

AVTCORE
Internet-Draft
Intended status: Informational
Expires: April 26, 2013

R. Ejzak
Alcatel-Lucent
October 23, 2012

Media multiplexing with Real-time Transport Protocol (RTP) subsessions
draft-ejzak-avtcore-rtp-subsessions-02

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 April 26, 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	4
2.1. RTP procedures	4
2.2. Summary of SSRC bit assignments	5
2.3. SDP procedures	5
2.4. Additional procedures for relays	7
2.5. When a system doesn't support RTP subsessions	8
2.6. Alternative SDP procedures considered	8
2.6.1. Signaling only MSID	8
2.6.2. MSID with implicit SSRC range reservation	8
3. Applicability of RTP subsessions	9
4. Examples	10
5. Evaluation	12
5.1. Comparison to BUNDLE	12
5.2. Comparison to SHIM	13
5.3. Comparison to other SSRC proposals	13
6. IANA Considerations	13
7. Security Considerations	13
8. Informative References	14
Author's Address	15

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], [I-D.westerlund-avtcore-max-ssrc] and [I-D.alvestrand-mmusic-msid]. Note in particular that the use of MSID, mandated in RTCWEB, uses SSRC reservation to assist in stream identification.

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. Also included is a description of a means of negotiating use of RTP subsessions in SDP, along with some alternatives. 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. Summary of SSRC bit assignments

RTP subsessions supports topologies including end systems, mixers and translators (relays), where relays are classified either as "master" or "slave". These end systems share responsibility for SSRC bit assignments as described in the following table, using the procedure described in the following section. There is a strict precedence order between these systems, where a master relay can override SSRC prefix assignments made by a slave relay, which can override SSRC prefix assignments made by a mixer or end system, but not the other way around.

SSRC bit range	Purpose
1	identifies direction of flow
2-8	subsession id (for QoS)
9-16	allocated by primary relay to slave relays (except one value reserved for self)
17-24	allocated by slave relays to non-relay systems
25-32	allocated by non-relay systems to individual RTP streams (e.g., via MSID)

SSRC bit assignments

2.3. 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.

2.4. 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.

2.5. 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.

2.6. Alternative SDP procedures considered

Since [I-D.alvestrand-mmusic-msid] already describes an SSRC reservation procedure for SDP, it is appropriate to consider the relationship to RTP subsessions and whether MSID can be used to simplify the signaling of RTP subsessions in SDP.

2.6.1. Signaling only MSID

If we consider using MSID without change to signal the use of RTP subsessions, there are two limitations to consider. MSID provides for explicit identification of the SSRC used for each stream, so a traffic filter could be constructed to search for RTP packets with particular SSRC values, but this is quite cumbersome once there is any significant amount of multiplexing per media line. Furthermore, MSID includes no SSRC collision resolution procedure, thus making the use of SDP offer/answer re-negotiation necessary for collision resolution. While SSRC collisions are fairly rare if SSRC assignment is completely random, collisions will occur and this form of resolution is expensive and preferably avoided.

2.6.2. MSID with implicit SSRC range reservation

RTP subsession negotiation could be easily added to MSID as a mandatory feature, as follows. If selection of an SSRC value for MSID implicitly reserves SSRC ranges according to the conventions described earlier in the document, then there would be no need to explicitly signal the SSRC prefix. The only aspect missing is to create a convention to negotiate precedence (master relay, slave relay, or other) to provide for a simple method of resolving any SSRC

or SSRC range collisions. Details are left for further study if this proposal for enhancing MSID is agreed.

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 that provide additional media processing functions before forwarding RTP streams 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 24 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 prefix 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 0x11111111 and 0xb3333333, 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 0x22222222 to 0xb3333333 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 24 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. SHIM 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 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.alvestrand-mmusic-msid]
Alvestrand, H., "Cross Session Stream Identification in the Session Description Protocol", draft-alvestrand-mmusic-msid-01 (work in progress), October 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-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: April 25, 2013

A. Williams
Audinate
K. Gross
AVA Networks
R. van Brandenburg
H. Stokking
TNO
October 22, 2012

RTP Clock Source Signalling
draft-ietf-avtcore-clksrc-01

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 April 25, 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 Quality	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	14
5.4. Signalling Grammar	15
5.5. Examples	16
6. IANA Considerations	18
7. References	19
7.1. Normative References	19
7.2. Informative References	20
Authors' Addresses	21

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 [16] or audio word clock [17]). 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 [8] 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 [18].

