

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: May 3, 2012

M. Westerlund
Ericsson
C. Perkins
University of Glasgow
October 31, 2011

Multiple RTP Session on a Single Lower-Layer Transport
draft-westerlund-avtcore-transport-multiplexing-01

Abstract

This document specifies how multiple RTP sessions are to be multiplexed on the same lower-layer transport, e.g. a UDP flow. It discusses various requirements that have been raised and their feasibility, which results in a solution with a certain applicability. A solution is recommended and that solution is provided in more detail, including signalling and examples.

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 May 3, 2012.

Copyright Notice

Copyright (c) 2011 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of

the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

- 1. Introduction 3
- 2. Conventions 3
 - 2.1. Terminology 3
 - 2.2. Requirements Language 3
- 3. Requirements 3
 - 3.1. Support Use of Multiple RTP Sessions 4
 - 3.2. Same SSRC Value in Multiple RTP Sessions 4
 - 3.3. SRTP 5
 - 3.4. Don't Redefine Used Bits 6
 - 3.5. Firewall Friendly 6
 - 3.6. Monitoring and Reporting 6
 - 3.7. Usable Also Over Multicast 6
 - 3.8. Incremental Deployment 7
- 4. Possible Solutions 7
 - 4.1. Header Extension 7
 - 4.2. Multiplexing Shim 8
 - 4.3. Single Session 9
 - 4.4. Use the SRTP MKI field 10
 - 4.5. Use an Octet in the Padding 11
 - 4.6. Redefine the SSRC field 11
- 5. Recommendation 12
- 6. Specification 12
 - 6.1. Shim Layer 12
 - 6.2. Signalling 16
 - 6.3. SRTP Key Management 17
 - 6.3.1. Security Description 17
 - 6.3.2. DTLS-SRTP 18
 - 6.3.3. MIKEY 18
 - 6.4. Examples 18
 - 6.4.1. RTP Packet with Transport Header 18
 - 6.4.2. SDP Offer/Answer example 19
- 7. Open Issues 21
- 8. IANA Considerations 22
- 9. Security Considerations 22
- 10. Acknowledgements 22
- 11. References 22
 - 11.1. Normative References 22
 - 11.2. Informational References 23
- Authors' Addresses 24

1. Introduction

There has been renewed interest for having a solution that allows multiple RTP sessions [RFC3550] to use a single lower layer transport, such as a bi-directional UDP flow. The main reason is the cost of doing NAT/FW traversal for each individual flow. ICE and other NAT/FW 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/FW resources. The latter might be an increasing problem in the IPv4 to IPv6 transition period.

This document draws up some requirements for consideration on how to transport multiple RTP sessions over a single lower-layer transport. These requirements will have to be weighted as the combined set of requirements result in that no known solution exist that can fulfill them completely.

A number of possible solutions are then considered and discussed with respect to their properties. Based on that, the authors recommends a shim layer variant as single solution, which is described in more detail including signalling solution and examples.

2. Conventions

2.1. Terminology

Some terminology used in this document.

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

2.2. Requirements Language

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

3. 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 fulfill and which can be compromised on to arrive at a solution.

3.1. Support Use of Multiple RTP Sessions

This may at first glance appear to be an obvious requirement. Although the authors are convinced it is a mandatory requirement for a solution, it warrants some discussion around the implications of not having multiple RTP sessions and instead use a single RTP session.

The main purpose of RTP sessions is to allow separation of streams that have different purposes, for example different media types. A big reason for establishing this is the knowledge that any SSRC within the session is supposed to be processed in a similar way.

For simpler cases, where the streams within each media type need the same processing, it is clearly possible to find other multiplex solutions, for example based on the Payload Type and the differences in encoding that the payload type allows to describe. This may anyhow be insufficient when you get into more advanced usages where you have multiple sources of the same media type, but for different purposes or as alternatives. For example when you have one set of video sources that shows session participants and another set of video sources that shares an application or slides, you likely want to separate those streams for various reasons such as control, prioritization, QoS, methods for robustification, etc. In those cases, using the RTP session for separation of properties is a powerful tool. A tool with properties that need to be preserved when providing a solution for how to use only a single lower-layer transport.

For more discussion of the usage of RTP sessions verses other multiplexing we recommend RTP Multiplexing Architecture [I-D.westerlund-avtcore-multiplex-architecture].

