress), July 2013.

[I-D.ietf-avtcore-multiplex-guidelines]
           Westerlund, M., Perkins, C., and H. Alvestrand,
           "Guidelines for using the Multiplexing Features of RTP to
           Support Multiple Media Streams", draft-ietf-avtcore-
           multiplex-guidelines-01 (work in progress), July 2013.

[I-D.lennox-rtcweb-rtp-media-type-mux]
           Rosenberg, J. and J. Lennox, "Multiplexing Multiple Media
           Types In a Single Real-Time Transport Protocol (RTP)
           Session", draft-lennox-rtcweb-rtp-media-type-mux-00 (work
           in progress), October 2011.

[I-D.rosenberg-rtcweb-rtpmux]
          Rosenberg, J., Jennings, C., Peterson, J., Kaufman, M.,
          Rescorla, E., and T. Terriberry, "Multiplexing of Real-
          Time Transport Protocol (RTP) Traffic for Browser based
          Real-Time Communications (RTC)", draft-rosenberg-rtcweb-
          rtpmux-00 (work in progress), July 2011.

[RFC0768] Postel, J., "User Datagram Protocol", STD 6, RFC 768,
          August 1980.

[RFC0793] Postel, J., "Transmission Control Protocol", STD 7, RFC
          793, September 1981.

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

[RFC4340] Kohler, E., Handley, M., and S. Floyd, "Datagram
          Congestion Control Protocol (DCCP)", RFC 4340, March 2006.

[RFC4567] Arkko, J., Lindholm, F., Naslund, M., Norrman, K., and E.
          Carrara, "Key Management Extensions for Session
          Description Protocol (SDP) and Real Time Streaming
          Protocol (RTSP)", RFC 4567, 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.

[RFC4960] Stewart, R., "Stream Control Transmission Protocol", RFC
          4960, September 2007.

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

[RFC5245] Rosenberg, J., "Interactive Connectivity Establishment
          (ICE): A Protocol for Network Address Translator (NAT)
          Traversal for Offer/Answer Protocols", RFC 5245, April
          2010.

[RFC5285] Singer, D. and H. Desineni, "A General Mechanism for RTP
          Header Extensions", RFC 5285, July 2008.

   [RFC5389] Rosenberg, J., Mahy, R., Matthews, P., and D. Wing,
              "Session Traversal Utilities for NAT (STUN)", RFC 5389,
              October 2008.

   [RFC5506] Johansson, I. and M. Westerlund, "Support for Reduced-Size
              Real-Time Transport Control Protocol (RTCP): Opportunities
              and Consequences", RFC 5506, April 2009.

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

   [RFC5764] McGrew, D. and E. Rescorla, "Datagram Transport Layer
              Security (DTLS) Extension to Establish Keys for the Secure
              Real-time Transport Protocol (SRTP)", RFC 5764, May 2010.

Appendix A. Possible Solutions

   This section documents the solutions explored when selecting a SHIM
   based one and discusses their feasibility.

A.1. Header Extension

   One proposal is to define an RTP header extension [RFC5285] that
   explicitly enumerates the session identifier in each packet. This
   proposal has some merits regarding RTP, since it uses an existing
   extension mechanism; it explicitly enumerates the session allowing
   for third parties to associate the packet to a given RTP session; and
   it works with SRTP as currently defined since a header extension is
   by default not encrypted, and is thus readable by the receiving stack
   without needing to guess which session it belongs to and attempt to
   decrypt it. This approach does, however, conflict with the
   requirement from [RFC5285] that "header extensions using this
   specification MUST only be used for data that can be safely ignored
   by the recipient", since correct processing of the received packet
   depends on using the header extension to demultiplex it to the
   correct RTP session.

