

AVTCORE  
Internet-Draft  
Intended status: Informational  
Expires: January 11, 2013

R. Ejzak  
Alcatel-Lucent  
July 10, 2012

Media multiplexing with Real-time Transport Protocol (RTP) subsessions  
draft-ejzak-avtcore-rtp-subsessions-01

Abstract

This document describes a means of multiplexing RTP streams having different media types within a single transport connection and a means of representing this multiplexing option in SDP so that network nodes can easily identify groups of RTP streams and their associated RTCP packets for differential treatment as necessary. This mechanism can be used to complement the BUNDLE multiplexing scheme, which uses the grouping framework to identify all media lines associated with a single transport connection, but provides no means for network nodes to group RTP streams and their RTCP packets for differential treatment. RTP subsessions is an alternative to the SHIM multiplexing proposal; it clearly partitions packets associated with different media lines without changing the format of individual RTP and RTCP packets, but it does not maintain a completely independent SSRC space for each media line, as does SHIM. RTP subsessions avoid SSRC conflicts by construction and can be used in all RTP topologies in which all systems implement RTP subsessions, although SSRC mapping might be needed when forwarding RTP streams from an unpartitioned RTP session into an RTP subsession.

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 January 11, 2013.

Copyright Notice

Copyright (c) 2012 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. Overview of RTP subsessions . . . . .	3
2.1. RTP procedures . . . . .	3
2.2. SDP procedures . . . . .	5
2.3. Additional procedures for relays . . . . .	6
2.4. When a system doesn't support RTP subsessions . . . . .	7
3. Applicability of RTP subsessions . . . . .	7
4. Examples . . . . .	9
5. Evaluation . . . . .	11
5.1. Comparison to BUNDLE . . . . .	11
5.2. Comparison to SHIM . . . . .	11
5.3. Comparison to other SSRC proposals . . . . .	12
6. IANA Considerations . . . . .	12
7. Security Considerations . . . . .	12
8. Informative References . . . . .	12
Author's Address . . . . .	14

## 1. Introduction

The subject of RTP multiplexing has received significant attention within RTCWEB and AVTCORE in the last year. It would be highly desirable to multiplex RTP streams associated with a communications session onto as few transport connections as possible to reduce the messaging required to complete ICE connectivity checks [RFC5245] and DTLS key exchange [RFC6347] prior to being able to send media. RTP is specified in [RFC3550]. This document focuses specifically on how to enable network nodes to provide differential treatment to multiplexed media types within the same transport connection, since means already exist to apply differential treatment to an entire transport connection and RTP is capable of multiplexing RTP streams of the same media type onto a single transport connection using the SSRC field. While an application may be able to request different DSCP markings for these flows, the treatment associated with a particular marking is network-specific and many networks routinely remap packet markings according to local policy unless they can independently verify the requested treatment.

The following drafts provide key background and context, which will not be duplicated here: [I-D.ietf-rtcweb-rtp-usage], [I-D.westerlund-avtcore-multiplex-architecture] and [I-D.westerlund-avtcore-max-ssrc]. The BUNDLE solution [I-D.ietf-mmusic-sdp-bundle-negotiation] uses SDP [RFC4566] to assign different sets of payload type values for each media line sharing an RTP session. The SHIM solution [I-D.westerlund-avtcore-transport-multiplexing] extends the RTP frame with an RTP session identifier to allow multiple RTP sessions to share a single transport connection. Schemes no longer under consideration, based on SSRC, are described in [I-D.rosenberg-rtcweb-rtpmux] and [I-D.peterson-rosenberg-avt-rtp-ssrc-demux].

This document describes an optional enhancement to BUNDLE that allows network nodes to identify RTP streams associated with an SDP media line and their associated RTCP packets for differential treatment in the network. This document assumes the use of unicast RTP sessions only and there is no discussion of multicast sessions.

## 2. Overview of RTP subsessions

### 2.1. RTP procedures

An RTP subsession is a set of RTP streams and corresponding RTCP packets that use a subset of the entire range of SSRC values. Each RTP subsession has most of the characteristics of a complete RTP

session with some exceptions described below. RTP subsessions that are assigned non-overlapping SSRC value ranges can share a single transport connection.

As described in [RFC3550], RTP topologies can include end systems, translators(relays) and mixers. RTP subsessions are particularly suited to supporting end systems and mixers, since these systems typically remap SSRC values when forwarding RTP streams and so can independently assign SSRC values on each transport connection. The remainder of this section describes use of RTP subsessions by end systems and mixers; a later section describes use of RTP subsessions by relays.

Each RTP subsession sharing a single transport connection is assigned a unique 24-bit SSRC prefix for each direction of transmission, i.e., the first (highest order) bit is reserved to indicate each direction of transmission and the next 23 bits of the 32-bit SSRC field have a fixed value for RTP streams associated with the RTP subsession. Each assigned SSRC prefix is also unique in the first 8 bits to allow for potential connection to relays that forward RTP streams from other sources. The network can use the first 8 bits of the SSRC prefix to filter IP packets on the transport connection to identify those associated with the RTP subsession.

In non-relay topologies, the SDP offerer selects the first 24 bits of the SSRC for RTP streams associated with a media line that it transmits to the SDP answerer, and the SDP answerer selects the other value of the 1st bit and the same values of the remaining 23 bits of the SSRC prefix selected by the SDP offerer for RTP streams associated with the same media line that it transmits in the other direction to the SDP offerer. The final 8 bits of the SSRC are available to specify separate RTP streams within the RTP subsession. The SDP offerer selects the 24 bits of the SSRC prefix randomly for each RTP subsession in such a way that the first 8 bits are also unique across known RTP subsessions in the transport connection, and each endpoint selects the last 8 bits of the SSRC randomly for each RTP stream within an RTP subsession. The SDP offerer is allowed to specify either value for the 1st bit of the SSRC to allow the offerer and answerer to change roles during subsequent session modifications while allowing continued use of already assigned SSRC values.

When concatenating multiple RTP or RTCP packets within a single IP packet, as allowed by existing specifications, RTP systems will only combine packets from the same RTP subsession. RTP subsessions are thus segregated into separate IP packets to allow differential treatment in the network.

## 2.2. SDP procedures

The use of RTP subsessions is negotiated during the SDP offer/answer exchange as follows. When signaled in combination with BUNDLE, BUNDLE defines each group of media lines to share a transport connection, and the SDP attribute for RTP subsessions defines the SSRC prefix for each media line. Use of RTP subsessions without BUNDLE is possible but not defined within this document. This draft assumes that only one port is assigned for transport of RTP streams on each m line, since use of multiple ports, though allowed in [RFC4566], is rarely used. SDP subsessions can be readily enhanced to support multiple media ports per m line if necessary.

BUNDLE specifies how the connection and port information is to be set for each media line in a BUNDLE group. When RTP subsessions is used with BUNDLE, there is no requirement for the payload type numbers for each media line in the group to be non-overlapping, as required for BUNDLE without RTP subsessions. The payload type field is not needed to identify the media line associated with an RTP stream if the SSRC prefix is known.

A system supporting RTP subsessions will include an "ssrc-prefix" attribute for each media line in a sharing group, where each ssrc-prefix attribute includes a parameter that indicates whether the system is a relay (supporting RTP stream forwarding) and a parameter that is a hex representation of the 24-bit SSRC prefix proposed for use by the RTP subsession associated with the m line. If the endpoint expects to transmit or receive more than 256 RTP streams of the same media type, it should allocate more than one m line for that media type. The system will also include the rtcp-mux attribute [RFC5761] for each media line sharing the same transport connection. A system supporting RTP subsessions that receives SDP with ssrc-prefix attributes will assume the presence of the rtcp-mux attribute for each associated media line, even in its absence. While rtcp-mux could be made optional, there is no clear use case for supporting RTP subsessions without rtcp-mux.

In addition to including the requisite BUNDLE attributes, the SDP offer includes the ssrc-prefix attribute and the rtcp-mux attribute for each media line in the sharing group. If the SDP answerer is not a relay and agrees to use RTP subsessions (regardless of the role of the SDP offerer), or if the SDP offerer is not a relay and the SDP answerer is a relay that agrees to use RTP subsessions with the value(s) of the SSRC prefix in the SDP offer, then the SDP answerer also includes in the SDP answer the BUNDLE attributes, the ssrc-prefix attribute and the rtcp-mux attribute for each media line in the sharing group, with the same ssrc-prefix parameter values as the corresponding ones from the SDP offer, but with the 1st bit changed,

i.e., '0' becomes '1' or '1' becomes '0'. The SDP answerer can disable the use of RTP subsessions by not including the ssrc-prefix attributes in the answer.

If the SDP offerer is not a relay and the SDP answerer is a relay that selects not to use the SSRC prefix in the SDP offer for one or more of the media lines, then it includes in the SDP answer the BUNDLE attributes, the ssrc-prefix attribute and the rtcp-mux attribute for each media line in the sharing group, with its own ssrc-prefix parameter values for those selected media lines and as described in the previous paragraph for the remaining media lines. The SDP offerer must then use the ssrc-prefix values from the selected media lines in the SDP answer with the 1st bit changed as above. Note that for these selected media lines, any SSRC values selected by other SDP attributes (such as ssrc for msid) in the SDP offer are assumed by the SDP answerer to be remapped to include the ssrc-prefix from the SDP answer with the 1st bit changed and the original final 8 bits from the attribute in the SDP offer. The SDP offerer will also assume the use of the remapped values and include the remapped values in any subsequent SDP messages.

### 2.3. Additional procedures for relays

RTP subsessions will work without SSRC collisions for RTP system configurations consisting of a single master relay directly connected to up to 256 slave relays, where each relay may be directly connected to up to 256 end systems or mixers. Note that a master relay incorporates the function of at least one slave relay so that it can connect to non-relay systems. The only requirements are that the master relay initiate all connections to separate slave relays using SDP offer/answer procedures and that all participating systems in the RTP session comprising the RTP subsessions support SDP offer/answer procedures to negotiate RTP subsessions. The master relay picks the first 8 bits of the SSRC for every RTP subsession (used for filtering in the network), thus allowing up to 128 RTP subsessions within an RTP session. Each RTP subsession multiplexes traffic that is to receive the same QoS treatment throughout the RTP session. The master relay picks the first 16 bits of the SSRC that a slave relay can use with a particular RTP subsession and the slave relay picks the first 24 bits of the SSRC that an end system or mixer can use with a particular RTP subsession. The non-relay systems are free to allocate the last 8 bits of the SSRC to multiplex RTP streams on an RTP subsession.

The previous sections already describe how SDP offer/answer occurs between relay and non-relay systems to ensure that the relay determines the 24-bit SSRC prefix that the non-relay system uses for an RTP subsession, regardless of which system initiates the SDP

offer/answer procedure.

Once a relay system chooses the role of a master relay by selecting an SSRC prefix for an RTP subsession, it refuses to accept an SDP offer from any other relay. When sending an SDP offer to another system, without knowing if it is another relay, it selects a 24-bit SSRC prefix for each media line as already discussed, ensuring that the first 16 bits of the SSRC have not been assigned to any other transport connection. If the SDP answerer is not a relay, then the 24-bit SSRC prefix is reserved for the non-relay system to use for the RTP subsession. If the SDP answerer is another relay (a slave relay), it can accept the RTP subsession in the same manner as a non-relay system by responding with the same ssrc-prefix with the first bit changed. A slave relay is not allowed to respond to a master relay with a different SSRC prefix except for the changed first bit. Since the SDP offerer and answerer both signal that they are relays, they both understand that the master relay will reserve the first 16 bits of the SSRC to the slave relay, thus allowing the master relay to delegate SSRC prefix assignment to up to 256 slave relays and allowing each slave relay to select the 24-bit SSRC prefix for the RTP subsessions for up to 256 non-relay systems.

#### 2.4. When a system doesn't support RTP subsessions

When a non-relay system fails to negotiate the use of RTP subsessions with a peer, it will be difficult for the network to identify flows for differential treatment in the presence of BUNDLE type multiplexing. Once a relay has committed to RTP subsessions with one or more other systems, it has only two choices if a new system being added to the RTP session does not support RTP subsessions: it can re-negotiate with all existing systems to disable RTP subsessions; or it can perform SSRC mapping with the non-conformant system.

### 3. Applicability of RTP subsessions

Each RTP subsession is able to provide most of the capabilities of a full RTP session with some limitations associated with constraints on SSRC assignment. No changes are required to RTP or SRTP packet formats to realize RTP subsessions. The RTP streams associated with an individual media line can be easily identified for separate handling (e.g., to provide separate QoS treatment) by filtering on the first 8 bits of the SSRC field, in addition to the five-tuple for the transport connection. RTP systems can enable successful filtering on the port and SSRC fields, which are above the IP layer, by adhering to MTU limits to avoid IP fragmentation.

[RFC3550] describes three kinds of systems that can use RTP: end

systems, translators (relays) and mixers. RTP subsessions can be freely used by end systems and mixers since these systems can freely assign any SSRC value to each RTP stream and thus are free to detect and resolve SSRC conflicts. For interworking with systems not supporting RTP subsessions in particular, a relay can act as a mixer to reassign the SSRC value associated with an RTP stream before forwarding. When performing SSRC mapping rather than SRC forwarding, RTCP is handled differently and SRTP forwarding is more complex, requiring the decrypting and re-encrypting of each packet, as discussed in [I-D.westerlund-avtcore-multiplex-architecture]. While there is some additional processing needed to change SSRC when forwarding SRTP, there seems to be no consensus in RTCWEB to require support for forwarding with SSRC preservation. Systems forwarding RTP streams that provide additional media processing functions need to decrypt SRTP packets anyway.

The relay procedures for RTP subsessions allow support of most relay topologies as long as all systems in an RTP session support RTP subsessions. This is a limitation in the short term when interworking with existing systems, but if RTP subsession support is made available early in the deployment of RTCWEB systems, then new applications where relay systems, multiplexing of different media types, and differential treatment of media flows are desirable can require support for RTP subsessions.

Some applications require the ability to allocate the same SSRC value across multiple ports or media lines. RTP streams in different RTP subsessions are considered to use identical SSRC values if they match on the last 8 bits of the SSRC, so RTP subsessions can also be used for these applications.

Most RTP capabilities and extensions depend primarily on the use of a different SSRC for each RTP stream. Since RTP subsessions retain this characteristic, they introduce no new compatibility issues. Examples of such extensions include FEC, interleaving and RTP retransmissions.

Another short term limitation of RTP subsessions is that most networks capable of assigning differential treatment do so with the granularity of a five-tuple. The availability of a simple filter to identify flows for differential treatment allows deployment of DPI or custom gateways to enforce or verify DSCP markings in the short term. In the longer term, network policy control systems can be enhanced to perform SSRC filtering.



#### 4. Examples

In the first example, a non-relay SDP offerer desiring to use a single transport connection for an audio m line and a video m line might construct the following SDP offer. BUNDLE attributes are present but not shown.

```
C=...
m=audio 49170 RTP/AVP 96 97
a=rtcp-mux
a=ssrc-prefix: non-relay 0x111111
...
m=video 49170 RTP/AVP 98 99
a=rtcp-mux
a=ssrc-prefix: non-relay 0x222222
...
```

The non-relay SDP answerer agrees to use SDP subsessions as described in the SDP offer and accepts the SSRC prefixes for the two m lines as shown below. BUNDLE attributes are present but not shown.

```
C=...
m=audio 36008 RTP/AVP 96 97
a=rtcp-mux
a=ssrc-prefix: non-relay 0x911111
...
m=video 36008 RTP/AVP 98 99
a=rtcp-mux
a=ssrc-prefix: non-relay 0xa22222
...
```

The endpoints will establish one audio RTP subsession and one video RTP subsession associated with the SDP offer/answer exchange. These SDP subsessions and their corresponding RTCP will each share a single transport connection using ports 49170 and 36008, respectively. Media flows associated with each SDP subsession can be identified by filtering on the first 8 bits of the SSRC field as necessary for differential handling, e.g., to assign separate QoS treatment. The SDP offerer will send RTP streams on the two RTP subsessions for the audio and video media m lines using 24-bit SSRC prefixes 0x111111 and 0x222222, respectively. The SDP answerer will send RTP streams using 24-bit SSRC prefixes 0x911111 and 0xa22222, respectively. The network can filter on 8-bit SSRC prefixes 0x11 and 0x22 for packets in the five-tuple from the SDP offerer and the network can filter on 8-bit SSRC prefixes 0x91 and 0xa2 for packets in the five-tuple sent to the SDP offerer.

In another example to highlight how a relay may override an SSRC

prefix selected by a non-relay system, a non-relay SDP offerer desiring to use a single transport connection for an audio m line and a video m line might construct the following SDP offer. BUNDLE attributes are present but not shown.

```
C=...
m=audio 49170 RTP/AVP 96 97
a=rtcp-mux
a=ssrc-prefix: non-relay 0x111111
...
m=video 49170 RTP/AVP 98 99
a=rtcp-mux
a=ssrc-prefix: non-relay 0x222222
...
```

The SDP answerer performing as an RTP relay agrees to use SDP subsessions as described in the SDP offer. It accepts the SSRC prefix for the audio m line but selects an alternate SSRC prefix for the video m line as shown below. BUNDLE attributes are present but not shown.

```
C=...
m=audio 36008 RTP/AVP 96 97
a=rtcp-mux
a=ssrc-prefix: relay 0x911111
...
m=video 36008 RTP/AVP 98 99
a=rtcp-mux
a=ssrc-prefix: relay 0x333333
...
```

The endpoints will establish one audio RTP subsession and one video RTP subsession associated with the SDP offer/answer exchange. These SDP subsessions and their corresponding RTCP will each share a single transport connection using ports 49170 and 36008, respectively. Media flows associated with each SDP subsession can be identified by filtering on the first 8 bits of the SSRC field as necessary for differential handling, e.g., to assign separate QoS treatment. The non-relay SDP offerer will send RTP streams on the two RTP subsessions for the audio and video media m lines using 24-bit SSRC prefixes 0x111111 and 0xb33333, respectively. The SDP answerer performing as an RTP relay will send RTP streams (forwarded from other systems) using 8-bit SSRC prefixes 0x91 and 0x33, respectively. The network can filter on 8-bit SSRC prefixes 0x11 and 0xb3 for packets in the five-tuple from the SDP offerer and the network can filter on 8-bit SSRC prefixes 0x91 and 0x33 for packets in the five-tuple sent to the SDP offerer. Note that the 32-bit prefix for any ssrc values selected for RTP streams associated with the video m line

are assumed to be changed from 0x222222 to 0xb33333 while ssrc values selected for the audio line can remain unchanged.

## 5. Evaluation

### 5.1. Comparison to BUNDLE

BUNDLE [I-D.ietf-mmusic-sdp-bundle-negotiation] effectively merges RTP sessions together in relay topologies, thus slightly increasing the probability of SSRC collisions, although the impact is minor. In the context of RTCWEB, an SSRC collision, though rare, will require the use of an additional SDP offer/answer transaction to change msid/ssrc mappings, so is clearly undesirable. This can be avoided with RTP subsessions, since SDP offer/answer is used to pre-allocate SSRC ranges for each m line, thus completely avoiding SSRC collisions. It must also be assured that different RTP streams do not share the same SSRC, even though they use non-overlapping payload type values, so that each RTCP packet can be uniquely associated with a particular RTP stream. It is difficult to define a filter to allow the network to identify the RTP streams associated with different media lines with BUNDLE since this requires filtering on sets of payload type values. RTP subsessions can enhance BUNDLE for this purpose, since it is only necessary to screen for a single value in the first 8 bits of the SSRC. It is also not possible with BUNDLE to associate RTP streams from different m lines using a single SSRC value (as it is possible to do using the last 8 bits of the SSRC with RTP subsessions), due to the need to separate the RTCP messages for each RTP stream.

The author proposes that RTP subsessions be adopted to augment BUNDLE to eliminate the restrictions described above.

### 5.2. Comparison to SHIM

SHIM [I-D.westerlund-avtcore-transport-multiplexing] is the most successful scheme in preserving the SSRC space of each RTP session when multiplexing those RTP sessions onto a single transport connection. By placing the SHIM at the end of the RTP packet, however, it is less desirable to use SHIM as the basis for filtering of packets for differential treatment. SHIM also changes RTP so it can potentially impact many different middleboxes. It also slightly increases the size of each media packet, which can be of concern in some bandwidth-limited deployments.

The author recommends RTP subsessions over SHIM to better address two key requirements: to support flow identification for differential treatment; and to minimize impact on the RTP packet format. Given

the increasing use of pre-selected SSRC values to identify individual RTP streams, which is somewhat inconsistent with the idea of random SSRC assignment and the use of SSRC collision detection and resolution schemes, the author also recommends the use of an effective SSRC collision avoidance scheme based on SSRC range reservation, such as the one defined for RTP subsessions.

### 5.3. Comparison to other SSRC proposals

Two expired drafts, [I-D.rosenberg-rtcweb-rtpmux] and [I-D.peterson-rosenberg-avt-rtp-ssrc-demux], propose other means of multiplexing based on the SSRC field. [I-D.rosenberg-rtcweb-rtpmux] uses a portion of the SSRC to identify the media type for the RTP stream. This scheme redefines the SSRC field and has significant limitations when multiple m lines share the same media type, since some other mechanism is still needed to separate them. [I-D.peterson-rosenberg-avt-rtp-ssrc-demux] assumes a single SSRC value per m line, so is not consistent with current RTP multiplexing requirements.

Neither earlier SSRC-based scheme fully addresses the requirements for multiplexing agreed in the working groups.

## 6. IANA Considerations

To be completed.

## 7. Security Considerations

To be completed.

## 8. Informative References

[I-D.ietf-mmusic-sdp-bundle-negotiation]  
Holmberg, C. and H. Alvestrand, "Multiplexing Negotiation Using Session Description Protocol (SDP) Port Numbers", draft-ietf-mmusic-sdp-bundle-negotiation-00 (work in progress), February 2012.

[I-D.ietf-rtcweb-rtp-usage]  
Perkins, C., Westerlund, M., and J. Ott, "Web Real-Time Communication (WebRTC): Media Transport and Use of RTP", draft-ietf-rtcweb-rtp-usage-02 (work in progress), March 2012.

- [I-D.peterson-rosenberg-avt-rtp-ssrc-demux]  
Peterson, J. and J. Rosenberg, "A Multiplexing Mechanism for the Real-Time Protocol (RTP)",  
draft-peterson-rosenberg-avt-rtp-ssrc-demux-00 (work in progress), July 2004.
- [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-max-ssrc]  
Westerlund, M., Burman, B., and F. Jansson, "Multiple Synchronization sources (SSRC) in RTP Session Signaling",  
draft-westerlund-avtcore-max-ssrc-01 (work in progress), April 2012.
- [I-D.westerlund-avtcore-multiplex-architecture]  
Westerlund, M., Burman, B., and C. Perkins, "RTP Multiplexing Architecture",  
draft-westerlund-avtcore-multiplex-architecture-01 (work in progress), March 2012.
- [I-D.westerlund-avtcore-transport-multiplexing]  
Westerlund, M. and C. Perkins, "Multiple RTP Sessions on a Single Lower-Layer Transport",  
draft-westerlund-avtcore-transport-multiplexing-02 (work in progress), March 2012.
- [RFC3550] Schulzrinne, H., Casner, S., Frederick, R., and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications", STD 64, RFC 3550, July 2003.
- [RFC4566] Handley, M., Jacobson, V., and C. Perkins, "SDP: Session Description Protocol", RFC 4566, July 2006.
- [RFC5245] Rosenberg, J., "Interactive Connectivity Establishment (ICE): A Protocol for Network Address Translator (NAT) Traversal for Offer/Answer Protocols", RFC 5245, April 2010.
- [RFC5761] Perkins, C. and M. Westerlund, "Multiplexing RTP Data and Control Packets on a Single Port", RFC 5761, April 2010.
- [RFC6347] Rescorla, E. and N. Modadugu, "Datagram Transport Layer

Security Version 1.2", RFC 6347, January 2012.

Author's Address

Richard Ejzak  
Alcatel-Lucent  
1960 Lucent Lane  
Naperville, Illinois 60563-1594  
US

Phone: +1 630 979 7036  
Email: richard.ejzak@alcatel-lucent.com



Audio/Video Transport Core  
Maintenance  
Internet-Draft  
Intended status: Standards Track  
Expires: January 4, 2013

A. Williams  
Audinate  
K. Gross  
AVA Networks  
R. van Brandenburg  
H. Stokking  
TNO  
July 3, 2012

RTP Clock Source Signalling  
draft-ietf-avtcore-clksrc-00

Abstract

NTP timestamps are used by several RTP protocols for synchronisation and statistical measurement. This memo specifies SDP signalling identifying NTP timestamp clock sources and SDP signalling identifying the media clock sources in a multimedia session.

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 [1].

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 January 4, 2013.

Copyright Notice

Copyright (c) 2012 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. Applications . . . . .	3
3. Definitions . . . . .	4
4. Timestamp Reference Clock Source Signalling . . . . .	5
4.1. Clock synchronization . . . . .	5
4.2. Identifying NTP Reference Clocks . . . . .	6
4.3. Identifying PTP Reference Clocks . . . . .	6
4.4. Identifying Global Reference Clocks . . . . .	7
4.5. Other Reference Clocks . . . . .	8
4.6. Traceable Reference Clocks . . . . .	8
4.7. Synchronisation Confidence . . . . .	8
4.8. SDP Signalling of Timestamp Clock Source . . . . .	9
4.8.1. Examples . . . . .	11
5. Media Clock Source Signalling . . . . .	12
5.1. Asynchronously Generated Media Clock . . . . .	13
5.2. Direct-Referenced Media Clock . . . . .	13
5.3. Stream-Referenced Media Clock . . . . .	13
5.4. Signalling Grammar . . . . .	14
5.5. Examples . . . . .	16
6. IANA Considerations . . . . .	17
7. References . . . . .	19
7.1. Normative References . . . . .	19
7.2. Informative References . . . . .	19
Authors' Addresses . . . . .	20

## 1. Introduction

RTP protocols use NTP format timestamps to facilitate media stream synchronisation and for providing estimates of round trip time (RTT) and other statistical parameters.

Information about media clock timing exchanged in NTP format timestamps may come from a clock which is synchronised to a global time reference, but this cannot be assumed nor is there a standardised mechanism available to indicate that timestamps are derived from a common reference clock. Therefore, RTP implementations typically assume that NTP timestamps are taken using unsynchronised clocks and must compensate for absolute time differences and rate differences. Without a shared reference clock, RTP can time align flows from the same source at a given receiver using relative timing, however tight synchronisation between two or more different receivers (possibly with different network paths) or between two or more senders is not possible.

High performance AV systems often use a reference media clock distributed to all devices in the system. The reference media clock is often distinct from the reference clock used to provide timestamps. A reference media clock may be provided along with an audio or video signal interface, or via a dedicated clock signal (e.g. genlock [12] or audio word clock [13]). If sending and receiving media clocks are known to be synchronised to a common reference clock, performance can be improved by minimising buffering and avoiding rate conversion.

This specification defines SDP signalling of timestamp clock sources and media reference clock sources.

## 2. Applications

Timestamp clock source and reference media clock signalling benefit applications requiring synchronised media capture or playout and low latency operation.

Examples include, but are not limited to:

Social TV RTCP for inter-destination media synchronization [6] defines social TV as the combination of media content consumption by two or more users at different devices and locations and real-time communication between those users. An example of Social TV, is where two or more users are watching the same television broadcast at different devices and/or locations, while communicating with each other using text, audio and/or video. A

skew in the media playout of the two or more users can have adverse effects on their experience. A well-known use case here is one friend experiencing a goal in a football match well before or after other friends.

**Video Walls** A video wall consists of multiple computer monitors, video projectors, or television sets tiled together contiguously or overlapped in order to form one large screen. Each of the screens reproduces a portion of the larger picture. In some implementations, each screen or projector may be individually connected to the network and receive its portion of the overall image from a network-connected video server or video scaler. Screens are refreshed at 50 or 60 hertz or potentially faster. If the refresh is not synchronized, the effect of multiple screens acting as one is broken.

**Networked Audio** Networked loudspeakers, amplifiers and analogue I/O devices transmitting or receiving audio signals via RTP can be connected to various parts of a building or campus network. Such situations can for example be found in large conference rooms, legislative chambers, classrooms (especially those supporting distance learning) and other large-scale environments such as stadiums. Since humans are more susceptible to differences in audio delay, this use case needs even more accuracy than the video wall use case. Depending on the exact application, the need for accuracy can then be in the range of microseconds [14].

**Sensor Arrays** Sensor arrays contain many synchronised measurement elements producing signals which are then combined to form an overall measurement. Accurate capture of the phase relationships between the various signals arriving at each element of the array is critically important for proper operation. Examples include towed or fixed sonar arrays, seismic arrays and phased arrays used in radar applications, for instance.

### 3. Definitions

The definitions of streams, sources and levels of information in SDP descriptions follow the definitions found in Source-Specific Media Attributes in the Session Description Protocol (SDP) [2].

**multimedia session** A set of multimedia senders and receivers as well as the data streams flowing from senders to receivers. The Session Description Protocol (SDP) [3] describes multimedia sessions.

**media stream** An RTP session potentially containing more than one RTP source. SDP media descriptions beginning with an "m"-line define the parameters of a media stream.

**media source** A media source is single stream of RTP packets, identified by an RTP SSRC.

**session level** Session level information applies to an entire multimedia session. In an SDP description, session-level information appears before the first "m"-line.

**media level** Media level information applies to a single media stream (RTP session). In an SDP description, media-level information appears after each "m"-line.

**source level** Source level information applies to a single stream of RTP packets, identified by an RTP SSRC. Source-Specific Media Attributes in the Session Description Protocol (SDP) [2] defines how source-level information is included into an SDP session description.

**traceable time** A clock is considered to provide traceable time if it can be proven to be synchronised to a global time reference. GPS [7] is commonly used to provide a traceable time reference. Some network time synchronisation protocols (e.g. PTP [8], NTP) can explicitly indicate that the master clock is providing a traceable time reference over the network.

#### 4. Timestamp Reference Clock Source Signalling

The NTP timestamps used by RTP are taken by reading a local real-time clock at the sender or receiver. This local clock may be synchronised to another clock (time source) by some means or it may be unsynchronised. A variety of methods are available to synchronise local clocks to a reference time source, including network time protocols (e.g. NTP [9]) and radio clocks like GPS [7].

The following sections describe and define SDP signalling, indicating whether and how the local timestamping clock in an RTP sender/receiver is synchronised to a reference clock.

##### 4.1. Clock synchronization

Two or more local clocks that are sufficiently synchronised will produce timestamps for a given RTP event can be used as if they came from the same clock. Providing they are sufficiently synchronised, timestamps produced in one RTP sender/receiver can be directly

compared to a local clock in another RTP sender/receiver. The timestamps produced by synchronized local clocks in two or more RTP senders/receivers can be directly compared.

The accuracy of synchronization required is application dependent. See Applications (Section 2) section for a discussion of applications and their corresponding requirements. To serve as a reference clock, clocks must minimally be syntonized (exactly frequency matched) to one another.

Sufficient synchronization can typically be achieving by using a network time protocol (e.g. NTP, 802.1AS, IEEE 1588-2008) to synchronize all devices to a single master clock.

Another approach is to use clocks providing a global time reference (e.g. GPS, Galileo). This concept may be used in conjunction with network time protocols as some protocols (e.g. PTP, NTP) allow master clocks to indicate explicitly that they are "traceable" back to a global time reference.

#### 4.2. Identifying NTP Reference Clocks

A single NTP server is identified by hostname (or IP address) and an optional port number. If the port number is not indicated, it is assumed to be the standard NTP port (123).

Two or more NTP servers may be listed at the same level in the session description to indicate that they are interchangeable. RTP senders/receivers can use any of the listed NTP servers to govern a local clock that is equivalent to a local clock slaved to a different server.

#### 4.3. Identifying PTP Reference Clocks

The IEEE 1588 Precision Time Protocol (PTP) family of clock synchronisation protocols provides a shared reference clock in an network - typically a LAN. IEEE 1588 provides sub-microsecond synchronisation between devices on a LAN and typically locks within seconds at startup. With support from Ethernet switches, IEEE 1588 protocols can achieve nanosecond timing accuracy in LANs. Network interface chips and cards supporting hardware time-stamping of timing critical protocol messages are also available.

Three flavours of IEEE 1588 are in use today:

- o IEEE 1588-2002 [10]: the original "Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems". This is also known as IEEE1588v1 or PTPv1.

- o IEEE 1588-2008 [8]: the second version of the "Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems". This is a revised version of the original IEEE1588-2002 standard and is also known as IEEE1588v2 or PTPv2. IEEE 1588-2008 is not protocol compatible with IEEE 1588-2002.
- o IEEE 802.1AS [11]: "Timing and Synchronization for Time Sensitive Applications in Bridged Local Area Networks". This is a Layer-2 only profile of IEEE 1588-2008 for use in Audio/Video Bridged LANs.

Each IEEE 1588 clock is identified by a globally unique EUI-64 called a "ClockIdentity". A slave clock using one of the IEEE 1588 family of network time protocols acquires the ClockIdentity/EUI-64 of the grandmaster clock that is the ultimate source of timing information for the network. A master clock which is itself slaved to another master clock passes the grandmaster ClockIdentity through to its slaves.

Several instances of the IEEE 1588 protocol may operate independently on a single network, forming distinct PTP network protocol domains, each of which may have a different grandmaster clock. As the IEEE 1588 standards have developed, the definition of PTP domains has changed. IEEE 1588-2002 identifies protocol subdomains by a textual name, but IEEE 1588-2008 identifies protocol domains using a numeric domain number. 802.1AS is a Layer-2 profile of IEEE 1588-2008 supporting a single numeric clock domain (0).

When PTP subdomains are signalled via SDP, senders and receivers SHOULD check that both grandmaster ClockIdentity and PTP subdomain match when determining clock equivalence.

The PTP protocols employ a distributed election protocol called the "Best Master Clock Algorithm" (BMCA) to determine the active clock master. The clock master choices available to BMCA can be restricted or favourably biased by setting stratum values, preferred master clock bits, or other parameters to influence the election process. In some systems it may be desirable to limit the number of possible PTP clock masters to avoid re-signalling timestamp clock sources when the clock master changes.

#### 4.4. Identifying Global Reference Clocks

Global reference clocks provide a source of traceable time, typically via a hardware radio receiver interface. Examples include GPS and Galileo. Apart from the name of the reference clock system, no further identification is required.

#### 4.5. Other Reference Clocks

At the time of writing, it is common for RTP senders/receivers not to synchronise their local timestamp clocks to a shared master. An unsynchronised clock such as a quartz oscillator is identified as a "local" reference clock.

In some systems, all RTP senders/receivers may use a timestamp clock synchronised to a reference clock that is not provided by one of the methods listed above. Examples may include the reference time information provided by digital television or cellular services. These sources are identified as "private" reference clocks. All RTP senders/receivers in a session using a private reference clock are assumed to have a mechanism outside this specification confirming that their local timestamp clocks are equivalent.

#### 4.6. Traceable Reference Clocks

A timestamp clock source may be labelled "traceable" if it is known to be sourced from a global time reference such as TAI or UTC. Providing adjustments are made for differing time bases, timestamps taken using clocks synchronised to a traceable time source can be directly compared even if the clocks are synchronised to different sources or via different mechanisms.

Since all NTP and PTP servers providing traceable time can be directly compared, it is not necessary to identify traceable time servers by protocol address or other identifiers.

#### 4.7. Synchronisation Confidence

Network time protocol services periodically exchange timestamped messages between servers and clients. Assuming RTP sender/receiver clocks are based on commonly available quartz crystal hardware which is subject to drift, tight synchronisation requires frequent exchange of synchronisation messages.

Unfortunately, in some implementations, it is not possible to control the frequency of synchronisation messages nor is it possible to discover when the last synchronisation message occurred. In order to provide a measure of confidence that the timestamp clock is sufficiently synchronised, an optional timestamp may be included in the SDP clock source signalling. In addition, the frequency of synchronisation message may also be signalled.

The optional timestamp and synchronisation frequency parameters provide an indication of synchronisation quality to the receiver of those parameters. If the synchronisation confidence timestamp is far

from the timestamp clock at the receiver of the parameters, it can be assumed that synchronisation has not occurred recently or the timestamp reference clock source cannot be contacted. In this case, the receiver can take action to prevent unsynchronised playout or may fall back to assuming that the timestamp clocks are not synchronised.

Synchronisation frequency is expressed as a signed (two's-complement) 8-bit field which is the base-2 logarithm of the frequency in Hz. The synchronisation frequencies represented by this field range from  $2^{-128}$  Hz to  $2^{+127}$  Hz. The field value of 0 corresponds to an update frequency of 1 Hz.

#### 4.8. SDP Signalling of Timestamp Clock Source

Specification of the timestamp reference clock source may be at any or all levels (session, media or source) of an SDP description (see level definitions (Section 3) earlier in this document for more information).

Timestamp clock source signalling included at session-level provides default parameters for all RTP sessions and sources in the session description. More specific signalling included at the media level overrides default session level signalling. Further, source-level signalling overrides timestamp clock source signalling at the enclosing media level and session level.

If timestamp clock source signalling is included anywhere in an SDP description, it must be properly defined for all levels in the description. This may simply be achieved by providing default signalling at the session level.

Timestamp reference clock parameters may be repeated at a given level (i.e. for a session or source) to provide information about additional servers or clock sources. If the attribute is repeated at a given level, all clocks described at that level are assumed to be equivalent. Traceable clock sources MUST NOT be mixed with non-traceable clock sources at any given level. Unless synchronisation confidence information is available for each of the reference clocks listed at a given level, it SHOULD only be included with the first reference clock source attribute at that level.

Note that clock source parameters may change from time to time, for example, as a result of a PTP clock master election. The SIP [4] protocol supports re-signalling of updated SDP information, however other protocols may require additional notification mechanisms.

Figure 1 shows the ABNF [5] grammar for the SDP reference clock source information.



```

timestamp-refclk = "a=ts-refclk:" clksrc [ SP sync-confidence ] CRLF

clksrc = ntp / ptp / gps / gal / local / private

ntp          = "ntp=" ntp-server-addr
ntp-server-addr = host [ ":" port ]
ntp-server-addr =/ "traceable" )

ptp          = "ptp=" ptp-version ":" ptp-gmid [ ":" ptp-domain]
ptp-version  = "IEEE1588-2002"
ptp-version  =/ "IEEE1588-2008"
ptp-version  =/ "IEEE802.1AS-2011"
ptp-gmid     = EUI64
ptp-gmid     =/ "traceable"
ptp-domain   = ptp-domain-name / ptp-domain-nmbr
ptp-domain-name = "domain-name=" 16ptp-domain-char
ptp-domain-char = %x21-7E / %x00
                ; allowed characters: 0x21-0x7E (IEEE 1588-2002)
ptp-domain-nmbr = "domain-nmbr=" %x00-7f
                ; allowed number range: 0-127 (IEEE 1588-2008)

gps          = "gps"
gal          = "gal"
local        = "local"
private      = "private" [ ":" "traceable" ]

sync-confidence = sync-timestamp [SP sync-frequency]

sync-timestamp = sync-date SP sync-time SP sync-UTCoffset

sync-date      = 4DIGIT "-" 2DIGIT "-" 2DIGIT
                ; yyyy-mm-dd (e.g., 1982-12-02)

sync-time      = 2DIGIT ":" 2DIGIT ":" 2DIGIT "." 3DIGIT
                ; 00:00:00.000 - 23:59:59.999

sync-UTCoffset = ( "+" / "-" ) 2DIGIT ":" 2DIGIT
                ; +HH:MM or -HH:MM

sync-frequency = 2HEXDIG
                ; If N is the field value, HZ=2^(N-127)

host           = hostname / IPv4address / IPv6reference

hostname       = *( domainlabel "." ) toplabel [ "." ]
toplabel       = ALPHA / ALPHA *( alphanum / "-" ) alphanum

```

```

domainlabel    = alphanum
                =/ alphanum *( alphanum / "-" ) alphanum

IPv4address    = 1*3DIGIT "." 1*3DIGIT "." 1*3DIGIT "." 1*3DIGIT
IPv6reference  = "[" IPv6address "]"
IPv6address    = hexpart [ ":" IPv4address ]
hexpart        = hexseq / hexseq "::" [ hexseq ] / "::" [ hexseq ]
hexseq         = hex4 *( ":" hex4 )
hex4           = 1*4HEXDIG

port           = 1*DIGIT

EUI64          = 7(2HEXDIG "-") 2HEXDIG

```

Figure 1: Timestamp Reference Clock Source Signalling

#### 4.8.1. Examples

Figure 2 shows an example SDP description with a timestamp reference clock source defined at the session level.

```

v=0
o=jdoe 2890844526 2890842807 IN IP4 10.47.16.5
s=SDP Seminar
i=A Seminar on the session description protocol
u=http://www.example.com/seminars/sdp.pdf
e=j.doe@example.com (Jane Doe)
c=IN IP4 224.2.17.12/127
t=2873397496 2873404696
a=recvonly
a=ts-refclk:ntp=traceable
m=audio 49170 RTP/AVP 0
m=video 51372 RTP/AVP 99
a=rtpmap:99 h263-1998/90000

```

Figure 2: Timestamp reference clock definition at the session level

Figure 3 shows an example SDP description with timestamp reference clock definitions at the media level overriding the session level defaults. Note that the synchronisation confidence timestamp appears on the first attribute at the media level only.

```
v=0
o=jdoe 2890844526 2890842807 IN IP4 10.47.16.5
s=SDP Seminar
i=A Seminar on the session description protocol
u=http://www.example.com/seminars/sdp.pdf
e=j.doe@example.com (Jane Doe)
c=IN IP4 224.2.17.12/127
t=2873397496 2873404696
a=recvonly
a=ts-refclk:local
m=audio 49170 RTP/AVP 0
a=ts-refclk:ntp=203.0.113.10 2011-02-19 21:03:20.345+01:00
a=ts-refclk:ntp=198.51.100.22
m=video 51372 RTP/AVP 99
a=rtpmap:99 h263-1998/90000
a=ts-refclk:ptp=IEEE802.1AS-2011:39-A7-94-FF-FE-07-CB-D0
```

Figure 3: Timestamp reference clock definition at the media level

Figure 4 shows an example SDP description with a timestamp reference clock definition at the source level overriding the session level default.

```
v=0
o=jdoe 2890844526 2890842807 IN IP4 10.47.16.5
s=SDP Seminar
i=A Seminar on the session description protocol
u=http://www.example.com/seminars/sdp.pdf
e=j.doe@example.com (Jane Doe)
c=IN IP4 224.2.17.12/127
t=2873397496 2873404696
a=recvonly
a=ts-refclk:local
m=audio 49170 RTP/AVP 0
m=video 51372 RTP/AVP 99
a=rtpmap:99 h263-1998/90000
a:ssrc:12345 ts-refclk:ptp=IEEE802.1AS-2011:39-A7-94-FF-FE-07-CB-D0
```

Figure 4: Timestamp reference clock signalling at the source level

## 5. Media Clock Source Signalling

The media clock source for a stream determines the timebase used to advance the RTP timestamps included in RTP packets. The media clock may be asynchronously generated by the sender, it may be generated in fixed relationship to the reference clock or it may be generated with respect to another stream on the network (which is presumably being

received by the sender).