3.2. Same SSRC Value in Multiple RTP Sessions

Two different RTP sessions being multiplexed on the same lower layer transport need to be able to use the same SSRC value. This is a strong requirement, for two reasons:

1. To avoid mandating SSRC assignment rules that are coordinated between the sessions. If the RTP sessions multiplexed together must 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 must also 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.

2. There are some few RTP extensions that currently rely on being able to use the same SSRC in different RTP sessions:

- * XOR FEC (RFC5109)
- * RTP Retransmission in session mode (RFC4588)
- * Certain Layered Coding

3.3. SRTP

SRTP [RFC3711] is one of the most commonly used security solutions for RTP. In addition, it is the only one recommended 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 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

must be considered when defining a mechanism.

3.4. Don't Redefine Used Bits

As the core of RTP is in use in many systems and has a really large deployment story and numerous implementations, changing any of the field definitions is highly problematic. First of all, the implementations need to change to support this new semantics. Secondly, you get a large transition issue when you have some session participants that support the new semantics and some that don't. Combining the two behaviors in the same session can force the deployment of costly and less than perfect translation devices.

3.5. Firewall Friendly

It is desirable that current firewalls will accept the solutions as 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 more difficult deployment story.

3.6. Monitoring and Reporting

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 may 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.

3.7. Usable Also Over Multicast

It is desirable that a solution should be possible to use also when 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.

3.8. 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 should 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 should also be as easy to do the reverse forwarding operation.

4. Possible Solutions

This section looks at a few possible solutions and discusses their feasibility.

4.1. 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 deployability 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.

4.2. 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 shall 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 may 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.

4.3. 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 demultiplex 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 must be taken in that inspection. For example, you must 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 purpose separator is likely not a big issue if the only difference between RTP Sessions is the media type. In this case, you can use the Payload Type field to identify the media type. The loss of the RTP Session functionality is more severe, however, if you actually use 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.

This approach has been proposed in the RTCWeb context in [I-D.lennox-rtcweb-rtp-media-type-mux] and [I-D.holmberg-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 7.2.9 of the RTP Multiplexing Architecture [I-D.westerlund-avtcore-multiplex-architecture] draft. However, they fail to discuss maybe the largest issue with this solution: how to do incremental deployment and transition.

Many transition scenarios use an RTP translator as a gateway between a single RTP session containing multiple media types multiplexed together, and several separate RTP sessions each using a single media type. In this scenario, it is possible that a legacy device that uses one RTP session for each media type will use the same SSRC in each session. When translating these into a single RTP session, it will be necessary to rewrite one of the SSRCs, so that each stream has a unique SSRC. This SSRC translation process is straight-forward for RTP packets, but is very complex for RTCP packets. It also hinders innovation, since such a gateway will not be able to translate new RTCP extensions that it is unaware of, even if they are supported by devices on both sides of the gateway.

This method has several limitations that makes it unsuitable as general mechanism to provide multiple RTP sessions on the same lower layer transport. However, we acknowledge that there are some uses for which this method may be sufficient and which can accept the method limitations and other downsides. The RTCWEB WG has a working assumption to support this method. For more details of this method, see the relevant drafts under development.

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

4.4. 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.

4.5. Use an Octet in the Padding

The basics of this proposal is to have the RTP packet and the last (required 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 required 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 hueristics 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.

4.6. 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 normal RTP is problematic. Such interoperability requires an SSRC translator function in the gateway to ensure that the SSRCs fulfill the requirements 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 any future RTCP extension might not work through the translator, causing a barrier for deployment of future extensions.

This solution has no per packet overhead.

5. Recommendation

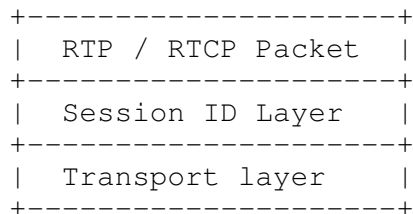
Considering these options, the authors would recommend that AVTCORE standardize a solution based on a postfixed multiplexing field, i.e. a shim approach combined with the appropriate signalling as described in Section 4.2.

6. Specification

This section contains the specification of the solution based on a SHIM, with the explicit session identifier at the end of the encapsulated payload.

6.1. Shim Layer

This solution is based on a shim layer that is inserted in the stack between the regular RTP and RTCP packets and the transport layer being used by the RTP sessions. Thus the layering looks like the following:



Stack View with Session ID 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 1 where four sessions, S1-S4 is 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.