   Using a header extension also result in the session ID is in the
   integrity protected part of the packet. Thus a translator between
   multiplexed and non-multiplexed has the options:

   1. to be part of the security context to verify the field

   2. to be part of the security context to verify the field and remove
       it before forwarding the packet

   3. to be outside of the security context and leave the header
       extension in the packet. However, that requires successful

negotiation of the header extension, but not of the
functionality, with the receiving end-points.

The biggest existing hurdle for this solution is that there exist no
header extension field in the RTCP packets. This requires defining a
solution for RTCP that allows carrying the explicit indicator,
preferably in a position that isn't encrypted by SRTCP. However, the
current SRTCP definition does not offer such a position in the
packet.

Modifying the RR or SR packets is possible using profile specific
extensions. However, that has issues when it comes to deployment and
in addition any information placed there would end up in the
encrypted part.

Another alternative could be to define another RTCP packet type that
only contains the common header, using the 5 bits in the first byte
of the common header to carry a session id. That would allow SRTCP
to work correctly as long it accepts this new packet type being the
first in the packet. Allowing a non-SR/RR packet as the first packet
in a compound RTCP packet is also needed if an implementation is to
support Reduced Size RTCP packets [RFC5506]. The remaining downside
with this is that all stack implementations supporting multiplexing
would need to modify its RTCP compound packet rules to include this
packet type first. Thus a translator box between supporting nodes
and non-supporting nodes needs to be in the crypto context.

This solution's per packet overhead is expected to be 64-bits for
RTCP. For RTP it is 64-bits if no header extension was otherwise
used, and an additional 16 bits (short header), or 24 bits plus (if
needed) padding to next 32-bits boundary if other header extensions
are used.

A.2. Multiplexing Shim

This proposal is to prefix or postfix all RTP and RTCP packets with a
session ID field. This field would be outside of the normal RTP and
RTCP packets, thus having no impact on the RTP and RTCP packets and
their processing. An additional step of demultiplexing processing
would be added prior to RTP stack processing to determine in which
RTP session context the packet is to be included. This has also no
impact on SRTP/SRTCP as the shim layer would be outside of its
protection context. The shim layer's session ID is however
implicitly integrity protected as any error in the field will result
in the packet being placed in the wrong or non-existing context, thus
resulting in a integrity failure if processed by SRTP/SRTCP.

This proposal is quite simple to implement in any gateway or translating device that goes from a multiplexed to a non-multiplexed domain or vice versa, as only an additional field needs to be added to or removed from the packet.

The main downside of this proposal is that it is very likely to trigger a firewall response from any deep packet inspection device. If the field is prefixed, the RTP fields are not matching the heuristics field (unless the shim is designed to look like an RTP header, in which case the payload length is unlikely to match the expected value) and thus are likely preventing classification of the packet as an RTP packet. If it is postfixed, it is likely classified as an RTP packet but might not correctly validate if the content validation is such that the payload length is expected to match certain values. It is expected that a postfixed shim will be less problematic than a prefixed shim in this regard, but we are lacking hard data on this.

This solution's per packet overhead is 1 byte.

A.3. Single Session

Given the difficulty of multiplexing several RTP sessions onto a single lower-layer transport, it's tempting to send multiple media streams in a single RTP session. Doing this avoids the need to de-multiplex several sessions on a single transport, but at the cost of losing the RTP session as a separator for different type of streams. Lacking different RTP sessions to demultiplex incoming packets, a receiver will have to dig deeper into the packet before determining what to do with it. Care has to be taken in that inspection. For example, it is important to be careful to ensure that each real media source uses its own SSRC in the session and that this SSRC doesn't change media type.

The loss of the RTP session as a separator for different usages or purpose would be an minor issue if the only difference between the RTP sessions is the media type. In this case, the application could use the Payload Type field to identify the media type. The loss of the RTP Session functionality is however severe, if the application uses the RTP Session for separating different treatments, contexts etc. Then you would need additional signalling to bind the different sources to groups which can help make the necessary distinctions.