#### 5.1. Asynchronously Generated Media Clock

In the simplest sender implementation, the sender generates media by sampling audio or video according to a free-running local clock. The RTP timestamps in media packets are advanced according to this media clock and packet transmission is typically timed to regular intervals on this timeline. The sender may or may not include an NTP timestamp in sender reports to allow mapping of this asynchronous media clock to a reference clock.

The asynchronously generated media clock is the assumed mode of operation when there is no signalling of media clock source. Alternatively, asynchronous media clock may be signaled.

```
a=mediaclock:sender
```

#### 5.2. Direct-Referenced Media Clock

A media clock may be directly derived from a reference clock. For this case it is required that a reference clock be specified. The signalling indicates a media clock offset value at the epoch (time of origin) of the reference clock. A rate for the media clock may also be specified. If include, the rate specification here overrides that specified or implied by the media description. If omitted, the rate is assumed to be the exact rate used by the media format. For example, the media clock for an 8 kHz G.711 audio stream will advance exactly 8000 units for each second advance in the reference clock from which it is derived.

A rate modifier may optionally be expressed as the ratio of two integers. This provision is useful for accommodating certain "oddball rates" associated with NTSC video (see Figure 7).

```
a=mediaclock:offset=<offset>[ rate=<rate numerator>/<rate denominator> ]
```

#### 5.3. Stream-Referenced Media Clock

The media clock for an outgoing stream may be generated based on the media clock received with an incoming stream. In this case, the signalling identifies the session and the stream source. The received media clock may be converted to a real-time clock which can then be used to generate outgoing media clocks. In this way, the format of the reference stream does not need to match the format of the outgoing stream.

A reference stream can be either another RTP stream or AVB stream based on the IEEE 1722 standard. An RTP stream is identified by destination IP address (for a multicast stream) or source IP address (for a unicast stream), destination port number and CNAME of the source.

```
a=mediaclock:rtp=<connection address>:<port> <CNAME>
```

An IEEE 1722 stream is identified by its StreamID, an EUI-64.

```
a=mediaclock:IEEE1722=<StreamID>
```

#### 5.4. Signalling Grammar

Specification of the media clock source may be at any or all levels (session, media or source) of an SDP description (see level definitions (Section 3) earlier in this document for more information).

Media clock source signalling included at session level provides default parameters for all RTP sessions and sources in the session description. More specific signalling included at the media level overrides default session level signalling. Further, source-level signalling overrides media clock source signalling at the enclosing media level and session level.

Media clock source signalling may be present or absent on a per-stream basis. In the absence of media clock source signals, receivers assume an asynchronous media clock generated by the sender.

Media clock source parameters may be repeated at a given level (i.e. for a session or source) to provide information about additional clock sources. If the attribute is repeated at a given level, all clocks described at that level are comparable clock sources and may be used interchangeably.

Figure 5 shows the ABNF [5] grammar for the SDP media clock source information.

```
timestamp-mediaclock = "a=mediaclock:" mediaclock
```

```
mediaclock = refclk / rtp / streamid / sender
```

```
refclk = "offset=" 1*DIGIT [ SP "rate=" 1*DIGIT "/" 1*DIGIT ]
```

```
rtp = "rtp=" nettype SP addrtype SP connection-address SP port SP cname
```

```
streamid = "IEEE1722=" EUI64
```

```
sender = "sender"

cname = non-ws-string

nettype = token
        ;typically "IN"

addrtype = token
        ;typically "IP4" or "IP6"

token-char = %x21 / %x23-27 / %x2A-2B / %x2D-2E / %x30-39 / %x41-5A
            / %x5E-7E

token = 1*(token-char)

connection-address = multicast-address / unicast-address

unicast-address = IP4-address / IP6-address / FQDN / extn-addr

multicast-address = IP4-multicast / IP6-multicast / FQDN / extn-addr

IP4-multicast = m1 3( "." decimal-uchar ) "/" ttl [ "/" integer ]
        ; IPv4 multicast addresses may be in the
        ; range 224.0.0.0 to 239.255.255.255

m1 = ("22" ("4"/"5"/"6"/"7"/"8"/"9")) / ("23" DIGIT )

IP6-multicast = hexpart [ "/" integer ]
        ; IPv6 address starting with FF

FQDN = 4*(alpha-numeric / "-" / ".")
        ; fully qualified domain name as specified
        ; in RFC 1035 (and updates)

IP4-address = b1 3("." decimal-uchar)

b1 = decimal-uchar
        ; less than "224"

; The following is consistent with RFC 2373 [30], Appendix B.
IP6-address = hexpart [ ":" IP4-address ]

hexpart = hexseq / hexseq "::" [ hexseq ] / "::" [ hexseq ]

hexseq = hex4 *( ":" hex4)

hex4 = 1*4HEXDIG
```

```

; Generic for other address families
extn-addr = non-ws-string

non-ws-string = 1*(VCHAR/%x80-FF)
               ;string of visible characters

port = 1*DIGIT

EUI64 = 7(2HEXDIG "-") 2HEXDIG

```

Figure 5: Media Clock Source Signalling

### 5.5. Examples

Figure 6 shows an example SDP description 8 channels of 24-bit, 48 kHz audio transmitted as a multicast stream. Media clock is derived directly from an IEEE 1588-2008 reference.

```

v=0
o=- 1311738121 1311738121 IN IP4 192.168.1.1
c=IN IP4 239.0.0.2/255
s=
t=0 0
m=audio 5004 RTP/AVP 96
a=rtpmap:96 L24/48000/8
a=sendonly
a=ts-refclk:ptp=IEEE1588-2008:39-A7-94-FF-FE-07-CB-D0:0
a=mediaclock:offset=963214424

```

Figure 6: Media clock directly referenced to IEEE 1588-2008

Figure 7 shows an example SDP description 2 channels of 24-bit, 44056 kHz NTSC "pull-down" media clock derived directly from an IEEE 1588-2008 reference clock

```

v=0
o=- 1311738121 1311738121 IN IP4 192.168.1.1
c=IN IP4 239.0.0.2/255
s=
t=0 0
m=audio 5004 RTP/AVP 96
a=rtpmap:96 L24/44100/2
a=sendonly
a=ts-refclk:ptp=IEEE1588-2008:39-A7-94-FF-FE-07-CB-D0:0
a=mediaclock:offset=963214424 rate=1000/1001

```

Figure 7: "Oddball" sample rate directly referenced to IEEE 1588-2008

Figure 8 shows the same 48 kHz audio transmission from Figure 6 with media clock derived from another RTP multicast stream. The stream providing the media clock must use the same reference clock as this stream that references it.

```
v=0
o=- 1311738121 1311738121 IN IP4 192.168.1.1
c=IN IP4 224.2.228.230/32
s=
t=0 0
m=audio 5004 RTP/AVP 96
a=rtpmap:96 L24/48000/2
a=sendonly
a=ts-refclk:ptp=IEEE1588-2008:39-A7-94-FF-FE-07-CB-D0:0
a=mediaclock:rtp=IN IP4 239.0.0.1 5004 00:60:2b:20:12:if
```

Figure 8: Stream media clock derived from another RTP multicast stream

Figure 9 shows the same 48 kHz audio transmission from Figure 6 with media clock derived from an IEEE 1722 AVB stream. The stream providing the media clock must be synchronized with the IEEE 1588-2008 reference clock used by this stream.

```
v=0
o=- 1311738121 1311738121 IN IP4 192.168.1.1
c=IN IP4 224.2.228.230/32
s=
t=0 0
m=audio 5004 RTP/AVP 96
a=rtpmap:96 L24/48000/2
a=sendonly
a=ts-refclk:ptp=IEEE1588-2008:39-A7-94-FF-FE-07-CB-D0:0
a=mediaclock:IEEE1722=38-D6-6D-8E-D2-78-13-2F
```

Figure 9: Stream media clock derived from another RTP multicast stream

## 6. IANA Considerations

The SDP attribute "ts-refclk" defined by this document is registered with the IANA registry of SDP Parameters as follows:



SDP Attribute ("att-field"):

Attribute name:       ts-refclk  
Long form:             Timestamp reference clock source  
Type of name:          att-field  
Type of attribute:    session, media and source level  
Subject to charset:   no  
Purpose:               See section 4 of this document  
Reference:             This document  
Values:                see this document and registrations below

The attribute has an extensible parameter field and therefore a registry for these parameters is required. This document creates an IANA registry called the Timestamp Reference Clock Source Parameters Registry. It contains the six parameters defined in Figure 1: "ntp", "ptp", "gps", "gal", "local", "private".

The SDP attribute "mediaclock" defined by this document is registered with the IANA registry of SDP Parameters as follows:

SDP Attribute ("att-field"):

Attribute name:       mediaclock  
Long form:             Media clock source  
Type of name:          att-field  
Type of attribute:    session and media level  
Subject to charset:   no  
Purpose:               See section 6 of this document  
Reference:             This document  
Values:                see this document and registrations below

The attribute has an extensible parameter field and therefore a

registry for these parameters is required. This document creates an IANA registry called the Media Clock Source Parameters Registry. It contains the three parameters defined in Figure 5: "refclk", "ssrc", "sender".

## 7. References

### 7.1. Normative References

- [1] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [2] Lennox, J., Ott, J., and T. Schierl, "Source-Specific Media Attributes in the Session Description Protocol (SDP)", RFC 5576, June 2009.
- [3] Handley, M., Jacobson, V., and C. Perkins, "SDP: Session Description Protocol", RFC 4566, July 2006.
- [4] Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston, A., Peterson, J., Sparks, R., Handley, M., and E. Schooler, "SIP: Session Initiation Protocol", RFC 3261, June 2002.
- [5] Crocker, D., Ed. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", RFC 2234, November 1997.

### 7.2. Informative References

- [6] Brandenburg, R., Stokking, H., Boronat, F., Montagud, M., and K. Gross, "RTCP for inter-destination media synchronization", draft-ietf-avtcore-idms-04 (work in progress), May 2012.
- [7] Global Positioning Systems Directorate, "Navstar GPS Space Segment/Navigation User Segment Interfaces", September 2011.
- [8] Institute of Electrical and Electronics Engineers, "1588-2008 - IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems", IEEE Std 1588-2008, 2008, <<http://standards.ieee.org/findstds/standard/1588-2008.html>>.
- [9] Mills, D., Martin, J., Burbank, J., and W. Kasch, "Network Time Protocol Version 4: Protocol and Algorithms Specification", RFC 5905, June 2010.
- [10] Institute of Electrical and Electronics Engineers, "1588-2002 - IEEE Standard for a Precision Clock Synchronization Protocol

for Networked Measurement and Control Systems", IEEE Std 1588-2002, 2002,  
<<http://standards.ieee.org/findstds/standard/1588-2002.html>>.

- [11] "Timing and Synchronization for Time-Sensitive Applications in Bridged Local Area Networks",  
<<http://standards.ieee.org/findstds/standard/802.1AS-2011.html>>.

#### URIs

- [12] <<http://en.wikipedia.org/wiki/Genlock>>  
[13] <[http://en.wikipedia.org/wiki/Word\\_clock](http://en.wikipedia.org/wiki/Word_clock)>  
[14] <<http://www.ieee802.org/1/files/public/docs2007/as-dolsen-time-accuracy-0407.pdf>>

#### Authors' Addresses

Aidan Williams  
Audinate  
Level 1, 458 Wattle St  
Ultimo, NSW 2007  
Australia

Phone: +61 2 8090 1000  
Fax: +61 2 8090 1001  
Email: [aidan.williams@audinate.com](mailto:aidan.williams@audinate.com)  
URI: <http://www.audinate.com/>

Kevin Gross  
AVA Networks  
Boulder, CO  
US

Email: [kevin.gross@avanw.com](mailto:kevin.gross@avanw.com)  
URI: <http://www.avanw.com/>

Ray van Brandenburg  
TNO  
Brassersplein 2  
Delft 2612CT  
the Netherlands

Phone: +31-88-866-7000  
Email: ray.vanbrandenburg@tno.nl

Hans Stokking  
TNO  
Brassersplein 2  
Delft 2612CT  
the Netherlands

Phone:  
Email: stokking@tno.nl



AVTCore  
Internet-Draft  
Intended status: Standards Track  
Expires: January 17, 2013

R. van Brandenburg  
H. Stokking  
O. van Deventer  
TNO  
F. Boronat  
M. Montagud  
Universitat Politecnica de  
Valencia  
K. Gross  
AVA Networks  
July 16, 2012

Inter-destination Media Synchronization using the RTP Control Protocol  
(RTCP)  
draft-ietf-avtcore-idms-06

Abstract

This document gives information on an RTP Control Protocol (RTCP) Packet Type and RTCP Extended Report (XR) Block Type including associated Session Description Protocol (SDP) parameters for Inter-Destination Media Synchronization (IDMS). This document mandates the use of the RTCP XR Block Type 12, which is used to collect media playout information from participants in a group playing out (watching, listening, etc.) a specific RTP media stream. This RTCP XR Block Type is specified and registered with IANA by ETSI. The RTCP packet type specified by this document is used to distribute a common target playout point to which all the distributed receivers, sharing a media experience, can synchronize.

Typical use cases in which IDMS is usefull are social TV, shared service control (i.e. applications where two or more geographically separated users are watching a media stream together), distance learning, networked video walls, networked loudspeakers, etc.

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 January 17, 2013.

#### Copyright Notice

Copyright (c) 2012 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 . . . . .	4
2. Rationale . . . . .	4
2.1. Applicability of RTCP to IDMS . . . . .	4
2.2. Applicability of SDP to IDMS . . . . .	5
2.3. This document and ETSI TISPAN . . . . .	5
3. Terminology . . . . .	5
4. Overview of IDMS operation . . . . .	5
5. Inter-Destination Media Synchronization use cases . . . . .	7
6. Architecture for Inter-Destination Media Synchronization . . . . .	8
6.1. Media Synchronization Application Server (MSAS) . . . . .	9
6.2. Synchronization Client (SC) . . . . .	9
6.3. Communication between MSAS and SCs . . . . .	9
7. RTCP XR Block Type for IDMS . . . . .	9
8. RTCP Packet Type for IDMS (IDMS Settings) . . . . .	12
9. Timing and NTP Considerations . . . . .	14
10. SDP Parameter for RTCP XR IDMS Block Type . . . . .	15
11. SDP Parameter for RTCP IDMS Packet Type . . . . .	16
12. Compatibility with ETSI TISPAN . . . . .	17
13. SDP rules . . . . .	17
13.1. Offer/Answer rules . . . . .	17
13.2. Declarative cases . . . . .	19
14. On the use of presentation timestamps . . . . .	19
15. Security Considerations . . . . .	19
16. IANA Considerations . . . . .	20
17. Contributors . . . . .	21
18. References . . . . .	21
18.1. Normative References . . . . .	21
18.2. Informative References . . . . .	22
Authors' Addresses . . . . .	22



## 1. Introduction

Inter-Destination Media Synchronization (IDMS) refers to the playout of media streams at two or more geographically distributed locations in a time synchronized manner. It can be applied to both unicast and multicast media streams and can be applied to any type and/or combination of streaming media, such as audio, video and text (subtitles). [Ishibashi2006] and [Boronat2009] provide an overview of technologies and algorithms for IDMS.

IDMS requires the exchange of information on media receipt and playout times among participants in an IDMS session. It may also require signaling for the initiation and maintenance of IDMS sessions and groups of receivers.

The presented RTCP specification for IDMS is independent of the used synchronization algorithm, which is out-of-scope of this document.

## 2. Rationale

### 2.1. Applicability of RTCP to IDMS

Currently, a large share of real-time applications make use of RTP and RTCP [RFC3550]. RTP provides end-to-end network transport functions suitable for applications requiring real-time data transport, such as audio, video or data, over multicast or unicast network services. The timestamps, sequence numbers, and payload (content) type identification mechanisms provided by RTP packets are very useful for reconstructing the original media timing, and for reordering and detecting packet loss at the client side.

The data transport is augmented by a control protocol (RTCP) to allow monitoring of the data delivery in a manner that is scalable to large groups, and to provide minimal control and identification functionality. RTP receivers and senders provide reception quality feedback by sending out RTCP Receiver Report (RR) and Sender Report (SR) packets [RFC3550], respectively, which may be augmented by eXtended Reports (XR) [RFC3611]. Both RTP and RTCP are intended to be tailored through modifications in order to include profile-specific information required by particular applications, and the guidelines on doing so are specified in [RFC5868].

IDMS involves the collection, summarizing and distribution of RTP packet arrival and playout times. As information on RTP packet arrival times and playout times can be considered reception quality feedback information, RTCP is well suited for carrying out IDMS, which may facilitate the implementation and deployment in typical

multimedia applications.

## 2.2. Applicability of SDP to IDMS

RTCP XR [RFC3611] defines the Extended Report (XR) packet type for the RTP Control Protocol (RTCP), and defines how the use of XR packets can be signaled by an application using the Session Description Protocol (SDP) [RFC4566].

SDP signaling is used to set up and maintain a synchronization group between Synchronization Clients (SCs). This document describes two SDP parameters for doing this, one for the RTCP XR block type and one for the new RTCP packet type.

## 2.3. This document and ETSI TISPAN

ETSI TISPAN [TS183063] has specified architecture and protocol for IDMS using RTCP XR exchange and SDP signaling. For more information on how this document relates to [TS183063], see Section 12.

This document mandates the use of an ETSI specified RTCP XR block for the purpose of collecting RTP packet arrival and playout times. For completeness, that XR block is contained in this document in section 7.

## 3. Terminology

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] and indicate requirement levels for compliant implementations.

## 4. Overview of IDMS operation

This section provides a brief example of how the RTCP functionality is used for achieving IDMS. The section is tutorial in nature and does not contain any normative statements.

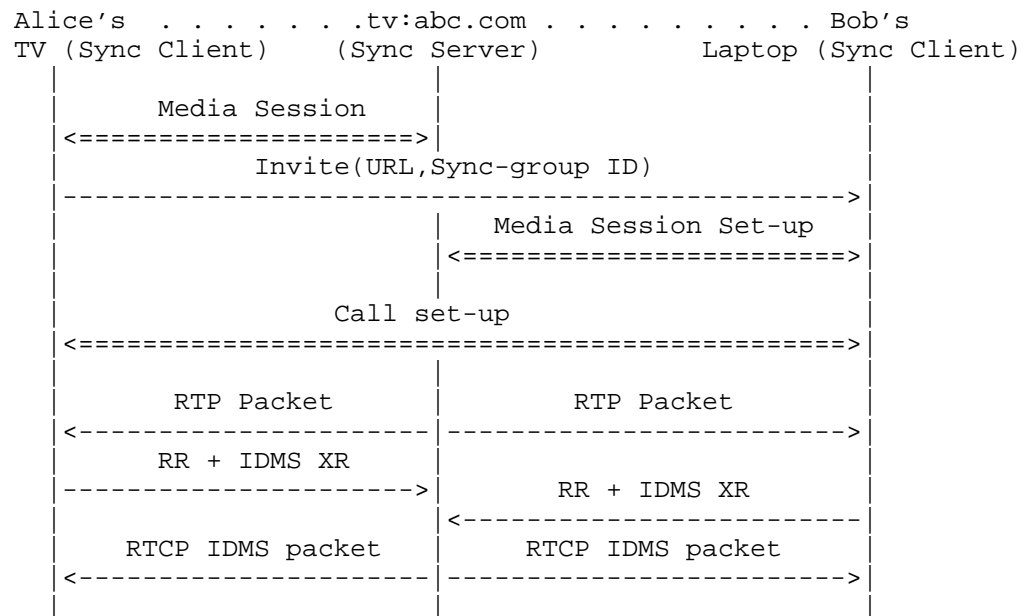


Figure 1: Example of a typical IDMS session

Alice is watching TV in her living room. At some point she sees that a football game of Bob's favorite team is on. She sends him an invite to watch the program together. Embedded in the invitation is the link to the media server and a unique sync-group identifier.

Bob, who is also at home, receives the invite on his laptop. He accepts Alice's invitation and the RTP client on his laptop sets up a session to the media server. A VoIP connection to Alice's TV is also set up, so that Alice and Bob can talk while watching the game together.

As is common with RTP, both the RTP client in Alice's TV as well as the one in Bob's laptop send periodic RTCP Receiver Reports (RR) to the media server. However, in order to make sure Alice and Bob see the events in the football game at (approximately) the same time, their clients also periodically send an IDMS XR block to the sync server function of the media server. Included in the XR blocks are timestamps on when both Alice and Bob received (and, optionally, when they played out) a particular RTP packet.

The sync server function in the media server calculates a reference client from the received IDMS XR blocks (e.g. by selecting whichever client received the packet the latest as the reference client). It then sends an RTCP IDMS packet containing the playout information of

this reference client to the sync clients of both Alice and Bob.

In this case Bob's connection has the longest delay and the reference client therefore includes a delay similar to the one experienced by Bob. Upon reception of this information, Alice's RTP client can choose what to do with this information. In this case it decreases its playout rate temporarily until the playout time matches with the reference client playout (and thus matches Bob's playout). Another option for Alice's TV would be to simply pause playback until it catches up. The exact implementation of the synchronization algorithm is up to the client.

Upon reception of the reference client RTCP IDMS packet, Bob's client does not have to do anything since it is already synchronized to the reference client (since it is based on Bob's delay). Note that other synchronization algorithms may introduce even more delay than the one experienced by the most delayed client, e.g. to account for delay variations, for new clients joining an existing synchronization group, etc.

For this functionality to work correctly, it is necessary that the wall clocks of the receivers are synchronized with each other. Alice and Bob both report when they receive, and optionally when they play out, certain RTP packets. In order to correlate their reports to each other, it is necessary that their wallclocks are synchronized.

## 5. Inter-Destination Media Synchronization use cases

There are a large number of use cases in which IDMS might be useful. This section will highlight some of them. It should be noted that this section is in no way meant to be exhaustive.

A first usage scenario for IDMS is Social TV. Social TV is the combination of media content consumption by two or more users at different devices and locations combined with the real-time communication between those users. An example of Social TV is when two or more users are watching the same television broadcast at different devices and locations, while communicating with each other using text, audio and/or video. A skew in their media playout processes can have adverse effects on their experience. A well-known use case here is one friend experiencing a goal in a football match well before or after other friend(s).

Another potential use case for IDMS is a networked video wall. A video wall consists of multiple computer monitors, video projectors, or television sets tiled together contiguously or overlapped in order to form one large screen. Each of the screens reproduces a portion

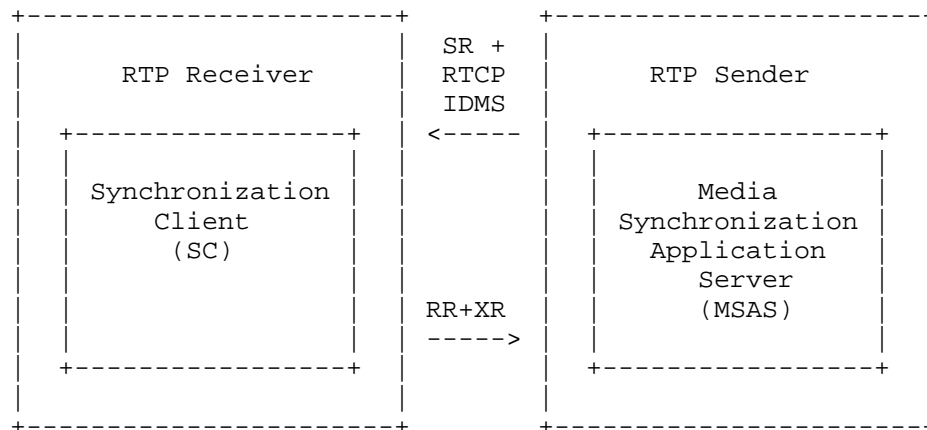
of the larger picture. In some implementations, each screen may be individually connected to the network and receive its portion of the overall image from a network-connected video server or video scaler. Screens are refreshed at 60 hertz (every 16-2/3 milliseconds) or potentially faster. If the refresh is not synchronized, the effect of multiple screens acting as one is broken.

A third usage scenario is that of the networked loudspeakers, in which two or more speakers are connected to the network individually. Such situations can for example be found in large conference rooms, legislative chambers, classrooms (especially those supporting distance learning) and other large-scale environments such as stadiums. Since humans are more susceptible to differences in audio delay, this use case needs even more accuracy than the video wall use case. Depending on the exact application, the need for accuracy can then be in the range of microseconds.

## 6. Architecture for Inter-Destination Media Synchronization

The architecture for IDMS, which is based on a sync-maestro architecture [Boronat2009], is sketched below. The Synchronization Client (SC) and Media Synchronization Application Server (MSAS) entities are shown as additional functionality for the RTP receiver and sender respectively.

It should be noted that a master/slave type of architecture is also supported by having one of the SC devices also act as an MSAS. In this case the MSAS functionality is thus embedded in an RTP receiver instead of an RTP sender.



### 6.1. Media Synchronization Application Server (MSAS)

An MSAS collects RTP packet arrival times and playout times from one or more SC(s) in a synchronization group. The MSAS summarizes and distributes this information to the SCs in the synchronization group as synchronization settings, e.g. by determining the SC with the most lagged playout and using its reported RTP packet arrival time and playout time as a summary.

### 6.2. Synchronization Client (SC)

An SC reports on RTP packet arrival times and playout times of a media stream. It can receive summaries of such information, and use that to adjust its playout buffer.

### 6.3. Communication between MSAS and SCs

Two different message types are used for the communication between MSAS and SCs. For the SC->MSAS message containing the playout information of a particular client, an RTCP XR Block Type is used (see Section 6). For the MSAS->SC message containing the synchronization settings instructions, a new RTCP Packet Type is defined (see Section 8).

## 7. RTCP XR Block Type for IDMS

For reporting IDMS information, SCs SHALL use the ETSI specified RTCP XR Block Type 12. For completeness, that RTCP XR Block Type is repeated here fully.

The definition of the XR block type is based on [RFC3611]. The RTCP XR is used to provide feedback information on receipt times and presentation times of RTP packets to e.g. a Sender [RFC3611], a Feedback Target [RFC5760] or a Third Party Monitor [RFC3611].

In most cases, a single RTP receiver will only be part of single IDMS session, i.e. it will report on receipt and presentation times of RTP packets from a single RTP stream in a certain synchronization group. In some cases however, an RTP receiver may be a member of multiple synchronization groups for the same RTP stream, e.g. watching a single television program simultaneously with different groups. In even further cases, a receiver may wish to synchronize different RTP streams at the same time, either as part of the same synchronization group or as part of multiple synchronization groups. These are all valid scenario's for IDMS, and will require multiple reports by an SC.

SCs SHOULD report on a recently received RTP packets. This document does not define new rules on when to sent RTCP reports, but uses the existing rules specified in [RFC3550] for sending RTCP reports. When the RTCP reporting timer allows an SC to send an IDMS report, the SC SHOULD report on the latest RTP packet received or played out (depending on whether the SC reports on presentation timestamps or receipt timestamps). For more details on which packet to report on, see below under 'Packet Received RTP timestamp'.

```

      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
+-----+-----+-----+-----+-----+-----+-----+-----+
|V=2|P| Resrv |   PT=XR=207   |           length           |
+-----+-----+-----+-----+-----+-----+-----+-----+
|                               SSRC of packet sender       |
+=====+=====+=====+=====+=====+=====+=====+=====+
|   BT=12   | SPST |Resrv|P|           block length=7       |
+-----+-----+-----+-----+-----+-----+-----+-----+
|   PT   |           Resrv           |
+-----+-----+-----+-----+-----+-----+-----+-----+
|                               Media Stream Correlation Identifier
+-----+-----+-----+-----+-----+-----+-----+-----+
|                               SSRC of media source        |
+-----+-----+-----+-----+-----+-----+-----+-----+
|   Packet Received NTP timestamp, most significant word   |
+-----+-----+-----+-----+-----+-----+-----+-----+
|   Packet Received NTP timestamp, least significant word  |
+-----+-----+-----+-----+-----+-----+-----+-----+
|   Packet Received RTP timestamp                          |
+-----+-----+-----+-----+-----+-----+-----+-----+
|   Packet Presented NTP timestamp                         |
+-----+-----+-----+-----+-----+-----+-----+-----+

```

The first 64 bits form the header of the RTCP XR, as defined in [RFC3611]. The SSRC of packet sender identifies the sender of the specific RTCP packet.

The IDMS report block consists of 8 32-bit words, with the following fields:

Block Type (BT): 8 bits. It identifies the block format. Its value SHALL be set to 12.

Synchronization Packet Sender Type (SPST): 4 bits. This field identifies the role of the packet sender for this specific eXtended Report. It can have the following values:

SPST=0 Reserved For future use.

SPST=1 The packet sender is an SC. It uses this XR to report synchronization status information. Timestamps relate to the SC input.

SPST=2 This setting is reserved in order to preserve compatibility with ETSI TISPA [TS183063]. See Section 12 for more information.

SPST=3-15 Reserved For future use.

Reserved bits (Resrv): 3 bits. These bits are reserved for future definition. In the absence of such a definition, the bits in this field MUST be set to zero and MUST be ignored by the receiver.

Packet Presented NTP timestamp flag (P): 1 bit. Bit set to 1 if the Packet Presented NTP timestamp field contains a value, 0 if it is empty. If this flag is set to zero, then the Packet Presented NTP timestamp SHALL be ignored.

Block Length: 16 bits. This field indicates the length of the block in 32 bit words minus one and SHALL be set to 7, as this RTCP Block Type has a fixed length.

Payload Type (PT): 7 bits. This field identifies the format of the media payload, according to [RFC3551]. This is the payload type of the RTP packet reported upon. The media payload is associated with an RTP timestamp clock rate. This clock rate provides the time base for the RTP timestamp counter. This clock rate is necessary for the MSAS to relate reports from different SCs on different RTP timestamp values.

Reserved bits (Resrv): 25 bits. These bits are reserved for future use and SHALL be set to 0.

Media Stream Correlation Identifier: 32 bits. This identifier is used to correlate synchronized media streams. The value 0 (all bits are set "0") indicates that this field is empty. The value  $2^{32}-1$  (all bits are set "1") is reserved for future use. If the RTCP Packet Sender is an SC (SPST=1) or an MSAS (SPST=2), then the Media Stream Correlation Identifier field contains the Synchronization Group Identifier (SyncGroupId) to which the report applies.

SSRC: 32 bits. The SSRC of the media source SHALL be set to the value of the SSRC identifier carried in the RTP header [RFC3550] of the RTP packet to which the XR relates.

Packet Received NTP timestamp: 64 bits. This timestamp reflects the



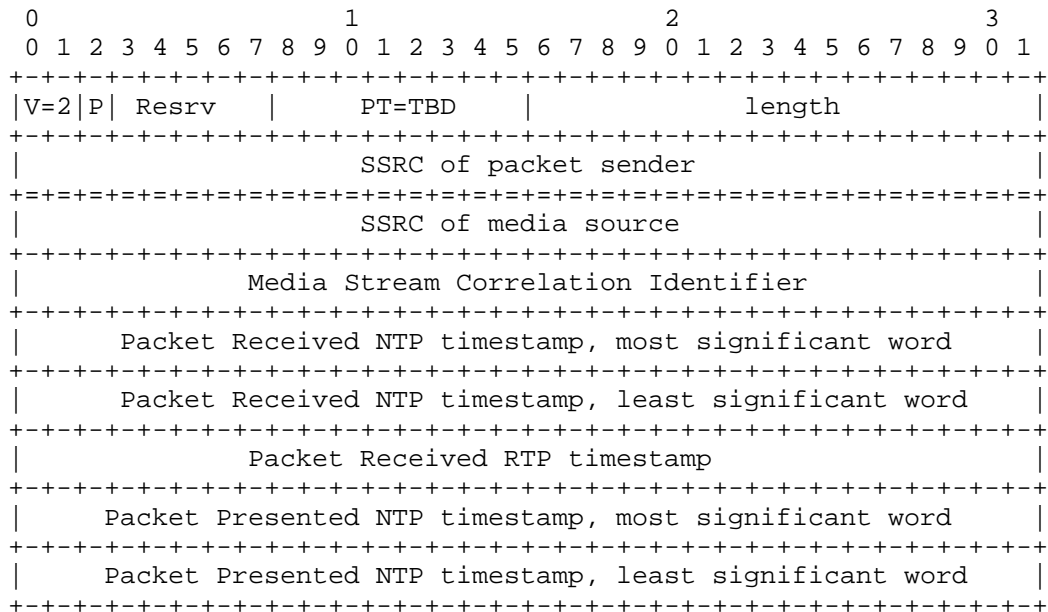
wall clock time at the moment of arrival of the first octet of the RTP packet to which the XR relates. It is formatted based on the NTP timestamp format as specified in [RFC5905]. See Section 9 for more information on how this field is used.

Packet Received RTP timestamp: 32 bits. This timestamp has the value of the RTP timestamp carried in the RTP header [RFC3550] of the RTP packet to which the XR relates. Several consecutive RTP packets will have equal timestamps if they are (logically) generated at once, e.g., belong to the same video frame. It may well be the case that one receiver reports on the first RTP packet having a certain RTP timestamp and a second receiver reports on the last RTP packet having that same RTP timestamp. This would lead to an error in the synchronization algorithm due to the faulty interpretation of considering both reports to be on the same RTP packet. When reporting on an RTP packet which is one of several consecutive RTP packets having equal timestamps, an SC SHOULD report on the RTP packet it received with the lowest sequence number. Note that with 'lowest sequence number' here is meant the first in the sequence of RTP packets just received, not from an earlier time before the last wrap-around of RTP timestamps (unless this wrap-around occurs during the sequence with equal RTP timestamps).

Packet Presented NTP timestamp: 32 bits. This timestamp reflects the wall clock time at the moment the rendered frame contained in the first byte of the associated RTP packet is presented to the user. It is based on the time format used by NTP and consists of the least significant 16 bits of the NTP seconds part and the most significant 16 bits of the NTP fractional second part. If this field is empty, then it SHALL be set to 0 and the Packet Presented NTP timestamp flag (P) SHALL be set to 0. Presented here means the moment the data is played out to the user of the system, i.e. sound played out through speakers, video images being displayed on some display, etc. The accuracy resulting from the synchronization algorithm will only be as good as the accuracy with which the receivers can determine the delay between receiving packets and presenting them to the end-user.

#### 8. RTCP Packet Type for IDMS (IDMS Settings)

This section specifies the RTCP Packet Type for indicating synchronization settings instructions to the receivers of the RTP media stream. Its definition is based on [RFC3550].



The first 64 bits form the header of the RTCP Packet Type, as defined in [RFC3550]. The SSRC of packet sender identifies the sender of the specific RTCP packet.

The RTCP IDMS packet consists of 7 32-bit words, with the following fields:

**SSRC:** 32 bits. The SSRC of the media source SHALL be set to the value of the SSRC identifier of the media source carried in the RTP header [RFC3550] of the RTP packet to which the RTCP IDMS packet relates.

**Media Stream Correlation Identifier:** 32 bits. This identifier is used to correlate synchronized media streams. The value 0 (all bits are set "0") indicates that this field is empty. The value  $2^{32}-1$  (all bits are set "1") is reserved for future use. The Media Stream Correlation Identifier contains the SyncGroupId of the group to which this packet is sent.

**Packet Received NTP timestamp:** 64 bits. This timestamp reflects the wall clock time at the reference client at the moment it received the first octet of the RTP packet to which this packet relates. It can be used by the synchronization algorithm on the receiving SC to adjust its playout timing in order to achieve synchronization, e.g. to set the required playout delay. The timestamp is formatted based on the NTP timestamp format as specified in [RFC5905]. See Section 9

for more information on how this field is used. Because RTP timestamps do wrap around, the sender of this packet SHOULD use recent values, i.e. choose NTP timestamps that reflect current time and not too far in the future or in the past.

Packet Received RTP timestamp: 32 bits. This timestamp has the value of the RTP timestamp carried in the RTP header [RFC3550] of the RTP packet to which the XR relates. This SHOULD relate to the first arriving RTP packet containing this particular RTP timestamp, in case multiple RTP packets contain the same RTP timestamp.

Packet Presented NTP timestamp: 64 bits. This timestamp reflects the wall clock time at the reference client at the moment it presented the rendered frame contained in the first octet of the associated RTP packet to the user. The timestamp is formatted based on the NTP timestamp format as specified in [RFC5905]. If this field is empty, then it SHALL be set to 0. This field MAY be left empty if none or only one of the receivers reported on presentation timestamps. Presented here means the moment the data is played out to the user of the system.

In some use cases (e.g. phased array transducers), the level of control an MSAS might need to have over the exact moment of playout is so precise that a 32bit Presented Timestamp will not suffice. For this reason, this RTCP Packet Type for IDMS includes a 64bit Presented Timestamp field. Since an MSAS will in practice always add some extra delay to the delay reported by the most lagged receiver (to account for packet jitter), it suffices for the IDMS XR Block Type with which the SCs report on their playout to have a 32bit Presented Timestamp field.

## 9. Timing and NTP Considerations

To achieve IDMS, the different receivers involved need synchronized (wall) clocks as a common timeline for synchronization. This synchronized clock is used for reporting the Packet Received NTP Timestamps and the Packet Presented NTP Timestamp, and for interpretation of these fields in received IDMS reports. Depending on the synchronization accuracy required, different clock synchronization methods can be used. For social TV, synchronization accuracy should be achieved on the order of hundreds of milliseconds. In that case, correct use of NTP on receivers will in most situations achieve the required accuracy. As a guideline, to deal with clock drift of receivers, receivers should synchronize their clocks at the beginning of a synchronized session. In case of high required accuracy, the synchronized clocks of different receivers should not drift beyond the accuracy required for the synchronization mechanism.

In practice, this can mean that receivers need to synchronize their clocks repeatedly during a synchronization session.

Because of the stringent synchronization requirements for achieving good audio in some use cases, a high accuracy will be needed. In this case, use of the global NTP system may not be sufficient. For improved accuracy, a local NTP server could be set up, or some other more accurate clock synchronization mechanism can be used, such as GPS time or the Precision Time Protocol [IEEE-1588].