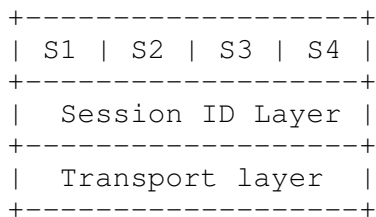
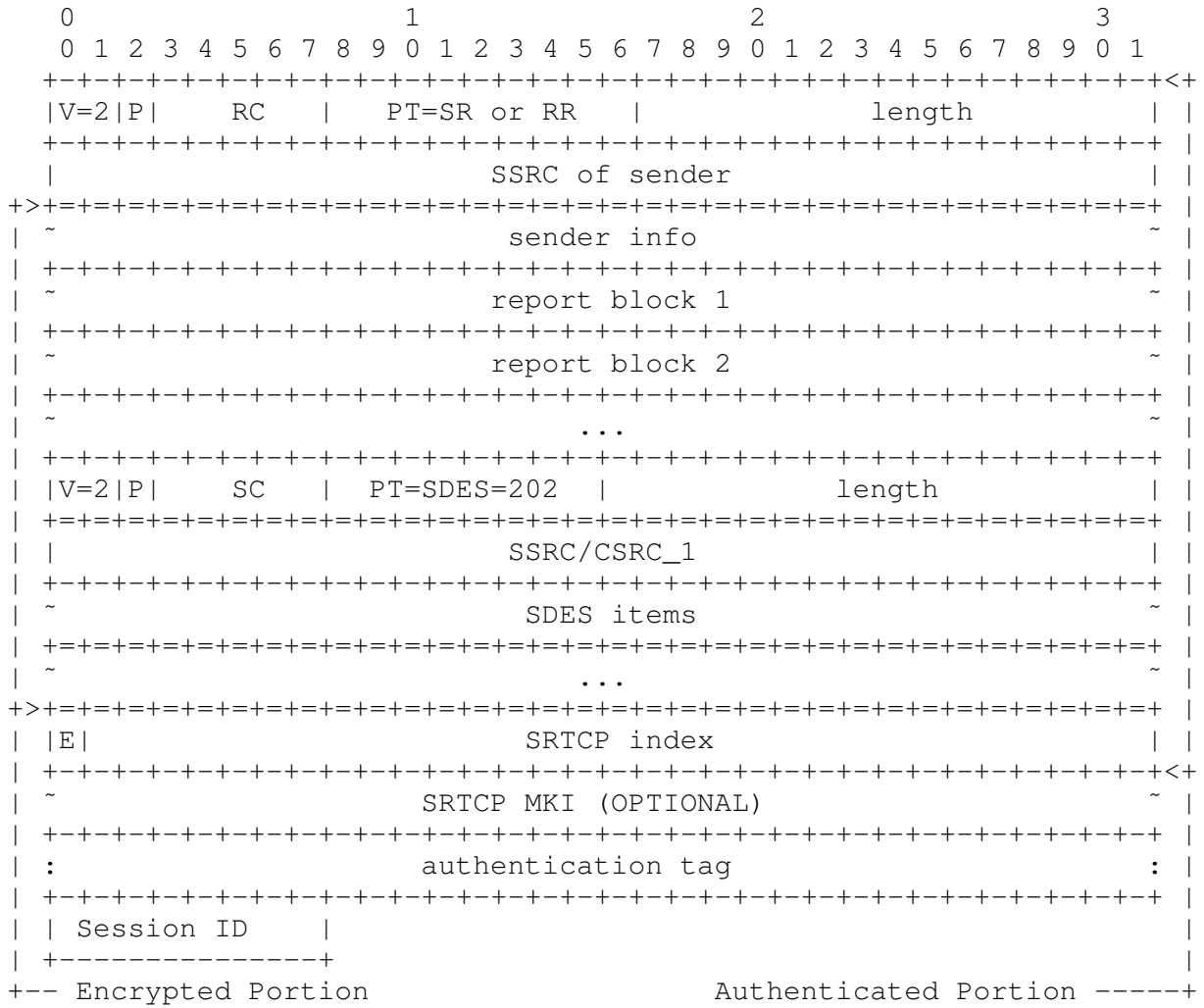


Figure 1: Multiple RTP Session On Top of Session ID Layer

The Session ID layer encapsulates one RTP or RTCP packet from a given RTP session and postfixes a one byte Session ID (SID) field to the packet. 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 0-255. 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 may 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, DCCP, TCP, and SCTP 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.