However, the loss of the RTP session as separator is not the only issue with this approach. The RTP Multiplexing Architecture [I-D.ietf-avtcore-multiplex-guidelines] discusses a number of issues in Section 6.7. These include RTCP bandwidth differences, limitations in the number of payload types, media aware RTP mixers and interactions with Legacy end-points.

Additional attention needs to be placed on this important aspect. In multi-party situations using central nodes there exist some difficulties in having a legacy implementation using multiple RTP sessions interworking with an end-point having only a single RTP session across the central node. The main reason is the fact that the one using single session with multiple media types has only one SSRC space, while the other end-points have multiple spaces. Thus translation might have to occur because there is several RTP sessions using the same SSRC value. This has both limitations, processing overhead and the possibility of becoming an deployment obstacle for new RTP/RTCP extensions.

This approach has been proposed in the RTCWeb context in [I-D.lennox-rtcweb-rtp-media-type-mux] and [I-D.ietf-mmusic-sdp-bundle-negotiation]. These drafts describe how to signal multiple media streams multiplexed into a single RTP session, and address some of the issues raised here and in Section 6.7 of the RTP Multiplexing Architecture [I-D.ietf-avtcore-multiplex-guidelines] draft.

This method has several limitations that limits its usage as solution in providing multiple RTP sessions on the same lower layer transport. However, we acknowledge that there are some uses for which this method can be sufficient and which can accept the methods limitations and downsides. The RTCWEB WG has a working assumption to support this method. For more details of this method, see the relevant drafts under development. We do include this method in the comparison to provide a more complete picture of the pro and cons of this method.

This solution has no per packet overhead. The signalling overhead will be a different question.

A.4. Use the SRTP MKI field

This proposal is to overload the MKI SRTP/SRTCP identifier to not only identify a particular crypto context, but also identify the actual RTP Session. This clearly is a miss use of the MKI field, however it appears to be with little negative implications. SRTP already supports handling of multiple crypto contexts.

The two major downsides with this proposal is first the fact that it
requires using SRTP/SRTCP to multiplex multiple sessions on a single
lower layer transport. The second issue is that the session ID
parameter needs to be put into the various key-management schemes and
to make them understand that the reason to establish multiple crypto
contexts is because they are connected to various RTP Sessions.
Considering that SRTP have at least 3 used keying mechanisms, DTLS-
SRTP [RFC5764], Security Descriptions [RFC4568], and MIKEY [RFC3830],
this is not an insignificant amount of work.

This solution has 32-bit per packet overhead, but only if the MKI was
not already used.

A.5. Use an Octet in the Padding

The basics of this proposal is to have the RTP packet and the last
(mandated by RFC3550) RTCP packet in a compound to include padding,
at least 2 bytes. One byte for the padding count (last byte) and one
byte just before the padding count containing the session ID.

This proposal uses bytes to carry the session ID that have no defined
value and is intended to be ignored by the receiver. From that
perspective it only causes packet expansion that is supported and
handled by all existing equipment. If an implementation fails to
understand that it is needs to interpret this padding byte to learn
the session ID, it will see a mostly coherent RTP session except
where SSRCs overlap or where the payload types overlap. However,
reporting on the individual sources or forwarding the RTCP RR are not
completely without merit.

There is one downside of this proposal and that has to do with SRTP.
To be able to determine the crypto context, it is necessary to access
to the encrypted payload of the packet. Thus, the only mechanism
available for a receiver to solve this issue is to try the existing
crypto contexts for any session on the same lower layer transport and
then use the one where the packet decrypts and verifies correctly.
Thus for transport flows with many crypto contexts, an attacker could
simply generate packets that don't validate to force the receiver to
try all crypto contexts they have rather than immediately discard it
as not matching a context. A receiver can mitigate this somewhat by
using heuristics based on the RTP header fields to determine which
context applies for a received packet, but this is not a complete
solution.

This solution has a 16-bit per packet overhead.