[I-D.draft-williams-avtcore-clksrc] defines a set of SDP parameters for signaling the clock synchronization source or sources available to and used by the individual receivers. SCs MAY use this draft to indicate their clock synchronization source or sourced in use and available. Using these parameters, an SC can indicate which synchronization source is being used at the moment, the last time the SC synchronized with this source and the synchronization frequency. An SC can also indicate any other synchronization sources available to it. This allows multiple SCs in an IDMS session to use the same or a similar clock source for their session.

Applications performing IDMS may or may not be able to choose a synchronization method for the system clock, because this may be a system-wide setting which the application cannot change. How applications deal with this is up to the implementation. The application might control the system clock, or it might use a separate application clock or even a separate IDMS session clock. It might also report on the system clock and the synchronization method used, without being able to change it.

[I-D.draft-gross-leap-second] presents some guidelines on how RTP senders and receivers should deal with leap seconds. When relying on NTP for clock synchronization, IDMS is particularly sensitive to leap second induced timing discrepancies. It is RECOMMENDED to take the guideline specified in [I-D.draft-gross-leap-second] into account when implementing IDMS.

#### 10. SDP Parameter for RTCP XR IDMS Block Type

The SDP parameter sync-group is used to signal the use of the RTCP XR block for IDMS, as specified by ETSI, included here for completeness. It is also used to carry an identifier of the synchronization group to which clients belong or will belong. This SDP parameter extends rtcp-xr-attr as follows, using Augmented Backus-Naur Form [RFC5234].

```
rtcp-xr-attr = "a=" "rtcp-xr" ":" [xr-format *(SP xr-format)] CRLF
```

; Original definition from [RFC3611], section 5.1

xr-format =/ grp-sync ; Extending xr-format for Inter-Destination Media Synchronization

grp-sync = "grp-sync" [",sync-group=" SyncGroupId]

SyncGroupId = 1\*10DIGIT ; Numerical value from 0 through 4294967294

DIGIT = %x30-39

SyncGroupId is a 32-bit unsigned integer represented in decimal. SyncGroupId identifies a group of SCs for IDMS. It maps on the Media Stream Correlation Identifier as described in Section 7 and Section 8. The value SyncGroupId=0 represents an empty SyncGroupId. The value 4294967294 ( $2^{32}-1$ ) is reserved for future use.

The following is an example of the SDP attribute for IDMS

a=rtcp-xr:grp-sync,sync-group=42

#### 11. SDP Parameter for RTCP IDMS Packet Type

The SDP parameter rtcp-idms is used to signal the use of the RTCP IDMS Packet Type for IDMS. It is also used to carry an identifier of the synchronization group to which clients belong or will belong. The SDP parameter is used as a media-level attribute during session setup. This means that in case of multiple related streams, IDMS is performed on one of them. The other streams will be synchronized to this first stream using existing inter-stream synchronization (i.e. lip-sync) solutions, i.e. using Sender Reports based on a common clock source. Basic guidelines for choosing the media stream for IDMS is to choose audio above video, as humans are more sensitive to degradation in audio quality than in video quality. When using mutli-description or multi-view codecs, the IDMS control should be performed on the base layer.

This SDP parameter is defined as follows, using Augmented Backus-Naur Form [RFC5234].

rtcp-idms = "a=" "rtcp-idms" ":" [sync-grp] CRLF

sync-grp = "sync-group=" SyncGroupId

SyncGroupId = 1\*10DIGIT ; Numerical value from 0 through 4294967294

DIGIT = %x30-39

SyncGroupId is a 32-bit unsigned integer and represented in decimal. SyncGroupId identifies a group of SCs for IDMS. The value SyncGroupId=0 represents an empty SyncGroupId. The value 4294967294 ( $2^{32}-1$ ) is reserved for future use.

The following is an example of the SDP attribute for IDMS.

```
a=rtcp-idms:sync-group=42
```

## 12. Compatibility with ETSI TISPAN

As described in Section 2.3, ETSI TISPAN has described its mechanism for IDMS in [TS183063]. One of the main differences between the TISPAN document and this document is the fact that the TISPAN solution uses an RTCP XR block for both the SC->MSAS message and the MSAS->SC message (by selecting SPST-type 2), while this document specifies a new RTCP Packet Type for the MSAS->SC message. The message from MSAS to SC is not in any way a report on how a receiver sees a session, and therefore a separate RTCP packet type is more appropriate than the XR block solution chosen in ETSI TISPAN. To achieve compatibility, MSAS implementations SHOULD implement both the TISPAN RTCP block and the new RTCP IDMS Settings packet for MSAS->SC messages. SCs MAY implement support for both types of messages. For the MSAS->SC signaling, it is recommended to use the RTCP IDMS Settings packet defined in this document. The TISPAN RTCP XR block with SPST=2 MAY be used for purposes of compatibility with the TISPAN solution, but MUST NOT be used if all nodes involved support the new RTCP IDMS Settings packet.

## 13. SDP rules

### 13.1. Offer/Answer rules

The SDP usage for IDMS follows the rules defined in RFC3611 in section 5 on SDP signalling, with the exception of what is stated here. The IDMS usage of RTCP is a (loosely coupled) collaborative parameter, in the sense that receivers sent their status information and in response (asynchronously) the MSAS sends synchronization instructions. Both the sync-group parameter (defined by ETSI TISPAN) and the rtcp-idms parameter (defined in this document) thus indicate the ability to sent and the ability to receive indicated RTCP messages. This section defines how these SDP parameters should be used.

Most of the times, the IDMS SDP parameters will be used in the offer/answer context. Receivers will indicate in their SDP which RTCP

messages they support.

For a unicast situation, three situations are possible in offer/answer context:

- If a receiver indicates at least the rtcp-idms SDP parameter, the MSAS SHOULD reply with only the rtcp-idms parameter and use only the RTCP IDMS Settings packet for MSAS->SC communication
- If a receiver indicates only the sync-group SDP parameter, and the MSAS also supports this, it SHOULD reply with only the sync-group parameter and use only the RTCP XR block with SPST=2 for MSAS->SC communication
- If a receiver indicates only the sync-group SDP parameter, and the MSAS does not support this, the media sender MUST ignore the parameter. This receiver will not become part of the synchronization session

Note that it is possible that for a certain synchronization group, that the MSAS sends RTCP IDMS packets to one receiver and RTCP XR IDMS blocks with SPST=2 to another receiver.

In a multicast situation using the offer/answer context, it will work a bit differently. The negotiation is the same as in the unicast situation. But, the MSAS will multicast all RTCP messages to all receivers. So:

- If all receivers support the RTCP IDMS Settings packet, the MSAS SHOULD only sent the RTCP IDMS Settings packet for MSAS->SC messages
- If not all receivers support the RTCP IDMS Settings packet, but all receivers support the TISPAN solution, the MSAS SHOULD only sent the RTCP XR block with SPST=2 for MSAS->SC messages.
- If some receivers support only the RTCP IDMS Settings packet and other receivers support only the TISPAN solution, the MSAS SHOULD sent both the RTCP IDMS Settings packet and the RTCP XR block with SPST=2 for MSAS->SC messages. This is less efficient, since the information sent is duplicated, but this is the only way to include all receivers in a synchronization session in this scenario.

In certain multicast situations, there is no offer/answer context, but only a declarative modus. In that case, the MSAS SHOULD use only the RTCP IDMS packet type and thus use only the SDP parameter rtcp-idms. Receivers that do not support the RTCP IDMS packet will just

ignore both the SDP parameter and the RTCP IDMS packets, and will thus not join the synchronization session. For compatability with the TISPAN solution, the MSAS MAY choose to use the RTCP XR IDMS block type instead, using the SDP parameter sync-group. The media sender SHOULD NOT use both parameters at the same time in this case of no offer/answer context.

### 13.2. Declarative cases

In certain multicast situations, there is no offer/answer context, but only a declarative modus. In that case, the MSAS SHOULD use only the RTCP IDMS packet type and thus use only the SDP parameter rtcp-idms. Receivers that do not support the RTCP IDMS packet will just ignore both the SDP parameter and the RTCP IDMS packets, and will thus not join the synchronization session. For compatability with the TISPAN solution, the MSAS MAY choose to use the RTCP XR IDMS block type instead, using the SDP parameter sync-group. The media sender SHOULD NOT use both parameters at the same time in this case of no offer/answer context.

## 14. On the use of presentation timestamps

A receiver can report on different timing events, i.e. on packet arrival times and on playout times. A receiver SHALL report on arrival times and a receiver MAY report on playout times. RTP packet arrival times are relatively easy to report on. Normally, the processing and playout of the same media stream by different receivers will take roughly the same amount of time. Synchronizing on packet arrival times, may lead to some accuracy loss, but it will be adequate for many applications, such as social TV.

Also, if the receivers are in some way controlled, e.g. having the same buffer settings and decoding times, high accuracy can be achieved. However, if all receivers in a synchronization session have the ability to report on, and thus synchronize on, actual playout times, or packet presentation times, this may be more accurate. It is up to applications and implementations of this RTCP extension whether to implement and use this.

## 15. Security Considerations

The security considerations described in [RFC3611] apply to this document as well.

The specified RTCP XR Block Type in this document is used to collect, summarize and distribute information on packet reception- and



playout-times of streaming media. The information may be used to orchestrate the media playout at multiple devices.

Errors in the information, either accidental or malicious, may lead to undesired behavior. For example, if one device erroneously reports a two-hour delayed playout, then another device in the same synchronization group could decide to delay its playout by two hours as well, in order to keep its playout synchronized. A user would likely interpret this two hour delay as a malfunctioning service.

Therefore, the application logic of both Synchronization Clients and Media Synchronization Application Servers should check for inconsistent information. Differences in playout time exceeding configured limits (e.g. more than ten seconds) could be an indication of such inconsistent information.

No new mechanisms are introduced in this document to ensure confidentiality. Encryption procedures, such as those being suggested for a Secure RTP (SRTP) at the time that this document was written, can be used when confidentiality is a concern to end hosts.

## 16. IANA Considerations

This document defines a new RTCP packet type called IDMS Settings packet in the IANA registry of RTP parameters, part of RTCP Control Packet types (PT), based on the specification in Section 11.

Further, this document defines a new SDP parameter "rtcp-idms" within the existing IANA registry of SDP Parameters, part of the "att-field (media level only)".

The SDP attribute "rtcp-idms" defined by this document is registered with the IANA registry of SDP Parameters as follows:

SDP Attribute ("att-field"):

Attribute name: rtcp-idms

Long form: RTCP report block for IDMS

Type of name: att-field

Type of attribute: media level

Subject to charset: no

Purpose: see sections 7 and 10 of this document

Reference: this document

Values: see this document

## 17. Contributors

The following people have participated as co-authors or provided substantial contributions to this document: Omar Niamut, Fabian Walraven, Ishan Vaishnavi, Rafael Mekuria and Rob Koenen.

## 18. References

### 18.1. Normative References

- [I-D.draft-williams-avtcore-clksrc]  
Williams, A., van Brandenburg, R., Stokking, H., and K. Gross, "RTP Clock Source Signalling, draft-williams-avtcore-clksrc-00", March 2012.
- [RFC2119] Bradner, S., "Key Words for use in RFCs to Indicate Requirement Levels, RFC 2119", March 1997.
- [RFC3550] Schulzrinne, H., Casner, S., Frederick, R., and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications, RFC3550", July 2003.
- [RFC3551] Schulzrinne, H. and S. Casner, "RTP Profile for Audio and Video conferences with Minimal Control, RFC3551", July 2003.
- [RFC3611] Friedman, T., Ed., Caceres, R., Ed., and A. Clark, Ed., "RTP Control Protocol Extended Reports (RTCP XR), RFC3611", November 2003.
- [RFC4566] Handley, M., Jacobson, V., and C. Perkins, "SDP: Session Description Protocol, RFC4566", July 2006.
- [RFC5234] Crocker, D., Ed. and P. Overell, "Augmented BNF for Syntax Specifications, RFC5234", January 2008.

- [RFC5760] Ott, J., Chesterfield, J., and E. Schooler, "RTP Control Protocol (RTCP) Extensions for Single-Source Multicast Sessions with Unicast Feedback, RFC5760", February 2010.
- [RFC5905] Mills, D., Martin, J., Ed., Burbank, J., and W. Kasch, "Network Time Protocol Version 4: Protocol and Algorithms Specifications, RFC5905", February 2010.
- [TS183063] "IMS-based IPTV stage 3 specification, TS 183 063 v3.4.1", June 2010.

## 18.2. Informative References

- [Boronat2009] Boronat, F., Lloret, J., and M. Garcia, "Multimedia group and inter-stream synchronization techniques: a comparative study, Elsevier Information Systems 34 (2009), pp. 108-131".
- [I-D.draft-gross-leap-second] Gross, K. and R. Brandenburg, van, "RTP and Leap Seconds, draft-gross-leap-seconds-01".
- [IEEE-1588] "1588-2008 - IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems", 2008.
- [Ishibashi2006] Ishibashi, Y., Nagasaka, M., and N. Fujiyoshi, "Subjective Assessment of Fairness among users in multipoint communications, Proceedings of the 2006 ACM SIGCHI international conference on Advances in computer entertainment technology, 2006".
- [RFC5868] Ott, J. and C. Perkins, "Guidelines for Extending the RTP Control Protocol (RTCP), RFC5968", September 2010.

Authors' Addresses

Ray van Brandenburg  
TNO  
Brassersplein 2  
Delft 2612CT  
the Netherlands

Phone: +31-88-866-7000  
Email: ray.vanbrandenburg@tno.nl

Hans Stokking  
TNO  
Brassersplein 2  
Delft 2612CT  
the Netherlands

Phone: +31-88-866-7000  
Email: hans.stokking@tno.nl

M. Oskar van Deventer  
TNO  
Brassersplein 2  
Delft 2612CT  
the Netherlands

Phone: +31-88-866-7000  
Email: oskar.vandeventer@tno.nl

Fernando Boronat  
Universitat Politecnica de Valencia  
IGIC Institute, Universitat Politecnica de Valencia-Campus de Gandia (UPV), C  
/ Paraninfo, 1, Grao de Gandia, C/ Paraninfo, 1, Grao de Gandia  
Valencia 46730  
Spain

Phone: +34 962 849 341  
Email: fboronat@dcom.upv.es

Mario Montagud  
Universitat Politecnica de Valencia  
IGIC Institute, Universitat Politecnica de Valencia-Campus de Gandia (UPV), C  
/ Paraninfo, 1, Grao de Gandia, C/ Paraninfo, 1, Grao de Gandia  
Valencia 46730  
Spain

Phone: +34 962 849 341  
Email: mamontor@posgrado.upv.es

Kevin Gross  
AVA Networks

Phone: +1-303-447-0517  
Email: Kevin.Gross@AVAnw.com



AVTCore  
Internet-Draft  
Updates: 3550 (if approved)  
Intended status: Standards Track  
Expires: December 23, 2012

K. Gross  
AVA Networks  
R. van Brandenburg  
TNO  
June 21, 2012

RTP and Leap Seconds  
draft-ietf-avtcore-leap-second-00

Abstract

This document discusses issues that arise when RTP sessions span (UTC) leap seconds. It updates RFC 3550 to describe how RTP senders and receivers should behave in the presence of leap seconds.

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 December 23, 2012.

Copyright Notice

Copyright (c) 2012 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. Terminology . . . . .	3
3. Leap seconds . . . . .	3
3.1. UTC behavior during leap second . . . . .	3
3.2. NTP behavior during leap second . . . . .	4
3.3. POSIX behavior during leap second . . . . .	4
4. Recommendations . . . . .	4
4.1. RTP Sender Reports and Receiver Reports . . . . .	5
4.2. RTP Packet Playout . . . . .	5
5. Security Considerations . . . . .	5
6. IANA Considerations . . . . .	5
7. Acknowledgements . . . . .	5
8. Normative References . . . . .	5
Authors' Addresses . . . . .	6



## 1. Introduction

In some applications, RTP streams are referenced to a wallclock time (absolute date and time). This is typically accomplished through use of the NTP timestamp field in the RTCP sender report (SR) to create a mapping between RTP timestamps and the wallclock. When a wallclock reference is used, the playout time for RTP packets is referenced to the wallclock. Smooth and continuous media playout requires a smooth and continuous timebase. The timebase used by the wallclock may include leap seconds which, in many cases, are not rendered smoothly.

This document provides recommendations for smoothly rendering streamed media referenced to common wallclocks which may not have smooth or continuous behavior in the presence of leap seconds.

## 2. Terminology

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 [RFC2119] and indicate requirement levels for compliant implementations.

## 3. Leap seconds

Leap seconds are intended to keep UTC time synchronized with the rotation of the earth. Leap seconds are scheduled by the International Earth Rotation and Reference Systems Service. Leap seconds may be scheduled at the last day of any month but are preferentially scheduled for December and June and secondarily March and September.[TF.460-6] Because earth's rotation is unpredictable, leap seconds are typically not scheduled more than six months in advance. Leap seconds can be scheduled to either add or remove a second from the day. All leap second events thus far have been scheduled in June or December and have all added seconds. This is a situation that is expected but not guaranteed to continue.

NOTE- The ITU is studying a proposal which could eventually eliminate leap seconds from UTC. As of January 2012, this proposal is expected to be decided no earlier than 2015.

### 3.1. UTC behavior during leap second

UTC clocks insert a 61st second at the end of the day when a leap second is scheduled. The leap second is designated "23h 59m 60s". The sequence of the second markers near the UTC leap second transition are:

Day 0, 23h 59m 59s

Day 0, 23h 59m 60s <-- leap second

Day 1, 0h 0m 0s

### 3.2. NTP behavior during leap second

Under NTP [RFC5905] a leap second is inserted at the beginning of the last second of the day. This results in the clock freezing or slowing for one second immediately prior to the last second of the affected day. This results in the last second of the day having a real-time duration of two seconds.

### 3.3. POSIX behavior during leap second

Most POSIX systems insert the leap second at the end of the last second of the day. This results in repetition of the last second. A timestamp within the last second of the day is therefore ambiguous in that it can refer to either of the last two seconds of a day containing a leap second.

## 4. Recommendations

Senders and receivers which are not referenced to a wallclock are not affected by issues associated with leap seconds and no special accommodation is required.

RTP implementation using a wallclock reference is simplified by using a clock with a timescale which does not include leap seconds. IEEE 1588 [IEEE1588-2008], GPS [IS-GPS-200F] and other TAI (International Atomic Time) [CircularT] references do not include leap seconds. NTP time, operating system clocks and other UTC (Coordinated Universal Time) references include leap seconds.

All participants working to a leap-second-bearing reference SHOULD recognize leap seconds and have a working communications channel to receive notification of leap second scheduling. Without prior knowledge of leap second schedule, NTP servers and clients may become offset by exactly one second with respect to their UTC reference. This potential discrepancy begins when a leap second occurs and ends when all participants receive a time update from a server or peer. Depending on the system implementation, the offset can last anywhere from a few seconds to a few days. A long-lived discrepancy can be particularly disruptive to RTP operation.

Because of the ambiguity leap seconds can introduce and the

inconsistent manner in which different systems accommodate leap seconds, generating or using NTP timestamps during the entire last second of a day on which a leap second has been scheduled SHOULD be avoided. Note that the period to be avoided has a real-time duration of two seconds.

#### 4.1. RTP Sender Reports and Receiver Reports

RTP Senders working to a leap-second-bearing reference SHOULD NOT generate sender reports containing an originating NTP timestamp in the vicinity of a leap second. Receivers SHOULD ignore timestamps in any such reports inadvertently generated.

#### 4.2. RTP Packet Payout

Receivers working to a leap-second-bearing reference SHOULD take leap seconds in their reference into account in determining payout time from RTP timestamps for data in RTP packets.

### 5. Security Considerations

It is believed that the recommendations herein introduce no new security considerations beyond those already discussed in [RFC3550].

### 6. IANA Considerations

This document has no actions for IANA."

### 7. Acknowledgements

The authors would like to thank Steve Allen for his valuable comments in helping to improve this document.

### 8. Normative References

[CircularT]

BIPM, "Circular T", May 2012.

[IEEE1588-2008]

IEEE, "IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems", July 2008.

[IS-GPS-200F]

Global Positioning Systems Directorate, "Navstar GPS Space Segment/Navigation User Segment Interfaces", September 2011.

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", March 1997.
- [RFC3550] Schulzrinne, H., Casner, S., Frederick, R., and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications, RFC3550", July 2003.
- [RFC5905] Mills, D., Delaware, U., Martin, J., Ed., Burbank, J., and W. Kasch, "Network Time Protocol Version 4: Protocol and Algorithms Specification", June 2010.
- [TF.460-6] ITU-R, "Recommendation ITU-R TF.460-4 - Standard-frequency and time-signal emissions", February 2002.

#### Authors' Addresses

Kevin Gross  
AVA Networks  
Boulder, CO  
US

Email: kevin.gross@avanw.com

Ray van Brandenburg  
TNO  
Brassersplein 2  
Delft 2612CT  
the Netherlands

Phone: +31-88-866-7000  
Email: ray.vanbrandenburg@tno.nl



AVT Working Group  
Internet-Draft  
Intended status: Standards Track  
Expires: January 10, 2013

D. McGrew  
F. Andreasen  
D. Wing  
Cisco  
K. Fischer  
Siemens Enterprise Communications  
July 9, 2012

Encrypted Key Transport for Secure RTP  
draft-ietf-avtcore-srtp-ekt-00

Abstract

Encrypted Key Transport (EKT) is an extension to Secure Real-time Transport Protocol (SRTP) that provides for the secure transport of SRTP master keys, Rollover Counters, and other information, within SRTP or SRTCP. This facility enables SRTP to work for decentralized conferences with minimal control.

This note defines EKT, and also describes how to use it with SDP Security Descriptions, DTLS-SRTP, and MIKEY. These other key management protocols provide an EKT key to everyone in a session, and EKT coordinates the keys within the session.

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 January 10, 2013.

Copyright Notice

Copyright (c) 2012 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 . . . . .	4
1.1. History . . . . .	5
1.2. Conventions Used In This Document . . . . .	5
2. Encrypted Key Transport . . . . .	6
2.1. EKT Field Formats . . . . .	6
2.2. Packet Processing and State Machine . . . . .	8
2.2.1. Outbound Processing . . . . .	8
2.2.2. Inbound Processing . . . . .	10
2.3. Ciphers . . . . .	11
2.3.1. The Default Cipher . . . . .	12
2.3.2. Other EKT Ciphers . . . . .	13
2.4. Synchronizing Operation . . . . .	13
2.5. Transport . . . . .	14
2.6. Timing and Reliability Consideration . . . . .	14
3. Use of EKT with SDP Security Descriptions . . . . .	16
3.1. SDP Security Descriptions Recap . . . . .	16
3.2. Relationship between EKT and SDP Security Descriptions . .	17
3.3. Overview of Combined EKT and SDP Security Description Operation . . . . .	19
3.4. EKT Extensions to SDP Security Descriptions . . . . .	19
3.4.1. EKT_Cipher . . . . .	19
3.4.2. EKT_Key . . . . .	20
3.4.3. EKT_SPI . . . . .	20
3.5. Offer/Answer Procedures . . . . .	20
3.5.1. Generating the Initial Offer - Unicast Streams . . . .	21
3.5.2. Generating the Initial Answer - Unicast Streams . . .	22
3.5.3. Processing of the Initial Answer - Unicast Streams . .	23
3.6. SRTP-Specific Use Outside Offer/Answer . . . . .	24
3.7. Modifying the Session . . . . .	24
3.8. Backwards Compatibility Considerations . . . . .	25
3.9. Grammar . . . . .	25
4. Use of EKT with DTLS-SRTP Key Transport . . . . .	27
4.1. EKT Extensions to DTLS-SRTP . . . . .	27
4.1.1. Scaling to Large Groups . . . . .	29
4.2. Offer/Answer Considerations . . . . .	30
4.2.1. Generating the Initial Offer . . . . .	30

4.2.2. Generating the Initial Answer . . . . .	31
4.2.3. Processing the Initial Answer . . . . .	31
4.2.4. Modifying the Session . . . . .	32
5. Use of EKT with MIKEY . . . . .	33
5.1. EKT extensions to MIKEY . . . . .	34
5.2. Offer/Answer considerations . . . . .	36
5.2.1. Generating the Initial Offer . . . . .	36
5.2.2. Generating the Initial Answer . . . . .	37
5.2.3. Processing the Initial Answer . . . . .	37
5.2.4. Modifying the Session . . . . .	37
6. Using EKT for interoperability between key management systems . . . . .	39
7. Design Rationale . . . . .	40
7.1. Alternatives . . . . .	41
8. Security Considerations . . . . .	42
9. IANA Considerations . . . . .	44
10. Acknowledgements . . . . .	45
11. References . . . . .	46
11.1. Normative References . . . . .	46
11.2. Informative References . . . . .	47
Appendix A. Using EKT to Optimize Interworking DTLS-SRTP with Security Descriptions . . . . .	48
Authors' Addresses . . . . .	51



## 1. Introduction

RTP is designed to allow decentralized groups with minimal control to establish sessions, such as for multimedia conferences. Unfortunately, Secure RTP (SRTP [RFC3711]) cannot be used in many minimal-control scenarios, because it requires that SSRC values and other data be coordinated among all of the participants in a session. For example, if a participant joins a session that is already in progress, the SRTP rollover counter (ROC) of each SRTP source in the session needs to be provided to that participant.

The inability of SRTP to work in the absence of central control was well understood during the design of that protocol; that omission was considered less important than optimizations such as bandwidth conservation. Additionally, in many situations SRTP is used in conjunction with a signaling system that can provide most of the central control needed by SRTP. However, there are several cases in which conventional signaling systems cannot easily provide all of the coordination required. It is also desirable to eliminate the layer violations that occur when signaling systems coordinate certain SRTP parameters, such as SSRC values and ROCs.

This document defines Encrypted Key Transport (EKT) for SRTP, an extension to SRTP that fits within the SRTP framework and reduces the amount of signaling control that is needed in an SRTP session. EKT securely distributes the SRTP master key and other information for each SRTP source, using SRTCP or SRTP to transport that information. With this method, SRTP entities are free to choose SSRC values as they see fit, and to start up new SRTP sources with new SRTP master keys (see Section 2.2) within a session without coordinating with other entities via signaling or other external means. This fact allows to reinstate the RTP collision detection and repair mechanism, which is nullified by the current SRTP specification because of the need to control SSRC values closely. An SRTP endpoint using EKT can generate new keys whenever an existing SRTP master key has been overused, or start up a new SRTP source to replace an old SRTP source that has reached the packet-count limit. EKT also solves the problem in which the burst loss of the N initial SRTP packets can confuse an SRTP receiver, when the initial RTP sequence number is greater than or equal to  $2^{16} - N$ . These features can simplify many architectures that implement SRTP.

EKT provides a way for an SRTP session participant, either a sender or a receiver, to securely transport its SRTP master key and current SRTP rollover counter to the other participants in the session. This data, possibly in conjunction with additional data provided by an external signaling protocol, furnishes the information needed by the receiver to instantiate an SRTP/SRTCP receiver context.

EKT does not control the manner in which the SSRC and master key are generated; it is only concerned with their secure transport. Those values may be generated on demand by the SRTP endpoint, or may be dictated by an external mechanism such as a signaling agent or a secure group controller.

EKT is not intended to replace external key establishment mechanisms such as SDP Security Descriptions [RFC4568], DTLS-SRTP [RFC5764], or MIKEY [RFC3830][RFC4563]. Instead, it is used in conjunction with those methods, and it relieves them of the burden of tightly coordinating every SRTP source among every SRTP participant.

This document is organized as follows. The complete normative definition of EKT is contained in Section 2. It mainly consists of packet processing algorithms (Section 2.2) and cryptographic definitions (Section 2.3). Section 3, Section 4, and Section 5 define the use of EKT with SDP Security Descriptions, DTLS-SRTP, and MIKEY, respectively. Section 7 provides a design rationale. Section 6 explains how EKT can interwork with keying in call signaling. Security Considerations are discussed in Section 8, and IANA considerations are provided in Section 9.

### 1.1. History

RFC Editor Note: please remove this section prior to publication as an RFC.

This version is substantially revised from earlier versions, in order to make it possible for the EKT data to be removed from a packet without affecting the ability of the receiver to correctly process the data that is present in that packet. This capability facilitates interoperability between SRTP implementations with different SRTP key management methods. The changes also greatly simplify the EKT processing rules, and make the EKT data that must be carried in SRTP and/or SRTCP packets somewhat larger.

### 1.2. Conventions Used In This Document

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 [RFC2119].

## 2. Encrypted Key Transport

In EKT, an SRTP master key is encrypted with a key encrypting key and the resulting ciphertext is transported in selected SRTCP or in selected SRTP packets. The key encrypting key is called an EKT key. A single such key suffices for a single SRTP session, regardless of the number of participants in that session. However, there can be multiple EKT keys used within a particular session.

EKT defines a new method of providing SRTP master keys to an endpoint. In order to convey the ciphertext of the SRTP master key, and other additional information, an additional EKT field is added to SRTP or SRTCP packets. When added to SRTCP, the EKT field appears at the end of the packet, after the authentication tag, if that tag is present, or after the MKI or the SRTCP index otherwise. When added to SRTP, the EKT field appears at the end of the packet, after the authentication tag, if that tag is present, or after the MKI or the ciphertext of the encrypted portion of the packet otherwise.

### 2.1. EKT Field Formats

The EKT Field uses one of the two formats defined below. These two formats can always be unambiguously distinguished on receipt by examining the final bit of the EKT Field, which is also the final bit of the SRTP or SRTCP packet. The first format is the Full EKT Field (or `Full_EKT_Field`), and the second is the Short EKT Field (or `Short_EKT_Field`). The formats are defined as

```

EKT_Plaintext = SRTP_Master_Key || SSRC || ROC || ISN

EKT_Ciphertext = EKT_Encrypt(EKT_Key, EKT_Plaintext)

Full_EKT_Field = EKT_Ciphertext || SPI || '1'

Short_EKT_Field = Reserved || '0'

```

Figure 1: EKT data formats

Here `||` denotes concatenation, and `'1'` and `'0'` denote single one and zero bits, respectively. These fields and data elements are defined as follows:

**EKT\_Plaintext:** The data that is input to the EKT encryption operation. This data never appears on the wire, and is used only in computations internal to EKT.

**EKT\_Ciphertext:** The data that is output from the EKT encryption operation, which is performed as as defined in Section 2.3. This field is included in SRTP and SRTCP packets when EKT is in use. The length of this field is variable, and is equal to the ciphertext size N defined in Section 2.3. Note that the length of the field is inferable from the SPI field, since the particular EKT cipher used by the sender of a packet can be inferred from that field.

**Rollover Counter (ROC):** The length of this field is fixed at 32 bits. It is included in the EKT plaintext, but does not appear on the wire. On the sender side, this field is set to the current value of the SRTP rollover counter in the SRTP context associated with the SSRC in the SRTP or SRTCP packet.

**Initial Sequence Number (ISN):** The length of this field is fixed at 16 bits. It is included the EKT plaintext, but does not appear on the wire. If this field is nonzero, then it indicates the RTP sequence number of the initial RTP packet that is protected using the SRTP master key conveyed (in encrypted form) by the EKT Ciphertext field of this packet. If this field is zero, it indicates that the initial RTP packet protected using the SRTP master key conveyed in this packet preceded, or was concurrent with, the last roll-over of the RTP sequence number.

**Security Parameter Index (SPI):** The length of this field is fixed at 15 bits. This field is included in SRTP and SRTCP packets when EKT is in use. It indicates the appropriate EKT key and other parameters for the receiver to use when processing the packet. It is an "index" into a table of possibilities (which are established via signaling or some other out-of-band means), much like the IPsec Security Parameter Index [RFC4301]. The parameters that are identified by this field are:

- \* The EKT key used to process the packet.
- \* The EKT cipher used to process the packet.
- \* The Secure RTP parameters associated with the SRTP Master Key carried by the packet and the SSRC value in the packet. Section 8.2. of [RFC3711] summarizes the parameters defined by that specification.
- \* The Master Salt associated with the Master Key. (This value is part of the parameters mentioned above, but we call it out for emphasis.) The Master Salt is communicated separately, via signaling, typically along with the EKT key.

Together, these data elements associated with an instance of EKT are called an EKT parameter set. Within each SRTP session, each distinct EKT parameter set that may be used MUST be associated with a distinct SPI value, to avoid ambiguity.

Reserved: MUST be all zeros on transmission, and MUST be ignored on reception.

Examples of the Full\_EKT\_Field and Short\_EKT\_Field formats are shown in (Figure 2) and (Figure 3), respectively. These figures show the on-the-wire data. The Ciphertext field holds encrypted data, and thus has no apparent inner structure.

```

      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
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
:
:                               EKT Ciphertext                               :
:
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|   Security Parameter Index   |1|
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+

```

Figure 2: An example of the Full EKT Field format

```

      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
+---+---+---+---+---+
|   Reserved   |0|
+---+---+---+---+---+

```

Figure 3: An example of the Short EKT Field format

## 2.2. Packet Processing and State Machine

At any given time, each SRTP/SRTCP source has associated with it a single EKT parameter set. This parameter set is used to process all outbound packets, and is called the outbound parameter set. There may be other EKT parameter sets that are used by other SRTP/SRTCP sources in the same session. All of these EKT parameter sets SHOULD be stored by all of the participants in an SRTP session, for use in processing inbound SRTCP traffic.

### 2.2.1. Outbound Processing

When an SRTP or SRTCP packet is to be sent, the EKT field for that packet is created as follows, or uses an equivalent set of steps.

The creation of the EKT field MUST precede the normal SRTP or SRTCP packet processing, so that the ROC used in EKT is the same as the one used in the SRTP or SRTCP processing.

First, the sender decides whether to use the Full or Short format. When sending EKT with SRTP, the Full format SHOULD be used on the initial SRTP packet in a session and after each rekeying event. When sending EKT with SRTCP, the Full format MUST be used. Not all SRTP or SRTCP packets need to include the EKT key, but it SHOULD be included with some regularity, e.g., every second or every ten seconds, though it need not be sent on a regular schedule.

If the Short format is used, an all-zero Reserved octet is appended to the packet. Otherwise, processing continues as follows.

The Rollover Counter field in the packet is set to the current value of the SRTP rollover counter (represented as an unsigned integer in network byte order).

The Initial Sequence Number field is set to zero, if the initial RTP packet protected using the current SRTP master key for this source preceded, or was concurrent with, the last roll-over of the RTP sequence number. Otherwise, that field is set to the value of the RTP sequence number of the initial RTP packet that was or will be protected by that key. When the SRTP master key corresponding to a source is changed, the new key SHOULD be communicated in advance via EKT. (Note that the ISN field allows the receiver to know when it should start using the new key to process SRTP packets.) This enables the rekeying event to be communicated before any RTP packets are protected with the new key. The rekeying event MUST NOT change the value of ROC (otherwise, the current value of the ROC would not be known to late joiners of existing sessions).

The Security Parameter Index field is set to the value of the Security Parameter Index that is associated with the outbound parameter set.

The EKT\_Plaintext field is computed from the SRTP Master Key, SSRC, ROC, and ISN fields, as shown in Figure 1.

The EKT\_Ciphertext field is set to the ciphertext created by encrypting the EKT\_Plaintext with the EKT cipher, using the KEK as the encryption key. The encryption process is detailed in Section 2.3. Implementations MAY cache the value of this field to avoid recomputing it for each packet that is sent.

### 2.2.2. Inbound Processing

When an SRTP or SRTCP packet containing an EKT field is received, it is processed as follows, or uses an equivalent set of steps. Inbound EKT processing MUST take place prior to the usual SRTP or SRTCP processing.

1. The final bit is checked to determine which EKT format is in use. If the packet contains a Short EKT Tag, then the EKT Tag is stripped off of the packet, and then the normal SRTP or SRTCP processing is applied. If the packet contains a Full EKT Tag, then processing continues as described below.
2. The Security Parameter Index (SPI) field is checked to determine which EKT parameter set should be used when processing the packet. If multiple parameter sets have been defined for the SRTP session, then the one that is associated with the value of the SPI field in the packet is used. This parameter set is called the matching parameter set below. If there is no matching SPI, then the verification function MUST return an indication of authentication failure, and the steps described below are not performed.
3. The EKT\_Ciphertext is decrypted using the EKT\_Key and EKT\_Cipher in the matching parameter set, as described in Section 2.3. If the EKT decryption operation returns an authentication failure, then the packet processing halts with an indication of failure. Otherwise, the resulting EKT\_Plaintext is parsed as described in Figure 1, to recover the SRTP Master Key, SSRC, ROC, and ISN fields.
4. The SSRC field output from the decryption operation is compared to the SSRC field from the SRTP header. If the values of the two fields do not match, then packet processing halts with an indication of failure. Otherwise, it continues as follows.
5. If the ROC from the EKT\_Plaintext is less than the ROC in the SRTP context, then packet processing halts. Otherwise, the ROC in the SRTP context is set to the value of the ROC from the EKT\_Plaintext, and the SRTP Master Key from the EKT\_Plaintext is accepted as the SRTP master key corresponding to the SRTP source that sent the packet. If an MKI is present in the packet, then the master key corresponds to the particular SSRC and MKI combination. If there is no SRTP crypto context corresponding to the SSRC in the packet, then a new crypto context is created. If the crypto context is not new, then the rollover counter in the context MUST NOT be set to a value lower than its current value. (If the replay protection step described above is performed, it

ensures that this requirement is satisfied.)

6. If the Initial Sequence Number field is nonzero, then the initial sequence number for the SRTP master key is set to the packet index created by appending that field to the current rollover counter and treating the result as a 48-bit unsigned integer. The initial sequence number for the master key is equivalent to the "From" value of the <From, To> pair of indices (Section 8.1.1 of [RFC3711]) that can be associated with a master key.
7. The newly accepted SRTP master key, the SRTP parameters from the matching parameter set, the SSRC from the packet, and the MKI from the packet, if one is present, are stored in the crypto context associated with the SRTP source. The SRTP Key Derivation algorithm is run in order to compute the SRTP encryption and authentication keys, and those keys are stored for use in SRTP processing of inbound packets. The Key Derivation algorithm takes as input the newly accepted SRTP master key, along with the Master Salt from the matching parameter set.

Implementation note: the receiver may want to retain old master keys for some brief period of time, so that out of order packets can be processed.

8. At this point, EKT processing has successfully completed, and the normal SRTP or SRTCP processing takes place.

Implementation note: the value of the EKT Ciphertext field is identical in successive packets protected by the same EKT parameter set and the same SRTP master key and ROC. This ciphertext value MAY be cached by an SRTP receiver to minimize computational effort by noting when the SRTP master key is unchanged and avoiding repeating Steps 2, 3, 4 5, and 6.

### 2.3. Ciphers

EKT uses an authenticated cipher to encrypt the SRTP master keys, ROC, and ISN. We first specify the interface to the cipher, in order to abstract the interface away from the details of that function. We then define the cipher that is used in EKT by default. This cipher MUST be implemented, but another cipher that conforms to this interface MAY be used, in which case its use MUST be coordinated by external means (e.g., call signaling).

An EKT cipher consists of an encryption function and a decryption function. The encryption function  $E(K, P)$  takes the following inputs:



- o a secret key  $K$  with a length of  $L$  bytes, and
- o a plaintext value  $P$  with a length of  $M$  bytes.

The encryption function returns a ciphertext value  $C$  whose length is  $N$  bytes, where  $N$  is at least  $M$ . The decryption function  $D(K, C)$  takes the following inputs:

- o a secret key  $K$  with a length of  $L$  bytes, and
- o a ciphertext value  $C$  with a length of  $N$  bytes.

The decryption function returns a plaintext value  $P$  that is  $M$  bytes long, or returns an indication that the decryption operation failed because the ciphertext was invalid (i.e. it was not generated by the encryption of plaintext with the key  $K$ ).

These functions have the property that  $D(K, E(K, P)) = P$  for all values of  $K$  and  $P$ . Each cipher also has a limit  $T$  on the number of times that it can be used with any fixed key value. For each key, the encryption function MUST NOT be invoked on more than  $T$  distinct values of  $P$ , and the decryption function MUST NOT be invoked on more than  $T$  distinct values of  $C$ .

The length of the EKT Plaintext is ten bytes, plus the length of the SRTP Master Key.

Security requirements for EKT ciphers are discussed in Section 8.

#### 2.3.1. The Default Cipher

The default EKT Cipher is the Advanced Encryption Standard (AES) [FIPS197] Key Wrap with Padding [RFC5649] algorithm, which can be used with plaintexts larger than 16 bytes in length, and is thus suitable for keys of any size. It requires a plaintext length  $M$  that is at least eight bytes, and it returns a ciphertext with a length of  $N = M + 8$  bytes. It can be used with key sizes of  $L = 16, 24$ , and  $32$ , and its use with those key sizes is indicated as AESKW\_128, AESKW\_192, and AESKW\_256, respectively. The key size determines the length of the AES key used by the Key Wrap algorithm. With this cipher,  $T=2^{48}$ .