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

The use of the Session ID layer needs to be explicitly agreed on between the communicating parties. Each RTP Session the application uses must in addition to the regular configuration such as payload types, RTCP extension etc, have both the underlying 5-tuple (source address and port, destination address and port, and transport protocol) and the Session ID used for the particular RTP session. The signalling requirement is to assign unique Session ID values to all RTP Sessions being sent over the same 5-tuple. The same Session ID shall be used for an RTP session independently of the traffic direction. Note that nothing prevents a multi-media application from using multiple 5-tuples if desired for some reason, in which case each 5-tuple has its own session ID value space.

This section defines how to negotiate the use of the Session ID layer, using the Session Description Protocol (SDP) Offer/Answer mechanism [RFC3264]. A new media-level SDP attribute, 'session-mux-id', is defined, in order to be used with the media BUNDLE mechanism defined in [I-D.holmberg-mmusic-sdp-bundle-negotiation]. The attribute allows each media description ("m=" line) associated with a 'BUNDLE' group to form a separate RTP session.

The 'session-mux-id' attribute is included for a media description, in order to indicate the Session ID for that particular media description. Every media description that shares a common attribute value is assumed to be part of a single RTP session. An SDP Offerer MUST include the 'session-mux-id' attribute for every media description associated with a 'BUNDLE' group. If the SDP Answer does not contain 'session-mux-id' attributes, the SDP Offerer MUST NOT assume that separate RTP sessions will be used. If the SDP Answer still describes a 'BUNDLE' group, the procedures in [I-D.holmberg-mmusic-sdp-bundle-negotiation] apply.

An SDP Answerer MUST NOT include the 'session-mux-id' attribute in an SDP Answer, unless included in the SDP Offer.

The attribute has the following ABNF [RFC5234] definition.

```

Session-mux-id-attr = "a=session-mux-id:" SID *SID-prop
SID                  = SID-value / SID-pairs
SID-value           = 1*3DIGIT / "NoN"
SID-pairs           = SID-value "/" SID-value ; RTP/RTCP SIDs
SID-prop            = SP assignment-policy / prop-ext
prop-ext            = token "=" value
assignment-policy   = "policy=" ("tentative" / "fixed")

```


The following parameters MUST be configured as specified:

- o RTP Profile SHOULD be the same, but MUST be compatible, like AVP and AVPF.
- o RTCP bandwidth parameters are the same
- o RTP Payload type values are not overlapping

In declarative SDP usage, there is clearly no method for fallback unless some other negotiation protocol is used.

The SID property "policy" is used in negotiation by an end-point to indicate if the session ID values are merely a tentative suggestion or if they must have these values. This is used when negotiating SID for multi-party RTP sessions to support shared transports such as multicast or RTP translators that are unable to produce renumbered SIDs on a per end-point basis. The normal behavior is that the offer suggest a tentative set of values, indicated by "policy=tentative". These SHOULD be accepted by the peer unless that peer negotiate session IDs on behalf of a centralized policy, in which case it MAY change the value(s) in the answer. If the offer represents a policy that does not allow changing the session ID values, it can indicate that to the answerer by setting the policy to "fixed". This enables the answering peer to either accept the value or indicate that there is a conflict in who is performing the assignment by setting the SID value to NoN (Not a Number). Offerer and answerer SHOULD always include the policy they are operating under. Thus, in case of no centralized behaviors, both offerer and answerer will indicate the tentative policy.

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. Thus each DTLS message must be associated with the SRTP and/or SRTCP flow it is keying.

The most direct solution is 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).

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].