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 [9] is commonly used to provide a traceable time reference. Some network time synchronisation protocols (e.g. PTP [10], 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 [11]) and radio clocks (e.g. GPS [9]).

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 or receiver can be directly

compared to a local clock in another RTP sender or receiver.

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 or 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 [12]: 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 [10]: 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 [13]: "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 as described in IEEE 802.1BA-2011 [14].

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 boundary clock which is itself slaved to another boundary clock or the grandmaster 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 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 domains are signalled via SDP, senders and receivers SHOULD check that both grandmaster ClockIdentity and PTP domain 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 biased by configuration parameters to influence the election process. In some systems it may be desirable to limit the number of possible PTP clock masters to avoid the need to re-signal 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

RFC 3550 allows senders and receivers to either use a local wallclock reference for their NTP timestamps or, by setting the timestamp field to 0, to supply no timestamps at all. Both are common practice in embedded RTP implementations. These clocks are identified as "local" and can only be assumed to be equivalent to clocks originating from the same device.

In other systems, all RTP senders and 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 and receivers in a session using a private reference clock are assumed to have a mechanism outside this specification for determining whether their 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 Quality

Network time protocol services periodically exchange timestamped messages between servers and clients. Assuming RTP device 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 synchronization quality, an optional timestamp may be included in the SDP clock source signalling. In addition, the frequency of synchronisation messages 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 quality timestamp is far from the timestamp clock at the receiver of the parameters, it can be assumed that synchronisation has not occurred recently, the timestamp reference clock source cannot be contacted or the SDP information has not been recently refreshed. To eliminate the latter possibility, updated SDP information should be retrieved. If the synchronisation quality timestamp is still far from the timestamp clock 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.

When multiple reference clocks are specified, a sender or receiver may wish to base selection of the clock used based on most recent or most frequent update as indicated by the synchronization quality signalling.

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.

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 quality 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-quality ] 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-server
ptp-version      = "IEEE1588-2002"
ptp-version      =/ "IEEE1588-2008"
ptp-version      =/ "IEEE802.1AS-2011"
ptp-server       = ptp-gmid [ ":" ptp-domain ] / "traceable"
ptp-gmid         = EUI64
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-quality     = 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 explicitly signalled.

```
a=mediaclk: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 with an `a=ts-refclk` attribute (Section 4.8).

The signalling optionally indicates a media clock offset value. The offset indicates the RTP timestamp value at the epoch (time of origin) of the reference clock. If no offset is signalled, the offset can be inferred at the receiver by examining RTCP sender reports which contain NTP and RTP timestamps which combined define a mapping.

A rate modifier may be specified. The modifier is expressed as the ratio of two integers and modifies the rate specified or implied by the media description by this ratio. If omitted, the rate is assumed to be the exact rate specified or implied by the media format. For example, without a rate specification, 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.

The rate modifier is primarily useful for accommodating certain "oddball" audio sample rates associated with NTSC video (see Figure 7). Modified rates are not advised for video streams which generally use a 90 kHz RTP clock regardless of frame rate or sample rate used for embedded audio.

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

5.3. Stream-Referenced Media Clock

A common synchronisation architecture for audio/visual systems involves distributing a reference media clock from a master device to a number of slave devices, typically by means of a cable. Examples include audio word clock distribution and video black burst distribution. In this case, the media clock is locally generated, often by a crystal oscillator and is not locked to a timestamp reference clock.

To support this architecture across a network, a master clock identifier is associated with media sources carrying media clock timing information from a master device. The master clock identifier represents a media clock source in the master device. Slave devices in turn associate the master media clock identifier with streams they transmit, signalling the synchronisation relationship between the master and slave devices.

Slave devices recover media clock timing from the clock master stream, using it to synchronise the slave media clock with the master. Timestamps in the master clock media stream are taken using the timestamp reference clock shared by the master and slave devices. The timestamps communicate information about media clock timing (rate, phase) from the master to the slave devices. Timestamps are communicated in the usual RTP fashion via RTCP SRs, or via the RFC6051 [6] header extension. The stream media format may indicate other clock information, such as the nominal rate.

Note that slaving of a device media clock to a master device does not affect the usual RTP lip sync / time alignment algorithms. Time aligned playout of two or more RTP sources still relies upon NTP timestamps supplied via RTCP SRs or by the RFC6051 timestamp header extension.

In a given system, master clock identifiers must be unique. Such identifiers MAY be manually configured, however 17 octet string identifiers SHOULD be generated according to the "short-term persistent RTCP CNAME" algorithm as described in RFC6222 [7].

A reference stream can be an RTP stream or AVB stream based on the IEEE 1722 [15] standard.

An RTP clock master stream SHOULD be identified at the source level by an SSRC and master clock identifier. If master clock identifiers are declared at the media or session level, all RTP sources at or below the level of declaration MUST provide equivalent timing to a slave receiver.