A.6. Redefine the SSRC field

The Rosenberg et. al. Internet draft "Multiplexing of Real-Time
Transport Protocol (RTP) Traffic for Browser based Real-Time
Communications (RTC)" [I-D.rosenberg-rtcweb-rtpmux] proposed to
redefine the SSRC field. This has the advantage of no packet
expansion. It also looks like regular RTP. However, it has a number
of implications. First of all it prevents any RTP functionality that
require the same SSRC in multiple RTP sessions.

Secondly its interoperability with end-point using multiple RTP
sessions are problematic. Such interoperability will requires an
SSRC translator function in the gateway node to ensure that the SSRCs
fulfil the semantic rules of the different domains. That translator
is actually far from easy as it needs to understand the semantics of
all RTP and RTCP extensions that include SSRC/CSRC. This as it is
necessary to know when a particular matching 32-bit pattern is an
SSRC field and when the field is just a combination of other fields
that create the same matching 32-bit pattern. Thus there is a
possibility that such a translator becomes a obstacle in deploying
future RTP/RTCP extensions. In addition the translator actually have
significant overhead when SRTP are in use. This as a verification
that the packet is authentic, decryption, SSRC translation,
encryption and finally generation of authentication tags are needed.
In addition the translator has to be part of the security context.

This solution has no per packet overhead.

Appendix B. Comparison

This section compares the above potential solutions with the
requirements. Motivations are provided in addition to a high level
metric of successfully, partially and failing to meet requirement.
In the end a summary table (Figure 6) of the high level value are
provided.

B.1. Support of Multiple RTP Sessions Over Single Transport

This one is easy to determine. Only the single session proposal
fails this requirement as it is not at all designed to meet it. The
rest fully support this requirement.

B.2. Enable Same SSRC Value in Multiple RTP Sessions

Based on the discussion in Section 4 two sub-requirements have been
derived.

B.2.1. Avoid SSRC Translation in Gateways/Translation

This sub-requirement is derived based on the desire to avoid having
gateways or translators perform full SSRC translation to minimize
complexity, avoid the requirement to have gateways in security
context, and as a hinder to long-term evolution. Two of the
proposals have issues with this, due to their lack of support for
multiple 32-bit SSRC spaces and lacking possibility to have the same
SSRC value in multiple RTP sessions. The proposals that have these
properties and thus are marked as failing are the Single Session and
Redefine the SSRC field. The other proposals are all successful in
meeting this requirement.

B.2.2. Support Existing Extensions

The second sub-requirement is how well the proposals support using
the existing RTP mechanisms. Here both Single Session and Redefine
the SSRC field will have clear issues as they cannot support the same
full 32-bit SSRC value in two different RTP sessions. This is
clearly an issue for the XOR based FEC. RTP retransmission and
scalable encoding are minor issues as there exist alternatives to
those mechanisms that works with the structure of these two
proposals. Thus we give them a fail. The Header Extension gets a
partial due to unclear interaction between putting in an header
extension and these mechanisms.

B.3. Ensure SRTP Functions

This requirement is about ensuring both secure and efficient usage of
SRTP. The Octet in Padding field proposal gets a fail as the
receiving end-point cannot determine the intended RTP session prior
to de-encryption of the padding field. Thus a catch-22 arises which
can only be resolved by trying all session contexts and see what
decrypts. This causes a security vulnerability as an attacker can
inject a packet which does not meet any of the session contexts. The
receiver will then attempt decryption and authentication of it using
all its session contexts, increasing the amount of wasted resources
by a factor equal to the number of multiplexed sessions. Thus this
proposal gets a fail.

The proposal of Overloading the SRTP MKI field as session identifier
gets a partial due to the fact that it cannot use SRTP's key-
management mechanism out of the box. It forces the key-management
mechanism and the SRTP implementations to maintain the MKI-to-RTP
session bindings to maintain secure and correct function.