When AES-128 is used in SRTP and/or SRTCP, AESKW\_128 SHOULD be used in EKT. In this case, the EKT Plaintext is 26 bytes long, the EKT Ciphertext is 40 bytes long, and the Full EKT field is 42 bytes long.

When AES-192 is used in SRTP and/or SRTCP, AESKW\_192 SHOULD be used in EKT. In this case, the EKT Plaintext is 34 bytes long, the EKT

Ciphertext is 48 bytes long, and the Full EKT field is 50 bytes long.

When AES-256 is used in SRTP and/or SRTCP, AESKW\_256 SHOULD be used in EKT. In this case, the EKT Plaintext is 42 bytes long, the EKT Ciphertext is 56 bytes long, and the Full EKT field is 58 bytes long.

### 2.3.2. Other EKT Ciphers

Other specifications may extend this one by defining other EKT ciphers per Section 9. This section defines how those ciphers interact with this specification.

An EKT cipher determines how the EKT Ciphertext field is written, and how it is processed when it is read. This field is opaque to the other aspects of EKT processing. EKT ciphers are free to use this field in any way, but they SHOULD NOT use other EKT or SRTP fields as an input. The values of the parameters L, M, N, and T MUST be defined by each EKT cipher, and those values MUST be inferable from the EKT parameter set.

### 2.4. Synchronizing Operation

A participant in a session MAY opt to use a particular EKT key to protect outbound packets after it accepts that EKT key for protecting inbound traffic. In this case, the fact that one participant has changed to using a new EKT key for outbound traffic can trigger other participants to switch to using the same key.

An SRTP/SRTCP source SHOULD change its SRTP master key after its EKT key has been changed. This will ensure that the set of participants able to decrypt the traffic will be limited to those who know the current EKT key.

EKT can be transported over SRTCP, but some of the information that it conveys is used for SRTP processing; some elements of the EKT parameter set apply to both SRTP and SRTCP. Furthermore, SRTCP packets can be lost and both SRTP and SRTCP packets may be delivered out of order. This can lead to various race conditions if EKT is transported over SRTCP but not SRTP, which we review below.

When joining an SRTP session, SRTP packets may be received before any EKT over SRTCP packets, which implies the crypto context has not been established, unless other external signaling mechanism has done so. Rather than automatically discarding such SRTP packets, the receiver MAY want to provisionally place them in a jitter buffer and delay discarding them until playout time.

When an SRTP source using EKT over SRTCP performs a rekeying

operation, there is a race between the actual rekeying signaled via SRTCP and the SRTP packets secured by the new keying material. If the SRTP packets are received first, they will fail authentication; alternatively, if authentication is not being used, they will decrypt to unintelligible random-looking plaintext. (Note, however, that [RFC3711] says that SRTP "SHOULD NOT be used without message authentication".) In order to address this problem, the rekeying event can be sent before packets using the new SRTP master key are sent (by use of the ISN field). Another solution involves using an MKI at the expense of added overhead in each SRTP packet. Alternatively, receivers MAY want to delay discarding packets from known SSRs that fail authentication in anticipation of receiving a rekeying event via EKT (SRTCP) shortly.

The ROC signaled via EKT over SRTCP may be off by one when it is received by the other party(ies) in the session. In order to deal with this, receivers should simply follow the SRTP packet index estimation procedures defined in Section 3.3.1 [RFC3711].

## 2.5. Transport

EKT MUST be used over SRTCP, whenever RTCP is in use. EKT MAY be used over SRTP. When EKT over SRTP is used in an SRTP session in which SRTCP is available, then EKT MUST be used for both SRTP and SRTCP.

The packet processing, state machine, and Authentication Tag format for EKT over SRTP are nearly identical to that for EKT over SRTCP. Differences are highlighted in Section 2.2.1 and Section 2.2.2.

## 2.6. Timing and Reliability Consideration

SRTCP communicates the master key and ROC for the SRTP session. Thus, as explained above, if SRTP packets are received prior to the corresponding SRTCP (EKT) packet, a race condition occurs. From an EKT point of view, it is therefore desirable for an SRTP sender to send an EKT packet containing the Base Authentication Tag as soon as possible, and in no case any later than when the initial SRTP packet is sent. It is RECOMMENDED that the Base Authentication Tag be transmitted 3 times (to accommodate packet loss) and to provide a reliable indication to the receiver that the sender is now using the EKT key. If the Base Authentication Tag sent in SRTCP, the SRTCP timing rules associated with the profile under which it runs (e.g., RTP/SAVP or RTP/SAVPF) MUST be obeyed. Subject to that constraint, SRTP senders using EKT over SRTCP SHOULD send an SRTCP packet as soon as possible after joining a session. Note that there is no need for SRTP receivers to do so. Also note, that per RFC 3550, Section 6.2, it is permissible to send a compound RTCP packet immediately after

joining a unicast session (but not a multicast session).

SRTP is not reliable and hence SRTP packets may be lost. This is obviously a problem for endpoints joining an SRTP session and receiving SRTP traffic (as opposed to SRTCP), or for endpoints receiving SRTP traffic following a rekeying event. To reduce the impact of lost packets, SRTP senders using EKT over SRTCP SHOULD send SRTCP packets as often as allowed by the profile under which they operate.

### 3. Use of EKT with SDP Security Descriptions

The SDP Security Descriptions (SDESC) [RFC4568] specification defines a generic framework for negotiating security parameters for media streams negotiated via the Session Description Protocol by use of a new SDP "crypto" attribute and the Offer/Answer procedures defined in [RFC3264]. In addition to the general framework, SDES also defines how to use that framework specifically to negotiate security parameters for Secure RTP. Below, we first provide a brief recap of the crypto attribute when used for SRTP and we then explain how it is complementary to EKT. In the rest of this Section, we provide extensions to the crypto attribute and associated offer/answer procedures to define its use with EKT.

#### 3.1. SDP Security Descriptions Recap

The SRTP crypto attribute defined for SDESC contains a tag followed by three types of parameters (refer to [RFC4568] for details):

- o Crypto-suite. Identifies the encryption and authentication transform
- o Key parameters. SRTP keying material and parameters.
- o Session parameters. Additional (optional) SRTP parameters such as Key Derivation Rate, Forward Error Correction Order, use of unencrypted SRTP, and other parameters defined by SDESC.

The crypto attributes in the example SDP in Figure 4 illustrate these parameters.

```
v=0
o=sam 2890844526 2890842807 IN IP4 192.0.2.5
s=SRTP Discussion
i=A discussion of Secure RTP
u=http://www.example.com/seminars/srtp.pdf
e=marge@example.com (Marge Simpson)
c=IN IP4 192.0.2.12
t=2873397496 2873404696
m=audio 49170 RTP/SAVP 0
a=crypto:1 AES_CM_128_HMAC_SHA1_80
    inline:WVNfXl9zZWljdGwgKCKgewkyMjA7fQp9CnVubGVz|2^20|1:4
    FEC_ORDER=FEC_SRTP
a=crypto:2 F8_128_HMAC_SHA1_80
    inline:MTIzNDU2Nzg5QUJDREUwMTIzNDU2Nzg5QUJjZGVm|2^20|1:4;
    inline:QUJjZGVmMTIzNDU2Nzg5QUJDREUwMTIzNDU2Nzg5|2^20|2:4
    FEC_ORDER=FEC_SRTP
```

Figure 4: SDP Security Descriptions example

For legibility the SDP shows line breaks that are not present on the wire.

The first crypto attribute has the tag "1" and uses the crypto-suite AES\_CM\_128\_HMAC\_SHA1\_80. The "inline" parameter provides the SRTP master key and salt, the master key lifetime (number of packets), and the (optional) Master Key Identifier (MKI) whose value is "1" and has a byte length of "4" in the SRTP packets. Finally, the FEC\_ORDER session parameter indicates the order of Forward Error Correction used (FEC is applied before SRTP processing by the sender of the SRTP media).

The second crypto attribute has the tag "2" and uses the crypto-suite F8\_128\_HMAC\_SHA1\_80. It includes two SRTP master keys and associated salts. The first one is used with the MKI value 1, whereas the second one is used with the MKI value 2. Finally, the FEC\_ORDER session parameter indicates the order of Forward Error Correction used.

### 3.2. Relationship between EKT and SDP Security Descriptions

SDP Security Descriptions [RFC4568] define a generic framework for negotiating security parameters for media streams negotiated via the Session Description Protocol by use of the Offer/Answer procedures defined in [RFC3264]. In addition to the general framework, SDESC also defines how to use it specifically to negotiate security parameters for Secure RTP.

EKT and SDESC are complementary. SDESC can negotiate several of the

SRTP security parameters (e.g., cipher and use of Master Key Identifier/MKI) as well as SRTP master keys. SDESC, however, does not negotiate SSRs and their associated Rollover Counter (ROC). Instead, SDESC relies on a so-called "late binding", where a newly observed SSR will have its crypto context initialized to a ROC value of zero. Clearly, this does not work for participants joining an SRTP session that has been established for a while and hence has a non-zero ROC. It is impossible to use SDESC to join an SRTP session that is already in progress. In this case, EKT on the endpoint running SDP Security can provide the additional signaling necessary to communicate the ROC (Section 6.4.1 of [RFC4568]). The use of EKT solves this problem by communicating the ROC associated with the SSR in the media plane.

SDP Security Descriptions negotiates different SRTP master keys in the send and receive direction. The offer contains the master key used by the offerer to send media, and the answer contains the master key used by the answerer to send media. Consequently, if media is received by the offerer prior to the answer being received, the offerer does not know the master key being used. Use of SDP security preconditions can solve this problem, however it requires an additional round-trip as well as a more complicated state machine. EKT solves this problem by simply sending the master key used in the media plane thereby avoiding the need for security preconditions.

If multiple crypto-suites were offered, the offerer also will not know which of the crypto-suites offered was selected until the answer is received. EKT solves this problem by using a correlator, the Security Parameter Index (SPI), which uniquely identifies each crypto attribute in the offer.

One of the primary call signaling protocols using offer/answer is the Session Initiation Protocol (SIP) [RFC3261]. SIP uses the INVITE message to initiate a media session and typically includes an offer SDP in the INVITE. An INVITE may be "forked" to multiple recipients which potentially can lead to multiple answers being received. SDESC, however, does not properly support this scenario, mainly because SDP and RTP/RTCP does not contain sufficient information to allow for correlation of an incoming RTP/RTCP packet with a particular answer SDP. Note that extensions providing this correlation do exist (e.g., Interactive Connectivity Establishment (ICE)). SDESC addresses this point-to-multipoint problem by moving each answer to a separate RTP transport address thereby turning a point-to-multipoint scenario into multiple point-to-point scenarios. There are however significant disadvantages to doing so. As long as the crypto attribute in the answer does not contain any declarative parameters that differ from those in the offer, EKT solves this problem by use of the SPI correlator and communication of the

answerer's SRTP master key in EKT.

As can be seen from the above, the combination of EKT and SDESC provides a better solution to SRTP negotiation for offer/answer than either of them alone. SDESC negotiates the various SRTP crypto parameters (which EKT does not), whereas EKT addresses the shortcomings of SDESC.

### 3.3. Overview of Combined EKT and SDP Security Description Operation

We define three session extension parameters to SDESC to communicate the EKT cipher, EKT key, and Security Parameter Index to the peer. The original SDESC parameters are used as defined in [RFC4568], however the procedures associated with the SRTP master key differ slightly, since both SDESC and EKT communicate an SRTP master key. In particular, the SRTP master key communicated via SDESC is used only if there is currently no crypto context established for the SSRC in question. This will be the case when an entity has received only the offer or answer, but has yet to receive a valid EKT message from the peer. Once a valid EKT message is received for the SSRC, the crypto context is initialized accordingly, and the SRTP master key will then be derived from the EKT message. Subsequent offer/answer exchanges do not change this: The most recent SRTP master key negotiated via EKT will be used, or, if none is available for the SSRC in question, the most recent SRTP master key negotiated via offer/answer will be used. Note that with these rules, once a valid EKT message has been received for a given SSRC, rekeying for that SSRC can only be done via EKT. The associated SRTP crypto parameters however can be changed via SDESC.

### 3.4. EKT Extensions to SDP Security Descriptions

In order to use EKT and SDESC in conjunction with each other, the following new SDES session parameters are defined. These MUST NOT appear more than once in a given crypto attribute:

EKT\_Cipher: The EKT cipher used to encrypt the SRTP Master Key

EKT\_Key: The EKT key used to encrypt the SRTP Master Key

EKT\_SPI: The EKT Security Parameter Index

Below are details on each of these attributes.

#### 3.4.1. EKT\_Cipher

The (optional) EKT\_Cipher parameter defines the EKT cipher used to encrypt the EKT key with in SRTCP packets. The default value is



"AESKW\_128" in accordance with Section 2.3.1. For the AES Key Wrap cipher, the values "AESKW\_128", "AESKW\_192", and "AESKW\_256" are defined for values of L=16, 24, and 32 respectively. In the Offer/Answer model, the EKT\_Cipher parameter is a negotiated parameter.

#### 3.4.2. EKT\_Key

The (mandatory) EKT\_Key parameter is the key K used to encrypt the SRTP Master Key in SRTCP packets. The value is base64 encoded as described in Section 4 [RFC4648]. When base64 decoding the key, padding characters (i.e., one or two "=" at the end of the base64 encoded data) are discarded (see [RFC4648] for details). Base64 encoding assumes that the base64 encoding input is an integral number of octets. If a given EKT cipher requires the use of a key with a length that is not an integral number of octets, said cipher MUST define a padding scheme that results in the base64 input being an integral number of octets. For example, if the length defined was 250 bits, then 6 padding bits would be needed, which could be defined to be the last 6 bits in a 256 bit input. In the Offer/Answer model, the EKT\_Key parameter is a negotiated parameter.

#### 3.4.3. EKT\_SPI

The (mandatory) EKT\_SPI parameter is the Security Parameter Index. It is encoded as an ASCII string representing the hexadecimal value of the Security Parameter Index. The SPI identifies the \*offer\* crypto attribute (including the EKT Key and Cipher) being used for the associated SRTP session. A crypto attribute corresponds to an EKT Parameter Set and hence the SPI effectively identifies a particular EKT parameter set. Note that the scope of the SPI is the SRTP session, which may or may not be limited to the scope of the associated SIP dialog. In particular, if one of the participants in an SRTP session is an SRTP translator, the scope of the SRTP session is not limited to the scope of a single SIP dialog. However, if all of the participants in the session are endpoints or mixers, the scope of the SRTP session will correspond to a single SIP dialog. In the Offer/Answer model, the EKT\_SPI parameter is a negotiated parameter.

#### 3.5. Offer/Answer Procedures

In this section, we provide the offer/answer procedures associated with use of the three new SDESC parameters defined in Section 3.4. Since SDESC is defined only for unicast streams, we provide only offer/answer procedures for unicast streams here as well.

### 3.5.1. Generating the Initial Offer - Unicast Streams

When the initial offer is generated, the offerer MUST follow the steps defined in [RFC4568] Section 7.1.1 as well as the following steps.

For each unicast media line using SDESC and where use of EKT is desired, the offerer MUST include one EKT\_Key parameter and one EKT\_SPI parameter in at least one "crypto" attribute (see [RFC4568]). The EKT\_SPI parameter serves to identify the EKT parameter set used for a particular SRTP packet. Consequently, within a single media line, a given EKT\_SPI value MUST NOT be used with multiple crypto attributes. Note that the EKT parameter set to use for the session is not yet established at this point; each offered crypto attribute contains a candidate EKT parameter set. Furthermore, if the media line refers to an existing SRTP session, then any SPI values used for EKT parameter sets in that session MUST NOT be remapped to any different EKT parameter sets. When an offer describes an SRTP session that is already in progress, the offer SHOULD use an EKT parameter set (incl. EKT\_SPI and EKT\_KEY) that is already in use.

If an EKT\_Cipher other than the default cipher is to be used, then the EKT\_Cipher parameter MUST be included as well.

If a given crypto attribute includes more than one set of SRTP key parameters (SRTP master key, salt, lifetime, MKI), they MUST all use the same salt. (EKT requires a single shared salt between all the participants in the direct SRTP session).

**Important Note:** The scope of the offer/answer exchange is the SIP dialog(s) established as a result of the INVITE, however the scope of EKT is the direct SRTP session, i.e., all the participants that are able to receive SRTP and SRTCP packets directly. If an SRTP session spans multiple SIP dialogs, the EKT parameter sets MUST be synchronized between all the SIP dialogs where SRTP and SRTCP packets can be exchanged. In the case where the SIP entity operates as an RTP mixer (and hence re-originates SRTP and SRTCP packets with its own SSRC), this is not an issue, unless the mixer receives traffic from the various participants on the same destination IP address and port, in which case further coordination of SPI values and crypto parameters may be needed between the SIP dialogs (note that SIP forking with multiple early media senders is an example of this). However if it operates as an RTP translator, synchronized negotiation of the EKT parameter sets on *\*all\** the involved SIP dialogs will be needed. This is non-trivial in a variety of use cases, and hence use of the combined SDES/EKT mechanism with RTP translators should be considered very carefully. It should be noted, that use of SRTP

with RTP translators in general should be considered very carefully as well.

The EKT session parameters can either be included as optional or mandatory parameters, however within a given crypto attribute, they MUST all be either optional or mandatory.

### 3.5.2. Generating the Initial Answer - Unicast Streams

When the initial answer is generated, the answerer MUST follow the steps defined in [RFC4568] Section 7.1.2 as well as the following steps.

For each unicast media line using SDESC, the answerer examines the associated crypto attribute(s) for the presence of EKT parameters. If mandatory EKT parameters are included with a "crypto" attribute, the answerer MUST support those parameters in order to accept that offered crypto attribute. If optional EKT parameters are included instead, the answerer MAY accept the offered crypto attribute without using EKT. However, doing so will prevent the offerer from processing any packets received before the answer. If neither optional nor mandatory EKT parameters are included with a crypto attribute, and that crypto attribute is accepted in the answer, EKT MUST NOT be used. If a given a crypto attribute includes a mixture of optional and mandatory EKT parameters, or an incomplete set of mandatory EKT parameters, that crypto attribute MUST be considered invalid.

When EKT is used with SDESC, the offerer and answerer MUST use the same SRTP master salt. Thus, the SRTP key parameter(s) in the answer crypto attribute MUST use the same master salt as the one accepted from the offer.

When the answerer accepts the offered media line and EKT is being used, the crypto attribute included in the answer MUST include the same EKT parameter values as found in the accepted crypto attribute from the offer (however, if the default EKT cipher is being used, it may be omitted). Furthermore, the EKT parameters included MUST be mandatory (i.e., no "-" prefix).

Acceptance of a crypto attribute with EKT parameters leads to establishment of the EKT parameter set for the corresponding SRTP session. Consequently, the answerer MUST send packets in accordance with that particular EKT parameter set only. If the answerer wants to enable the offerer to process SRTP packets received by the offerer before it receives the answer, the answerer MUST NOT include any declarative session parameters that either were not present in the offered crypto attribute, or were present but with a different value.

Otherwise, the offerer's view of the EKT parameter set would differ from the answerer's until the answer is received. Similarly, unless the offerer and answerer has other means for correlating an answer with a particular SRTP session, the answer SHOULD NOT include any declarative session parameters that either were not present in the offered crypto attribute, or were present but with a different value. If this recommendation is not followed and the offerer receives multiple answers (e.g., due to SIP forking), the offerer may not be able to process incoming media stream packets correctly.

### 3.5.3. Processing of the Initial Answer - Unicast Streams

When the offerer receives the answer, it MUST perform the steps in [RFC4568] Section 7.1.3 as well as the following steps for each SRTP media stream it offered with one or more crypto lines containing EKT parameters in it.

If the answer crypto line contains EKT parameters, and the corresponding crypto line from the offer contained the same EKT values, use of EKT has been negotiated successfully and MUST be used for the media stream. When determining whether the values match, optional and mandatory parameters MUST be considered equal. Furthermore, if the default EKT cipher is being used, it MAY be either present or absent in the offer and/or answer.

If the answer crypto line does not contain EKT parameters, then EKT MUST NOT be used for the corresponding SRTP session. Note that if the accepted crypto attribute contained mandatory EKT parameters in the offer, and the crypto attribute in the answer does not contain EKT parameters, then negotiation has failed (Section 5.1.3 of [RFC4568]).

If the answer crypto line contains EKT parameters but the corresponding offered crypto line did not, or if the parameters don't match or are invalid, then the offerer MUST consider the crypto line invalid (see Section 7.1.3 of [RFC4568] for further operation).

The EKT parameter set is established when the answer is received, however there are a couple of special cases to consider here. First of all, if an SRTCP packet is received prior to the answer, then the EKT parameter set is established provisionally based on the SPI included. Once the answer (which may include declarative session parameters) is received, the EKT parameter set is fully established. The second case involves receipt of multiple answers due to SIP forking. In this case, there will be multiple EKT parameter sets; one for each SRTP session. As mentioned earlier, reliable correlation of SIP dialogs to SRTP sessions requires extensions, and hence if one or more of the answers include declarative session

parameters, it may be difficult to fully establish the EKT parameter set for each SRTP session. In the absence of a specific correlation mechanism, it is RECOMMENDED, that such correlation be done based on the signaled receive IP-address in the SDP and the observed source IP-address in incoming SRTP/SRTCP packets, and, if necessary, the signaled receive UDP port and the observed source UDP port.

### 3.6. SRTP-Specific Use Outside Offer/Answer

Security Descriptions use for SRTP is not defined outside offer/answer and hence neither does Security Descriptions with EKT.

### 3.7. Modifying the Session

When a media stream using the SRTP security descriptions has been established, and a new offer/answer exchange is performed, the offerer and answerer MUST follow the steps in Section 7.1.4 of [RFC4568] as well as the following steps. SDESC allows for all parameters of the session to be modified, and the EKT session parameters are no exception to that, however, there are a few additional rules to be adhered to when using EKT.

It is permissible to start a session without the use of EKT, and then subsequently start using EKT, however the converse is not. Thus, once use of EKT has been negotiated on a particular media stream, EKT MUST continue to be used on that media stream in all subsequent offer/answer exchanges.

The reason for this is that both SDESC and EKT communicate the SRTP Master Key with EKT Master Keys taking precedence. Reverting back to an SDESC-controlled master key in a synchronized manner is difficult.

Once EKT is being used, the salt for the direct SRTP session MUST NOT be changed. Thus, a new offer/answer which does not create a new SRTP session (e.g., because it reuses the same IP address and port) MUST use the same salt for all crypto attributes as is currently used for the direct SRTP session.

Finally, subsequent offer/answer exchanges MUST NOT remap a given SPI value to a different EKT parameter set until  $2^{32}$  other mappings have been used within the SRTP session. In practice, this requirements is most easily met by using a monotonically increasing SPI value (modulo  $2^{32}$  and starting with zero) per direct SRTP session. Note that a direct SRTP session may span multiple SIP dialogs, and in such cases coordination of SPI values across those SIP dialogs will be required. In the simple point-to-point unicast case without translators, the requirement simply applies within each media line in the SDP. In the point-to-multipoint case, the requirement applies across all the

associated SIP dialogs.

### 3.8. Backwards Compatibility Considerations

Backwards compatibility can be achieved in a couple of ways. First of all, SDESC allows for session parameters to be prefixed with "-" to indicate that they are optional. If the answerer does not support the EKT session parameters, such optional parameters will simply be ignored. When the answer is received, absence of the parameters will indicate that EKT is not being used. Receipt of SRTCP packets prior to receipt of such an answer will obviously be problematic (as is normally the case for SDESC without EKT).

Alternatively, SDESC allows for multiple crypto lines to be included for a particular media stream. Thus, two crypto lines that differ in their use of EKT parameters (presence in one, absence in the other) can be used as a way to negotiate use of EKT. When the answer is received, the accepted crypto attribute will indicate whether EKT is being used or not.

### 3.9. Grammar

The ABNF [RFC5234] syntax for the one new SDP Security Descriptions session parameter, EKT, comprising three parts is shown in Figure 5.

```

ekt      = "EKT=" cipher "|" key "|" spi
cipher   = cipher-extension / "AES_128" / "AESKW_128" /
          "AESKW_192" / "AESKW_256"
cipher-extension = 1*(ALPHA / DIGIT / "_")
key       = 1*(base64)      ; See Section 4 of [RFC4648]
base64    = ALPHA / DIGIT / "+" / "/" / "="
spi       = 4HEXDIG        ; See [RFC5234]
```

Figure 5: ABNF for the EKT session parameters

Using the example from Figure 5 with the EKT extensions to SDP Security Descriptions results in the following example SDP:

```
v=0
o=sam 2890844526 2890842807 IN IP4 192.0.2.5
s=SRTP Discussion
i=A discussion of Secure RTP
u=http://www.example.com/seminars/srtp.pdf
e=marge@example.com (Marge Simpson)
c=IN IP4 192.0.2.12
t=2873397496 2873404696
m=audio 49170 RTP/SAVP 0
a=crypto:1 AES_CM_128_HMAC_SHA1_80
    inline:WVNfx19zZW1jdGwgKCKgewkyMjA7fQp9CnVubGVz|2^20|1:4
    FEC_ORDER=FEC_SRTP EKT=AES_128|FE9C|AAE0
a=crypto:2 F8_128_HMAC_SHA1_80
    inline:MTIzNDU2Nzg5QUJDREUwMTIzNDU2Nzg5QUJjZGVm|2^20|1:4;
    inline:QUJjZGVmMTIzNDU2Nzg5QUJDREUwMTIzNDU2Nzg5|2^20|2:4
    FEC_ORDER=FEC_SRTP EKT=AES_128|FE9C|AAE0
```

For legibility the SDP shows line breaks that are not present on the wire.

#### 4. Use of EKT with DTLS-SRTP Key Transport

This document defines an extension to DTLS-SRTP called Key Transport. Using EKT with the DTLS-SRTP Key Transport extensions allows securely transporting SRTP keying material from one DTLS-SRTP peer to another, so the same SRTP keying material can be used by those peers and so those peers can process EKT keys. This combination of protocols is valuable because it combines the advantages of DTLS (strong authentication of the endpoint and flexibility) with the advantages of EKT (allowing secure multiparty RTP with loose coordination and efficient communication of per-source keys).

##### 4.1. EKT Extensions to DTLS-SRTP

This document adds a new TLS negotiated extension called "ekt". This adds a new TLS content type, EKT, and a new negotiated extension EKT. The negotiated extension MUST only be requested in conjunction with the "use\_srtp" extension (Section 3.2 of [RFC5764]). The DTLS server indicates its support for EKT by including "dtls-srtp-ekt" in its SDP and "ekt" in its TLS ServerHello message. If a DTLS client includes "ekt" in its ClientHello, but does not receive "ekt" in the ServerHello, the DTLS client MUST NOT send DTLS packets with the "ekt" content-type.



Using the syntax described in DTLS [I-D.ietf-tls-rfc4347-bis], the following structures are used:

```
enum {
    ekt_key(0),
    ekt_key_ack(1),
    ekt_key_error(254),
    (255)
} SRTPKeyTransportType;

struct {
    SRTPKeyTransportType keytrans_type;
    uint24 length;
    uint16 message_seq;
    uint24 fragment_offset;
    uint24 fragment_length;
    select (SRTPKeyTransportType) {
        case ekt_key:
            EKTkey;
    };
} KeyTransport;

enum {
    AES_128(0),
    AESKW_128(1),
    AESKW_192(2),
    AESKW_256(3),
} ektcipher;

struct {
    ektcipher EKT_Cipher;
    uint EKT_Key_Value<1..256>;
    uint EKT_Master_Salt<1..256>;
    uint16 EKT_SPI;
} EKTkey;
```

Figure 6: Additional TLS Data Structures

The diagram below shows a message flow of DTLS client and DTLS server using the DTLS-SRTP Key Transport extension. SRTP packets exchanged prior to the `ekt_message` are encrypted using the SRTP master key derived from the normal DTLS-SRTP key derivation function. After the `ekt_key` message, they can be encrypted using the EKT key.

Editor's note: do we need reliability for the `ekt_key` messages?

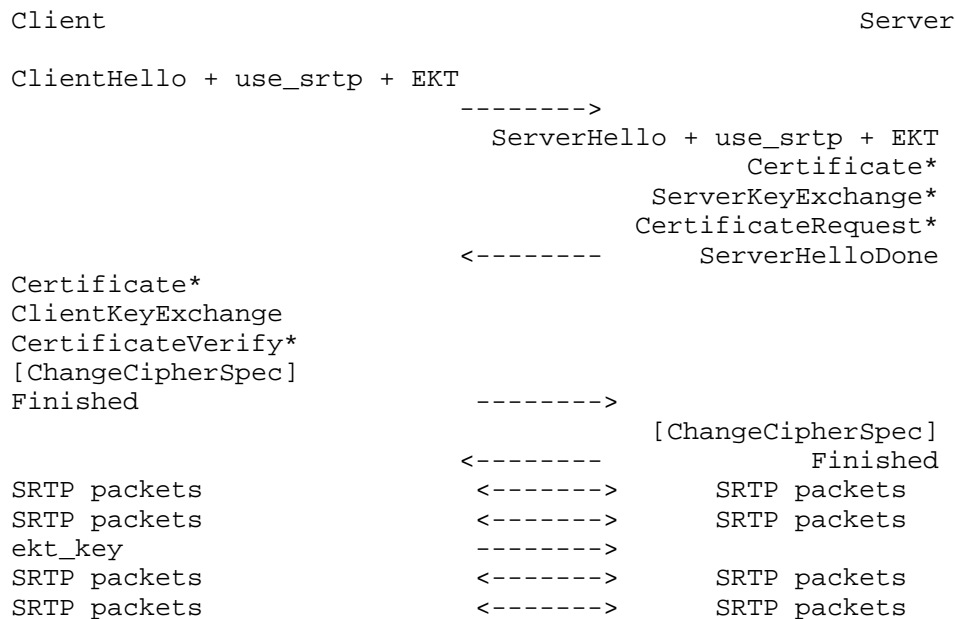


Figure 7: Handshake Message Flow

#### 4.1.1.1. Scaling to Large Groups

In certain scenarios it is useful to perform DTLS-SRTP with a device that is not the RTP peer. A common scenario is multicast, where it is necessary to distribute the DTLS-SRTP (and EKT distribution) to several devices. To allow for this, a new SDP attribute, `dtls-srtp-host`, is defined which follows the general syntax specified in Section 5.13 of [RFC4566]. When signaled, it indicates this host controls the EKT keying for all group members. For the `dtls-srtp-host` attribute:

- o the name is the ASCII string "dtls-srtp-host" (lowercase)
- o the value is the IP address and port number used for DTLS-SRTP
- o This is a media-level attribute and MUST NOT appear at the session level

The formal description of the attribute is defined by the following ABNF [RFC5234] syntax:

```

attribute = "a=dtls-srtp-host:"
           dtls-srtp-host-info *(SP dtls-srtp-host-info)
host-info = nettype space addrtype space
  
```

connection-address space port CRLF

Multiple IP/port pairs are provided for IPv6/IPv4 interworking, and to allow failover. The receiving host SHOULD attempt to use them in the order provided.

An example of SDP containing the dtls-srtp-host attribute:

```
v=0
o=sam 2890844526 2890842807 IN IP4 192.0.2.5
s=SRTP Discussion
i=A discussion of Secure RTP
u=http://www.example.com/seminars/srtp.pdf
e=marge@example.com (Marge Simpson)
c=IN IP4 192.0.2.12
t=2873397496 2873404696
m=audio 49170 UDP/TLS/RTP/SAVP 0
a=fingerprint:SHA-1
    4A:AD:B9:B1:3F:82:18:3B:54:02:12:DF:3E:5D:49:6B:19:E5:7C:AB
a=dtls-srtp-ekt
a=dtls-srtp-host:IN IP4 192.0.2.13 56789
```

For legibility the SDP shows line breaks that are not present on the wire.

#### 4.2. Offer/Answer Considerations

This section describes Offer/Answer considerations for the use of EKT together with DTLS-SRTP for unicast and multicast streams. The offerer and answerer MUST follow the procedures specified in [RFC5764] as well as the following ones.

As most DTLS-SRTP processing is performed on the media channel, rather than in SDP, there is little processing performed in SDP other than informational and to redirect DTLS-SRTP to an alternate host. Advertising support for the extension is necessary in SDP because in some cases it is required to establish an SRTP call. For example, a mixer may be able to only support SRTP listeners if those listeners implement DTLS Key Transport (because it lacks the CPU cycles necessary to encrypt SRTP uniquely for each listener).

##### 4.2.1. Generating the Initial Offer

The initial offer contains a new SDP attribute, "dtls-srtp-ekt", which contains no value. This indicates the offerer is capable of supporting DTLS-SRTP with EKT extensions, and indicates the desire to use the "ekt" extension during the DTLS-SRTP handshake. If the offerer wants another host to perform DTLS-SRTP-EKT processing, it

also includes the dtls-srtp-host attribute in its offer (Section 4.1).

An example of SDP containing the dtls-srtp-ekt attribute::

```
v=0
o=sam 2890844526 2890842807 IN IP4 192.0.2.5
s=SRTP Discussion
i=A discussion of Secure RTP
u=http://www.example.com/seminars/srtp.pdf
e=marge@example.com (Marge Simpson)
c=IN IP4 192.0.2.12
t=2873397496 2873404696
m=audio 49170 UDP/TLS/RTP/SAVP 0
a=fingerprint:SHA-1
    4A:AD:B9:B1:3F:82:18:3B:54:02:12:DF:3E:5D:49:6B:19:E5:7C:AB
a=dtls-srtp-ekt
```

For legibility the SDP shows line breaks that are not present on the wire.

#### 4.2.2. Generating the Initial Answer

Upon receiving the initial offer, the presence of the dtls-srtp-ekt attribute indicates a desire to receive the EKT extension in the DTLS-SRTP handshake. The presence of the dtls-srtp-host attribute indicates an alternate host to send the DTLS-SRTP handshake (instead of the host on the c/m lines). DTLS messages should be constructed according to those two attributes.

The SDP answer SHOULD contain the dtls-srtp-ekt attribute to indicate the answerer understands dtls-srtp. It should only contain the dtls-srtp-host attribute if the answerer also wishes to offload its DTLS-SRTP processing to another host.

#### 4.2.3. Processing the Initial Answer

The presence of the dtls-srtp-ekt attribute indicates a desire by the answerer to perform DTLS-SRTP with EKT extensions, and the dtls-srtp-host attribute indicates an alternate host for DTLS-SRTP processing.

After successful negotiation of the key\_transport extension, the DTLS client and server MAY exchange SRTP packets, encrypted using the KDF described in [RFC5764]. This is normal and expected, even if Key Transport was negotiated by both sides, as neither side may (yet) have a need to alter the SRTP key. However, it is also possible that one (or both) peers will immediately send new\_srtp\_key message before sending any SRTP, and also possible that SRTP, encrypted with an

unknown key, may be received before the `new_srtp_key` message is received.

#### 4.2.4. Modifying the Session

As DTLS-SRTP-EKT processing is done on the DTLS-SRTP channel (media channel) rather than signaling, no special processing for modifying the session is necessary.

## 5. Use of EKT with MIKEY

The advantages outlined in Section 1 are useful in some scenarios in which MIKEY is used to establish SRTP sessions. In this section, we briefly review MIKEY and related work, and discuss these scenarios.

An SRTP sender or a group controller can use MIKEY to establish a SRTP cryptographic context. This capability includes the distribution of a TEK generation key (TGK) or the TEK itself, security policy payload, crypto session bundle ID (CSB\_ID) and a crypto session ID (CS\_ID). The TEK directly maps to an SRTP master key, whereas the TGK is used along with the CSB\_ID and a CS\_ID to generate a TEK. The CS\_ID is used to generate multiple TEKs (SRTP master keys) from a single TGK. For a media stream in SDP, MIKEY allocates two consecutive numbers for the crypto session IDs, so that each direction uses a different SRTP master key (see [RFC4567]).

The MIKEY specification [RFC3830] defines three modes to exchange keys, associated parameters and to protect the MIKEY message: pre-shared key, public-key encryption and Diffie-Hellman key exchange. In the first two modes the MIKEY initiator only chooses and distributes the TGK or TEK, whereas in the third mode both MIKEY entities (the initiator and responder) contribute to the keys. All three MIKEY modes have in common that for establishing a SRTP session the exchanged key is valid for the send and receive direction. Especially for group communications it is desirable to update the SRTP master key individually per direction. EKT provides this property by distributing the SRTP master key within the SRTP/SRTCP packet.

MIKEY already supports synchronization of ROC values between the MIKEY initiator and responder. The SSRC / ROC value pair is part of the MIKEY Common Header payload. This allows providing the current ROC value to late joiners of a session. However, in some scenarios a key management based ROC synchronization is not sufficient. For example, in mobile and wireless environments, members may go in and out of coverage and may miss a sequence number overrun. In point-to-multipoint translator scenarios it is desirable to not require the group controller to track the ROC values of each member, but to provide the ROC value by the originator of the SRTP packet. A better alternative to synchronize the ROC values is to send them directly via SRTP/SRTCP, as EKT does. A separate SRTP extension is being proposed [RFC4771] to include the ROC as part of a modified authentication tag. Unlike EKT, this extension uses only SRTP and not SRTCP as its transport and does not allow updating the SRTP master key.

Besides the ROC, MIKEY synchronizes also the SSRC values of the SRTP

streams. Each sender of a stream sends the associated SSRC within the MIKEY message to the other party. If a SRTP session participant starts a new SRTP source or a new participant is added to a group, subsequent SDP offer/answer and MIKEY exchanges are necessary to update the SSRC values. EKT improves these scenarios by updating the keys and SSRC values without coordination on the signaling channel. With EKT, SRTP can handle early media, since the EKT SPI allows the receiver to identify the cryptographic keys and parameters used by the source.

The MIKEY specification [RFC3830] suggests the use of unicast for rekeying. This method does not scale well to large groups or interactive groups. The EKT extension of SRTP/SRTCP provides a solution for rekeying the SRTP master key and for ROC/SSRC synchronization. EKT is not a substitution for MIKEY, but rather a complementary addition to address the above described limitations of MIKEY.

In the next section we provide an extension to MIKEY for support of EKT. EKT can be used only with the pre-shared key or public-key encryption MIKEY mode of [RFC3830]. The Diffie-Hellman exchange mode is not suitable in conjunction with EKT, because it is not possible to establish one common EKT key over multiple EKT entities. Additional MIKEY modes specified in separate documents are not considered for EKT.

#### 5.1. EKT extensions to MIKEY

In order to use EKT with MIKEY, the EKT cipher, EKT key and EKT SPI must be negotiated in the MIKEY message exchange.

For EKT we specify a new SRTP Policy Type in the Security Policy (SP) payload of MIKEY (see Section 6.10 of [RFC3830]). The SP payload contains a set of policies. Each policy consists of a number Policy Param TLVs.

Prot type	Value
EKT	TBD (will be requested from IANA)

For legibility the SDP shows line breaks that are not present on the wire.

Figure 8: EKT Security Policy

The EKT Security Policy has one parameter representing the EKT cipher.

Type	Meaning	Possible values
0	EKT cipher	see below

Figure 9: EKT Security Policy Parameters

EKT cipher	Value
AES_128	0
AESKW_128	1
AESKW_192	2
AESKW_256	3

Figure 10: EKT Cipher Parameters

AES\_128 is the default value for the EKT cipher.

The two mandatory EKT parameters (EKT\_Key and EKT\_SPI) are transported in the MIKEY KEMAC payload within one separate Key Data sub-payload. As specified in Section 6.2 of [RFC3830], the KEMAC payload carries the TEK Generation Key (TGK) or the Traffic Encryption Key (TEK). One or more TGKs or TEKs are carried in individual Key Data sub-payloads within the KEMAC payload. The KEMAC payload is encrypted as part of MIKEY. The Key Data sub-payload, specified in Section 6.13 of [RFC3830], has the following format:

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								
Next Payload										Type										KV										Key data length									
:										Key data										:																			
Salt length (optional)										! Salt data (optional)										:																			
:										KV data (optional)										:																			

Figure 11: Key Data Sub-Payload of MIKEY

These fields are described below:

Type: 4 bits in length, indicates the type of key included in the payload. We define Type = TBD (will be requested from IANA) to indicate transport of the EKT key.



KV: (4 bits): indicates the type of key validity period specified. KV=1 is currently specified as an SPI. We use that value to indicate the KV\_data contains the ETK\_SPI for the key type EKT\_Key. KV\_data would be 16 bits in length, but it is also possible to interpret the length from the 'Key data len' field. KV data MUST NOT be optional for the key type EKT\_Key when KV = 1.