```
a=ssrc:12345 mediaclock:master id=<media-clock-master-id>
```

An RTP media source indicates that it is slaved to a clock master via a clock master identifier:

```
a=mediaclock:slave id=<media-clock-master-id>
```

An RTP media source indicates that it is slaved to an IEEE 1722 clock master via a stream identifier (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-mediactk = "a=mediactk:" mediactk

mediactk = refclk / streamid / sender

rate = [ SP "rate=" 1*DIGIT "/" 1*DIGIT ]

refclk = "direct" [ "=" 1*DIGIT ] rate

streamid = "master-id=" clk-master-id

streamid =/ "slave-to=" clk-master-id

streamid =/ "IEEE1722=" EUI64

clk-master-id = EUI48

sender = "sender"

EUI48 = 5(2HEXDIG ":") 2HEXDIG
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=mediactk:direct=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:direct=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 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:slave id=00:60:2b:20:12:1f
```

Figure 8: RTP stream with media clock slaved to a master device

Figure 9 shows the same 48 kHz audio transmission from Figure 6 with media clock derived from an IEEE 1722 AVB 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: RTP stream with media clock slaved to an IEEE1722 master device

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: mediaclk
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: "sender", "direct", "master", "slave" and "IEEE1722".

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.
- [6] Perkins, C. and T. Schierl, "Rapid Synchronisation of RTP

Flows", RFC 6051, November 2010.

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

7.2. Informative References

- [8] Brandenburg, R., Stokking, H., Deventer, O., Boronat, F., Montagud, M., and K. Gross, "Inter-destination Media Synchronization using the RTP Control Protocol (RTCP)", draft-ietf-avtcore-idms-07 (work in progress), October 2012.
- [9] Global Positioning Systems Directorate, "Navstar GPS Space Segment/Navigation User Segment Interfaces", September 2011.
- [10] 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>>.
- [11] Mills, D., Martin, J., Burbank, J., and W. Kasch, "Network Time Protocol Version 4: Protocol and Algorithms Specification", RFC 5905, June 2010.
- [12] 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>>.
- [13] "Timing and Synchronization for Time-Sensitive Applications in Bridged Local Area Networks",
<<http://standards.ieee.org/findstds/standard/802.1AS-2011.html>>.
- [14] "Audio Video Bridging (AVB) Systems",
<<http://standards.ieee.org/findstds/standard/802.1BA-2011.html>>.
- [15] "IEEE Standard for Layer 2 Transport Protocol for Time Sensitive Applications in a Bridged Local Area Network",
<<http://standards.ieee.org/findstds/standard/1722-2011.html>>.