The Redefine the SSRC field gets a partial due to its need to modify the key-management mechanisms to correctly identify the partial SSRC space the parameters applies to. Similarly, the SRTP implementation also needs to be updated to correctly support this security context differentiation.

The header extension based solution gets a less severe partial than Redefine the SSRC and the MKI. It will however have an issue when using a gateway to a domain that does not multiplex multiple RTP sessions over the same transport. Then the gateway will require to be in the security context to be able to add or remove the header extension as it is in the part of the packet that is integrity protected by SRTP.

The remaining two proposals do not affect SRTP mechanisms and thus successfully meet this requirement.

B.4. Don't Redefine Used Bits

This requirement is all about RTP and RTCP header fields having a given definition ought not be changed as it can cause interoperability problems between modified and non-modified implementations. This becomes especially problematic in RTP sessions used for multi-party sessions.

Redefine the SSRC field gets a big fail on this as it redefines the SSRC field, a core field in RTP. It has been identified that such a change will have issues since if it gets connected to a non-modified end-point that randomly assigns the SSRC, as supposed by RFC 3550, those SSRCs will be distributed over different RTP sessions at the modified end-point. Also other functions using the SSRC field, not understanding the additional semantics of the SSRC field, is likely to have issues.

Using the SRTP MKI field to identify a session is overloading that field with double semantics. This likely has minimal negative impact in RTP since it ought to be possible to have the SRTP stack use the MKI field to both look up the security context and which output RTP session the processed packet belongs to. However, this redefinition clearly creates issues with the key-management scheme. That will have to be modified to handle both this change and deal with the interoperability issues when negotiating its usage. This gets a full fail due to that it makes the problem someone else's, namely the RTP implementers.

Defining an Octet in the Padding field redefines a field, whose definition is to have zero value and is expected to be ignored by the receiver according to the original semantics. Thus this is one of

the more benign modifications one can do, however this can still
cause issues in implementations that unnecessarily check the field
values, or in Firewalls. This is judged to be partially meeting the
requirement.

The Header Extension proposal does in fact not redefine any currently
used bits in RTP. The header extension would be a correctly
identified extension with its own definition. However, it does
redefine a rule on what header extensions are for. The RTCP solution
however would have more severe impact as it would need to redefine
the standard meaning of an RTCP packet header in addition to the
default compound packet rules. Due to these issues the proposal
fails to meet this requirement.

The multiplexing shim and the single session both successfully meet
this requirement.

B.5. Firewall Friendly

This requirement is clearly difficult to judge as firewall
implementations are highly different in both implementation, scope of
what it investigates in packets, and set policies. A reasonable goal
is to minimize the likeliness that rules and policies intended to let
RTP media streams pass, will also let these streams through when
multiplexing RTP sessions over a single transport. The below
analysis shows that no solution is truly firewall friendly and all
are judged as being partially meeting this goal. However, the reason
why it is believed that a firewall might react to the streams are
quite different.

The Single Session and Redefine the SSRC field are likely the least
suspect solutions from a firewall perspective. However, as their
transport flows contain multiple SSRCs with payloads that indicate
likely multiple different media types they are still likely to make a
picky firewall block the transport. This is especially true for
Firewalls that take signalling messages into account where it will
expect a particular media type in a given context. A non upgraded
firewall might in fact produce two different contexts with
overlapping transport parameters where both rules will receive media
streams of the other media type that are outside of the allowed rule.
However, to be clear if these proposals doesn't get through, none of
the other will either as they all will have this behaviour.

The header extension proposal is potentially problematic for two
reasons. The first reason, which also other proposals has, is
related to that the same SSRC value can exist in two RTP sessions
over the same underlying flow. Anyone tracking the sequence number
and timestamp will react badly as the second media stream with the

same SSRC causes constant jumps back and forth in these fields
compared to the first stream, if packets are transmitted
simultaneously for both SSRCs. This issue can likely only be solved
by having the Firewalls that like to track flows to also use the
session identifier to create context. This is possible as the header
extension will be in the clear and in the front. The second issue is
that the header extension itself can get the firewall to react.
Especially very picky ones that expect packets with certain media
types to have certain packet lengths. They are not compatible with a
header extension.