Salt length, Salt Data: These optional fields SHOULD be omitted for the key type EKT\_Key, if the SRTP master salt is already present in the TKG or TEK Key Data sub-payload. The EKT\_Key sub-payload MUST contain a SRTP master salt, if the SRTP master salt is not already present in the TKG or TEK Key Data sub-payload.

KV Data: length determined by Key Data Length field.

## 5.2. Offer/Answer considerations

This section describes Offer/Answer considerations for the use of EKT together with MIKEY for unicast streams. The offerer and answerer MUST follow the procedures specified in [RFC3830] and [RFC4567] as well as the following ones.

### 5.2.1. Generating the Initial Offer

If it is intended to use MIKEY together with EKT, the offerer MUST include at least one MIKEY key-mgmt attribute with one EKT\_Key Key Data sub-payload and the EKT\_Cipher Security Policy payload. MIKEY can be used on session or media level. On session level, MIKEY provides the keys for multiple SRTP sessions in the SDP offer. The EKT SPI references a EKT parameter set including the Secure RTP parameters as specified in Section 8.2 in [RFC3711]. If MIKEY is used on session level, it is only possible to use one EKT SPI value. Therefore, the session-level MIKEY message MUST contain one SRTP Security Policy payload only, which is valid for all related SRTP media lines. If MIKEY is used on media level, different SRTP Security Policy parameters (and consequently different EKT SPI values) can be used for each media line. If MIKEY is used on session and media level, the media level content overrides the session level content.

EKT requires a single shared SRTP master salt between all participants in the direct SRTP session. If a MIKEY key-mgmt attribute contains more than one TKG or TEK Key Data sub-payload, all the sub-payloads MUST contain the same master salt value. Consequently, the EKT\_Key Key Data sub-payload MAY also contain the same salt or MAY omit the salt value. If the SRTP master salt is not present in the TKG and TEK Key Data sub-payloads, the EKT\_Key sub-payload MUST contain a master salt.

#### 5.2.2. Generating the Initial Answer

For each media line in the offer using MIKEY, provided on session or/and on media level, the answerer examines the related MIKEY key-mgmt attributes for the presence of EKT parameters. In order to accept the offered key-mgmt attribute, the MIKEY message MUST contain one EKT\_Key Key Data sub-payload and the EKT\_Cipher Security Policy payload. The answerer examines also the existence of a SRTP master salt in the TGK/TEK and/or the EKT\_Key sub-payloads. If multiple salts are available, all values MUST be equal. If the salt values differ or no salt is present, the key-mgmt attribute MUST be considered as invalid.

The MIKEY responder message in the SDP answer does not contain a MIKEY KEMAC or Security Policy payload and consequently does not contain any EKT parameters. If the key-mgmt attribute for a media line was accepted by the answerer, the EKT parameter set of the offerer is valid for both directions of the SRTP session.

#### 5.2.3. Processing the Initial Answer

On reception of the answer, the offerer examines if EKT has been accepted for the offered media lines. If a MIKEY key-mgmt attribute is received containing a valid MIKEY responder message, EKT has been successfully negotiated. On receipt of a MIKEY error message, EKT negotiation has failed. For example, this may happen if an EKT extended MIKEY initiator message is sent to a MIKEY entity not supporting EKT. A MIKEY error code 'Invalid SP' or 'Invalid DT' is returned to indicate that the EKT\_Cipher Security Policy payload or the EKT\_Key sub-payload is not supported. In this case, the offerer may send a second SDP offer with a MIKEY key-mgmt attribute without the additional EKT extensions.

This behavior can be improved by defining an additional key-mgmt prtcl-id value 'mikeyekt' and offering two key-mgmt SDP attributes. One attribute offers MIKEY together with EKT and the other one offers MIKEY without EKT. This is for further discussion.

#### 5.2.4. Modifying the Session

Once a SRTP stream has been established, a new offer/answer exchange can modify the session including the EKT parameters. If the EKT key or EKT cipher is modified (i.e., a new EKT parameter set is created) the offerer MUST also provide a new EKT SPI value. The offerer MUST NOT remap an existing EKT SPI value to a new EKT parameter set. Similar, a modification of the SRTP Security Policy leads to a new EKT parameter set and requires a fresh EKT SPI, even the EKT key or cipher did not change.

Once EKT is being used, the SRTP master salt for the SRTP session MUST NOT be changed. The salt in the Key Data sub-payloads within the subsequent offers MUST be the same as the one already used.

After EKT has been successfully negotiated for a session and a SRTP master key has been transported by EKT, it is difficult to switch back to a pure MIKEY based key exchange in a synchronized way. Therefore, once EKT is being used for a session, EKT MUST be used also in all subsequent offer/answer exchanges for that session.

## 6. Using EKT for interoperability between key management systems

A media gateway (MGW) can provide interoperability between an SRTP-EKT endpoint and a non-EKT SRTP endpoint. When doing this function, the MGW can perform non-cryptographic transformations on SRTP packets outlined above. However, there are some uses of cryptography that will be required for that gateway. If a new SRTP master key is communicated to the MGW (via EKT from the EKT leg, or via Security Descriptions from the Security Descriptions leg), the MGW needs to convert that information for the other leg, and that process will incur some cryptographic operations. Specifically, if the new key arrived via EKT, that must be decrypted and then sent in Security Descriptions; likewise, if a new key arrives via Security Descriptions that must be encrypted via EKT and sent in SRTP/SRTCP.

Additional non-normative information can be found in Appendix A.

## 7. Design Rationale

From [RFC3550], a primary function of RTCP is to carry the CNAME, a "persistent transport-level identifier for an RTP source" since "receivers require the CNAME to keep track of each participant." EKT works in much the same way, using SRTCP to carry information needed for the proper processing of the SRTTP traffic.

With EKT, SRTTP gains the ability to synchronize the creation of cryptographic contexts across all of the participants in a single session. This feature provides some, but not all, of the functionality that is present in IKE phase two (but not phase one). Importantly, EKT does not provide a way to indicate SRTTP options.

With EKT, external signaling mechanisms provide the SRTTP options and the EKT Key, but need not provide the key(s) for each individual SRTTP source. EKT provides a separation between the signaling mechanisms and the details of SRTTP. The signaling system need not coordinate all SRTTP streams, nor predict in advance how many streams will be present, nor communicate SRTTP-level information (e.g., rollover counters) of current sessions.

EKT is especially useful for multi-party sessions, and for the case where multiple RTP sessions are sent to the same destination transport address (see the example in the definition of "RTP session" in [RFC3550]). A SIP offer that is forked in parallel (sent to multiple endpoints at the same time) can cause multiple RTP sessions to be sent to the same transport address, making EKT useful for use with SIP.

EKT can also be used in conjunction with a scalable group-key management system like GDOI [RFC3547]. Such a system provides a secure entity authentication method and a way to revoke group membership, both of which are out of scope of EKT.

It is natural to use SRTCP to transport encrypted keying material for SRTTP, as it provides a secure control channel for (S)RTP. However, there are several different places in SRTCP in which the encrypted SRTTP master key and ROC could be conveyed. We briefly review some of the alternatives in order to motivate the particular choice used in this specification. One alternative is to have those values carried as a new SDESC item or RTCP packet. This would require that the normal SRTCP encryption be turned off for the packets containing that SDESC item, since on the receiver's side, SRTCP processing completes before the RTCP processing starts. This tension between encryption and the desire for RTCP privacy is highly undesirable. Additionally, this alternative makes SRTCP dependent upon the parsing of the RTCP compound packet, which adds complexity. It is simpler to carry the

encrypted key in a new SRTCP field. One way to do this and to be backwards compatible with the existing specification is to define a new crypto function that incorporates the encrypted key. We define a new authentication transform because EKT relies on the normal SRTCP authentication to provide implicit authentication of the encrypted key.

An SRTP packet containing an SSRC that has not been seen will be discarded. This practice may induce a burst of packet loss at the outset of an SRTP stream, due to the loss or reorder of the first SRTCP packet with the EKT containing the key and rollover counter for that stream. However, this practice matches the conservative RTP memory-allocation strategy; many existing applications accept this risk of initial packet loss. Alternatively, implementations may wish to delay discarding such packets for a short period of time as described in Section 2.4.

The main motivation for the use of the variable-length format is bandwidth conservation. If EKT is used of SRTP, there will be a loss of bandwidth due to the additional 24 bytes in each RTP packet. For some applications, this bandwidth loss is significant.

#### 7.1. Alternatives

In its current design, EKT requires that the Master Salt be established out of band. That requirement is undesirable. In an offer/answer environment, it forces the answerer to re-use the same Master Salt value used by the offerer. The Master Salt value could be carried in EKT packets though that would consume yet more bandwidth.

In some scenarios, two SRTP sessions may be combined into a single session. When using EKT in such sessions, it is desirable to have an SPI value that is larger than 15 bits, so that collisions between SPI values in use in the two different sessions are unlikely (since each collision would confuse the members of one of the sessions.)

An alternative that addresses both of these needs is as follows: the SPI value can be lengthened from 15 bits to 63 bits, and the Master Salt can be identical to, or constructed from, the SPI value. SRTP conventionally uses a 14-byte Master Salt, but shorter values are acceptable. This alternative would add six bytes to each EKT packet; that overhead may be a reasonable tradeoff for addressing the problems outlined above.

## 8. Security Considerations

With EKT, each SRTP sender and receiver can generate distinct SRTP master keys. This property avoids any security concern over the re-use of keys, by empowering the SRTP layer to create keys on demand. Note that the inputs of EKT are the same as for SRTP with key-sharing: a single key is provided to protect an entire SRTP session. However, EKT provides complete security, even in the absence of further out-of-band coordination of SSRCs, and even when SSRC values collide.

In order to avoid potential security issues, the SRTP authentication tag length used by the base authentication method MUST be at least ten octets.

The presence of the SSRC in the EKT\_Plaintext ensures that an attacker cannot substitute an EKT\_Ciphertext from one SRTP stream into another SRTP stream, even if those two streams are using the same SRTP master key. This is important because some applications may use the same master key for multiple streams.

An attacker who strips a Full\_EKT\_Field from an SRTP packet may prevent the intended receiver of that packet from being able to decrypt it. This is a minor denial of service vulnerability. Similarly, an attacker who adds a Full\_EKT\_Field can disrupt service.

An attacker could send packets containing either Short EKT Tag or Full EKT Tag, in an attempt to consume additional CPU resources of the receiving system. In the case of the Short EKT Tag, this field is stripped and normal SRTP or SRTCP processing is performed. In the case of the Full EKT Tag, the attacker would have to have guessed or otherwise determined the SPI being used by the receiving system. If an invalid SPI is provided by the attacker, processing stops. If a valid SPI is provided by the attacker, the receiving system will decrypt the EKT ciphertext and return an authentication failure (Step 3 of Section 2.2.2).

An attacker learns from EKT when SRTP Master Keys change.

The EKT Cipher MUST be at least as strong as the encryption and authentication operations used in SRTP.

Part of the EKT\_Plaintext is known, or easily guessable to an attacker. Thus, the EKT Cipher MUST resist known plaintext attacks. In practice, this requirement does not impose any restrictions on our choices, since the ciphers in use provide high security even when much plaintext is known.

An EKT cipher MUST resist attacks in which both ciphertexts and plaintexts can be adaptively chosen. For each randomly chosen key, the encryption and decryption functions cannot be distinguished from a random permutation and its inverse with non-negligible advantage. This must be true even for adversaries that can query both the encryption and decryption functions adaptively. The advantage is defined as the difference between the probability that the adversary will identify the cipher as such and the probability that the adversary will identify the random permutation as the cipher, when each case is equally likely.



## 9. IANA Considerations

IANA is requested to register EKT into the SRTP Session Parameter registry [iana-sdp-sdesc].

IANA is requested to register dtls-srtp-ekt and dtls-srtp-host into the att-field table of the SDP Attributes registry [iana-sdp-attr].

We request the following IANA assignments from existing MIKEY IANA tables:

- o From the Key Data payload name spaces, a value to indicate the type as the 'EKT\_Key'.
- o From the Security Policy table name space, a new value to be assigned for 'EKT' (see Figure 8).

Furthermore, we need the following two new IANA registries created, populated with the initial values in this document. New values for both of these registries can be defined via Specification Required [RFC5226].

- o EKT parameter type (initially populated with the list from Figure 9)
- o EKT cipher (initially populated with the list from Figure 10)

## 10. Acknowledgements

Thanks to Lakshminath Dondeti for assistance with earlier versions of this document. Thanks to Nermeen Ismail, Eddy Lem, and Rob Raymond for fruitful discussions and comments. Thanks to Romain Biehlmann for his encouragement to add support DTLS-SRTP-EKT key servers for multicast. Thanks to Felix Wyss for his review and comments regarding ciphers.

## 11. References

### 11.1. Normative References

- [FIPS197] "The Advanced Encryption Standard (AES)", FIPS-197 Federal Information Processing Standard.
- [I-D.ietf-tls-rfc4347-bis]  
Rescorla, E. and N. Modadugu, "Datagram Transport Layer Security version 1.2", draft-ietf-tls-rfc4347-bis-06 (work in progress), July 2011.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC3261] Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston, A., Peterson, J., Sparks, R., Handley, M., and E. Schooler, "SIP: Session Initiation Protocol", RFC 3261, June 2002.
- [RFC3264] Rosenberg, J. and H. Schulzrinne, "An Offer/Answer Model with Session Description Protocol (SDP)", RFC 3264, June 2002.
- [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.
- [RFC4563] Carrara, E., Lehtovirta, V., and K. Norrman, "The Key ID Information Type for the General Extension Payload in Multimedia Internet KEYing (MIKEY)", RFC 4563, June 2006.
- [RFC4566] Handley, M., Jacobson, V., and C. Perkins, "SDP: Session Description Protocol", RFC 4566, July 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] Andreassen, F., Baugher, M., and D. Wing, "Session Description Protocol (SDP) Security Descriptions for Media Streams", RFC 4568, July 2006.

- [RFC4648] Josefsson, S., "The Base16, Base32, and Base64 Data Encodings", RFC 4648, October 2006.
- [RFC4771] Lehtovirta, V., Naslund, M., and K. Norrman, "Integrity Transform Carrying Roll-Over Counter for the Secure Real-time Transport Protocol (SRTP)", RFC 4771, January 2007.
- [RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 5226, May 2008.
- [RFC5234] Crocker, D. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", STD 68, RFC 5234, January 2008.
- [RFC5649] Housley, R. and M. Dworkin, "Advanced Encryption Standard (AES) Key Wrap with Padding Algorithm", RFC 5649, September 2009.
- [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.

#### 11.2. Informative References

- [RFC3547] Baugher, M., Weis, B., Hardjono, T., and H. Harney, "The Group Domain of Interpretation", RFC 3547, July 2003.
- [RFC3830] Arkko, J., Carrara, E., Lindholm, F., Naslund, M., and K. Norrman, "MIKEY: Multimedia Internet KEYing", RFC 3830, August 2004.
- [RFC4301] Kent, S. and K. Seo, "Security Architecture for the Internet Protocol", RFC 4301, December 2005.
- [iana-sdp-attr]  
IANA, "SDP Parameters", 2011, <<http://www.iana.org/assignments/sdp-parameters/sdp-parameters.xml>>.
- [iana-sdp-sdesc]  
IANA, "SDP Security Descriptions", 2011, <<http://www.iana.org/assignments/sdp-security-descriptions/sdp-security-descriptions.xml>>.

## Appendix A. Using EKT to Optimize Interworking DTLS-SRTP with Security Descriptions

Today, SDP Security Descriptions [RFC4568] is used for distributing SRTP keys in several different IP PBX systems and is expected to be used by 3GPP's Long Term Evolution (LTE). The IP PBX systems are typically used within a single enterprise, and LTE is used within the confines of a mobile operator's network. A Session Border Controller is a reasonable solution to interwork between Security Descriptions in one network and DTLS-SRTP in another network. For example, a mobile operator (or an Enterprise) could operate Security Descriptions within their network and DTLS-SRTP towards the Internet.

However, due to the way Security Descriptions and DTLS-SRTP manage their SRTP keys, such an SBC has to authenticate, decrypt, re-encrypt, and re-authenticate the SRTP (and SRTCP) packets in one direction, as shown in Figure 12, below. This is computationally expensive.

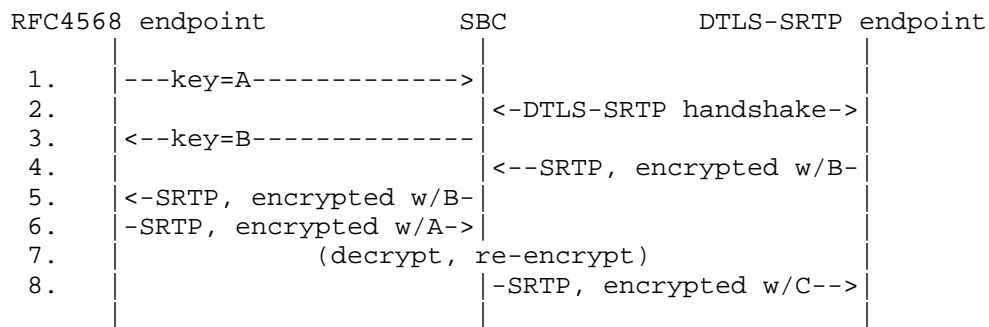


Figure 12: Interworking Security Descriptions and DTLS-SRTP

The message flow is as follows (similar steps occur with SRTCP):

1. The Security Descriptions [RFC4568] endpoint discloses its SRTP key to the SBC, using a=crypto in its SDP.
2. SBC completes DTLS-SRTP handshake. From this handshake, the SBC derives the SRTP key for traffic from the DTLS-SRTP endpoint (key B) and to the DTLS-SRTP endpoint (key C).
3. The SBC communicates the SRTP encryption key (key B) to the Security Descriptions endpoint (using a=crypto). (There is no way, with DTLS-SRTP, to communicate the Security Descriptions key to the DTLS-SRTP key endpoint.)

4. The DTLS-SRTP endpoint sends an SRTP key, encrypted with its key B. This is received by the SBC.
5. The received SRTP packet is simply forwarded; the SBC does not need to do anything with this packet as its key (key B) was already communicated in step 3.
6. The Security Descriptions endpoint sends an SRTP packet, encrypted with its key A.
7. The SBC has to authenticate and decrypt the SRTP packet (using key A), and re-encrypt it and generate an HMAC (using key C).
8. The SBC sends the new SRTP packet.

If EKT is deployed on the DTLS-SRTP endpoints, EKT helps to avoid the computationally expensive operation so the SBC does not need not perform any per-packet operations on the SRTP (or SRTCP) packets in either direction. With EKT the SBC can simply forward the SRTP (and SRTCP) packets in both directions without per-packet HMAC or cryptographic operations.

To accomplish this interworking, DTLS-SRTP EKT must be supported on the DTLS-SRTP endpoint, which allows the SBC to transport the Security Description key to the EKT endpoint and send the DTLS-SRTP key to the Security Descriptions endpoint. This works equally well for both incoming and outgoing calls. An abbreviated message flow is shown in Figure 13, below.

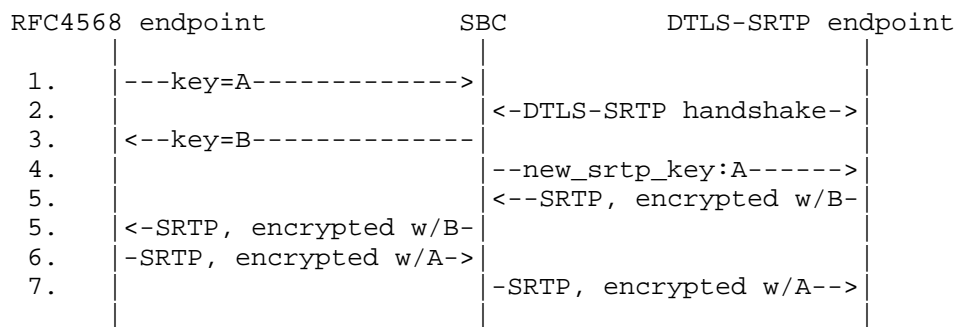


Figure 13: Interworking Security Descriptions and EKT

The message flow is as follows (similar steps occur with SRTCP):

1. Security Descriptions endpoint discloses its SRTP key to the SBC (a=crypto).

2. SBC completes DTLS-SRTP handshake. From this handshake, the SBC derives the SRTP key for traffic from the DTLS-SRTP endpoint (key B) and to the DTLS-SRTP endpoint (key C).
3. The SBC communicates the SRTP encryption key (key B) to the Security Descriptions endpoint.
4. The SBC uses the EKT to indicate that SRTP packets will be encrypted with 'key A' towards the DTLS-SRTP endpoint.
5. The DTLS-SRTP endpoint sends an SRTP key, encrypted with its key B. This is received by the SBC.
6. The received SRTP packet is simply forwarded; the SBC does not need to do anything with this packet as its key (key B) was communicated in step 3.
7. The Security Descriptions endpoint sends an SRTP packet, encrypted with its key A.
8. The received SRTP packet is simply forwarded; the SBC does not need to do anything with this packet as its key (key A) was communicated in step 4.

Authors' Addresses

David A. McGrew  
Cisco Systems, Inc.  
510 McCarthy Blvd.  
Milpitas, CA 95035  
US

Phone: (408) 525 8651  
Email: [mcgrew@cisco.com](mailto:mcgrew@cisco.com)  
URI: <http://www.mindspring.com/~dmcgrew/dam.htm>

Flemming Andreason  
Cisco Systems, Inc.  
499 Thornall Street  
Edison, NJ 08837  
US

Email: [fandreas@cisco.com](mailto:fandreas@cisco.com)

Dan Wing  
Cisco Systems, Inc.  
510 McCarthy Blvd.  
Milpitas, CA 95035  
US

Phone: (408) 853 4197  
Email: [dwing@cisco.com](mailto:dwing@cisco.com)

Kai Fischer  
Siemens Enterprise Communications GmbH & Co. KG  
Hofmannstr. 51  
Munich, Bavaria 81739  
Germany

Email: [kai.fischer@siemens-enterprise.com](mailto:kai.fischer@siemens-enterprise.com)





AVTCORE  
Internet-Draft  
Updates: 3550 (if approved)  
Intended status: Standards Track  
Expires: January 10, 2013

J. Lennox  
Vidyo  
M. Westerlund  
Ericsson  
July 9, 2012

Real-Time Transport Protocol (RTP) Considerations for Endpoints Sending  
Multiple Media Streams  
draft-lennox-avtcore-rtp-multi-stream-00

Abstract

This document expands and clarifies the behavior of the Real-Time Transport Protocol (RTP) endpoints when they are sending multiple media streams in a single RTP session. In particular, issues involving Real-Time Transport Control Protocol (RTCP) messages are described.

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 January 10, 2013.

Copyright Notice

Copyright (c) 2012 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. Terminology . . . . .	3
3. Use Cases For Multi-Stream Endpoints . . . . .	3
3.1. Multiple-Capturer Endpoints . . . . .	3
3.2. Multi-Media Sessions . . . . .	4
3.3. Multi-Stream Mixers . . . . .	4
4. Multi-Stream Endpoint RTP Media Recommendations . . . . .	4
5. Multi-Stream Endpoint RTCP Recommendations . . . . .	5
5.1. Transmission of RTCP Reception Statistics . . . . .	6
5.2. Consequences of Restricted RTCP Reception Statistics . . . . .	7
5.3. Alternate Restriction Proposal . . . . .	7
6. Security Considerations . . . . .	8
7. Open Issues . . . . .	8
8. IANA Considerations . . . . .	8
9. References . . . . .	9
9.1. Normative References . . . . .	9
9.2. Informative References . . . . .	9
Authors' Addresses . . . . .	10

## 1. Introduction

At the time The Real-Time Transport Protocol (RTP) [RFC3550] was originally written, and for quite some time after, endpoints in RTP sessions typically only transmitted a single media stream per RTP session, where separate RTP sessions were typically used for each distinct media type.

Recently, however, a number of scenarios have emerged (discussed further in Section 3) in which endpoints wish to send multiple RTP media streams, distinguished by distinct RTP synchronization source (SSRC) identifiers, in a single RTP session. Although RTP's initial design did consider such scenarios, the specification was not consistently written with such use cases in mind. The specifications are thus somewhat unclear.

The purpose of this document is to expand and clarify [RFC3550]'s language for these use cases. The authors believe this does not result in any major normative changes to the RTP specification, however this document defines how the RTP specification shall be interpreted. In these cases, this document updates RFC3550.

## 2. Terminology

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

## 3. Use Cases For Multi-Stream Endpoints

This section discusses several use cases that have motivated the development of endpoints that send multiple streams in a single RTP session.

### 3.1. Multiple-Capturer Endpoints

The most straightforward motivation for an endpoint to send multiple media streams in a session is the scenario where an endpoint has multiple capture devices of the same media type and characteristics. For example, telepresence endpoints, of the type described by the CLUE Telepresence Framework [I-D.ietf-clue-framework] is designed, often have multiple cameras or microphones covering various areas of a room.

### 3.2. Multi-Media Sessions

Recent work has been done in RTP [I-D.westerlund-avtcore-multi-media-rtp-session] and SDP [I-D.ietf-mmusic-sdp-bundle-negotiation] to update RTP's historical assumption that media streams of different media types would always be sent on different RTP sessions. In this work, a single endpoint's audio and video media streams (for example) are instead sent in a single RTP session.

### 3.3. Multi-Stream Mixers

There are several RTP topologies which can involve a central box which itself generates multiple media streams in a session.

One example is a mixer providing centralized compositing for a multi-capturer scenario like the one described in Section 3.1. In this case, the centralized node is behaving much like a multi-capturer endpoint, generating several similar and related sources.

More complicated is the Source Projecting Mixer, which is a central box that receives media streams from several endpoints, and then selectively forwards modified versions of some of the streams toward the other endpoints it is connected to. Toward one destination, a separate media source appears in the session for every other source connected to the mixer, "projected" from the original streams, but at any given time many of them may appear to be inactive (and thus receivers, not senders, in RTP). This box is an RTP mixer, not an RTP translator, in that it terminates RTCP reporting about the mixed streams, and it can re-write SSRCs, timestamps, and sequence numbers, as well as the contents of the RTP payloads, and can turn sources on and off at will without appearing to be generating packet loss. Each projected stream will typically preserve its original RTCP source description (SDS) information.

## 4. Multi-Stream Endpoint RTP Media Recommendations

While an endpoint MUST (of course) stay within its share of the available session bandwidth, as determined by signalling and congestion control, this need not be applied independently or uniformly to each media stream. In particular, session bandwidth MAY be reallocated among an endpoint's media streams, for example by varying the bandwidth use of a variable-rate codec, or changing the codec used by the media stream, up to the constraints of the session's negotiated (or declared) codecs. This includes enabling or disabling media streams as more or less bandwidth becomes available.

## 5. Multi-Stream Endpoint RTCP Recommendations

The Real-Time Transport Control Protocol (RTCP) is defined in Section 6 of [RFC3550], but it is largely documented in terms of "participants". For multi-media-stream endpoints, it is generally most useful to interpret the specification such that each media stream is a separate "participant".

For each of an endpoint's media streams, whether or not it is currently being sent, SR/RR and SDES packets MUST be sent at least once per RTCP report interval. (For discussion of the content of SR or RR packets' reception statistic reports, see Section 5.1.)

When a new media stream is added to a unicast session, the sentence in [RFC3550]'s Section 6.2 applies: "For unicast sessions ... the delay before sending the initial compound RTCP packet MAY be zero." Thus, endpoints MAY send an initial RTCP packet for the media stream immediately upon adding to the session.

Similarly, [RFC3550] Section 6.1 gives the following advice to RTP translators and mixers:

It is RECOMMENDED that translators and mixers combine individual RTCP packets from the multiple sources they are forwarding into one compound packet whenever feasible in order to amortize the packet overhead (see Section 7). An example RTCP compound packet as might be produced by a mixer is shown in Fig. 1. If the overall length of a compound packet would exceed the MTU of the network path, it SHOULD be segmented into multiple shorter compound packets to be transmitted in separate packets of the underlying protocol. This does not impair the RTCP bandwidth estimation because each compound packet represents at least one distinct participant. Note that each of the compound packets MUST begin with an SR or RR packet.

Note: To avoid confusion, an RTCP packet is an individual item, such as an Sender Report (SR), Receiver Report (RR), Source Description (SDES), Goodbye (BYE), Application Defined (APP), Feedback [RFC4585] or Extended Report (XR) [RFC3611] packet. A compound packet is the combination of two or more such RTCP packets where the first packet must be an SR or an RR packet, and which contains a SDES packet containing a CNAME item. Thus the above results in compound RTCP packets that contain multiple SR or RR packets from different sources as well as any of the other packet types. There are no restrictions on the order the packets may occur within the compound packet, except the regular compound rule, i.e. starting with an SR or RR.

This advice applies to multi-media-stream endpoints as well, with the same restrictions and considerations. (Note, however, that the last

sentence does not apply to AVPF [RFC4585] or SAVPF [RFC5124] feedback packets if Reduced-Size RTCP [RFC5506] is in use.)

Open Issue: Any clarifications on how one handle the scheduling of RTCP transmissions when having multiple sources? Alternatives include delaying one source to the next source's transmission, or to group multiple sources to use only one scheduling.

#### 5.1. Transmission of RTCP Reception Statistics

As required by [RFC3550], an endpoint MUST send reception reports about every active media stream it is receiving, from at least one local source.

However, a naive application of the RTP specification's rules could be quite inefficient. In particular, if a session has  $N$  media sources (active and inactive), and had  $S$  senders in each reporting interval, there would either be  $N*S$  report blocks per reporting interval, or (per the round-robinning recommendations of [RFC3550] Section 6.1) reception sources would be unnecessarily round-robbined. In a session where most media sources become senders reasonably frequently, this results in quadratically many reception report blocks in the conference, or reporting delays proportional to the number of session members.

Since traffic is received by endpoints, however, rather than by media sources, there is not actually any need for this quadratic expansion. All that is needed is for each endpoint to report all the remote sources it is receiving.

Thus, an endpoint SHOULD NOT send reception reports from one of its own media sources about another one of its own ("self-reports"). Similarly, an endpoint with multiple media sources SHOULD NOT send reception reports about a remote media source from more than one of its local sources ("cross-reports"). Instead, it SHOULD pick one of its local media sources as the "reporting" source for each remote media source, and use it to send reception reports for that remote source; all its other media sources SHOULD NOT send any reception reports for that remote media source.

An endpoint MAY choose different local media sources as the reporting source for different remote media sources (for example, it could choose to send reports about remote audio sources from its local audio source, and reports about remote video sources from its local video source), or it MAY choose a single local source for all its reports. If the reporting source leaves the session (sends BYE), another reporting source MUST be chosen. This "reporting" source SHOULD also be the source for any AVPF feedback messages about its

remote sources, as well.

## 5.2. Consequences of Restricted RTCP Reception Statistics

The RTCP traffic generated by receivers following the rules in Section 5.1 might appear, to observers unaware of the recommendations of this specification or knowledge about which end-points are associated with which SSRCs, to be generated by receivers who are experiencing a network disconnection.

This could be a potentially critical problem when one uses RTCP for congestion control, as a sender might think that it is sending so much traffic that it is causing complete congestion collapse. At the same time, however, a congestion control solution is likely not interested in performing unnecessary processing based on multiple reporting sources having identical statistics. A congestion control algorithm is likely more interested in frequent reporting from one specific source than multiple sources at the same end-point based on common statistics. That would reduce the uncertainty that sources are from the same end-point, and likely improve the interarrival time of the reporting, compared to multiple SSRCs which, by the RTCP algorithm, are deliberately desynchronized. However, this would clearly require clarifications on how the RTCP timer rules are to be treated.

However, such an interpretation of the session statistics would require a fairly sophisticated RTCP analysis. Any receiver of RTCP statistics which is just interested in information about itself needs to be prepared that any given reception report might not contain information about a specific media source, because reception reports in large conferences can be round-robin.

Thus, it is unclear to what extent this restriction would actually cause trouble in practice.

## 5.3. Alternate Restriction Proposal

If there are indeed scenarios in which the rules of Section 5.1 do cause troubles, an alternative solution would be to explicitly signal, in RTCP, which groups of media sources originate from a single endpoint. Thus, within a group of sources, receivers could know that there would not be self-reports, and only a single SSRC would be providing cross-reports. In such a mode, the signaling protocol would need to negotiate, or declare, that the mode was in use.

The next question would be to determine how to indicate the groups of sources for this purpose. The sources' CNAMEs would probably not be



sufficient, as some of the use cases described in Section 3, notably the source-projecting mixer, result in a single endpoint generating sources with multiple CNAME values. Thus, a new SDES item would be needed for these purposes.

TBD: If this solution is indeed taken, define the specifics of this SDES item, and the signaling needed to indicate its use.

## 6. Security Considerations

In the secure RTP protocol (SRTP) [RFC3711], the cryptographic context of a compound SRTCP packet is the SSRC of the sender of the first RTCP (sub-)packet. This could matter in some cases, especially for keying mechanisms such as Mikey [RFC3830] which use per-SSRC keying.

Other than that, the standard security considerations of RTP apply; sending multiple media streams from a single endpoint does not appear to have different security consequences than sending the same number of streams.

## 7. Open Issues

At this stage this document contains a number of open issues. The below list tries to summarize the issues:

1. Any clarifications on how to handle the RTCP scheduler when sending multiple sources in one compound packet.
2. Shall suppression of self-reporting, i.e. reporting one's other SSRCs in any SR/RR, be applied?
3. Shall suppression of cross-reporting be used, i.e. each end-point uses only one SSRC to report on any non-local SSRCs being received? If so what method should be applied:
  1. Implicit, by just not report using any other SSRC
  2. Explicit binding of SSRCs that are being commonly reported, either using SDES or another packet type, to explicitly indicate the SSRCs on whose behalf the report applies.
  3. Add any specific RTCP scheduler considerations.

## 8. IANA Considerations

This document makes no requests of IANA.

Note to the RFC Editor: please remove this section before publication.

(Note: This section may change if the alternative proposal of Section 5.3 is adopted.)

## 9. References

### 9.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC3550] Schulzrinne, H., Casner, S., Frederick, R., and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications", STD 64, RFC 3550, July 2003.
- [RFC3711] Baugher, M., McGrew, D., Naslund, M., Carrara, E., and K. Norrman, "The Secure Real-time Transport Protocol (SRTP)", RFC 3711, March 2004.
- [RFC4585] Ott, J., Wenger, S., Sato, N., Burmeister, C., and J. Rey, "Extended RTP Profile for Real-time Transport Control Protocol (RTCP)-Based Feedback (RTP/AVPF)", RFC 4585, July 2006.
- [RFC5124] Ott, J. and E. Carrara, "Extended Secure RTP Profile for Real-time Transport Control Protocol (RTCP)-Based Feedback (RTP/SAVPF)", RFC 5124, February 2008.
- [RFC5506] Johansson, I. and M. Westerlund, "Support for Reduced-Size Real-Time Transport Control Protocol (RTCP): Opportunities and Consequences", RFC 5506, April 2009.

### 9.2. Informative References

- [I-D.ietf-clue-framework]  
Romanow, A., Duckworth, M., Pepperell, A., and B. Baldino, "Framework for Telepresence Multi-Streams", draft-ietf-clue-framework-06 (work in progress), July 2012.
- [I-D.ietf-mmusic-sdp-bundle-negotiation]  
Holmberg, C. and H. Alvestrand, "Multiplexing Negotiation Using Session Description Protocol (SDP) Port Numbers", draft-ietf-mmusic-sdp-bundle-negotiation-00 (work in progress), February 2012.
- [I-D.westerlund-avtcore-multi-media-rtp-session]  
Westerlund, M., Perkins, C., and J. Lennox, "Multiple

Media Types in an RTP Session",  
draft-westerlund-avtcore-multi-media-rtp-session-00 (work  
in progress), July 2012.

[RFC3611] Friedman, T., Caceres, R., and A. Clark, "RTP Control  
Protocol Extended Reports (RTCP XR)", RFC 3611,  
November 2003.

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

#### Authors' Addresses

Jonathan Lennox  
Vidyo, Inc.  
433 Hackensack Avenue  
Seventh Floor  
Hackensack, NJ 07601  
US

Email: [jonathan@vidyo.com](mailto:jonathan@vidyo.com)

Magnus Westerlund  
Ericsson  
Farogatan 6  
SE-164 80 Kista  
Sweden

Phone: +46 10 714 82 87  
Email: [magnus.westerlund@ericsson.com](mailto:magnus.westerlund@ericsson.com)



Network Working Group  
Internet-Draft  
Updates: 6222 (if approved)  
Intended status: Standards Track  
Expires: January 10, 2013

E. Rescorla  
RTFM, Inc.  
July 09, 2012

Random algorithm for RTP CNAME generation  
draft-rescorla-avtcore-random-cname-00

Abstract

RFC 6222 describes a number of mechanisms for generating a unique CNAME. Unfortunately, these algorithms are rather complicated and also produce CNAMEs which in some cases are potentially linkable over multiple RTCP sessions even if a new CNAME is generated for each session. This document specifies a replacement algorithm for the algorithm in Section 5 which does not have this limitation and is also simpler to implement.

Legal

THIS DOCUMENT AND THE INFORMATION CONTAINED THEREIN ARE PROVIDED ON AN "AS IS" BASIS AND THE CONTRIBUTOR, THE ORGANIZATION HE/SHE REPRESENTS OR IS SPONSORED BY (IF ANY), THE INTERNET SOCIETY, THE IETF TRUST, AND THE INTERNET ENGINEERING TASK FORCE, DISCLAIM ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION THEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

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 January 10, 2013.

## Copyright Notice

Copyright (c) 2012 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.

This document may contain material from IETF Documents or IETF Contributions published or made publicly available before November 10, 2008. The person(s) controlling the copyright in some of this material may not have granted the IETF Trust the right to allow modifications of such material outside the IETF Standards Process. Without obtaining an adequate license from the person(s) controlling the copyright in such materials, this document may not be modified outside the IETF Standards Process, and derivative works of it may not be created outside the IETF Standards Process, except to format it for publication as an RFC or to translate it into languages other than English.

## Table of Contents

1. Terminology . . . . .	4
2. Introduction . . . . .	4
2.1. Linkability of the RFC 6222 algorithm . . . . .	5
3. Alternative Algorithm . . . . .	6
3.1. Comparison to RFC 6222 Algorithm . . . . .	6
3.1.1. Ease of implementation . . . . .	6
3.1.2. Format . . . . .	6
3.1.3. Uniqueness . . . . .	7
3.1.4. Linkability . . . . .	7
4. Security Considerations . . . . .	7
5. References . . . . .	7
5.1. Normative References . . . . .	7
5.2. Informative References . . . . .	8
Author's Address . . . . .	8

## 1. Terminology

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

## 2. Introduction

[RFC6222] defines a set of algorithms for generating unique RTP CNAMEs [RFC3550]. Although these algorithms attempt to provide some privacy, the CNAMEs they generate are still potentially linkable, as acknowledged in the security considerations section of RFC 6222. This document describes a simpler algorithm which produces an identifier which is compatible with RFC 6222 identifiers and is in fact indistinguishable from them without significant computational effort,

RFC 6222 Section 4.2 requires:

" An RTP endpoint that wishes to generate a per-session RTCP CNAME MUST use the following method:

- o For every new RTP session, a new CNAME is generated following the procedure described in Section 5. After performing that procedure, the least significant 96 bits are used to generate an identifier (to compromise between packet size and security), which is converted to ASCII using Base64 encoding [RFC4648]. This results in a 16-octet string representation. The RTCP CNAME cannot change over the life of an RTP session [RFC3550]; hence, only the initial SSRC value chosen by the endpoint is used. The "user@" part of the RTCP CNAME is omitted when generating per-session RTCP CNAMEs."

The algorithm in Section 5 of RFC 6222 is a cryptographic hash of the following input values:

- o The current time in 64-bit NTP format
- o An EUI-64 or 48-bit MAC address [RFC4291].
- o The initial SSRC and source and destination address/port quartets

The result of this process is a random-appearing binary value which can then be converted to a CNAME by the process described above. Unfortunately, in many settings the input values do not provide sufficient entropy, thus making it possible to determine if multiple CNAME values were generated on the same machine.



### 2.1. Linkability of the RFC 6222 algorithm

While the output of the RFC6222 algorithm is with high probability unique, it is not clearly unlinkable. Consider the case where we have two CNAMEs C1 and C2 and we wish to determine whether they were generated by the same endpoint. This situation might occur if multiple calls were made from some anonymous location like a domestic violence shelter. For instance, the attacker receives a call from an unknown location and then calls a number of candidate locations in an attempt to determine if they are the same. Starting with C1, the attacker exhaustively searches all the potential input values to find a set which hashes to C1. He then can simply search the nearby input space and if the result is C2, he knows that the calls involve the same endpoint.