URIs

- [16] <<http://en.wikipedia.org/wiki/Genlock>>

- [17] <http://en.wikipedia.org/wiki/Word_clock>
- [18] <<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
URI: <http://www.audinate.com/>

Kevin Gross
AVA Networks
Boulder, CO
US

Email: 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
Updates: 3550 (if approved)
Intended status: Standards Track
Expires: April 22, 2013

K. Gross
AVA Networks
R. van Brandenburg
TNO
October 19, 2012

RTP and Leap Seconds
draft-ietf-avtcore-leap-second-01

Abstract

This document discusses issues that arise when RTP sessions span Universal Coordinate Time (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 April 22, 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. Leap seconds	3
3.1. UTC behavior during leap second	4
3.2. NTP behavior during leap second	4
3.3. POSIX behavior during leap second	4
3.4. Summary of leap-second behaviors	4
4. Recommendations	5
4.1. RTP Sender Reports and Receiver Reports	5
4.2. RTP Packet Playout	5
5. Security Considerations	5
6. IANA Considerations	6
7. Acknowledgements	6
8. Normative References	6
Authors' Addresses	6

1. Introduction

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

This document provides recommendations for smoothly rendering streamed media referenced to common wall clocks which do 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 [1] and indicate requirement levels for compliant implementations.

3. Leap seconds

The world time standard is International Atomic Time (TAI) which is based on vibrations of cesium atoms in an atomic clock. The more common Universal Coordinated Time (UTC) is based on the rotation of the Earth. In 1971 UTC was redefined in terms of TAI and the concept of leap seconds was introduced to allow UTC to remain 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.[2] 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 since their introduction in 1971 have been scheduled in June or December and all have added seconds. This is a situation that is expected to 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]

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".

3.2. NTP behavior during leap second

Under NTP [4] 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 a moment in time in either of the last two seconds of a day containing a leap second.

3.4. Summary of leap-second behaviors

Table 1 summarizes behavior across a leap second for the wall clocks discussed above.

The table illustrates the leap second that occurred June 30, 2012 when the offset between International Atomic time (TAI) and UTC changed from 34 to 35 seconds. The first column shows RTP timestamps for an 8 kHz audio stream. The second column shows the TAI reference. Following columns show behavior for the leap-second-bearing wall clocks described above. Time values are shown at half-second intervals.

RTP	TAI	UTC	POSIX	NTP
8000	00:00:32.500	23:59:58.500	23:59:58.500	23:59:58.500
12000	00:00:33.000	23:59:59.000	23:59:59.000	23:59:59.000
16000	00:00:33.500	23:59:59.500	23:59:59.500	23:59:59.500
20000	00:00:34.000	23:59:60.000	23:59:59.000	00:00:00.000
24000	00:00:34.500	23:59:60.500	23:59:59.500	00:00:00.000
28000	00:00:35.000	00:00:00.000	00:00:00.000	00:00:00.000
32000	00:00:35.500	00:00:00.500	00:00:00.500	00:00:00.500

Table 1

4. Recommendations

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

RTP implementation using a wall-clock reference is simplified by using a clock with a timescale which does not include leap seconds. IEEE 1588 [5], GPS [6] and other TAI [7] 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. In the Table 1 example, the region to be avoided is indicated by RTP timestamps 12000 through 28000

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 Playout

Receivers working to a leap-second-bearing reference SHOULD take leap seconds in their reference into account in determining play-out 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 [8].

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

- [1] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", March 1997.
- [2] ITU-R, "Recommendation ITU-R TF.460-4 - Standard-frequency and time-signal emissions", February 2002.
- [3] ITU-R Working Party 7A, "Question SG07.236", February 2012.
- [4] Mills, D., Delaware, U., Martin, J., Ed., Burbank, J., and W. Kasch, "Network Time Protocol Version 4: Protocol and Algorithms Specification", June 2010.
- [5] IEEE, "IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems", July 2008.
- [6] Global Positioning Systems Directorate, "Navstar GPS Space Segment/Navigation User Segment Interfaces", September 2011.
- [7] BIPM, "Circular T", May 2012.
- [8] Schulzrinne, H., Casner, S., Frederick, R., and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications, RFC3550", July 2003.