The Multiplexing Shim shares the issue with multiple flows for the
same SSRC. Firewalls and deep packet inspection cause the shim
placement to be in question. If it is a pre-fixed shim, it prevents
the packet from looking like regular IP/UDP/RTP packets and be
correctly classified in Firewalls and DPI engines. However, if one
puts it last, it is unlikely that any firewall or DPI ever will be
able to take the session context into account as it is at the end of
the packet. This as many line rate processing devices only take a
certain amount of the headers into account.

The SRTP MKI field is likely the solution that has least firewall and
DPI issues, after the single RTP session. There is no additional
suspect field. The only difference from a single RTP session in the
transport flow is the fact that multiple MKI are guaranteed to be
used. However, that can occur also in a single RTP session usage.
Thus the only issues are the one shared with single session and the
one that several RTP media streams can use the same SSRC.

The octet in the padding field has, in addition to the issues the
SRTP MKI field has, the single issue that it redefines something that
is supposed to be zero into a value. Thus potentially causing a
deeply inspecting firewall to clamp the flow in fear of covert
channel or non-compliance.

B.6. Monitoring and Reporting

The monitoring and reporting requirement considers several aspects.
How useful monitoring can one get from an existing legacy monitor,
and secondary any issues in upgrading them to handle the selected
solution. Thirdly, packet selector filters and packet sniffers
concerns are considered.

In general one can expect the proposals that have only a single SSRC
space to work better with legacy. Thus both Single Session and
Redefine SSRC space can gather and report data on media flows most
likely. The only potential issue is that due to the different media
types and clock rates, some failure can occur. In particular a third

party monitor can be targeted to a specific media type, like
monitoring VoIP. That monitor will have problems processing any
video packets correctly and generate the VoIP specific metrics for
any video sending SSRC. In general, no legacy solution for
monitoring will be able to correctly create the sub-contexts that
each RTP session has in the solutions, without update to handle the
new semantics. Also when it comes to the packet filtering and
selector filters, fine grained control can only be accomplished
implementing the new semantics. Therefore only the Single Session
meets this requirement fully.

Redefine the SSRC field is close to fully meeting the requirement,
however due to that there exist a session structure that is hidden to
anyone that is not upgraded to understand the semantics, this only
gets a partial.

The other proposals all can have multiple RTP sessions using the same
SSRC. This will create significant issues for any legacy third party
monitor. Only an updated monitor, or for that matter packet
selector, can pick out the individual media streams and their
associated RTCP traffic. Thus all these proposals gets a failure to
meet the requirement.

B.7. Usable over Multicast

As discussed earlier the goal with having the option usable also over
multicast is to remove the need to produce different media streams
for transport over unicast and multicast. All of the proposals
successfully meet the requirement.

B.8. Incremental Deployment

The possibility to deploy the usage of the multiplexing of multiple
RTP sessions over a single transport, especially in the context of
multi-party sessions, is a great benefit for any of the proposals.
Thus not all end-point implementations needs to be upgraded before
one start enabling it in the central node and any signalling.

Considering a centralized multi-party application where some
participants are using multiple transport flows and you want to
enable one particular participant to use the single transport to the
central node, one criteria stands out. The possibility to have one
RTP session per transport in one leg, and in the next multiplex them
together with minimal complexity and packet changes. Here there are
significant differences.

The Multiplexing Shim has the least overhead for this. As the
central node or gateway between deployments only needs to either add

or remove the shim identifier and then forward the packet over the corresponding transport, either a joint one on the single transport side, or over the individual one on the multiple transport side.