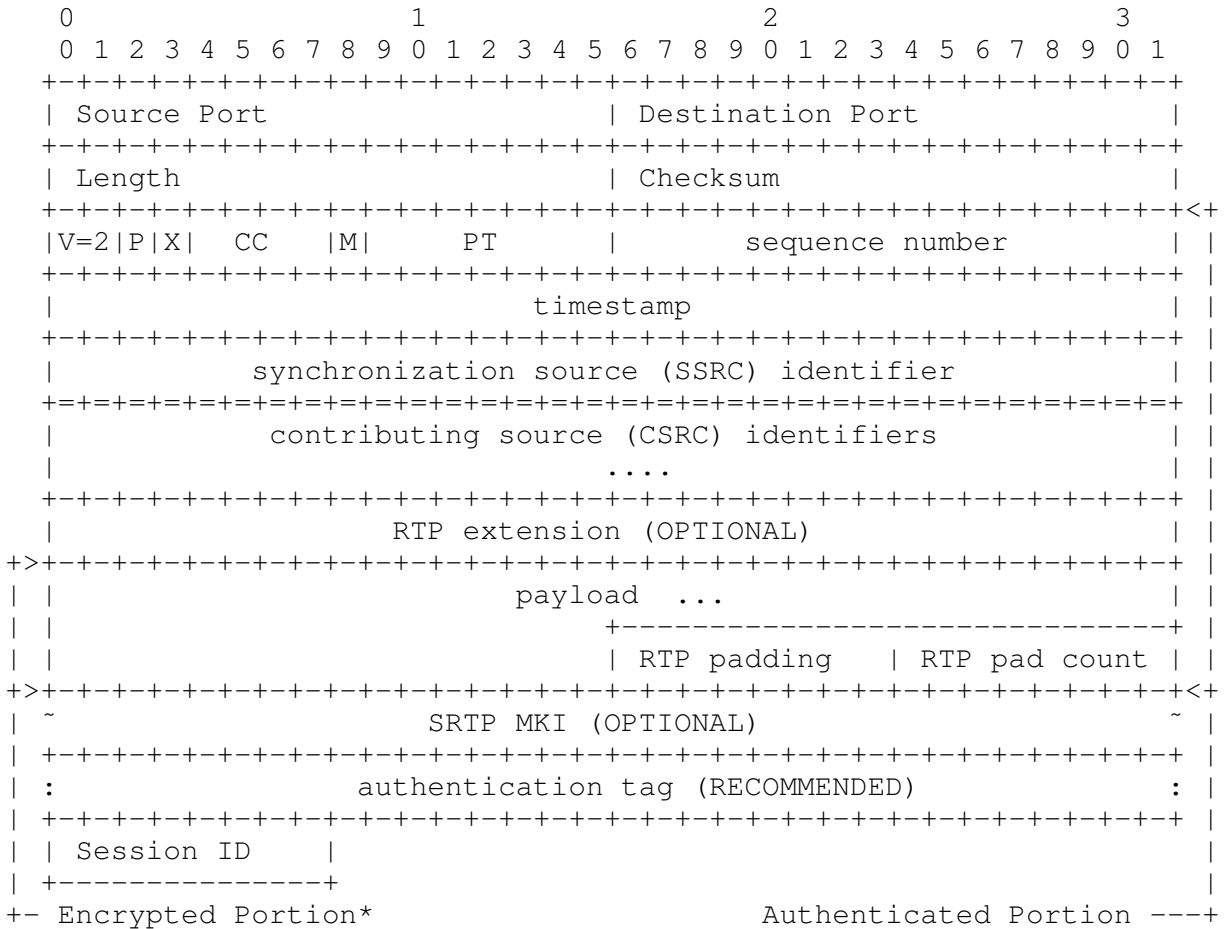
Lets start with the later, i.e. the SDP transport, which shares the properties with Security Description in that is can be associated with a particular media description in a SDP. As long as one avoids using the session level attribute one can be certain to correctly associate the key exchange with a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

The below figure contains an RTP packet with SID field encapsulated by a UDP packet (added UDP header).



SRTP Packet Encapsulated by Session ID Layer

6.4.2. SDP Offer/Answer example

This section contains SDP offer/answer examples. First one example of successful BUNDLEing, and then two where fallback occurs.

In the below SDP offer, one audio and one video is being offered. The audio is using SID 0, and the video is using SID 1 to indicate that they are different RTP sessions despite being offered over the same 5-tuple.

```
v=0
o=alice 2890844526 2890844526 IN IP4 atlanta.example.com
s=
c=IN IP4 atlanta.example.com
t=0 0
a=group:BUNDLE foo bar
m=audio 10000 RTP/AVP 0 8 97
b=AS:200
a=mid:foo
a=session-mxu-id:0 policy=suggest
a=rtpmap:0 PCMU/8000
a=rtpmap:8 PCMA/8000
a=rtpmap:97 iLBC/8000
m=video 10000 RTP/AVP 31 32
b=AS:1000
a=mid:bar
a=session-mxu-id:1 policy=suggest
a=rtpmap:31 H261/90000
a=rtpmap:32 MPV/90000
```