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

AVTCORE WG
Internet-Draft
Updates: 3550, 3551 (if approved)
Intended status: Standards Track
Expires: April 25, 2013

M. Westerlund
Ericsson
C. Perkins
University of Glasgow
J. Lennox
Vidyo
October 22, 2012

Multiple Media Types in an RTP Session
draft-ietf-avtcore-multi-media-rtp-session-01

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 and RFC 3551 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 April 25, 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	7
5.3. Homogeneous Multi-party	7
5.4. Reduced number of Payload Types	8
5.5. Stream Differentiation	8
5.6. Non-compatible Extensions	9
6. RTP Session Specification	9
6.1. RTP Session	9
6.2. Sender Source Restrictions	11
6.3. Payload Type Applicability	12
6.4. RTCP	12
7. Extension Considerations	14
7.1. RTP Retransmission	14
7.2. Generic FEC	14
8. Signalling	15
8.1. SDP-Based Signalling	15
9. IANA Considerations	16
10. Security Considerations	16
11. Acknowledgements	16
12. References	16
12.1. Normative References	16
12.2. Informative References	17
Authors' Addresses	18

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] recommends against sending several different types of media, for example audio and video, in a single RTP session. The RTP profile for Audio and Video Conferences with Minimal Control (RTP/AVP) [RFC3551] mandates a similar restriction. The motivation for these limitations is partly to allow lower layer Quality of Service (QoS) mechanisms to be used, and partly due to limitations of the RTCP timing rules that require all media in a session to have similar bandwidth. The Session Description Protocol (SDP) [RFC4566], as one of the dominant signalling method for establishing RTP session, has enforced this rule, simply 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 when allowing this behavior. 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 require different lower-layer QoS for different media types, and that have no special requirements for RTP extensions or RTCP reporting, the requirement to separate different media into different RTP sessions may seem unnecessary. Provided the media flows have similar bandwidth requirements, so that the RTCP timing rules work, using the same RTP session for several types of media at once appears a reasonable choice. The architecture should be agnostic about the type of media being carried in an RTP session to the extent possible given the constraints of the protocol.

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]. Similar change is needed to a sentence in Section 6 of [RFC3551] that states that "different media types SHALL NOT be

interleaved or multiplexed within a single RTP Session". This is resolved by appropriate exception clauses given that this specification and its applicability is followed.

Within an RTP session where multiple media types have been configured for use, an SSRC may send only one type of media during its lifetime (i.e., it can switch between different audio codecs, since those are both the same type of media, but cannot switch between audio and video). 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 it must ensure that their application and usage meets the below criteria.

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 Guidelines for using the Multiplexing Features of RTP [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].

The second important consideration is that all media flows to be sent within a single RTP session need to have similar bandwidth. This is due to limitations of the RTCP timing rules, and the need for a common RTCP reporting interval across all participants in a session to avoid problems with premature SSRC timeouts. If an RTP session contains flows with very different bandwidths, for example low-rate audio coupled with high-quality video, this will result in either excessive or insufficient RTCP for some flows, depending how the RTCP session bandwidth, and hence reporting interval, is configured. This is discussed further in Section 6.4.

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 use 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.

Second paragraph of Section 6 in RTP Profile for Audio and Video Conferences with Minimal Control [RFC3551] says:

The payload types currently defined in this profile are assigned to exactly one of three categories or media types: audio only, video only and those combining audio and video. The media types are marked in Tables 4 and 5 as "A", "V" and "AV", respectively. Payload types of different media types SHALL NOT be interleaved or multiplexed within a single RTP session, but multiple RTP sessions MAY be used in parallel to send multiple media types. An RTP source MAY change payload types within the same media type during a session. See the section "Multiplexing RTP Sessions" of RFC 3550 for additional explanation.

This specifications purpose is to violate that existing SHALL NOT under certain conditions. Thus also this sentence must be changed to allow for multiple media type's payload types in the same session. The above sentence is changed to:

Payload types of different media types SHALL NOT be interleaved or multiplexed within a single RTP session unless as specified and under the restriction in Multiple Media Types in an RTP Session [RFCXXXX]. Multiple RTP sessions MAY be used in parallel to send multiple media types.

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

We can now go on and discuss the five bullets that are motivating the previous in Section 5.2 of the RTP Specification [RFC3550]. They are repeated here for the reader's convenience:

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 of RFC 3550) 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.

Bullets 1 to 3 are all related to that each media source must use one or more unique SSRCs to avoid these issues as mandated below (Section 6.2). Bullet 4 can be served by two arguments, first of all each SSRC will commonly a native media type, communicated through the RTP payload type, allowing a middlebox to do media type specific operations. The second argument is that in many contexts blind combining without additional contexts are anyway not suitable. Regarding bullet 5 this is a understood and explicitly stated applicability limitations for the method described in this document.