The SRTP MKI field proposal is almost as good, as the only main difference is the need to coordinate the used MKIs on the non-multiplexed legs so that there is no overlap between the RTP sessions. And if there is, the MKI can be translated in gateway as SRTP has no integrity protection over the MKI. Thus both multiplexing shim and SRTP MKI field does successfully meet this requirement.

The Header Extension supports multiple full 32-bit SSRC spaces and can thus handle all the RTP sessions without need for any SSRC translation, however this proposal does run into the problem that the gateway needs to be in the security context to be able to add or remove the header extension when SRTP is used. In addition to the security implications of that, there is a complexity overhead due to the need to redo the authentication tags on all RTP/RTCP packets. Thus it gets a partial.

The Octet in the Padding field share issues with the header extension but have even higher complexities for this. The reason is that the padding field is also encrypted. Thus to add or remove it (although removing it might be unnecessary) forces the end-point to encrypt at least that byte also, and for ciphers that are not stream-ciphers, the whole packet needs to be re-encrypted. Thus this proposal gets a very weak partially meeting the requirement.

The Single Session and Redefine the SSRC field do not allow several vanilla RTP sessions to be connected to these proposals. The reason is the single 32-bit SSRC space they have. Single Session only has one session and the Redefine the SSRC fields uses some of the bits as session identifier. This forces the gateway to translate the SSRC whenever it does not fulfil the rules or semantics of the multiplexed side. For Redefine SSRC field this becomes almost constant as the session identifier part of the SSRC has to be the same over all SSRCs from the same session. For Single Session it might only be needed when there otherwise would be an SSRC collision between the sessions. This further assumes that the non-multiplexed side would never use any of the RTP mechanisms that require the same SSRC in multiple RTP sessions, as they cannot be gatewayed at all. When translating an SSRC there is first of all an overhead, with SRTP that includes a complete authenticate, decrypt, encrypt and create a new authentication tag cycle. In addition, the SSRC translation could potentially be a deployment obstacle for new RTP/RTCP extensions that has to be understood by the translator to be correctly translated. Therefore these two proposals gets a fail to meet the requirements.

B.9. **Summary and Conclusion**

   This section contains a summary table of the high level outcome
   against the different requirements.

   A table mapping the requirements against the ID numbers used in the
   table is the following:

   1: Support multiple RTP sessions over one transport flow

   2: Enable same SSRC value in multiple RTP sessions

      2.1: Avoid SSRC translation in gateways/translators

      2.2: Support existing extensions

   3: Ensure SRTP functions

   4: Don't Redefine used bits

   5: Firewall Friendly

   6: Monitoring and Reporting still needs to function

   7: Usable over Multicast

   8: Incremental deployment

   OH: Overhead in Bytes. + means variable

```
        --------------+---+---+---+---+---+---+---+---+---+----
        Solution      | 1 |2.1|2.2| 3 | 4 | 5 | 6 | 7 | 8 | OH
        --------------+---+---+---+---+---+---+---+---+---+----
        Header Ext.   | S | S | P | P | F | P | F | S | P | 8+
        Multiplex Shim| S | S | S | S | S | P | F | S | S | 1
        Single Session| F | F | F | S | S | P | S | S | F | 0
        SRTP MKI Field| S | S | S | P | F | P | F | S | S | 4
        Padding Field | S | S | S | F | P | P | F | S | P | 2
        Redefine SSRC | S | F | F | P | F | P | P | S | S | 0
        --------------+---+---+---+---+---+---+---+---+---+----
```

    Figure 6: Summary Table of Evaluation (Successfully (S), Partially
                 (P) or Fails (F) to meet requirement)

   Considering these options, the authors would recommend that AVTCORE
   standardize a solution based on a post or prefixed multiplexing
   field, i.e. a shim approach combined with the appropriate signalling
   as described in Appendix A.2.

Authors' Addresses

    Magnus Westerlund
    Ericsson
    Farogatan 6
    SE-164 80 Kista
    Sweden

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

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

    Email: csp@csperkins.org
    URI: http://csperkins.org/