The SDP answer from an end-point that supports this BUNDLEing:

```
v=0
o=bob 2808844564 2808844564 IN IP4 biloxi.example.com
s=
c=IN IP4 biloxi.example.com
t=0 0
a=group:BUNDLE foo bar
m=audio 20000 RTP/AVP 0
b=AS:200
a=mid:foo
a=session-mux-id:0 policy=suggest
a=rtpmap:0 PCMU/8000
m=video 20000 RTP/AVP 32
b=AS:1000
a=mid:bar
a=session-mux-id:1 policy=suggest
a=rtpmap:32 MPV/90000
```

The SDP answer from an end-point that does not support this BUNDLEing or the general signalling of [I-D.holmberg-mmusic-sdp-bundle-negotiation].

```
v=0
o=bob 2808844564 2808844564 IN IP4 biloxi.example.com
s=
c=IN IP4 biloxi.example.com
t=0 0
m=audio 20000 RTP/AVP 0
b=AS:200
a=rtpmap:0 PCMU/8000
m=video 30000 RTP/AVP 32
b=AS:1000
a=rtpmap:32 MPV/90000
```

The SDP answer of a client supporting [I-D.holmberg-mmusic-sdp-bundle-negotiation] but not this BUNDLEing would look like this:

```
v=0
o=bob 2808844564 2808844564 IN IP4 biloxi.example.com
s=
c=IN IP4 biloxi.example.com
t=0 0
a=group:BUNDLE foo bar
m=audio 20000 RTP/AVP 0
a=mid:foo
b=AS:200
a=rtpmap:0 PCMU/8000
m=video 20000 RTP/AVP 32
a=mid:bar
b=AS:1000
a=rtpmap:32 MPV/90000
```

In this last case, the result is a sing RTP session with both media types being established. If that isn't supported or desired, the offerer will have to either re-invite without the BUNDLE grouping to force different 5-tuples, or simply terminate the session.

7. Open Issues

This is the first version of this draft. It will obviously have a number of open issues. This section contains a list of open issues where the author desires some input.

1. Should RTP and RTCP multiplexing without RFC 5761 support be included?

8. IANA Considerations

This document request the registration of one SDP attribute. Details of the registration to be filled in.

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 document 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.holmberg-mmusic-sdp-bundle-negotiation]. The authors would like to thank Christer Holmberg for assistance in utilizing the BUNDLE grouping mechanism.

The proposal in Section 4.5 is original suggested by Colin Perkins. The idea in Section 4.6 is from an Internet Draft [I-D.rosenberg-rtcweb-rtpmux] written by Jonathan Rosenberg et. al. The proposal in Section 4.3 is a result of discussion by a group of people at IETF meeting #81 in Quebec.

11. References

11.1. Normative References

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

draft-holmberg-mmusic-sdp-bundle-negotiation-00 (work in progress), October 2011.

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC3550] Schulzrinne, H., Casner, S., Frederick, R., and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications", STD 64, RFC 3550, July 2003.
- [RFC3711] Baugher, M., McGrew, D., Naslund, M., Carrara, E., and K. Norrman, "The Secure Real-time Transport Protocol (SRTP)", RFC 3711, March 2004.
- [RFC5234] Crocker, D. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", STD 68, RFC 5234, January 2008.

11.2. Informational References

- [I-D.lennox-rtcweb-rtp-media-type-mux]
Lennox, J. and J. Rosenberg, "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.
- [I-D.westerlund-avtcore-multiplex-architecture]
Westerlund, M., Burman, B., and C. Perkins, "RTP Multiplexing Architecture", draft-westerlund-avtcore-multiplex-architecture-00 (work in progress), October 2011.
- [RFC3264] Rosenberg, J. and H. Schulzrinne, "An Offer/Answer Model with Session Description Protocol (SDP)", RFC 3264, June 2002.
- [RFC3830] Arkko, J., Carrara, E., Lindholm, F., Naslund, M., and K. Norrman, "MIKEY: Multimedia Internet KEYing", RFC 3830, August 2004.
- [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] Andreassen, F., Baugher, M., and D. Wing, "Session Description Protocol (SDP) Security Descriptions for Media Streams", RFC 4568, July 2006.
- [RFC5285] Singer, D. and H. Desineni, "A General Mechanism for RTP Header Extensions", RFC 5285, July 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.

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