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. In fact this document (Section 7.2) suggest that for usage of Generic FEC (XOR-based) as defined in RFC 5109 can actually use a single media type when used with independent RTP sessions for source and repair data.

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 is 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.

6.4. RTCP

An RTP session has a single set of parameters that configure the session bandwidth, the RTCP sender and receiver fractions (e.g., via the SDP "b=RR:" and "b=RS:" lines), and the parameters of the RTP/AVPF profile [RFC4585] (e.g., trr-int) if that profile (or its secure extension, RTP/SAVPF [RFC5124]) is used. As a consequence, the RTCP reporting interval will be the same for every SSRC in an RTP session. This uniform RTCP reporting interval can result in RTCP reports being sent more often than is considered desirable for a particular media type. For example, if an audio flow is multiplexed with a high quality video flow where the session bandwidth is configured to match the video bandwidth, this can result in the RTCP packets having a

greater bandwidth allocation than the audio data rate. If the reduced minimum RTCP interval described in Section 6.2 of [RFC3550] is used in the session, which might be appropriate for video where rapid feedback is wanted, the audio sources could be required to send RTCP packets more often than they send audio data packets. This is clearly undesirable, and while the mismatch can be reduced through careful tuning of the RTCP parameters, particularly `trr_int` in RTP/AVPF sessions, it is inherent in the design of the RTCP timing rules, and affects all RTP sessions containing flows with mismatched bandwidth.

(tbd: A future version of this draft needs to provide details of the extent of this problem, recommendations for how to tune the RTCP bandwidth fraction and `trr_int`, and when the mismatch is so great that it's better to use separate RTP sessions. The recommendations will likely be different for RTP/AVP and RTP/AVPF sessions, since `trr_int` offers a potential solution that is not suitable in legacy session.)

Having multiple media types in one RTP session also results in more SSRCs being present in this RTP session. This increasing 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

The RTP Payload Format for Generic Forward Error Correction [RFC5109], and also its predecessor [RFC2733], requires some considerations, and they are different depending on what type of configuration of usage one has.

Independent RTP Sessions, i.e. where source and repair data are sent in different RTP sessions. As this mode of configuration requires different RTP session, there must be at least one RTP session for source data, this session can be one using multiple media types. The repair session only needs one RTP Payload type indicating repair data, i.e. x/ulpfec or x/parityfec depending if RFC 5109 or RFC 2733 is used. The media type in this session is not relevant and can in theory be any of the defined ones. It is recommended that one uses "Application".

In stream, using RTP Payload for Redundant Audio Data [RFC2198] combining repair and source data in the same packets. This is possible to use within a single RTP session. However, the usage and configuration of the payload types can create an issue. First of all it might be required to have one payload type per media type for the FEC repair data payload format, i.e. one for audio/ulpfec and one for text/ulpfec if audio and text are combined in an RTP session. Secondly each combination of source payload and its FEC repair data must be an explicit configured payload type. This has potential for making the limitation of RTP payload types available into a real issue.

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-01 (work in progress), August 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.

- [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.
- [RFC3551] Schulzrinne, H. and S. Casner, "RTP Profile for Audio and Video Conferences with Minimal Control", STD 65, RFC 3551, July 2003.

12.2. Informative References

- [I-D.westerlund-avtcore-multiplex-architecture]
Westerlund, M., Burman, B., Perkins, C., and H. Alvestrand, "Guidelines for using the Multiplexing Features of RTP",
draft-westerlund-avtcore-multiplex-architecture-02 (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-03 (work in progress), July 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.
- [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.
- [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.
- [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.
- [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

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: April 25, 2013

C. Perkins
University of Glasgow
V. Singh
Aalto University
October 22, 2012

RTP Congestion Control: Circuit Breakers for Unicast Sessions
draft-ietf-avtcore-rtp-circuit-breakers-01

Abstract

The Real-time Transport Protocol (RTP) is widely used in telephony, video conferencing, and telepresence applications. Such applications are often run on best-effort UDP/IP networks. If congestion control is not implemented in the applications, then network congestion will deteriorate the user's multimedia experience. This document does not propose a congestion control algorithm; rather, it defines a minimal set of "circuit-breakers". Circuit-breakers are conditions under which an RTP flow is expected to stop transmitting media to protect the network from excessive congestion. It is expected that all RTP applications running on best-effort networks will be able to run without triggering these circuit breakers in normal operation. Any future RTP congestion control specification is expected to operate within the envelope defined by these circuit breakers.