The complexity of this attack is directly related to the entropy of the input variables. At minimum the attacker knows:

- o The destination IP address and port exactly.
- o The timestamp (from the RTP header) to within a few seconds. With a typical 100 ticks/second clock, this represents about 10 bits of entropy at most (and potentially more like 2-3 bits)
- o The SSRC (from the RTP header).

This leaves the primary sources of entropy as the source IP address/port and the MAC/EUI-64 address. RFC 6222 is unclear on which IP address/port is to be used, but there are three main possibilities:

- o A relayed address/port (known to the attacker) by looking at the RTP. [Note we are assuming that a media relay is used otherwise linkability is trivial.]
- o The local IP address (most likely chosen from a very small number of local addresses in the the 10.0.x.x. or 192.168.x.x range.). As residential NATs generally assign addresses in sequence and phones are often the first item to reboot addresses 10.0.0.1, 192.168.0.1, and 192.168.1.1 are very common with the first 5 addresses in each range representing a large fraction of all devices.
- o The public IP address of the peer--hard to guess but easy to determine with a scan and not really a natural choice.

Similarly, the port in use is often not chosen randomly but often from a small set of initial ports chosen by the implementation (by default Cisco devices often use 16000-16004. Thus, while in principle there are 48 bits of randomness in the IP and port, in practice they may offer no entropy (in the case where the relayed address is used as the RFC 6222 input) or only 7 bits (where the local address is used but the client is behind a residential NAT and

uses a limited port range.)

Similarly, while in principle the MAC address has 48 bits of entropy, in practice devices are easily fingerprinted and once the manufacturer is known, the MAC address is restricted to the much narrower range assigned to the manufacturer, which are again often assigned in sequence (on the order of 20-32 bits).

Thus, in order to mount the initial attack, the attacker need search somewhere between 20-30 bits (if the relayed address is known) and 70 bits. On the upper end, there is no real linkability problem, but on the lower end linkability is practical. The lower-end case is relevant to many residential and small business settings (exactly the kind operated by DV shelters) with "natural" implementations of RFC 6222.

### 3. Alternative Algorithm

In this document, we propose an alternative approach based on simply generating a cryptographically pseudorandom value. Implementations conformant with this specification MAY replace the algorithm in Section 5 with a random value generated using a cryptographic random number generator [RFC4096]. This value MUST be at least 96 bits but MAY be longer (see Section 4 for analysis of the length).

#### 3.1. Comparison to RFC 6222 Algorithm

##### 3.1.1. Ease of implementation

The biggest bottleneck to implementation of this algorithm is the availability of an appropriate cryptographically secure PRNG (CSPRNG). In any setting which already has a secure PRNG, this algorithm described is far simpler, and many implementations already have this capability. SIP stacks [RFC3261] are required to use cryptographically random numbers to generate To and From tags (Section 19.3). RTCWEB implementations [I-D.ietf-rtcweb-security-arch] will need to have secure PRNGs to implement ICE [RFC5245] and DTLS-SRTP [RFC5764]. And of course essentially every Web browser already supports TLS, which requires a secure PRNG.

##### 3.1.2. Format

The output produced by this algorithm is a string of random bits. If it is of length 96 bits, it is indistinguishable from the output of the RFC6222 algorithm without significant computation (see Section 2.1).

### 3.1.3. Uniqueness

One concern that is often raised whenever random numbers are proposed is that of uniqueness. However, for the purposes of statistical uniqueness, the RFC6222 algorithm has equivalent properties to a PRNG, since the chance of the hashes of any two arbitrarily chosen strings colliding are the same as those of any two random strings colliding (or else this constitutes a weakness in the hash.)

### 3.1.4. Linkability

A basic design criterion of a good CSPRNG is that it not be possible to distinguish its output from random values. Clearly, identifying two outputs as being from the same CSPRNG would violate this requirement. In order to mount the attack described in Section 2.1 would require exhaustively searching the seed space of the PRNG. Any conditions under which this was practical would represent a severe threat to the security of the CSPRNG if used in any communications security setting.

## 4. Security Considerations

The privacy properties of the algorithm described here are as strong or stronger than those of the RFC6222 algorithm. Because of the properties of the PRNG, there is no significant privacy/linkability difference between long and short CNAMEs. However, the requirement to generate unique CNAMEs implies a certain minimum length. A length of 96-bits allows on the order of  $2^{40}$  CNAMEs globally before there is a large chance of collision (there is about a 50% chance of one collision after  $2^{48}$  CNAMEs).

## 5. References

### 5.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC3550] Schulzrinne, H., Casner, S., Frederick, R., and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications", STD 64, RFC 3550, July 2003.
- [RFC4096] Malamud, C., "Policy-Mandated Labels Such as "Adv:" in Email Subject Headers Considered Ineffective At Best", RFC 4096, May 2005.

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

## 5.2. Informative References

- [I-D.ietf-rtcweb-security-arch]  
Rescorla, E., "RTCWEB Security Architecture",  
draft-ietf-rtcweb-security-arch-02 (work in progress),  
June 2012.
- [RFC3261] Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston, A., Peterson, J., Sparks, R., Handley, M., and E. Schooler, "SIP: Session Initiation Protocol", RFC 3261, June 2002.
- [RFC4291] Hinden, R. and S. Deering, "IP Version 6 Addressing Architecture", RFC 4291, February 2006.
- [RFC5245] Rosenberg, J., "Interactive Connectivity Establishment (ICE): A Protocol for Network Address Translator (NAT) Traversal for Offer/Answer Protocols", RFC 5245, 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.

## Author's Address

Eric Rescorla  
RTFM, Inc.  
2064 Edgewood Drive  
Palo Alto, CA 94303  
USA

Phone: +1 650 678 2350  
Email: [ekr@rtfm.com](mailto:ekr@rtfm.com)



AVTCORE WG  
Internet-Draft  
Updates: 3550 (if approved)  
Intended status: Standards Track  
Expires: January 10, 2013

M. Westerlund  
Ericsson  
C. Perkins  
University of Glasgow  
J. Lennox  
Vidyo  
July 9, 2012

Multiple Media Types in an RTP Session  
draft-westerlund-avtcore-multi-media-rtp-session-00

Abstract

This document specifies how an RTP session can contain media streams with media from multiple media types such as audio, video, and text. This has been restricted by the RTP Specification, and thus this document updates RFC 3550 to enable this behavior for applications that satisfy the applicability for using multiple media types in a single RTP session.

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 January 10, 2013.

Copyright Notice

Copyright (c) 2012 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. Definitions . . . . .	3
2.1. Requirements Language . . . . .	4
2.2. Terminology . . . . .	4
3. Motivation . . . . .	4
3.1. NAT and Firewalls . . . . .	4
3.2. No Transport Level QoS . . . . .	5
3.3. Architectural Equality . . . . .	5
4. Overview of Solution . . . . .	5
5. Applicability . . . . .	6
5.1. Usage of the RTP session . . . . .	6
5.2. Signalled Support . . . . .	6
5.3. Homogeneous Multi-party . . . . .	7
5.4. Reduced number of Payload Types . . . . .	8
5.5. Stream Differentiation . . . . .	8
5.6. Non-compatible Extensions . . . . .	8
6. RTP Session Specification . . . . .	9
6.1. RTP Session . . . . .	9
6.2. Sender Source Restrictions . . . . .	10
6.3. Payload Type Applicability . . . . .	10
6.4. RTCP . . . . .	11
7. Extension Considerations . . . . .	11
7.1. RTP Retransmission . . . . .	12
7.2. Generic FEC . . . . .	12
8. Signalling . . . . .	12
8.1. SDP-Based Signalling . . . . .	13
9. IANA Considerations . . . . .	13
10. Security Considerations . . . . .	13
11. Acknowledgements . . . . .	13
12. References . . . . .	14
12.1. Normative References . . . . .	14
12.2. Informative References . . . . .	14
Authors' Addresses . . . . .	15

## 1. Introduction

When the Real-time Transport Protocol (RTP) [RFC3550] was designed, close to 20 years ago, IP networks were very different compared to the ones in 2012 when this is written. The almost ubiquitous deployment of Network Address Translators (NAT) and Firewalls has increased the cost and likely-hood of communication failure when using many different transport flows. Thus there exists a pressure to reduce the number of concurrent transport flows.

RTP [RFC3550] as defined recommends against having multiple media types, like audio and video, in the same RTP session. The motivation for this is dependent on particular usage or dependencies on lower layer Quality of Service (QoS). When these aren't present, there are no strong RTP reasons for not allowing multiple media types in one RTP session. However, the Session Description Protocol (SDP) [RFC4566], as one of the dominant signalling method for establishing RTP session, has enforced this rule, by not allowing multiple media types for a given receiver destination or set of ICE candidates, which is the most common method to determine which RTP session the packets are intended for.

The fact that these limitations have been in place for so long a time, in addition to RFC 3550 being written without fully considering multiple media types in an RTP session, does result in a number of considerations being needed. This document provides such considerations regarding applicability as well as functionality, including normative specification of behavior.

First, some basic definitions are provided. This is followed by a background that discusses the motivation in more detail. A overview of the solution of how to provide multiple media types in one RTP session is then presented. Next is the formal applicability this specification have followed by the normative specification. This is followed by a discussion how some RTP/RTCP Extensions should function in the case of multiple media types in one RTP session. A specification of the requirements on signalling from this specification and a look how this is realized in SDP using Bundle [I-D.ietf-mmusic-sdp-bundle-negotiation]. The document ends with the security considerations.

## 2. Definitions



### 2.1. Requirements Language

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

### 2.2. Terminology

The following terms are used with supplied definitions:

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

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

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

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

## 3. Motivation

This section discusses in more detail the main motivations why allowing multiple media types in the same RTP session is suitable.

### 3.1. NAT and Firewalls

The existence of NATs and Firewalls at almost all Internet access has had implications on protocols like RTP that were designed to use multiple transport flows. First of all, the NAT/FW traversal solution one uses needs to ensure that all these transport flows are established. This has three different 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/FWs reaches their limits, unexpected behaviors usually occur.
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 flows reduces the risk of communication failure, improved establishment behavior and less load on NAT and Firewalls.

### 3.2. No Transport Level QoS

Many RTP-using applications don't utilize any network level Quality of Service functions. Nor do they expect or desire any separation in network treatment of its media packets, independent of whether they are audio, video or text. When an application has no such desire, it doesn't need to provide a transport flow structure that simplifies flow based QoS.

### 3.3. Architectural Equality

For applications that don't desire any type of different treatment, neither on the transport level nor in RTP or RTCP reporting, using the same RTP session for both media types appears a reasonable choice. The architecture should be neutral to media type, rather look at what it provides based on the application users choice. Therefore this bias should be removed and let the application designer make the choice if they need multiple RTP sessions or not based on other aspects.

## 4. Overview of Solution

The goal of the solution is to enable having one or more RTP sessions, where each RTP session may contain two or more media types. This includes having multiple RTP sessions containing a given media type, for example having three sessions containing video and audio.

The solution is quite straightforward. The first step is to override the SHOULD and SHOULD NOT language of the RTP specification [RFC3550]. This is done by appropriate exception clauses given that this specification is followed.

Within an RTP session where multiple media types have been configured for use, a SSRC may only send one media type during its lifetime. Different SSRCs must be used for the different media sources, the same way multiple media sources of the same media type already have to do. The payload type will inform a receiver which media type the SSRC is being used for. Thus the payload type must be unique across all of the payload configurations independent of media type that may be used in the RTP session.

Some few extra considerations within the RTP sessions also needs to be considered. RTCP bandwidth and regular reporting suppression (AVPF and SAVPF) should be considered to be configured. Certain payload types like FEC also need additional rules.

The final important part of the solution to this is to use signalling and ensure that agreement on using multiple media types in an RTP session exists, and how that then is configured. Thus document documents some existing requirements, while an external reference defines how this is accomplished in SDP.

## 5. Applicability

This specification has limited applicability and any one intending to use must ensure that their application and usage meets the below criteria for usage.

### 5.1. Usage of the RTP session

Before choosing to use this specification, an application implementer needs to ensure that they don't have a need for different RTP sessions between the media types for some reason. The main rule is that if one expects to have equal treatment of all media packets, then this specification might be suitable. The equal treatment include anything from network level up to RTCP reporting and feedback. The document *Guidance on RTP Multiplexing Architecture* [I-D.westerlund-avtcore-multiplex-architecture] gives more detailed guidance on aspects to consider when choosing how to use RTP and specifically sessions. RTP-using applications that need or would prefer multiple RTP sessions, but do not require the functionalities or behaviors that multiple transport flows give, can consider using *Multiple RTP Sessions on a Single Lower-Layer Transport* [I-D.westerlund-avtcore-transport-multiplexing].

### 5.2. Signalled Support

Usage of this specification is not compatible with anyone following RFC 3550 and intending to have different RTP sessions for each media

type. Therefore there must be mutual agreement to use multiple media types in one RTP session by all participants within an RTP session. This agreement must in most cases be determined using signalling.

This requirement can be a problem for signalling solutions that can't negotiate with all participants. For declarative signalling solutions, mandating that the session is using multiple media types in one RTP session can be a way of attempting to ensure that all participants in the RTP session follow the requirement. However, for signalling solutions that lack methods for enforcing that a receiver supports a specific feature, this can still cause issues.

### 5.3. Homogeneous Multi-party

In multiparty communication scenarios it is important to separate two different cases. One case is where the RTP session contains multiple participants in a common RTP session. This occurs for example in Any Source Multicast (ASM) and Transport Translator topologies as defined in RTP Topologies [RFC5117]. It may also occur in some implementations of RTP mixers that share the same SSRC/CSRC space across all participants. The second case is when the RTP session is terminated in a middlebox and the other participants sources are projected or switched into each RTP session and rewritten on RTP header level including SSRC mappings.

For the first case, with a common RTP session or at least shared SSRC/CSRC values, all participants in multiparty communication are required to support multiple media types in an RTP session. An participant using two or more RTP sessions towards a multiparty session can't be collapsed into a single session with multiple media types. The reason is that in case of multiple RTP sessions, the same SSRC value can be used in both RTP sessions without any issues, but when collapsed to a single session there is an SSRC collision. In addition some collisions can't be represented in the multiple separate RTP sessions. For example, in a session with audio and video, an SSRC value used for video will not show up in the Audio RTP session at the participant using multiple RTP sessions, and thus not trigger any collision handling. Thus any application using this type of RTP session structure must have a homogeneous support for multiple media types in one RTP session, or be forced to insert a translator node between that participant and the rest of the RTP session.

For the second case of separate RTP sessions for each multiparty participant and a central node it is possible to have a mix of single RTP session users and multiple RTP session users as long as one is willing to remap the SSRCs used by a participant with multiple RTP sessions into non-used values in the single RTP session SSRC space for each of the participants using a single RTP session with multiple

media types. It can be noted that this type of implementation is required to understand any type of RTP/RTCP extension being used in the RTP sessions to correctly be able to translate them between the RTP sessions.

#### 5.4. Reduced number of Payload Types

An RTP session with multiple media types in it have only a single 7-bit Payload Type range for all its payload types. Within the 128 available values, only 96 or less if "Multiplexing RTP Data and Control Packets on a Single Port" [RFC5761] is used, all the different RTP payload configurations for all the media types must fit. For most applications this will not be a real problem, but the limitation exists and could be encountered.

#### 5.5. Stream Differentiation

If network level differentiation of the media streams of different media types are desired using this specification can cause severe limitations. All media streams in an RTP session, independent of the media type, will be sent over the same underlying transport flow. Any flow-based Quality of Service (QoS) mechanism will be unable to provide differentiated treatment between different media types, e.g. to prioritize audio over video. If that is desired, separate RTP sessions over different underlying transport flows needs to be used. Any marking-based QoS scheme like DiffServ is not affected unless a network ingress marks based on flows.

#### 5.6. Non-compatible Extensions

There exist some RTP and RTCP extensions that rely on the existence of multiple RTP sessions. If the goal of using an RTP session with multiple media types is to have only a single RTP session, then these extensions can't be used. If one has no need to have different RTP sessions for the media types but is willing to have multiple RTP sessions, one for the main media transmission and one for the extension, they can be used. It should be noted that this assumes that it is possible to get the extension working when the related RTP session contains multiple media types.

Identified RTP/RTCP extensions that require multiple RTP Sessions are:

RTP Retransmission: RTP Retransmission [RFC4588] has a session multiplexed mode. It also has a SSRC multiplexed mode that can be used instead. So use the mode that is suitable for the RTP application.

XOR-Based FEC: The RTP Payload Format for Generic Forward Error Correction [RFC5109] and its predecessor [RFC2733] requires a separate RTP session unless the FEC data is carried in RTP Payload for Redundant Audio Data [RFC2198] which has another set of restrictions.

Note that the Source-Specific Media Attributes [RFC5576] specification defines an SDP syntax (the "FEC" semantic of the "ssrc-group" attribute) to signal FEC relationships between multiple media streams within a single RTP session. However, this can't be used as the FEC repair packets are required to have the same SSRC value as the source packets being protected. [RFC5576] does not normatively update and resolve that restriction.

## 6. RTP Session Specification

This section defines what needs to be done or avoided to make an RTP session with multiple media types function without issues.

### 6.1. RTP Session

Section 5.2 of "RTP: A Transport Protocol for Real-Time Applications" [RFC3550] states:

For example, in a teleconference composed of audio and video media encoded separately, each medium SHOULD be carried in a separate RTP session with its own destination transport address.

Separate audio and video streams SHOULD NOT be carried in a single RTP session and demultiplexed based on the payload type or SSRC fields.

This specification changes both of these sentences. The first sentence is changed to:

For example, in a teleconference composed of audio and video media encoded separately, each medium SHOULD be carried in a separate RTP session with its own destination transport address, unless specification [RFCXXXX] is followed and the application meets the applicability constraints.

The second sentence is changed to:

Separate audio and video streams SHOULD NOT be carried in a single RTP session and demultiplexed based on the payload type or SSRC fields, unless multiplexed based on both SSRC and payload type and usage meets what Multiple Media Types in an RTP Session [RFCXXXX]

specifies.

RFC-Editor Note: Please replace RFCXXXX with the RFC number of this specification when assigned.

TBD: Discussion of the motivations in Section 5.2 of the RTP Specification [RFC3550].

## 6.2. Sender Source Restrictions

A SSRC in the RTP session MUST only send one media type (audio, video, text etc.) during the SSRC's lifetime. The main motivation is that a given SSRC has its own RTP timestamp and sequence number spaces. The same way that you can't send two streams of encoded audio on the same SSRC, you can't send one audio and one video encoding on the same SSRC. Each media encoding when made into an RTP stream needs to have the sole control over the sequence number and timestamp space. If not, one would not be able to detect packet loss for that particular stream. Nor can one easily determine which clock rate a particular SSRCs timestamp shall increase with.

## 6.3. Payload Type Applicability

Most Payload Types have a native media type, like an audio codec is natural belonging to the audio media type. However, there exist a number of RTP payload types that don't have a native media type. For example, transport robustification mechanisms like RTP Retransmission [RFC4588] and Generic FEC [RFC5109] inherit their media type from what they protect. RTP Retransmission is explicitly bound to the payload type it is protecting, and thus will inherit it. However Generic FEC is a excellent example of an RTP payload type that has no natural media type. The media type for what it protects is not relevant as it is the recovered RTP packets that have a particular media type, and thus Generic FEC is best categorized as an application media type.

The above discussion is relevant to what limitations exist for RTP payload type usage within an RTP session that has multiple media types. When it comes to Generic FEC, is an configured payload type allowed to be used to protect both audio SSRCs and Video SSRCs? Note a particular SSRC carrying Generic FEC will clearly only protect a specific SSRC and thus that instance is bound to the SSRC's media type. For this specific case, it appears possible to have one be applicable to both. However, in cases when the signalling is setup to enable fallback to using separate RTP sessions, then using a different media type, e.g. application, than the media being protected can create issues.

TBD: What recommendations are needed here?

#### 6.4.    RTCP

All SSRCs in an RTP session fall under the same set of RTCP configuration parameters, such as the RR and RS bandwidth and the trr-int parameter if AVPF or SAVPF is used. This means that at least the regular reporting period by, and on, a source will be equal, independent of the media type for that source. This should in most cases not be an issue, but it may result in more frequent reporting than is considered necessary for a particular media type or set of media sources. Having multiple media types in one RTP session also results in more SSRCs being present in this RTP session. This increases the amount of cross reporting between the SSRCs. From an RTCP perspective, two RTP sessions with half the number of SSRCs in each will be slightly more efficient. If someone needs either the higher efficiency due to the lesser number of SSRCs or the fact that one can't tailor RTCP usage per media type, they need to use independent RTP sessions.

When it comes to handling multiple SSRCs in an RTP session there is a clarification under discussion in Real-Time Transport Protocol (RTP) Considerations for Multi-Stream Endpoints [I-D.lennox-avtcore-rtp-multi-stream]. When it comes to configuring RTCP the need for regular periodic reporting needs to be weighted against any feedback or control messages being sent. The applications using AVPF or SAVPF are RECOMMENDED to consider setting trr-int parameter to a value suitable for the applications needs, thus potentially reducing the need for regular reporting and thus releasing more bandwidth for use for feedback or control.

Another aspect of an RTP session with multiple media types is that the used RTCP packets, RTCP Feedback Messages, or RTCP XR metrics used may not be applicable to all media types. Instead all RTP/RTCP endpoints need to correlate the media type of the SSRC being referenced in an messages/packet and only use those that apply to that particular SSRC and its media type. Signalling solutions may have shortcomings when it comes to indicate that a particular set of RTCP reports or feedback messages only apply to a particular media type within an RTP session.

#### 7.    Extension Considerations

This section discusses the impact on some RTP/RTCP extensions due to usage of multiple media types in on RTP session. Only extensions where something worth noting has been included.



### 7.1. RTP Retransmission

SSRC-multiplexed RTP retransmission [RFC4588] is actually very straightforward. Each retransmission RTP payload type is explicitly connected to an associated payload type. If retransmission is only to be used with a subset of all payload types, this is not a problem, as it will be evident from the retransmission payload types which payload types that have retransmission enabled for them.

Session-multiplexed RTP retransmission is also possible to use where an retransmission session contains the retransmissions of the associated payload types in the source RTP session. The only difference to previously is that the source RTP session is one which contains multiple media types. Thus it is even more likely that only a subset of the source RTP session's payload types and SSRCS are actually retransmitted.

Open Issue: When using SDP to signal retransmission for one RTP session with multiple media types and one RTP session for the retransmission data will cause a situation where one will have multiple m= lines grouped using FID and the ones belonging to respective RTP session being grouped using BUNDLE. This usage may contradict both the FID semantics [RFC5888] and an assumption in the RTP retransmission specification [RFC4588].

### 7.2. Generic FEC

TBW:

## 8. Signalling

The Signalling requirements

Establishing an RTP session with multiple media types requires signalling. This signalling needs to fulfill the following requirements:

1. Ensure that any participant in the RTP session is aware that this is an RTP session with multiple media types.
2. Ensure that the payload types in use in the RTP session are using unique values, with no overlap between the media types.
3. Configure the RTP session level parameters, such as RTCP RR and RS bandwidth, AVPF trr-int, underlying transport, the RTCP extensions in use, and security parameters, commonly for the RTP session.

4. RTP and RTCP functions that can be bound to a particular media type should be reused when possible also for other media types, instead of having to be configured for multiple code-points.

Note: In some cases one will not have a choice but to use multiple configurations.

#### 8.1. SDP-Based Signalling

The signalling of multiple media types in one RTP session in SDP is specified in "Multiplexing Negotiation Using Session Description Protocol (SDP) Port Numbers"

[I-D.ietf-mmusic-sdp-bundle-negotiation].

#### 9. IANA Considerations

This document makes no request of IANA.

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

#### 10. Security Considerations

Having an RTP session with multiple media types doesn't change the methods for securing a particular RTP session. One possible difference is that the different media have often had different security requirements. When combining multiple media types in one session, their security requirements must also be combined by selecting the most demanding for each property. Thus having multiple media types may result in increased overhead for security for some media types to ensure that all requirements are met.

Otherwise, the recommendations for how to configure an RTP session do not add any additional requirements compared to normal RTP, except for the need to be able to ensure that the participants are aware that it is a multiple media type session. If not that is ensured it can cause issues in the RTP session for both the unaware and the aware one. Similar issues can also be produced in a normal RTP session by creating configurations for different end-points that doesn't match each other.

#### 11. Acknowledgements

The authors would like to thank Christer Holmberg for the feedback on the document.

## 12. References

### 12.1. Normative References

- [I-D.ietf-mmusic-sdp-bundle-negotiation]  
Holmberg, C. and H. Alvestrand, "Multiplexing Negotiation Using Session Description Protocol (SDP) Port Numbers", draft-ietf-mmusic-sdp-bundle-negotiation-00 (work in progress), February 2012.
- [I-D.lennox-avtcore-rtp-multi-stream]  
Lennox, J. and M. Westerlund, "Real-Time Transport Protocol (RTP) Considerations for Multi-Stream Endpoints", July 2012.
- [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.

### 12.2. Informative References

- [I-D.westerlund-avtcore-multiplex-architecture]  
Westerlund, M., Burman, B., and C. Perkins, "RTP Multiplexing Architecture", draft-westerlund-avtcore-multiplex-architecture-01 (work in progress), March 2012.
- [I-D.westerlund-avtcore-transport-multiplexing]  
Westerlund, M. and C. Perkins, "Multiple RTP Sessions on a Single Lower-Layer Transport", draft-westerlund-avtcore-transport-multiplexing-02 (work in progress), March 2012.
- [RFC2198] Perkins, C., Kouvelas, I., Hodson, O., Hardman, V., Handley, M., Bolot, J., Vega-Garcia, A., and S. Fosse-Parisis, "RTP Payload for Redundant Audio Data", RFC 2198, September 1997.
- [RFC2733] Rosenberg, J. and H. Schulzrinne, "An RTP Payload Format for Generic Forward Error Correction", RFC 2733, December 1999.
- [RFC4566] Handley, M., Jacobson, V., and C. Perkins, "SDP: Session Description Protocol", RFC 4566, July 2006.

- [RFC4588]    Rey, J., Leon, D., Miyazaki, A., Varsa, V., and R. Hakenberg, "RTP Retransmission Payload Format", RFC 4588, July 2006.
- [RFC5109]    Li, A., "RTP Payload Format for Generic Forward Error Correction", RFC 5109, December 2007.
- [RFC5117]    Westerlund, M. and S. Wenger, "RTP Topologies", RFC 5117, January 2008.
- [RFC5576]    Lennox, J., Ott, J., and T. Schierl, "Source-Specific Media Attributes in the Session Description Protocol (SDP)", RFC 5576, June 2009.
- [RFC5761]    Perkins, C. and M. Westerlund, "Multiplexing RTP Data and Control Packets on a Single Port", RFC 5761, April 2010.
- [RFC5888]    Camarillo, G. and H. Schulzrinne, "The Session Description Protocol (SDP) Grouping Framework", RFC 5888, June 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@csp Perkins.org

Jonathan Lennox  
Vidyo, Inc.  
433 Hackensack Avenue  
Seventh Floor  
Hackensack, NJ 07601  
US

Email: [jonathan@vidyo.com](mailto:jonathan@vidyo.com)



Network Working Group  
Internet-Draft  
Intended status: Informational  
Expires: January 17, 2013

M. Westerlund  
B. Burman  
Ericsson  
C. Perkins  
University of Glasgow  
H. Alvestrand  
Google  
July 16, 2012

Guidelines for using the Multiplexing Features of RTP  
draft-westerlund-avtcore-multiplex-architecture-02

Abstract

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

Status of this Memo

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

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

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

This Internet-Draft will expire on January 17, 2013.

Copyright Notice

Copyright (c) 2012 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 . . . . .	4
2. Definitions . . . . .	5
2.1. Terminology . . . . .	5
2.2. Subjects Out of Scope . . . . .	7
3. RTP Concepts . . . . .	7
3.1. Session . . . . .	7
3.2. SSRC . . . . .	8
3.3. CSRC . . . . .	10
3.4. Payload Type . . . . .	10
4. Multiple Streams Alternatives . . . . .	12
5. RTP Topologies and Issues . . . . .	13
5.1. Point to Point . . . . .	13
5.1.1. Translators & Gateways . . . . .	14
5.2. Point to Multipoint Using Multicast . . . . .	15
5.3. Point to Multipoint Using an RTP Transport Translator . . . . .	17
5.4. Point to Multipoint Using an RTP Mixer . . . . .	18
5.4.1. Media Mixing . . . . .	19
5.4.2. Media Switching . . . . .	22
5.4.3. RTP Source Projecting . . . . .	24
5.5. Point to Multipoint using Multiple Unicast flows . . . . .	26
5.6. De-composite Endpoint . . . . .	27
6. Multiple Streams Discussion . . . . .	28
6.1. Introduction . . . . .	28
6.2. RTP/RTCP Aspects . . . . .	28
6.2.1. The RTP Specification . . . . .	29
6.2.2. Multiple SSRCs in a Session . . . . .	31
6.2.3. Handling Varying sets of Senders . . . . .	32
6.2.4. Cross Session RTCP Requests . . . . .	32
6.2.5. Binding Related Sources . . . . .	33
6.2.6. Forward Error Correction . . . . .	35
6.2.7. Transport Translator Sessions . . . . .	36
6.3. Interworking . . . . .	36
6.3.1. Types of Interworking . . . . .	36
6.3.2. RTP Translator Interworking . . . . .	36
6.3.3. Gateway Interworking . . . . .	37
6.3.4. Multiple SSRC Legacy Considerations . . . . .	38



6.4. Network Aspects . . . . .	38
6.4.1. Quality of Service . . . . .	39
6.4.2. NAT and Firewall Traversal . . . . .	39
6.4.3. Multicast . . . . .	41
6.4.4. Multiplexing multiple RTP Session on a Single Transport . . . . .	41
6.5. Security Aspects . . . . .	42
6.5.1. Security Context Scope . . . . .	42
6.5.2. Key Management for Multi-party session . . . . .	42
6.5.3. Complexity Implications . . . . .	43
7. Arch-Types . . . . .	43
7.1. Single SSRC per Session . . . . .	43
7.2. Multiple SSRCs of the Same Media Type . . . . .	45
7.3. Multiple Sessions for one Media type . . . . .	46
7.4. Multiple Media Types in one Session . . . . .	48
7.5. Summary . . . . .	49
8. Summary considerations and guidelines . . . . .	50
8.1. Guidelines . . . . .	50
9. IANA Considerations . . . . .	51
10. Security Considerations . . . . .	51
11. References . . . . .	51
11.1. Normative References . . . . .	51
11.2. Informative References . . . . .	51
Appendix A. Dismissing Payload Type Multiplexing . . . . .	55
Appendix B. Proposals for Future Work . . . . .	57
Appendix C. RTP Specification Clarifications . . . . .	57
C.1. RTCP Reporting from all SSRCs . . . . .	58
C.2. RTCP Self-reporting . . . . .	58
C.3. Combined RTCP Packets . . . . .	58
Appendix D. Signalling considerations . . . . .	58
D.1. Signalling Aspects . . . . .	59
D.1.1. Session Oriented Properties . . . . .	59
D.1.2. SDP Prevents Multiple Media Types . . . . .	60
D.1.3. Signalling Media Stream Usage . . . . .	60
Appendix E. Changes from -01 to -02 . . . . .	61
Authors' Addresses . . . . .	61

## 1. Introduction

Real-time Transport Protocol (RTP) [RFC3550] is a commonly used protocol for real-time media transport. It is a protocol that provides great flexibility and can support a large set of different applications. RTP has several multiplexing points designed for different purposes. These enable support of multiple media streams and switching between different encoding or packetization of the media. By using multiple RTP sessions, sets of media streams can be structured for efficient processing or identification. Thus the question for any RTP application designer is how to best use the RTP session, the SSRC and the payload type to meet the application's needs.

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

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

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

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

Finally, some recommendations and examples are provided.

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

## 2. Definitions

### 2.1. Terminology

The following terms and abbreviations are used in this document:

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

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

**Media Source:** The source of a stream of data of one Media Type, It can either be a single media capturing device such as a video camera, a microphone, or a specific output of a media production function, such as an audio mixer, or some video editing function. Sending data from a Media Source may cause multiple RTP sources to send multiple Media Streams.

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

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

**Media Sink:** A recipient of a Media Stream. The endpoint sinking media are Identified using one or more SSRCS. There may be more than one Media Sink for one RTP source.

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

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

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

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

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

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

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

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

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

(tbd: The terms "SSRC multiplexing" and "session multiplexing" are confusing, with unclear historical meanings; they need to be removed from this document in the interests of clarity)

## 2.2. Subjects Out of Scope

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

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

## 3. RTP Concepts

This section describes the existing RTP tools that are particularly important when discussing multiplexing of different media streams.

### 3.1. Session

The RTP Session is the highest semantic level in RTP and contains all of the RTP functionality. RTP itself has no normative statements about the relationship between different RTP sessions.

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

Position: Depending on underlying transport and signalling protocol. For example, when running RTP on top of UDP, an RTP endpoint can identify and delimit an RTP Session from other RTP Sessions through the UDP source and destination transport address, consisting of network address and port number(s).

Commonly, RTP and RTCP use separate ports and the destination transport address is in fact an address pair, but in the case of RTP/RTCP multiplex [RFC5761] there is only a single port. Another example is SDP signalling [RFC4566], where the grouping framework [RFC5888] uses an identifier per "m="-line. If there is a one-to-one mapping between "m="-line and RTP Session, that grouping framework identifier can identify a single RTP Session.

Usage: Identify separate RTP Sessions.

Uniqueness: Globally unique, but identity can only be detected by the general communication context for the specific endpoint.

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

Special Restrictions: None.

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

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

### 3.2. SSRC

An SSRC identifies a RTP source or a media sink. For end-points that both source and sink media streams its SSRCs are used in both roles. At any given time, a RTP source has one and only one SSRC - although that may change over the lifetime of the RTP source or sink. An RTP Session serves one or more RTP sources, each sending a Media Stream.

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

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

Usage: Identify individual RTP sources and media sinks within an RTP Session. Refer to individual RTP sources and media sinks in RTCP messages and SDP signalling.

Uniqueness: Randomly chosen, intended to be globally unique within an RTP Session and not dependent on network address. SSRC value collisions may occur and must be handled as specified in RTP [RFC3550].

Inter-relation: SSRC belonging to the same synchronisation context (originating from the same endpoint), within or between RTP Sessions, are indicated through use of identical SDES CNAME items in RTCP compound packets with those SSRC as originating source. SDP signalling can provide explicit SSRC grouping [RFC5576]. When CNAME is inappropriate or insufficient, there exist a few other methods to relate different SSRC. One such case is session-based RTP retransmission [RFC4588]. In some cases, the same SSRC Identifier value is used to relate streams in two different RTP Sessions, such as in Multi-Session Transmission of scalable video [RFC6190].

Special Restrictions: All RTP implementations must be prepared to use procedures for SSRC collision handling, which results in an SSRC number change. A RTP source that changes its RTP Session identifier (e.g. source transport address) during a session must also choose a new SSRC identifier to avoid being interpreted as looped source.

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

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

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

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

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

Note that a endpoint that generates more than one media type, e.g. a conference participant sending both audio and video, need not (and commonly should not) use the same SSRC value across RTP sessions.

RTCP Compound packets containing the CNAME SDP item is the designated method to bind an SSRC to a CNAME, effectively cross-correlating SSRCs within and between RTP Sessions as coming from the same endpoint. The main property attributed to SSRCs associated with the same CNAME is that they are from a particular synchronisation context and may be synchronised at playback.

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

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

### 3.3. CSRC

The Contributing Source (CSRC) is not a separate identifier, but an usage of the SSRC identifier. It is optionally included in the RTP header as list of up to 15 contributing RTP sources. CSRC shares the SSRC number space and specifies which set of SSRCs that has contributed to the RTP payload. However, even though each RTP packet and SSRC can be tagged with the contained CSRCs, the media representation of an individual CSRC is in general not possible to extract from the RTP payload since it is typically the result of a media mixing (merge) operation (by an RTP mixer) on the individual media streams corresponding to the CSRC identifiers. The exception is the case when only a single CSRC is indicated as this represent forwarding of a media stream, possibly modified. The RTP header extension for Mixer-to-Client Audio Level Indication [RFC6465] expands on the receivers information about a packet with CSRC list. Due to these restrictions, CSRC will not be considered a fully qualified multiplex point and will be disregarded in the rest of this document.

### 3.4. Payload Type

Each Media Stream utilises one or more encoding formats, identified by the Payload Type.

The Payload Type is not a multiplexing point. Appendix A gives some of the many reasons why attempting to use it as a multiplexing point will have bad results.



Identifier: Payload Type number.

Position: In every RTP header and in SDP signalling.

Usage: Identify a specific Media Stream encoding format. The format definition may be taken from [RFC3551] for statically allocated Payload Types, but should be explicitly defined in signalling, such as SDP, both for static and dynamic Payload Types. The term "format" here includes whatever can be described by out-of-band signalling means. In SDP, the term "format" includes media type, RTP timestamp sampling rate, codec, codec configuration, payload format configurations, and various robustness mechanisms such as redundant encodings [RFC2198].

Uniqueness: Scoped by sending endpoint within an RTP Session. To avoid any potential for ambiguity, it is desirable that payload types are unique across all sending endpoints within an RTP session, but this is often not true in practice. All SSRC in an RTP session sent from an single endpoint share the same Payload Types definitions. The RTP Payload Type is designed such that only a single Payload Type is valid at any time instant in the SSRC's RTP timestamp time line, effectively time-multiplexing different Payload Types if any change occurs. Used Payload Type may change on a per-packet basis for an SSRC, for example a speech codec making use of generic Comfort Noise [RFC3389].

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

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

If there is a true need to send multiple Payload Types for the same SSRC that are valid for the same RTP Timestamps, then redundant encodings [RFC2198] can be used. Several additional constraints than the ones mentioned above need to be met to enable this use, one of which is that the combined payload sizes of the different Payload Types must not exceed the transport MTU.

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

#### 4. Multiple Streams Alternatives

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

- o Multiple Media Sources
- o Multiple Media Streams may be needed to represent one Media Source (for instance when using layered encodings)
- o A Retransmission stream may repeat the content of another Media Stream
- o An FEC stream may provide material that can be used to repair another Media Stream
- o Alternative Encodings, for instance different codecs for the same audio stream
- o Alternative formats, for instance multiple resolutions of the same video stream

Thus the choice made due to one reason may not be the choice suitable for another reason. In the above list, the different items have different levels of maturity in the discussion on how to solve them. The clearest understanding is associated with multiple media sources of the same media type. However, all warrant discussion and clarification on how to deal with them.

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

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

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

As the below discussion will show, in reality we cannot choose a single one of the two solutions. To utilise RTP well and as

efficiently as possible, both are needed. The real issue is finding the right guidance on when to create RTP sessions and when additional SSRCs in an RTP session is the right choice.

## 5. RTP Topologies and Issues

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

### 5.1. Point to Point

This is the most basic use case with two endpoints directly interconnected and no additional entities are involved in the communication.

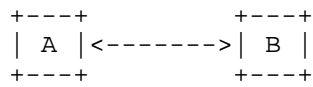


Figure 1: Point to Point

A number of applications are using a single RTP session with one RTP source per endpoint. This is likely the simplest case, as you basically doesn't have to make any choices regarding multiplexing. When you add an additional source to either endpoint you immediately create the question do one send the media stream in the existing RTP session or should I use an additional RTP session.

This raises a number of considerations that are discussed in detail below (Section 6). But the range over such aspects as:

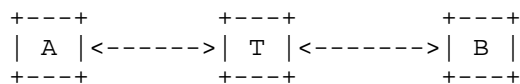
- o Does my communication peer support RTP as defined with multiple SSRCs?
- o Do I need network differentiation in form of QoS?
- o Can the application more easily process and handle the media streams if they are in different RTP sessions?

- o etc.

The application designer will have to make choices here. The point to point topology can contain one to many RTP sessions with one to many RTP sources (SSRC) per session.

#### 5.1.1. Translators & Gateways

A point to point communication can end up in a situation when the peer it is communicating with is not compatible with the other peer for various reasons. This is in many situation resolved by the inclusion of a translator in-between the two peers.



Point to Point with Translator

The translator main purpose is to make the peer look to the other peer like something it is compatible with. An RTP translator will commonly not be distinguishable from the actual end-point. It is intentional not identifiable on RTP level. Reasons a translator can be required are:

- o No common media codec for a media type thus requiring transcoding.
- o Different usages of the RTP multiplexing points
- o Usage of different media transport protocols
- o Usage of different transport protocols
- o Different security solutions

The RTP translator will rewrite RTP and RTCP as required to provide a consistent view to each peer of the traffic the translator forwards and the feedback being provided back to the RTP source.