Status of this Memo

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

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

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

This Internet-Draft will expire on April 25, 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. Background	3
4. RTP Circuit Breakers for Systems Using the RTP/AVP Profile . .	6
4.1. RTP/AVP Circuit Breaker #1: Media Timeout	7
4.2. RTP/AVP Circuit Breaker #2: RTCP Timeout	8
4.3. RTP/AVP Circuit Breaker #3: Congestion	9
5. RTP Circuit Breakers for Systems Using the RTP/AVPF Profile .	11
6. Impact of RTCP XR	12
7. Impact of Explicit Congestion Notification (ECN)	12
8. Security Considerations	13
9. IANA Considerations	13
10. Acknowledgements	13
11. References	13
11.1. Normative References	13
11.2. Informative References	14
Authors' Addresses	15

1. Introduction

The Real-time Transport Protocol (RTP) [RFC3550] is widely used in voice-over-IP, video teleconferencing, and telepresence systems. Many of these systems run over best-effort UDP/IP networks, and can suffer from packet loss and increased latency if network congestion occurs. Designing effective RTP congestion control algorithms, to adapt the transmission of RTP-based media to match the available network capacity, while also maintaining the user experience, is a difficult but important problem. Many such congestion control and media adaptation algorithms have been proposed, but to date there is no consensus on the correct approach, or even that a single standard algorithm is desirable.

This memo does not attempt to propose a new RTP congestion control algorithm. Rather, it proposes a minimal set of "circuit breakers"; conditions under which there is general agreement that an RTP flow is causing serious congestion, and ought to cease transmission. It is expected that future standards-track congestion control algorithms for RTP will operate within the envelope defined by this memo.

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 RFC 2119 [RFC2119]. This interpretation of these key words applies only when written in ALL CAPS. Mixed- or lower-case uses of these key words are not to be interpreted as carrying special significance in this memo.

3. Background

We consider congestion control for unicast RTP traffic flows. This is the problem of adapting the transmission of an audio/visual data flow, encapsulated within an RTP transport session, from one sender to one receiver, so that it matches the available network bandwidth. Such adaptation needs to be done in a way that limits the disruption to the user experience caused by both packet loss and excessive rate changes. Congestion control for multicast flows is outside the scope of this memo.

Congestion control for unicast RTP traffic can be implemented in one of two places in the protocol stack. One approach is to run the RTP traffic over a congestion controlled transport protocol, for example over TCP, and to adapt the media encoding to match the dictates of the transport-layer congestion control algorithm. This is safe for

the network, but can be suboptimal for the media quality unless the transport protocol is designed to support real-time media flows. We do not consider this class of applications further in this memo, as their network safety is guaranteed by the underlying transport.

Alternatively, RTP flows can be run over a non-congestion controlled transport protocol, for example UDP, performing rate adaptation at the application layer based on RTP Control Protocol (RTCP) feedback. With a well-designed, network-aware, application, this allows highly effective media quality adaptation, but there is potential to disrupt the network's operation if the application does not adapt its sending rate in a timely and effective manner. We consider this class of applications in this memo.

Congestion control relies on monitoring the delivery of a media flow, and responding to adapt the transmission of that flow when there are signs that the network path is congested. Network congestion can be detected in one of three ways: 1) a receiver can infer the onset of congestion by observing an increase in one-way delay caused by queue build-up within the network; 2) if Explicit Congestion Notification (ECN) [RFC3168] is supported, the network can signal the presence of congestion by marking packets using ECN Congestion Experienced (CE) marks; or 3) in the extreme case, congestion will cause packet loss that can be detected by observing a gap in the received RTP sequence numbers. Once the onset of congestion is observed, the receiver has to send feedback to the sender to indicate that the transmission rate needs to be reduced. How the sender reduces the transmission rate is highly dependent on the media codec being used, and is outside the scope of this memo.