In some case security policies or the need for monitoring the media streams the direct communication are directed to a pass through a specific middlebox, commonly called a gateway. This is often placed on the border of administrative domain where the security policies are in effect. Many gateways simple relay the RTP and RTCP traffic between the domains, but some may do more by including above mentioned translator functions or even go as far as terminating the RTP session and do application level forwarding of the media traffic. The later places requirements on the gateway to have full

understanding of the application logic and especially be able to cope with any congestion control or media adaptation.

A variant of translator behaviour worth pointing out is when an endpoint A sends a media flow to B. On the path there is a device T that on A's behalf does something with the media streams, for example adds an RTP session with FEC information for A's media streams. T will in this case need to bind the new FEC streams to A's media stream by using the same CNAME as A.

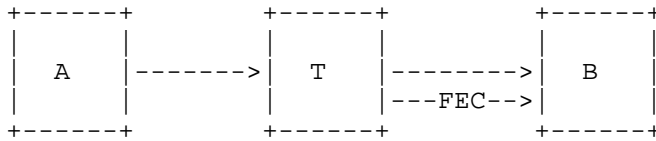


Figure 2: When De-composition is a Translator

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

## 5.2. Point to Multipoint Using Multicast

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

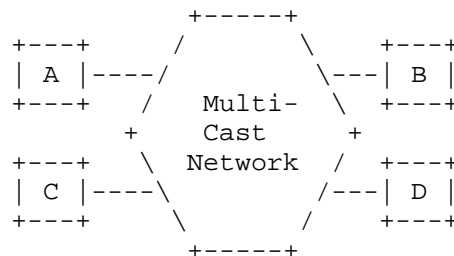
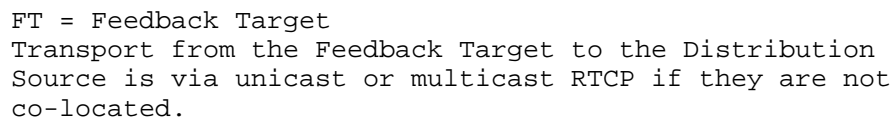


Figure 3: Point to Multipoint Using Any Source Multicast

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



In the SSM topology (Figure 4) a number of RTP sources (1 to M) are allowed to send media to the SSM group. These send media to the distribution source which then forwards the media streams to the multicast group. The media streams reach the Receivers (R(1) to R(n)). The Receivers' RTCP cannot be sent to the multicast group. To support RTCP, an RTP extension for SSM [RFC5760] was defined to use unicast transmission to send RTCP from the receivers to one or more Feedback Targets (FT).

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

1. Multicast applications use a group of RTP sessions, not one. Each endpoint will need to be a member of a number of RTP sessions in order to perform well.
2. Within each RTP session, the number of media sinks is likely to be much larger than the number of RTP sources.
3. Multicast applications need signalling functions to identify the relationships between RTP sessions.
4. Multicast applications need signalling functions to identify the relationships between SSRCs in different RTP sessions.

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

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

### 5.3. Point to Multipoint Using an RTP Transport Translator

This mode is described in section 3.3 of RFC 5117.

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

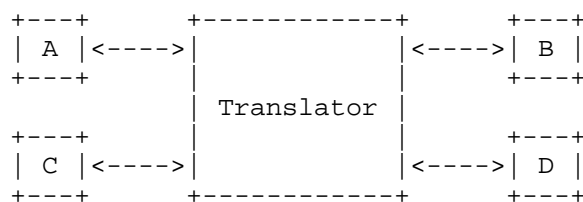


Figure 5: Transport Translator (Relay)

An RTP translator forwards both RTP and RTCP packets from all participants to all other participants.

One of the most important aspects with the simple relay is that it is only rewriting transport headers, no RTP modifications nor media transcoding occur. The most obvious downside of this basic relaying is that the translator has no control over how many streams need to be delivered to a receiver. Nor can it simply select to deliver only certain streams, as this creates session inconsistencies: If the translator temporarily stops a stream, this prevents some receivers from reporting on it. From the sender's perspective it will look like a transport failure. Applications having needs to stop or switch streams in the central node should consider using an RTP mixer to avoid this issue.

The Transport Translator does not need to have an SSRC of itself, nor does it need to send any RTCP reports on the flows that pass it. This as the RTP source will receive feedback for the full path in the RTCP being sent back. However the transport translator may choose to send RTCP reports using its own SSRC, as if it itself contained a media sink, in order to make information about the source-to-translator link available to monitors.

Use of a transport translator results in that all the endpoints will receive multiple SSRCs over a single unicast transport flow from the translator. That is independent of whether the other endpoints have only a single or several SSRCs.

The Transport Translator has the same signalling requirement as multicast: All participants must have the same payload type configuration. Also most of the ASM security issues also arise here. Some alternative when it comes to solution do exist as there after all exist a central node to communicate with. One that also can enforce some security policies depending on the level of trust placed in the node.

#### 5.4. Point to Multipoint Using an RTP Mixer

An mixer (Figure 6) is a centralised node that selects or mixes content in a conference to optimise the RTP session so that each endpoint only needs connect to one entity, the mixer. The mixer can also reduce the bit-rate needed from the mixer down to a conference participants as the media sent from the mixer to the end-point can be optimised in different ways. These optimisations include methods like only choosing media from the currently most active speaker or mixing together audio so that only one audio stream is required instead of three in the depicted scenario (Figure 6).



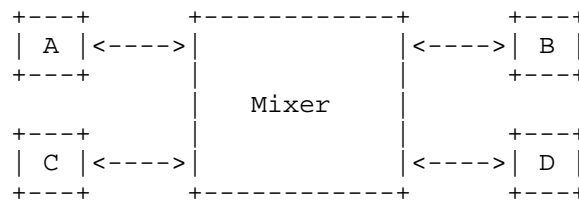


Figure 6: RTP Mixer

Mixers has some downsides, the first is that the mixer must be a trusted node as they either performs media operations or at least repacketize the media. Both type of operations requires when using SRTP that the mixer verifies integrity, decrypts the content, perform its operation and form new RTP packets, encrypts and integrity protect them. This applies to all types of mixers described below. The second downside is that all these operations and optimisation of the session requires processing. How much depends on the implementation as will become evident below.

A mixer, unlike a pure transport translator, is always application specific: the application logic for stream mixing or stream selection has to be embedded within the mixer, and controlled using application specific signalling. The implementation of an mixer can take several different forms and we will discuss the main themes available that doesn't break RTP.

Please note that a Mixer could also contain translator functionalities, like a media transcoder to adjust the media bit-rate or codec used on a particular RTP media stream.

#### 5.4.1. Media Mixing

This type of mixer is one which clearly can be called RTP mixer is likely the one that most thinks of when they hear the term mixer. Its basic pattern of operation is that it will receive the different participants RTP media stream. Select which that are to be included in a media domain mix of the incoming RTP media streams. Then create a single outgoing stream from this mix.

The most commonly deployed media mixer is probably the audio mixer, used in voice conferencing, where the output consists of some mixture of all the input streams; this needs minimal signalling to be successful. Audio mixing is straight forward and commonly possible to do for a number of participants. Lets assume that you want to mix N number of streams from different participants. Then the mixer need to perform N decodings. Then it needs to produce N or N+1 mixes, the reasons that different mixes are needed are so that each contributing

source get a mix which don't contain themselves, as this would result in an echo. When N is lower than the number of all participants one may produce a Mix of all N streams for the group that are currently not included in the mix, thus N+1 mixes. These audio streams are then encoded again, RTP packetised and sent out.

Video can't really be "mixed" and produce something particular useful for the users, however creating an composition out of the contributed video streams can be done. In fact it can be done in a number of ways, tiling the different streams creating a chessboard, selecting someone as more important and showing them large and a number of other sources as smaller overlays is another. Also here one commonly need to produce a number of different compositions so that the contributing part doesn't need to see themselves. Then the mixer re-encodes the created video stream, RTP packetise it and send it out

The problem with media mixing is that it both consume large amount of media processing and encoding resources. The second is the quality degradation created by decoding and re-encoding the RTP media stream. Its advantage is that it is quite simplistic for the clients to handle as they don't need to handle local mixing and composition.

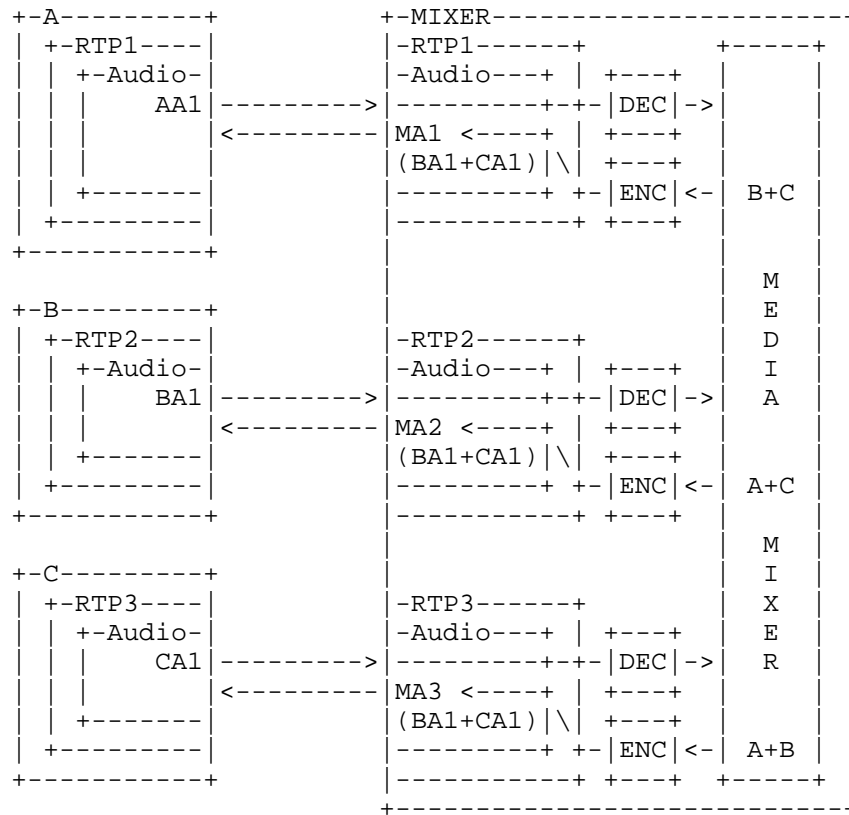


Figure 7: Session and SSRC details for Media Mixer

From an RTP perspective media mixing can be very straight forward as can be seen in Figure 7. The mixer present one SSRC towards the peer client, e.g. MA1 to Peer A, which is the media mix of the other participants. As each peer receives a different version produced by the mixer there are no actual relation between the different RTP sessions in the actual media or the transport level information. There is however one connection between RTP1-RTP3 in this figure. It has to do with the SSRC space and the identity information. When A receives the MA1 stream which is a combination of BA1 and CA1 streams in the other PeerConnections RTP could enable the mixer to include CSRC information in the MA1 stream to identify the contributing source BA1 and CA1.

The CSRC has in its turn utility in RTP extensions, like the in Mixer to Client audio levels RTP header extension [RFC6465]. If the SSRC from endpoint to mixer leg are used as CSRC in another PeerConnection then RTP1, RTP2 and RTP3 becomes one joint session as they have a

common SSRC space. At this stage the mixer also need to consider which RTCP information it need to expose in the different legs. For the above situation commonly nothing more than the Source Description (SDS) information and RTCP BYE for CSRC need to be exposed. The main goal would be to enable the correct binding against the application logic and other information sources. This also enables loop detection in the RTP session.

#### 5.4.1.1. RTP Session Termination

There exist an possible implementation choice to have the RTP sessions being separated between the different legs in the multi-party communication session and only generate RTP media streams in each without carrying on RTP/RTCP level any identity information about the contributing sources. This removes both the functionality that CSRC can provide and the possibility to use any extensions that build on CSRC and the loop detection. It may appear a simplification if SSRC collision would occur between two different end-points as they can be avoided to be resolved and instead remapped between the independent sessions if at all exposed. However, SSRC/CSRC remapping requires that SSRC/CSRC are never used in the application level as reference. This as they only have local importance, if they are used on a multi-party session scope the result would be miss-referencing.

Session termination may appear to resolve some issues, it however creates other issues that needs resolving, like loop detection, identification of contributing sources and the need to handle mapped identities and ensure that the right one is used towards the right identities and never used directly between multiple end-points.

#### 5.4.2. Media Switching

An RTP Mixer based on media switching avoids the media decoding and encoding cycle in the mixer, but not the decryption and re-encryption cycle as one rewrites RTP headers. This both reduces the amount of computational resources needed in the mixer and increases the media quality per transmitted bit. This is achieve by letting the mixer have a number of SSRCs that represents conceptual or functional streams the mixer produces. These streams are created by selecting media from one of the by the mixer received RTP media streams and forward the media using the mixers own SSRCs. The mixer can then switch between available sources if that is required by the concept for the source, like currently active speaker.

To achieve a coherent RTP media stream from the mixer's SSRC the mixer is forced to rewrite the incoming RTP packet's header. First the SSRC field must be set to the value of the Mixer's SSRC. Secondly, the sequence number must be the next in the sequence of

outgoing packets it sent. Thirdly the RTP timestamp value needs to be adjusted using an offset that changes each time one switch media source. Finally depending on the negotiation the RTP payload type value representing this particular RTP payload configuration may have to be changed if the different endpoint mixer legs have not arrived on the same numbering for a given configuration. This also requires that the different end-points do support a common set of codecs, otherwise media transcoding for codec compatibility is still required.

Lets consider the operation of media switching mixer that supports a video conference with six participants (A-F) where the two latest speakers in the conference are shown to each participants. Thus the mixer has two SSRs sending video to each peer.

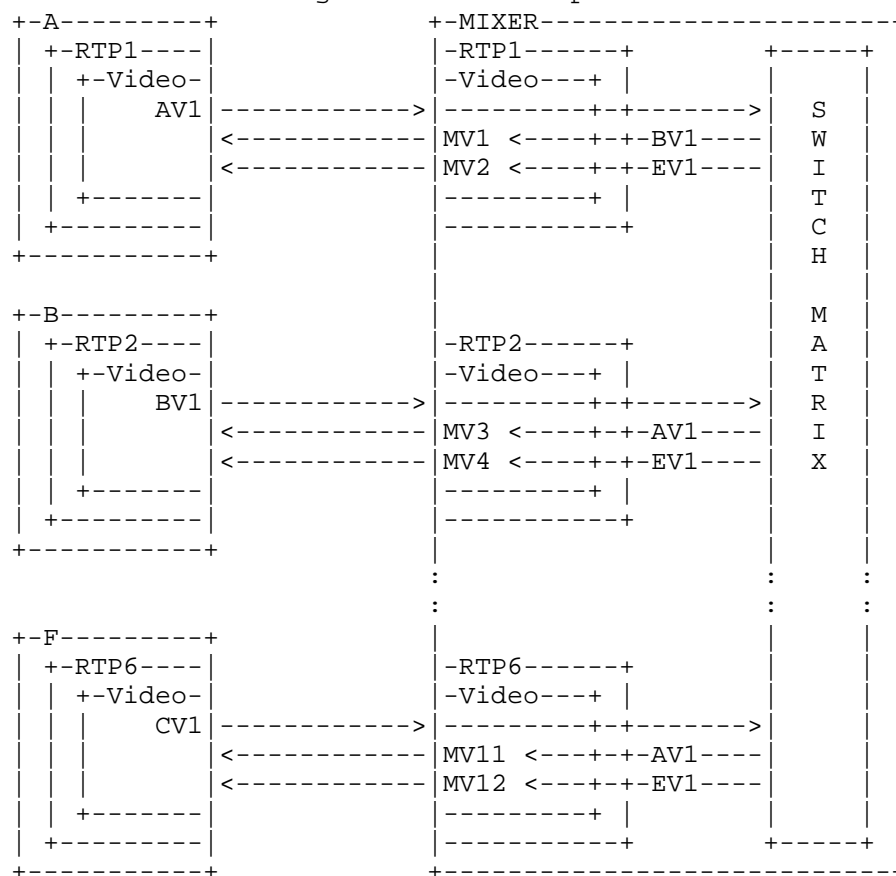


Figure 8: Media Switching RTP Mixer

The Media Switching RTP mixer can similar to the Media Mixing one reduce the bit-rate needed towards the different peers by selecting and switching in a sub-set of RTP media streams out of the ones it receives from the conference participations.

To ensure that a media receiver can correctly decode the RTP media stream after a switch, it becomes necessary to ensure for state saving codecs that they start from default state at the point of switching. Thus one common tool for video is to request that the encoding creates an intra picture, something that isn't dependent on earlier state. This can be done using Full Intra Request [RFC5104] RTCP codec control message.

Also in this type of mixer one could consider to terminate the RTP sessions fully between the different end-point and mixer legs. The same arguments and considerations as discussed in Section 5.4.1.1 applies here.

#### 5.4.3. RTP Source Projecting

Another method for handling media in the RTP mixer is to project all potential RTP sources (SSRCs) into a per end-point independent RTP session. The mixer can then select which of the potential sources that are currently actively transmitting media, despite that the mixer in another RTP session receives media from that end-point. This is similar to the media switching Mixer but have some important differences in RTP details.



sequence number will need to be consecutively incremented based on the packet actually being transmitted in each RTP session. Thus the RTP sequence number offset will change each time a source is turned on in a RTP session.

As the RTP sessions are independent the SSRC numbers used can be handled independently also thus working around any SSRC collisions by having remapping tables between the RTP sessions. This will result that each endpoint may have a different view of the application usage of a particular SSRC. Thus the application must not use SSRC as references to RTP media streams when communicating with other peers directly.

The mixer will also be responsible to act on any RTCP codec control requests coming from an end-point and decide if it can act on it locally or needs to translate the request into the RTP session that contains the media source. Both end-points and the mixer will need to implement conference related codec control functionalities to provide a good experience. Full Intra Request to request from the media source to provide switching points between the sources, Temporary Maximum Media Bit-rate Request (TMMBR) to enable the mixer to aggregate congestion control response towards the media source and have it adjust its bit-rate in case the limitation is not in the source to mixer link.

This version of the mixer also puts different requirements on the end-point when it comes to decoder instances and handling of the RTP media streams providing media. As each projected SSRC can at any time provide media the end-point either needs to handle having thus many allocated decoder instances or have efficient switching of decoder contexts in a more limited set of actual decoder instances to cope with the switches. The WebRTC application also gets more responsibility to update how the media provides is to be presented to the user.

#### 5.5. Point to Multipoint using Multiple Unicast flows

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



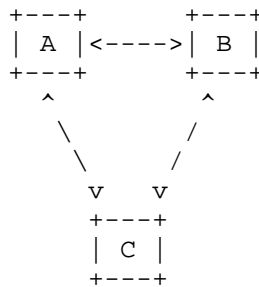


Figure 10: Point to Multi-Point using Multiple Unicast Transports

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

There exists an alternative structure for establishing the above communication scenario (Figure 10) which uses independent RTP sessions between each pair of peers, i.e. three different RTP sessions. Unless independently adapted the same RTP media stream could be sent in both of the RTP sessions an endpoint has. The difference exists in the behaviours around RTCP, for example common RTCP bandwidth for one joint session, rather than three independent pools, and the awareness based on RTCP reports between the peers of how that third leg is doing.

#### 5.6. De-composite Endpoint

The implementation of an application may desire to send a subset of the application's data to each of multiple devices, each with their own network address. A very basic use case for this would be to separate audio and video processing for a particular endpoint, like a conference room, into one device handling the audio and another handling the video, being interconnected by some control functions allowing them to behave as a single endpoint in all aspects except for transport Figure 11.

Which decomposition that is possible is highly dependent on the RTP session usage. It is not really feasible to decomposed one logical end-point into two different transport node in one RTP session. From a third party monitor of such an attempt the two entities would look like two different end-points with a CNAME collision. This put a requirement on that the only type of de-composited endpoint that RTP really supports is one where the different parts have separate RTP sessions to send and/or receive media streams intended for them.

Figure 11: De-composite End-Point

In the above usage, let us assume that the RTP sessions are different for audio and video. The audio and video parts will use a common CNAME and also have a common clock to ensure that synchronisation and clock drift handling works despite the decomposition. Also the RTCP handling works correctly as long as only one part of the de-composite is part of each RTP session. That way any differences in the path between A's audio entity and B and A's video and B are related to different SSRCs in different RTP sessions.

The requirements that can be derived from the above usage is that the transport flows for each RTP session might be under common control but still go to what looks like different endpoints based on addresses and ports. This geometry cannot be accomplished using one RTP session, so in this case, multiple RTP sessions are needed.

## 6.1. Introduction

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

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

### 6.2.1. The RTP Specification

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

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

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

1. If, say, two audio streams shared the same RTP session and the same SSRC value, and one were to change encodings and thus acquire a different RTP payload type, there would be no general way of identifying which stream had changed encodings.
2. An SSRC is defined to identify a single timing and sequence number space. Interleaving multiple payload types would require different timing spaces if the media clock rates differ and would require different sequence number spaces to tell which payload type suffered packet loss.
3. The RTCP sender and receiver reports (see Section 6.4) can only describe one timing and sequence number space per SSRC and do not carry a payload type field.
4. An RTP mixer would not be able to combine interleaved streams of incompatible media into one stream.
5. Carrying multiple media in one RTP session precludes: the use of different network paths or network resource allocations if appropriate; reception of a subset of the media if desired, for example just audio if video would exceed the available bandwidth; and receiver implementations that use separate processes for the different media, whereas using separate RTP sessions permits either single- or multiple-process implementations.

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

two.

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

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

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

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

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

The fifth argument discusses network aspects that we will discuss more below in Section 6.4. It also goes into aspects of implementation, like decomposed endpoints where different processes or inter-connected devices handle different aspects of the whole multi-media session.

A summary of RFC 3550's view on multiplexing is to use unique SSRCs for anything that is its own media/packet stream, and to use different RTP sessions for media streams that don't share media type. The first this document support as very valid. The later is one thing which is further discussed in this document as something the application developer needs to make a continuous choice for.

#### 6.2.1.1. Different Media Types Recommendations

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

"For example, in a teleconference composed of audio and video media encoded separately, each medium SHOULD be carried in a separate RTP session with its own destination transport address."

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

#### 6.2.2. Multiple SSRCs in a Session

In cases when an endpoint uses multiple SSRCs, we have found two closely related issues. The first is if every SSRC shall report on all other SSRC, even the ones originating from the same endpoint. The reason for this would be to ensure that no monitoring function should suspect a breakage in the RTP session. No monitoring function that gives an alert on non-reporting of an endpoint's own SSRCs has been identified.

The second issue around RTCP reporting arise when an endpoint receives one or more media streams, and when the receiving endpoint itself sends multiple SSRC in the same RTP session. As transport statistics are gathered per endpoint and shared between the nodes, all the endpoint's SSRC will report based on the same received data, the only difference will be which SSRCs sends the report. This could be considered unnecessary overhead, but for consistency it might be simplest to always have all sending SSRCs send RTCP reports on all media streams the endpoint receives.

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

Although this issue is valid already for the simple point to point case, it needs to be considered in all topologies. From the perspective of an endpoint, any solution needs to take into account

what a particular endpoint can determine without explicit information of the topology. For example, a Transport Translator (Relay) topology will look quite similar to point to point on a transport level but is different on RTP level. Assume a first scenario with two SSRC being sent from an endpoint to a Transport Translator, and a second scenario with two single SSRC remote endpoints sending to the same Transport Translator. The main differences between those two scenarios are that in the second scenario, the RTT may vary between the SSRCs (but it is not guaranteed), and the SSRCs may also have different CNAMEs.

When an endpoint has multiple SSRCs and it needs to send RTCP packets on behalf of these SSRCs, the question arises how RTCP packets with different source SSRCs can be sent in the same compound packet. It appears allowed, however some consideration of the transmission scheduling is needed.

These issues are currently being discussed and a recommendation for how to handle them are developed in "Real-Time Transport Protocol (RTP) Considerations for Endpoints Sending Multiple Media Streams" [I-D.lennox-avtcore-rtp-multi-stream].

#### 6.2.3. Handling Varying sets of Senders

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

This behaviour will cause similar issues independent of SSRC or Session multiplexing.

#### 6.2.4. Cross Session RTCP Requests

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

In Session multiplexed cases, the RTCP timing rules in the sessions

and the transport aspects, such as packet loss and jitter, prevents a receiver from relying on atomic operations, forcing it to use more robust and forgiving mechanisms.

#### 6.2.5. Binding Related Sources

A common problem in a number of various RTP extensions has been how to bind related sources together. This issue is common to SSRC multiplexing and Session Multiplexing.

Most, if not all, solutions to this problem are implemented in the signalling plane, providing metadata information using SDP.

There exists one solution for grouping RTP sessions together in SDP [RFC5888] to know which RTP session contains for example the FEC data for the source data in another session. However, this mechanism does not work on individual media flows and is thus not directly applicable to the problem. The other solution is also SDP based and can group SSRCs within a single RTP session [RFC5576]. Thus this mechanism can bind media streams in SSRC multiplexed cases. Both solutions have the shortcoming of being restricted to SDP based signalling and also do not work in cases where the session's dynamic properties are such that it is difficult or resource consuming to keep the list of related SSRCs up to date.

One possible solution could be to mandate the same SSRC value being used in all RTP session in case of session multiplexing. We do note that Section 8.3 of the RTP Specification [RFC3550] recommends using a single SSRC space across all RTP sessions for layered coding. However this recommendation has some downsides and is less applicable beyond the field of layered coding. To use the same sender SSRC in all RTP sessions from a particular endpoint can cause issues if an SSRC collision occurs. If the same SSRC is used as the required binding between the streams, then all streams in the related RTP sessions must change their SSRC. This is extra likely to cause problems if the participant populations are different in the different sessions. For example, in case of large number of receivers having selected totally random SSRC values in each RTP session as RFC 3550 specifies, a change due to a SSRC collision in one session can then cause a new collision in another session. This cascading effect is not severe but there is an increased risk that this occurs for well populated sessions (the birthday paradox ensures that if you populate a single session with 9292 SSRCs at random, the chances are approximately 1% that at least one collision will occur). In addition, being forced to change the SSRC affects all the related media streams; instead of having to resynchronise only the originally conflicting stream, all streams will suddenly need to be resynchronised with each other. This will prevent also the media

streams not having an actual collision from being usable during the resynchronisation and also increases the time until synchronisation is finalised. In addition, it requires exception handling in the SSRC generation.

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

Existing mechanisms with known issues:

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

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



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

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

#### 6.2.6. Forward Error Correction

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

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

In usages involving multicast, having the FEC information on its own multicast group, and therefore in its own RTP session, allows for flexibility, for example when using Rapid Acquisition of Multicast Groups (RAMS) [RFC6285]. During the RAMS burst where data is received over unicast and where it is possible to combine with unicast based retransmission [RFC4588], there is no need to burst the FEC data related to the burst of the source media streams needed to catch up with the multicast group. This saves bandwidth to the receiver during the burst, enabling quicker catch up. When the receiver has caught up and joins the multicast group(s) for the source, it can at the same time join the multicast group with the FEC information. Having the source stream and the FEC in separate groups allows for easy separation in the Burst/Retransmission Source (BRS)

without having to individually classify packets.

#### 6.2.7. Transport Translator Sessions

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

#### 6.3. Interworking

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

##### 6.3.1. Types of Interworking

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

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

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

##### 6.3.2. RTP Translator Interworking

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

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

#### 6.3.3. Gateway Interworking

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

- o Generating appropriate RTCP reports for all media streams (possibly based on incoming RTCP reports), originating from SSRCs controlled by the gateway.
- o Handling SSRC collision resolution in each application's RTP sessions.
- o Signalling, choosing and policing appropriate bit-rates for each session.

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

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

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

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

If one uses security functions, like SRTP, they can as seen above

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

#### 6.3.4. Multiple SSRC Legacy Considerations

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

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

1. Need to handle more than one stream simultaneously rather than replacing an already existing stream with a new one.
2. Be capable of decoding multiple streams simultaneously.
3. Be capable of rendering multiple streams simultaneously.

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

#### 6.4. Network Aspects

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

#### 6.4.1. Quality of Service

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

However, for a flow based scheme there is a clear difference between the methods. SSRC multiplexing will result in all media streams being part of the same 5-tuple (protocol, source address, destination address, source port, destination port) which is the most common selector for flow based QoS. Thus, separation of the level of QoS between media streams is not possible. That is however possible for session based multiplexing, where each media stream for which a separate QoS handling is desired can be in a different RTP session that can be sent over different 5-tuples.

#### 6.4.2. NAT and Firewall Traversal

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

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

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

**NAT State:** If an endpoint sits behind a NAT, each flow it generates to an external address will result in a state that has to be kept in the NAT. That state is a limited resource. In home or Small Office/Home Office (SOHO) NATs, memory or processing are usually the most limited resources. For large scale NATs serving many

internal endpoints, available external ports are typically the scarce resource. Port limitations is primarily a problem for larger centralised NATs where endpoint independent mapping requires each flow to use one port for the external IP address. This affects the maximum number of internal users per external IP address. However, it is worth pointing out that a real-time video conference session with audio and video is likely using less than 10 UDP flows, compared to certain web applications that can use 100+ TCP flows to various servers from a single browser instance.

**NAT Traversal Excess Time:** Making the NAT/FW traversal takes a certain amount of time for each flow. It also takes time in a phase of communication between accepting to communicate and the media path being established which is fairly critical. The best case scenario for how much extra time it takes after finding the first valid candidate pair following the specified ICE procedures are:  $1.5 \cdot \text{RTT} + T_a \cdot (\text{Additional\_Flows} - 1)$ , where  $T_a$  is the pacing timer, which ICE specifies to be no smaller than 20 ms. That assumes a message in one direction, and then an immediate triggered check back. The reason it isn't more is that ICE first finds one candidate pair that works prior to attempting to establish multiple flows. Thus, there is no extra time until one has found a working candidate pair. Based on that working pair the needed extra time is to in parallel establish the, in most cases 2-3, additional flows. However, packet loss causes extra delays, at least 100 ms which is the minimal retransmission timer for ICE.

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

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

SSRC multiplexing keeps the additional media streams within one RTP Session and does not introduce any additional NAT traversal complexities per media stream. This can be compared with normally one or two additional transport flows per RTP session when using session multiplexing. Additional lower layer transport flows will be required, unless an explicit de-multiplexing layer is added between

RTP and the transport protocol. A proposal for how to multiplex multiple RTP sessions over the same single lower layer transport exist in [I-D.westerlund-avtcore-transport-multiplexing].

#### 6.4.3. Multicast

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

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

RTP can't function correctly if media streams sent over different multicast groups where considered part of the same RTP session. First of all the different layers needs different SSRCs or the sequence number space seen for a receiver of any sub set of the layers would have sender side holes. Thus triggering packet loss reactions. Also any RTCP reporting of such a session would be non consistent and making it difficult for the sender to determine the sessions actual state.

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

#### 6.4.4. Multiplexing multiple RTP Session on a Single Transport

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

## 6.5. Security Aspects

When dealing with point-to-point, 2-member RTP sessions only, there are few security issues that are relevant to the choice of having one RTP session or multiple RTP sessions. However, there are a few aspects of multiparty sessions that might warrant consideration.

### 6.5.1. Security Context Scope

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

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

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

SRTP [RFC3711] has not special functions for dealing with different sets of master keys for different SSRs. The key-management functions has different capabilities to establish different set of keys, normally on a per end-point basis. DTLS-SRTP [RFC5764] and Security Descriptions [RFC4568] for example establish different keys for outgoing and incoming traffic from an end-point. This key usage must be written into the cryptographic context, possibly associated with different SSRs.

### 6.5.2. Key Management for Multi-party session

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

Multi-party sessions, such as transport translator based sessions and multicast sessions, cannot use Security Description [RFC4568] nor



DTLS-SRTP [RFC5764] without an extension as each endpoint provides its set of keys. In centralised conference, the signalling counterpart is a conference server and the media plane unicast counterpart (to which DTLS messages would be sent) is the transport translator. Thus an extension like Encrypted Key Transport [I-D.ietf-avt-srtp-ekt] is needed or a MIKEY [RFC3830] based solution that allows for keying all session participants with the same master key.

#### 6.5.3. Complexity Implications

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

### 7. Arch-Types

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

#### 7.1. Single SSRC per Session

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

The Pros:

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

2. The signalling has good possibilities to negotiate and describe the exact formats and bit-rates for each media stream, especially using today's tools in SDP.
3. It does not matter if usage or purpose of the media stream is signalled on media stream level or session level as there is no difference.
4. It is possible to control security association per RTP session with current key-management.

The Cons:

- a. The number of required RTP sessions cannot really be higher, which has the implications:
  - \* Linear growth of the amount of NAT/FW state with number of media streams.
  - \* Increased delay and resource consumption from NAT/FW traversal.
  - \* Likely larger signalling message and signalling processing requirement due to the amount of session related information.
  - \* Higher potential for a single media stream to fail during transport between the endpoints.
- b. When the number of RTP sessions grows, the amount of explicit state for relating media stream also grows, linearly or possibly exponentially, depending on how the application needs to relate media streams.
- c. The port consumption may become a problem for centralised services, where the central node's port consumption grows rapidly with the number of sessions.
- d. For applications where the media streams are highly dynamic in their usage, i.e. entering and leaving, the amount of signalling can grow high. Issues arising from the timely establishment of additional RTP sessions can also arise.
- e. Cross session RTCP requests needs is likely to exist and may cause issues.
- f. If the same SSRC value is reused in multiple RTP sessions rather than being randomly chosen, interworking with applications that uses another multiplexing structure than this application will

have issues and require SSRC translation.

- g. Cannot be used with Any Source Multicast (ASM) as one cannot guarantee that only two endpoints participate as packet senders. Using SSM, it is possible to restrict to these requirements if no RTCP feedback is used.
- h. For most security mechanisms, each RTP session or transport flow requires individual key-management and security association establishment thus increasing the overhead.
- i. Does not support multiparty session within a session. Instead each multi-party participant will require an individual RTP session to a given endpoint, even if a central node is used.

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

## 7.2. Multiple SSRCs of the Same Media Type

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

The Pros:

- 1. Low number of RTP sessions needed compared to single SSRC case. This implies:
  - \* Reduced NAT/FW state
  - \* Lower NAT/FW Traversal Cost in both processing and delay.
- 2. Allows for early de-multiplexing in the processing chain in RTP applications where all media streams of the same type have the same usage in the application.

3. Works well with media type de-composite endpoints.
4. Enables Flow-based QoS with different prioritisation between media types.
5. For applications with dynamic usage of media streams, i.e. they come and go frequently, having much of the state associated with the RTP session rather than an individual SSRC can avoid the need for in-session signalling of meta-information about each SSRC.
6. Low overhead for security association establishment.

The Cons:

- a. May have some need for cross session RTCP requests for things that affect both media types in an asynchronous way.
- b. Some potential for concern with legacy implementations that does not support the RTP specification fully when it comes to handling multiple SSRC per endpoint.
- c. Will not be able to control security association for sets of media streams within the same media type with today's key-management mechanisms, only between SDP media descriptions.

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

### 7.3. Multiple Sessions for one Media type

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

The Pros:

1. More suitable for Multicast usage where receivers can individually select which RTP sessions they want to participate in, assuming each RTP session has its own multicast group.
2. Detailed indication of the application's usage of the media stream, where multiple different usages exist.
3. Less need for SSRC specific explicit signalling for each media stream and thus reduced need for explicit and timely signalling.
4. Enables detailed QoS prioritisation for flow based mechanisms.
5. Works well with de-composite endpoints.
6. Handles dynamic usage of media streams well.
7. For transport translator based multi-party sessions, this structure allows for improved control of which type of media streams an endpoint receives.
8. The scope for who is included in a security association can be structured around the different RTP sessions, thus enabling such functionality with existing key-management.

The Cons:

- a. Increases the amount of RTP sessions compared to Multiple SSRCS of the Same Media Type.
- b. Increased amount of session configuration state.
- c. May need synchronised cross-session RTCP requests and require some consideration due to this.
- d. For media streams that are part of scalability, simulcast or transport robustness it will be needed to bind sources, which must support multiple RTP sessions.
- e. Some potential for concern with legacy implementations that does not support the RTP specification fully when it comes to handling multiple SSRC per endpoint.
- f. Higher overhead for security association establishment.
- g. If the applications need finer control than on media type level over which session participants that are included in different sets of security associations, most of today's key-management will have difficulties establishing such a session.

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

#### 7.4. Multiple Media Types in one Session

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

The Pros:

1. Single RTP session which implies:
  - \* Minimal NAT/FW state.
  - \* Minimal NAT/FW Traversal Cost.
  - \* Fate-sharing for all media flows.
2. Enables separation of the different media types based on the payload types so media type specific endpoint or central processing can still be supported despite single session.
3. Can handle dynamic allocations of media streams well on an RTP level. Depends on the application's needs for explicit indication of the stream usage and how timely that can be signalled.
4. Minimal overhead for security association establishment.

The Cons:

- a. Less suitable for interworking with other applications that uses individual RTP sessions per media type or multiple sessions for a single media type, due to need of SSRC translation.
- b. Negotiation of bandwidth for the different media types is currently not possible in SDP. This requires SDP extensions to enable payload or source specific bandwidth. Likely to be a problem due to media type asymmetry in required bandwidth.

- c. Not suitable for de-composite end-points as it requires higher bandwidth and processing.
- d. Flow based QoS cannot provide separate treatment to some media streams compared to other in the single RTP session.
- e. If there is significant asymmetry between the media streams RTCP reporting needs, there are some challenges in configuration and usage to avoid wasting RTCP reporting on the media stream that does not need that frequent reporting.
- f. Not suitable for applications where some receivers like to receive only a subset of the media streams, especially if multicast or transport translator is being used.
- g. Additional concern with legacy implementations that does not support the RTP specification fully when it comes to handling multiple SSRC per endpoint, as also multiple simultaneous media types needs to be handled.
- h. If the applications need finer control over which session participants that are included in different sets of security associations, most key-management will have difficulties establishing such a session.

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

#### 7.5. Summary

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

The two other arch-types "Multiple SSRCs of the Same Media Type" and "Multiple Sessions for one Media Type" are examples of two other cases that first of all allows for some implicit mapping of the role or usage of the media streams based on which RTP session they appear in. It thus potentially allows for less signalling and in particular reduced need for real-time signalling in dynamic sessions. They also

represent points in between the first two when it comes to amount of RTP sessions established, i.e. representing an attempt to reduce the amount of sessions as much as possible without compromising the functionality the session provides both on network level and on signalling level.

## 8. Summary considerations and guidelines

### 8.1. Guidelines

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

**Do not Require the same SSRC across Sessions:** As discussed in Section 6.2.5 there exist drawbacks in using the same SSRC in multiple RTP sessions as a mechanism to bind related media streams together. It is instead recommended that a mechanism to explicitly signal the relation is used, either in RTP/RTCP or in the used signalling mechanism that establishes the RTP session(s).

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

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

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

**RTP/RTCP Extensions May Support SSRC and Session Multiplexing:** When defining an RTP or RTCP extension, the creator needs to consider if this extension is applicable in both SSRC multiplexed and Session multiplexed usages. Any extension intended to be generic is recommended to support both. Applications that are not as



generally applicable will have to consider if interoperability is better served by defining a single solution or providing both options.

Transport Support Extensions: When defining new RTP/RTCP extensions intended for transport support, like the retransmission or FEC mechanisms, they are recommended to include support for both SSRC and Session multiplexing so that application developers can choose freely from the set of mechanisms without concerning themselves with which of the multiplexing choices a particular solution supports.

## 9. IANA Considerations

This document makes no request of IANA.

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

## 10. Security Considerations

There is discussion of the security implications of choosing SSRC vs Session multiplexing in Section 6.5.

## 11. References

### 11.1. Normative References

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

### 11.2. Informative References

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

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

- [I-D.ietf-avt-srtp-ekt]  
Wing, D., McGrew, D., and K. Fischer, "Encrypted Key Transport for Secure RTP", draft-ietf-avt-srtp-ekt-03 (work in progress), October 2011.
- [I-D.ietf-avtcore-rtp-security-options]  
Westerlund, M. and C. Perkins, "Options for Securing RTP Sessions", draft-ietf-avtcore-rtp-security-options-00 (work in progress), July 2012.
- [I-D.ietf-avtext-multiple-clock-rates]  
Petit-Huguenin, M. and G. Zorn, "Support for Multiple Clock Rates in an RTP Session", draft-ietf-avtext-multiple-clock-rates-05 (work in progress), May 2012.
- [I-D.ietf-mmusic-sdp-bundle-negotiation]  
Holmberg, C. and H. Alvestrand, "Multiplexing Negotiation Using Session Description Protocol (SDP) Port Numbers", draft-ietf-mmusic-sdp-bundle-negotiation-00 (work in progress), February 2012.
- [I-D.ietf-payload-rtp-howto]  
Westerlund, M., "How to Write an RTP Payload Format", draft-ietf-payload-rtp-howto-02 (work in progress), July 2012.
- [I-D.lennox-avtcore-rtp-multi-stream]  
Lennox, J. and M. Westerlund, "Real-Time Transport Protocol (RTP) Considerations for Endpoints Sending Multiple Media Streams", draft-lennox-avtcore-rtp-multi-stream-00 (work in progress), July 2012.
- [I-D.lennox-mmusic-sdp-source-selection]  
Lennox, J. and H. Schulzrinne, "Mechanisms for Media Source Selection in the Session Description Protocol (SDP)", draft-lennox-mmusic-sdp-source-selection-04 (work in progress), March 2012.
- [I-D.westerlund-avtcore-max-ssrc]  
Westerlund, M., Burman, B., and F. Jansson, "Multiple Synchronization sources (SSRC) in RTP Session Signaling", draft-westerlund-avtcore-max-ssrc-01 (work in progress), April 2012.
- [I-D.westerlund-avtcore-multi-media-rtp-session]  
Westerlund, M., Perkins, C., and J. Lennox, "Multiple

Media Types in an RTP Session",  
draft-westerlund-avtcore-multi-media-rtp-session-00 (work  
in progress), July 2012.

- [I-D.westerlund-avtcore-transport-multiplexing]  
Westerlund, M. and C. Perkins, "Multiple RTP Sessions on a  
Single Lower-Layer Transport",  
draft-westerlund-avtcore-transport-multiplexing-02 (work  
in progress), March 2012.
- [I-D.westerlund-avtext-rtcp-sdes-srcname]  
Westerlund, M., Burman, B., and P. Sandgren, "RTCP SDSE  
Item SRCNAME to Label Individual Sources",  
draft-westerlund-avtext-rtcp-sdes-srcname-00 (work in  
progress), October 2011.
- [RFC2198] Perkins, C., Kouvelas, I., Hodson, O., Hardman, V.,  
Handley, M., Bolot, J., Vega-Garcia, A., and S. Fosse-  
Parisis, "RTP Payload for Redundant Audio Data", RFC 2198,  
September 1997.
- [RFC2205] Braden, B., Zhang, L., Berson, S., Herzog, S., and S.  
Jamin, "Resource ReSerVation Protocol (RSVP) -- Version 1  
Functional Specification", RFC 2205, September 1997.
- [RFC2326] Schulzrinne, H., Rao, A., and R. Lanphier, "Real Time  
Streaming Protocol (RTSP)", RFC 2326, April 1998.
- [RFC2474] Nichols, K., Blake, S., Baker, F., and D. Black,  
"Definition of the Differentiated Services Field (DS  
Field) in the IPv4 and IPv6 Headers", RFC 2474,  
December 1998.
- [RFC2974] Handley, M., Perkins, C., and E. Whelan, "Session  
Announcement Protocol", RFC 2974, October 2000.
- [RFC3261] Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston,  
A., Peterson, J., Sparks, R., Handley, M., and E.  
Schooler, "SIP: Session Initiation Protocol", RFC 3261,  
June 2002.
- [RFC3264] Rosenberg, J. and H. Schulzrinne, "An Offer/Answer Model  
with Session Description Protocol (SDP)", RFC 3264,  
June 2002.
- [RFC3389] Zopf, R., "Real-time Transport Protocol (RTP) Payload for  
Comfort Noise (CN)", RFC 3389, September 2002.

- [RFC3551] Schulzrinne, H. and S. Casner, "RTP Profile for Audio and Video Conferences with Minimal Control", STD 65, RFC 3551, July 2003.
- [RFC3711] Baugher, M., McGrew, D., Naslund, M., Carrara, E., and K. Norrman, "The Secure Real-time Transport Protocol (SRTP)", RFC 3711, March 2004.
- [RFC3830] Arkko, J., Carrara, E., Lindholm, F., Naslund, M., and K. Norrman, "MIKEY: Multimedia Internet KEYing", RFC 3830, August 2004.
- [RFC4103] Hellstrom, G. and P. Jones, "RTP Payload for Text Conversation", RFC 4103, June 2005.
- [RFC4566] Handley, M., Jacobson, V., and C. Perkins, "SDP: Session Description Protocol", RFC 4566, July 2006.
- [RFC4568] Andreasen, F., Baugher, M., and D. Wing, "Session Description Protocol (SDP) Security Descriptions for Media Streams", RFC 4568, July 2006.
- [RFC4588] Rey, J., Leon, D., Miyazaki, A., Varsa, V., and R. Hakenberg, "RTP Retransmission Payload Format", RFC 4588, July 2006.
- [RFC4607] Holbrook, H. and B. Cain, "Source-Specific Multicast for IP", RFC 4607, August 2006.
- [RFC5104] Wenger, S., Chandra, U., Westerlund, M., and B. Burman, "Codec Control Messages in the RTP Audio-Visual Profile with Feedback (AVPF)", RFC 5104, February 2008.
- [RFC5117] Westerlund, M. and S. Wenger, "RTP Topologies", RFC 5117, January 2008.
- [RFC5576] Lennox, J., Ott, J., and T. Schierl, "Source-Specific Media Attributes in the Session Description Protocol (SDP)", RFC 5576, June 2009.
- [RFC5583] Schierl, T. and S. Wenger, "Signaling Media Decoding Dependency in the Session Description Protocol (SDP)", RFC 5583, July 2009.
- [RFC5760] Ott, J., Chesterfield, J., and E. Schooler, "RTP Control Protocol (RTCP) Extensions for Single-Source Multicast Sessions with Unicast Feedback", RFC 5760, February 2010.

- [RFC5761] Perkins, C. and M. Westerlund, "Multiplexing RTP Data and Control Packets on a Single Port", RFC 5761, April 2010.
- [RFC5764] McGrew, D. and E. Rescorla, "Datagram Transport Layer Security (DTLS) Extension to Establish Keys for the Secure Real-time Transport Protocol (SRTP)", RFC 5764, May 2010.
- [RFC5888] Camarillo, G. and H. Schulzrinne, "The Session Description Protocol (SDP) Grouping Framework", RFC 5888, June 2010.
- [RFC6190] Wenger, S., Wang, Y., Schierl, T., and A. Eleftheriadis, "RTP Payload Format for Scalable Video Coding", RFC 6190, May 2011.
- [RFC6222] Begen, A., Perkins, C., and D. Wing, "Guidelines for Choosing RTP Control Protocol (RTCP) Canonical Names (CNAMEs)", RFC 6222, April 2011.
- [RFC6285] Ver Steeg, B., Begen, A., Van Caenegem, T., and Z. Vax, "Unicast-Based Rapid Acquisition of Multicast RTP Sessions", RFC 6285, June 2011.
- [RFC6465] Iovov, E., Marocco, E., and J. Lennox, "A Real-time Transport Protocol (RTP) Header Extension for Mixer-to-Client Audio Level Indication", RFC 6465, December 2011.

#### Appendix A. Dismissing Payload Type Multiplexing

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

1. Putting restraint on RTP timestamp rate for the multiplexed media. For example, media streams that use different RTP timestamp rates cannot be combined, as the timestamp values need to be consistent across all multiplexed media frames. Thus streams are forced to use the same rate. When this is not possible, Payload Type multiplexing cannot be used.
2. Many RTP payload formats may fragment a media object over multiple packets, like parts of a video frame. These payload formats need to determine the order of the fragments to

correctly decode them. Thus it is important to ensure that all fragments related to a frame or a similar media object are transmitted in sequence and without interruptions within the object. This can relatively simple be solved on the sender side by ensuring that the fragments of each media stream are sent in sequence.

3. Some media formats require uninterrupted sequence number space between media parts. These are media formats where any missing RTP sequence number will result in decoding failure or invoking of a repair mechanism within a single media context. The text/T140 payload format [RFC4103] is an example of such a format. These formats will need a sequence numbering abstraction function between RTP and the individual media stream before being used with Payload Type multiplexing.
4. Sending multiple streams in the same sequence number space makes it impossible to determine which Payload Type and thus which stream a packet loss relates to.
5. If RTP Retransmission [RFC4588] is used and there is a loss, it is possible to ask for the missing packet(s) by SSRC and sequence number, not by Payload Type. If only some of the Payload Type multiplexed streams are of interest, there is no way of telling which missing packet(s) belong to the interesting stream(s) and all lost packets must be requested, wasting bandwidth.
6. The current RTCP feedback mechanisms are built around providing feedback on media streams based on stream ID (SSRC), packet (sequence numbers) and time interval (RTP Timestamps). There is almost never a field to indicate which Payload Type is reported, so sending feedback for a specific media stream is difficult without extending existing RTCP reporting.
7. The current RTCP media control messages [RFC5104] specification is oriented around controlling particular media flows, i.e. requests are done addressing a particular SSRC. Such mechanisms would need to be redefined to support Payload Type multiplexing.
8. The number of payload types are inherently limited. Accordingly, using Payload Type multiplexing limits the number of streams that can be multiplexed and does not scale. This limitation is exacerbated if one uses solutions like RTP and RTCP multiplexing [RFC5761] where a number of payload types are blocked due to the overlap between RTP and RTCP.

9. At times, there is a need to group multiplexed streams and this is currently possible for RTP Sessions and for SSRC, but there is no defined way to group Payload Types.
10. It is currently not possible to signal bandwidth requirements per media stream when using Payload Type Multiplexing.
11. Most existing SDP media level attributes cannot be applied on a per Payload Type level and would require re-definition in that context.
12. A legacy endpoint that doesn't understand the indication that different RTP payload types are different media streams may be slightly confused by the large amount of possibly overlapping or identically defined RTP Payload Types.

#### Appendix B. Proposals for Future Work

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

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

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

#### Appendix C. RTP Specification Clarifications

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

All of the below proposals are under consideration in [I-D.lennox-avtcore-rtp-multi-stream].

### C.1. RTCP Reporting from all SSRCs

When one has multiple SSRC in an RTP node, all these SSRC must send some RTP or RTCP packet as long as the SSRC exist. It is not sufficient that only one SSRC in the node sends report blocks on the incoming RTP streams; any SSRC that intends to remain in the session must send some packets to avoid timing out according to the rules in RFC 3550 section 6.3.5.

It has been hypothesised that a third party monitor may be confused by not necessarily being able to determine that all these SSRC are in fact co-located and originate from the same stack instance; if this hypothesis is true, this may argue for having all the sources send full reception reports, even though they are reporting the same packet delivery.

The contrary argument is that such double reporting may confuse the third party monitor even more by making it seem that utilisation of the last-hop link to the recipient is (number of SSRCs) times higher than what it actually is.

### C.2. RTCP Self-reporting

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

### C.3. Combined RTCP Packets

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

## Appendix D. Signalling considerations

Signalling is not an architectural consideration for RTP itself, so this discussion has been moved to an appendix. However, it is hugely



important for anyone building complete applications, so it is deserving of discussion.

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

#### D.1. Signalling Aspects

There exist various signalling solutions for establishing RTP sessions. Many are SDP [RFC4566] based, however SDP functionality is also dependent on the signalling protocols carrying the SDP. Where RTSP [RFC2326] and SAP [RFC2974] both use SDP in a declarative fashion, while SIP [RFC3261] uses SDP with the additional definition of Offer/Answer [RFC3264]. The impact on signalling and especially SDP needs to be considered as it can greatly affect how to deploy a certain multiplexing point choice.

##### D.1.1. Session Oriented Properties

One aspect of the existing signalling is that it is focused around sessions, or at least in the case of SDP the media description. There are a number of things that are signalled on a session level/ media description but those are not necessarily strictly bound to an RTP session and could be of interest to signal specifically for a particular media stream (SSRC) within the session. The following properties have been identified as being potentially useful to signal not only on RTP session level:

- o Bitrate/Bandwidth exist today only at aggregate or a common any media stream limit, unless either codec-specific bandwidth limiting or RTCP signalling using TMMBR is used.
- o Which SSRC that will use which RTP Payload Types (this will be visible from the first media packet, but is sometimes useful to know before packet arrival).

Some of these issues are clearly SDP's problem rather than RTP limitations. However, if the aim is to deploy an SSRC multiplexed solution that contains several sets of media streams with different properties (encoding/packetization parameter, bit-rate, etc), putting each set in a different RTP session would directly enable negotiation of the parameters for each set. If insisting on SSRC multiplexing only, a number of signalling extensions are needed to clarify that there are multiple sets of media streams with different properties and that they shall in fact be kept different, since a single set will not satisfy the application's requirements.

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

#### D.1.1.2. SDP Prevents Multiple Media Types

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

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

#### D.1.1.3. Signalling Media Stream Usage

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

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

If all media streams in an RTP session are to be treated in the same way, identifying the session is enough. If SSRCs in a session are to be treated differently, signalling must identify both the session and the SSRC.

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

fashion.

#### Appendix E. Changes from -01 to -02

- o Added Harald Alvestrand as co-author.
- o Removed unused term "Media aggregate".
- o Added term "RTP session group", noted that CNAMEs are assumed to bind across the sessions of an RTP session group, and used it when appropriate (TODO)
- o Moved discussion of signalling aspects to appendix
- o Removed all suggestion that PT can be a multiplexing point
- o Normalised spelling of "endpoint" to follow RFC 3550 and not use a hyphen.
- o Added CNAME to definition list.
- o Added term "Media Sink" for the thing that is identified by a listen-only SSRC.
- o Added term "RTP source" for the thing that transmits one media stream, separating it from "Media Source". [[OUTSTANDING: Whether to use "RTP Source" or "Media Sender" here]]
- o Rewrote section on distributed endpoint, noting that this, like any endpoint that wants a subset of a set of RTP streams, needs multiple RTP sessions.
- o Removed all substantive references to the undefined term "purpose" from the main body of the document when it referred to the purpose of an RTP stream.
- o Moved the summary section of section 6 to the guidelines section that it most closely supports.
- o

## Authors' Addresses

Magnus Westerlund  
Ericsson  
Farogatan 6  
SE-164 80 Kista  
Sweden

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

Bo Burman  
Ericsson  
Farogatan 6  
SE-164 80 Kista  
Sweden

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

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

Email: csp@csp Perkins.org

Harald Tveit Alvestrand  
Google  
Kungsbron 2  
Stockholm, 11122  
Sweden

Phone:  
Fax:  
Email: harald@alvestrand.no  
URI:



Network Working Group  
Internet-Draft  
Intended status: Standards Track  
Expires: January 14, 2013

M. Westerlund  
Ericsson  
C. Perkins  
University of Glasgow  
July 13, 2012

Multiple RTP Sessions on a Single Lower-Layer Transport  
draft-westerlund-avtcore-transport-multiplexing-03

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 January 14, 2013.

Copyright Notice

Copyright (c) 2012 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 . . . . .	4
2. Conventions . . . . .	4
2.1. Terminology . . . . .	4
2.2. Requirements Language . . . . .	5
3. Motivations . . . . .	5
3.1. NAT and Firewalls . . . . .	5
3.2. No Transport Level QoS . . . . .	5
3.3. Multiple RTP sessions . . . . .	6
3.4. Usage of RTP Extensions . . . . .	6
3.5. Incremental Deployment . . . . .	7
3.6. Summary . . . . .	7
4. Requirements . . . . .	7
4.1. Support Use of Multiple RTP Sessions . . . . .	7
4.2. Same SSRC Value in Multiple RTP Sessions . . . . .	8
4.3. SRTP . . . . .	8
4.4. Don't Redefine Used Bits . . . . .	9
4.5. Firewall Friendly . . . . .	9
4.6. Monitoring and Reporting . . . . .	9
4.7. Usable Also Over Multicast . . . . .	10
4.8. Incremental Deployment . . . . .	10
5. Design Considerations . . . . .	10
5.1. Location of SHIM . . . . .	10
5.2. Signalling Fallback . . . . .	12
6. Specification . . . . .	13
6.1. Shim Layer . . . . .	13
6.2. Signalling . . . . .	16
6.3. SRTP Key Management . . . . .	17
6.3.1. Security Description . . . . .	18
6.3.2. DTLS-SRTP . . . . .	18
6.3.3. MIKEY . . . . .	18
6.4. Examples . . . . .	19
6.4.1. RTP Packet with Transport Header . . . . .	19
6.4.2. SDP Offer/Answer example . . . . .	19
7. Open Issues . . . . .	24
8. IANA Considerations . . . . .	25
9. Security Considerations . . . . .	25
10. Acknowledgements . . . . .	25
11. References . . . . .	26
11.1. Normative References . . . . .	26
11.2. Informational References . . . . .	26
Appendix A. Possible Solutions . . . . .	27
A.1. Header Extension . . . . .	27

A.2. Multiplexing Shim . . . . .	29
A.3. Single Session . . . . .	29
A.4. Use the SRTP MKI field . . . . .	31
A.5. Use an Octet in the Padding . . . . .	31
A.6. Redefine the SSRC field . . . . .	32
Appendix B. Comparison . . . . .	33
B.1. Support of Multiple RTP Sessions Over Single Transport . . . . .	33
B.2. Enable Same SSRC Value in Multiple RTP Sessions . . . . .	33
B.2.1. Avoid SSRC Translation in Gateways/Translation . . . . .	33
B.2.2. Support Existing Extensions . . . . .	33
B.3. Ensure SRTP Functions . . . . .	34
B.4. Don't Redefine Used Bits . . . . .	34
B.5. Firewall Friendly . . . . .	35
B.6. Monitoring and Reporting . . . . .	37
B.7. Usable over Multicast . . . . .	37
B.8. Incremental Deployment . . . . .	38
B.9. Summary and Conclusion . . . . .	39
Authors' Addresses . . . . .	40



## 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.

There is ongoing work on specifying how and when one RTP session may contain multiple media types [I-D.westerlund-avtcore-multi-media-rtp-session]. That addresses certain use cases, while this proposal addresses a different set of use cases and motivations. This is further discussed in the section on Motivations (Section 3). The classical method of having one RTP session 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 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 were 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. The proposals and the comparison is available as appendices.

## 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. Motivations

This section looks at the motivations why an additional solution is needed assuming that you can do both the classical method of having one RTP session per transport flow as defined by the RTP specification [RFC3550] and when you have multiple media types within one RTP session [I-D.westerlund-avtcore-multi-media-rtp-session].

First we look at the motivations why a single transport flow is of sufficient interest, namely NATs and Firewalls. Then

### 3.1. NAT and Firewalls

The existence of NATs and Firewalls at almost all Internet access has had implications on protocols like RTP that were designed to use multiple transport flows. First of all, the NAT/FW traversal solution one uses needs to ensure that all these transport flows are established. This has three different 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/FWs reaches their limits, unexpected behaviors usually occur.
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 flows reduces the risk of communication failure, improved establishment behavior and less load on NAT and Firewalls.

### 3.2. No Transport Level QoS

Many RTP-using applications don't utilize any network level Quality of Service functions. Nor do they expect or desire any separation in network treatment of its media packets, independent of whether they are audio, video or text. When an application has no such desire, it doesn't need to provide a transport flow structure that simplifies

flow based QoS.

### 3.3. Multiple RTP sessions

The usage of multiple RTP sessions allow separation of media streams that have different usages or purposes in an RTP based application, for example to separate the video of a presenter or most important current talker from those of the listeners that not all end-points receiver. Also separation for different processing based on media types such as audio and video in end-points and central nodes. Thus providing the node with the knowledge that any SSRC within the session is supposed to be processed in a similar or same 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 usages 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].

### 3.4. Usage of RTP Extensions

Applications uses different sets of RTP extensions. The solution for multiple media types in one RTP session [I-D.westerlund-avtcore-multi-media-rtp-session] is known to have limitations that prevent the usage of the following RTP mechanisms and extensions:

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

A developed solution should minimize the number of RTP/RTCP extension

and mechanism that can't be used.

### 3.5. Incremental Deployment

In various multi-party communication scenarios deployment can become an issue if all session participants are required to have the functionality before enabling its usage. This is especially difficult in communication scenarios where not all possible participants and their capabilities are known ahead of establishing the communication session with some sub-set of the participants. At least for centralized communication sessions it is desirable to have a solution that enables allows the solution to be used on a single leg without affecting any other leg, nor require advanced functionality in any central node.

### 3.6. Summary

The center of the motivation is to ensure that the RTP session is a available and usable tool also for applications that has no need for network level separation of its media streams and wants to reduce its exposure to any NAT or Firewall inconsistencies and minimize the resource consumption. As a benefit a well designed solution will enable incremental deployment and minimal limitations in what existing RTP mechanisms or extensions that can be used by the RTP using application.

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

### 4.1. Support Use of Multiple RTP Sessions

Section 3.3 discusses a number of reasons why an application may like to have multiple RTP sessions. Considering the motivations for this work this must be an absolute requirement. We also are of the opinion that the session provided by the solution must fulfill the definition in the RTP [RFC3550] specification:

"The distinguishing feature of an RTP session is that each maintains a full, separate space of SSRC identifiers (defined next). 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 (also defined below) or in RTCP."

#### 4.2. 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 fulfill 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

#### 4.3. 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.

#### 4.4. 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.

#### 4.5. 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.

#### 4.6. 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.

#### 4.7. 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.

#### 4.8. 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.

### 5. Design Considerations

When defining a SHIM solution for identifying RTP sessions over a single transport layer there has been some special considerations that is discussed in this section.

#### 5.1. Location of SHIM

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 post fixed solution will most likely forever prevent this type of devices to process it.

**Legacy Processing:** Packets or at least flows of the type IP/UDP/RTP can in many cases be identified in Deep Packet Inspection, Firewalls or other network entities that concern themselves with trying determine what traffic that flows in a particular packet. These nodes can clearly be updated but until they have they may create a hinder against deployment. Thus a post fix gives likely the least resistance for initial deployment. However, also for postfix location the deployment can be hindered in cases multiple RTP sessions using the same SSRC values due to irregular behavior of the fields for what the third party believes is one media stream rather than multiple ones. The prefixed 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 will not work. Instead only IP/UDP header compression can 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.

The question of a prefixed or a postfix header comes down to a trade-off between long term usability and deployment issues:

**Prefixed:** Long term good possibility to adapt any network function that needs to take the SHIM header into account. At the same time any function that tries to analyze packets and because of that may block the packets will be a hinder to deployment.

**Postfixed:** This solution will likely short term have the best possibilities to deploy successfully. However, 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.



Open Issue: Which should be chosen? The below specification uses prefix but that can easily be changed. But appears to be the best long term choice without too badly affecting deployability.

## 5.2. Signalling Fallback

There exist an important aspect in how the SDP signalling functions, especially Offer/Answer [RFC3264]. The initial idea for the signalling was to build on top of bundle [I-D.ietf-mmusic-sdp-bundle-negotiation] which in its default function negotiate multiple media types over one RTP session [I-D.westerlund-avtcore-multi-media-rtp-session]. If the signalling for the solution that main purpose is to enable multiple RTP sessions results in those cases the peer doesn't support this specification the communicating peer can end up in single RTP session if the peer supports that.

We consider it important that in the signalling design that the application developer can decide what type of fallback that will occur. It is also important to consider that one have to signal SHIM based multiplexing of RTP sessions that are in fact of the type with multiple media types. Thus the signalling for SHIM must be able to describe multiple different scenarios:

1. Multiple RTP sessions multiplexed together using SHIM over one transport
2. Like 1 but where at least one RTP session is containing multiple media types
3. Like 1, but where the peer doesn't support SHIM and the initiator wants to fallback to independent transports
4. Like 2, but where the peer doesn't support SHIM and wants to fallback to multiple BUNDLED sessions over independent transports.

In addition it must be possible to have multiple different transports where each is a SHIM multiplex.

To enable all of these scenarios we propose a solution where each indicates SHIM multiplex is indicated as its own grouping attribute across all media blocks that are included in some form in the multiplex. This resulting in that these media blocks fall under a form of BUNDLE super set. This super set will also have some of bundles restrictions on the transport layer, but not on higher layer. Which Session ID pair a particular media block is associated is signalled using a SDP attribute (a=session-mux-id) in each media

block. When multiple media block are assigned the same session ID pair, they form a RTP session with multiple media types and have the full restriction of bundle between them.

The method of fallback is indicated by providing explicit BUNDLE grouping in addition to the SHIM when the fallback from SHIM is to BUNDLE.

## 6. Specification

This section contains the specification of the solution based on a SHIM, with the 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 regular RTP and RTCP packets and the transport layer being used by the RTP sessions. Thus the layering looks like the following:

```
+-----+
| RTP / RTCP Packet |
+-----+
| Session ID Layer   |
+-----+
| Transport layer    |
+-----+
```

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.

```
+-----+
| S1 | S2 | S3 | S4 |
+-----+
| Session ID Layer   |
+-----+
| Transport layer    |
+-----+
```

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 prefixes 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 or in case any application would need more than 128 RTP sessions.

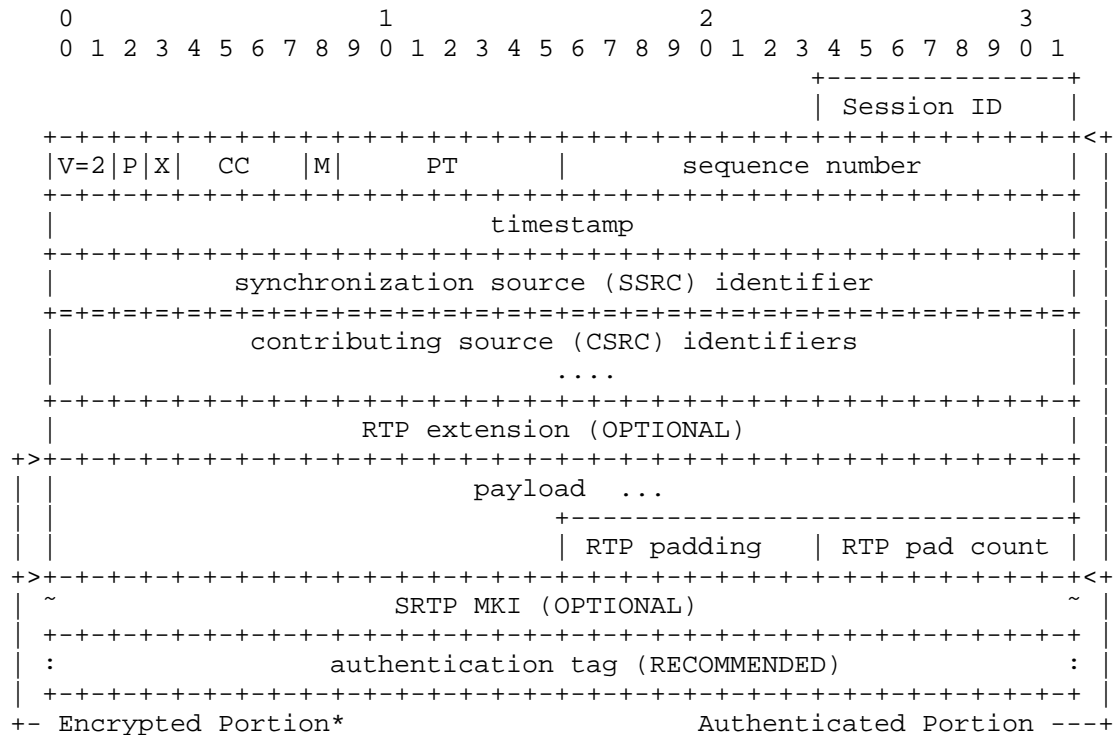


Figure 2: SRTP Packet encapsulated by Session ID Layer

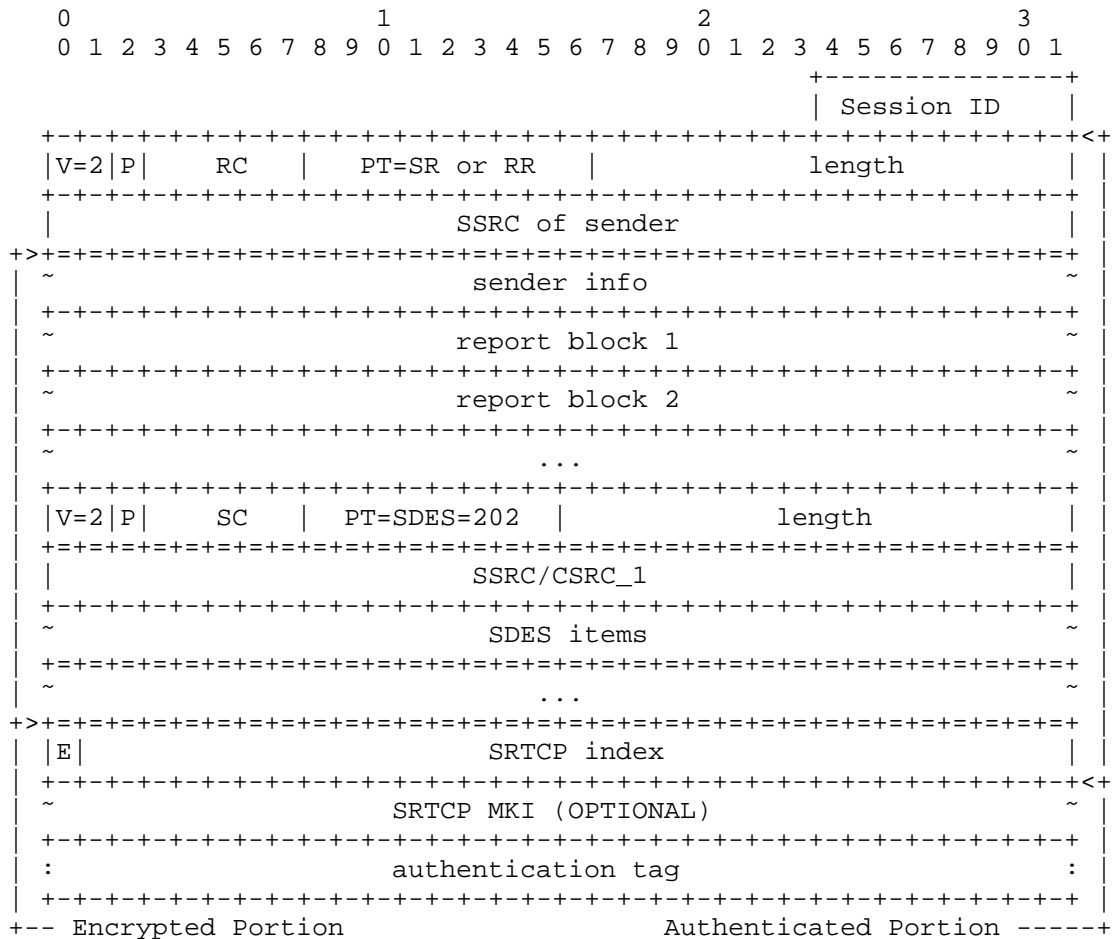


Figure 3: SRTCP packet encapsulated by Session ID layer

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 SDP grouping semantics is defined "SHIM" and a new media-level SDP attribute, 'session-mux-id'. The attribute allows each media description ("m=" line) associated with a 'SHIM' group to be identified in which RTP session it belongs.

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 'SHIM' group. If the SDP Answer does not contain the SHIM group, the SDP Offerer MUST NOT use SHIM based layering. However, if that is separate RTP sessions or BUNDLE is determined on what was present in the offer and answer. This will depend on what the offering party likes to happen. If they want a failure to negotiate a SHIM, instead may be one or more bundle groups then also the BUNDLE grouping is included in the offer. If the SDP Answer still describes a 'BUNDLE' group, the procedures in [I-D.ietf-mmusic-sdp-bundle-negotiation] apply. If not independent transports and sessions are used.

An SDP Answerer MUST NOT include the 'SHIM' group and 'session-mux-id' attribute in an SDP Answer, unless they were 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 SHIM group SHALL contain all media descriptions that are intended to be sent over the same transport flow, independent of Session ID. For all media descriptions part of the same SHIM group the transport parameters, i.e. ports, ICE-candidates etc MUST be the same and handled as described by BUNDLE. Note, the parameters related to the RTP session does not need to be same.

For media descriptions that have the same value of the Session ID SHALL be treated the same way as if they where part of a BUNDLE group, independently if that is indicated or not in the SDP.

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). Thus this behavior doesn't gain you anything in regards to key-management when using SHIM.

### 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].

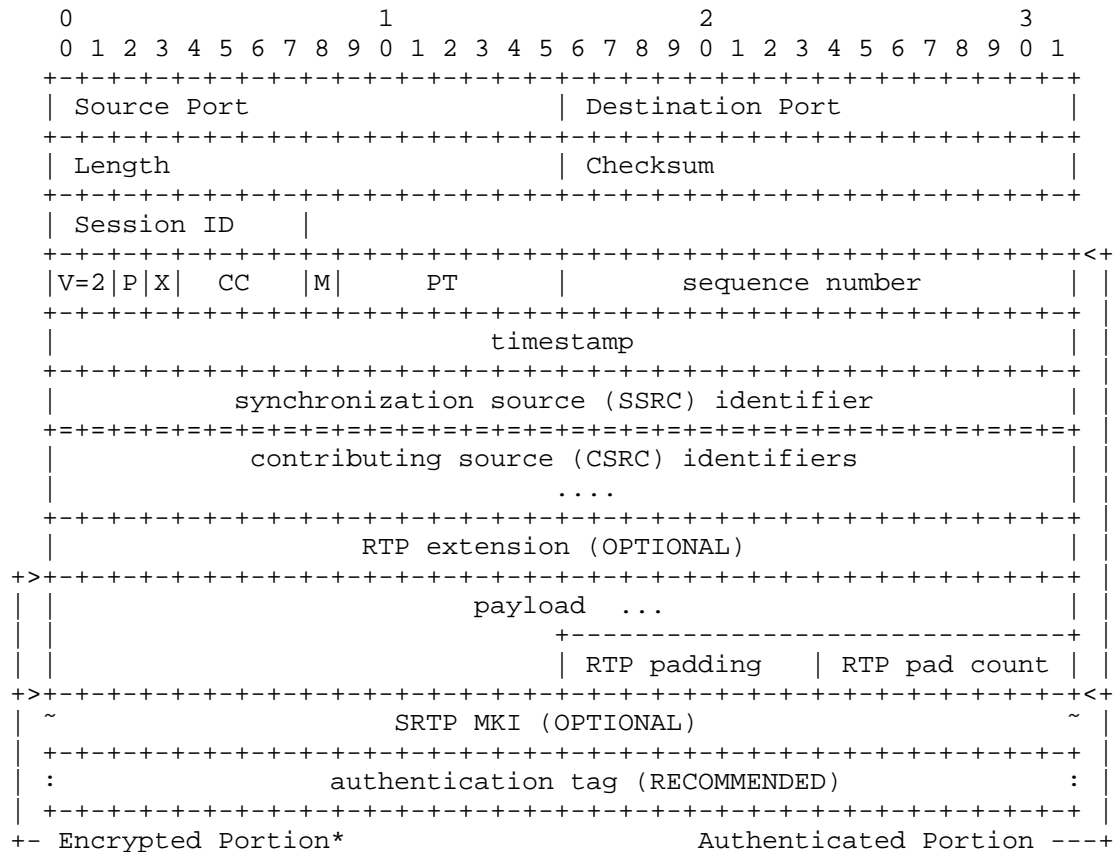
Lets start with the later, i.e. the SDP transport, which shares the properties with Security Description in that it can be associated with a particular media description in a SDP. As long as one avoids using the session level attribute one can be certain to correctly associate the key exchange with a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

#### 6.4.2.1. Basic Example

This section contains SDP offer/answer examples. First one example of successful SHIMing, and then two where fallback occurs. The fallback option here is to fallback to individual transports, thus no BUNDLE group.



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:SHIM foo bar
m=audio 10000 RTP/AVP 0 8 97
b=AS:200
a=mid:foo
a=session-mux-id:0 policy=tentative
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-mux-id:1 policy=tentative
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:SHIM foo bar
m=audio 20000 RTP/AVP 0
b=AS:200
a=mid:foo
a=session-mux-id:0 policy=tentative
a=rtpmap:0 PCMU/8000
m=video 20000 RTP/AVP 32
b=AS:1000
a=mid:bar
a=session-mux-id:1 policy=tentative
a=rtpmap:32 MPV/90000
```

The SDP answer from an end-point that does not support this SHIMing.

```
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
```

#### 6.4.2.2. Advanced Example

In this example we have two BUNDLED sessions, one with audio and video and one with XOR based FEC [RFC5109] for the audio and the video. These two RTP session are then SHIMed into a single transport flow.

```
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 1 2
a=group:BUNDLE 1 2
a=group:BUNDLE foo bar
a=group:FEC foo 1
a=group:FEC bar 2
m=audio 10000 RTP/AVP 0 8 97
b=AS:200
a=mid:foo
a=session-mux-id:0 policy=tentative
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-mux-id:0 policy=tentative
a=rtpmap:31 H261/90000
a=rtpmap:32 MPV/90000
m=audio 10000 RTP/AVP 100
b=AS:100
a=rtpmap:100 ulpfec/8000
a=mid:1
a=session-mux-id:1 policy=tentative
m=video 10000 RTP/AVP 101
b=AS:500
a=mid:2
a=session-mux-id:1 policy=tentative
a=rtpmap:101 ulpfec/90000
```

The SDP answer of a client supporting  
[I-D.ietf-mmusic-sdp-bundle-negotiation] but not this SHIMing 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 1 2
a=group:BUNDLE foo bar
a=group:FEC foo 1
a=group:FEC bar 2
m=audio 20000 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 20000 RTP/AVP 31 32
b=AS:1000
a=mid:bar
a=rtpmap:31 H261/90000
a=rtpmap:32 MPV/90000
m=audio 20002 RTP/AVP 100
b=AS:100
a=rtpmap:100 ulpfec/8000
a=mid:1
m=video 20002 RTP/AVP 101
b=AS:500
a=mid:2
a=rtpmap:101 ulpfec/90000
```

In the above case two different RTP sessions, both being of a BUNDLE type with multiple media types in each. The two established flows will be Alice:10000<->Bob:20000, and Alice:10000<->Bob:20002.

If the peer did support neither of the SHIM or BUNDLE extension the answer 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:FEC foo 1
a=group:FEC bar 2
m=audio 20000 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 20002 RTP/AVP 31 32
b=AS:1000
a=mid:bar
a=rtpmap:31 H261/90000
a=rtpmap:32 MPV/90000
m=audio 20004 RTP/AVP 100
b=AS:100
a=rtpmap:100 ulpfec/8000
a=mid:1
m=video 20006 RTP/AVP 101
b=AS:500
a=mid:2
a=rtpmap:101 ulpfec/90000
```

In this case four different transport flows would be established for RTP, each with a different RTP session over them. The answer also knows the binding between the sessions with FEC and their source data thanks to the FEC specification.

## 7. Open Issues

This work is still in the early phase of specification. 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 must 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 fallback to single RTP session when Multiplexing shim fails to be supported but Bundle is supported ends up with a better alternative?

3. Is there any issues with using DTLS-SRTP individually per RTP session?
4. Shall the SHIM header be prefixed or postfixed in relation to the RTP/RTCP packets?

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

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. and H. Alvestrand, "Multiplexing Negotiation Using Session Description Protocol (SDP) Port Numbers", draft-ietf-mmusic-sdp-bundle-negotiation-00 (work in progress), February 2012.
- [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]  
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.
- [I-D.westerlund-avtcore-multi-media-rtp-session]  
Westerlund, M., Perkins, C., and J. Lennox, "Multiple Media Types in an RTP Session", draft-westerlund-avtcore-multi-media-rtp-session-00 (work in progress), July 2012.
- [I-D.westerlund-avtcore-multiplex-architecture]  
Westerlund, M., Burman, B., and C. Perkins, "RTP

Multiplexing Architecture",  
draft-westerlund-avtcore-multiplex-architecture-01 (work  
in progress), March 2012.

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

## Appendix A. Possible Solutions

This section looks at a few possible solutions 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 SRTP. However, the current SRTP 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 SRTP 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 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.

#### 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 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 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.westerlund-avtcore-multiplex-architecture] 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 should be place 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 may 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.westerlund-avtcore-multiplex-architecture] 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 may 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 (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 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 gatewaying node to ensure that the SSRCs fulfill 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 required. In addition the translator must 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 4) 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. The main question around this requirement is how important it is to have as discussed in Section 4.1.

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

Based on the discussion in Section 4.2 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 being gatewayed 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 should 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 should 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 implementors.

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 behavior.

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 may 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 may 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 may 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 may occur. In particular a third party monitor may 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 may 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 fulfill the rules or semantics of the multiplexed side. For Redefine SSRC field this becomes almost constant as the session identifier part of the SSRC must be the same over all SSRCs from the same session. For Single Session it may 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 required 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 should still 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 4: 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@csp@perkins.org