There are several ways in which a receiver can send feedback to a media sender within the RTP framework:

- o The base RTP specification [RFC3550] defines RTCP Reception Report (RR) packets to convey reception quality feedback information, and Sender Report (SR) packets to convey information about the media transmission. RTCP SR packets contain data that can be used to reconstruct media timing at a receiver, along with a count of the total number of octets and packets sent. RTCP RR packets report on the fraction of packets lost in the last reporting interval, the cumulative number of packets lost, the highest sequence number received, and the inter-arrival jitter. The RTCP RR packets also contain timing information that allows the sender to estimate the network round trip time (RTT) to the receivers. RTCP reports are sent periodically, with the reporting interval being determined by the number of participants in the session and a configured session bandwidth estimate. The interval between reports sent from each receiver tends to be on the order of a few seconds on average, and

it is randomised to avoid synchronisation of reports from multiple receivers. RTCP RR packets allow a receiver to report ongoing network congestion to the sender. However, if a receiver detects the onset of congestion partway through a reporting interval, the base RTP specification contains no provision for sending the RTCP RR packet early, and the receiver has to wait until the next scheduled reporting interval.

- o The RTCP Extended Reports (XR) [RFC3611] allow reporting of more complex and sophisticated reception quality metrics, but do not change the RTCP timing rules. RTCP extended reports of potential interest for congestion control purposes are the extended packet loss, discard, and burst metrics [RFC3611], [I-D.ietf-xrblock-rtcp-xr-discard], [I-D.ietf-xrblock-rtcp-xr-discard-rle-metrics], [I-D.ietf-xrblock-rtcp-xr-burst-gap-discard], [I-D.ietf-xrblock-rtcp-xr-burst-gap-loss]; and the extended delay metrics [I-D.ietf-xrblock-rtcp-xr-delay], [I-D.ietf-xrblock-rtcp-xr-pdv]. Other RTCP Extended Reports that could be helpful for congestion control purposes might be developed in future.
- o Rapid feedback about the occurrence of congestion events can be achieved using the Extended RTP Profile for RTCP-Based Feedback (RTP/AVPF) [RFC4585] in place of the more common RTP/AVP profile [RFC3551]. This modifies the RTCP timing rules to allow RTCP reports to be sent early, in some cases immediately, provided the average RTCP reporting interval remains unchanged. It also defines new transport-layer feedback messages, including negative acknowledgements (NACKs), that can be used to report on specific congestion events. The use of the RTP/AVPF profile is dependent on signalling, but is otherwise generally backwards compatible, as it keeps the same average RTCP reporting interval as the base RTP specification. The RTP Codec Control Messages [RFC5104] extend the RTP/AVPF profile with additional feedback messages that can be used to influence that way in which rate adaptation occurs. The dynamics of how rapidly feedback can be sent are unchanged.
- o Finally, Explicit Congestion Notification (ECN) for RTP over UDP [RFC6679] can be used to provide feedback on the number of packets that received an ECN Congestion Experienced (CE) mark. This RTCP extension builds on the RTP/AVPF profile to allow rapid congestion feedback when ECN is supported.

In addition to these mechanisms for providing feedback, the sender can include an RTP header extension in each packet to record packet transmission times. There are two methods: [RFC5450] represents the transmission time in terms of a time-offset from the RTP timestamp of

the packet, while [RFC6051] includes an explicit NTP-format sending timestamp (potentially more accurate, but a higher header overhead). Accurate sending timestamps can be helpful for estimating queuing delays, to get an early indication of the onset of congestion.

Taken together, these various mechanisms allow receivers to provide feedback on the senders when congestion events occur, with varying degrees of timeliness and accuracy. The key distinction is between systems that use only the basic RTCP mechanisms, without RTP/AVPF rapid feedback, and those that use the RTP/AVPF extensions to respond to congestion more rapidly.

4. RTP Circuit Breakers for Systems Using the RTP/AVP Profile

The feedback mechanisms defined in [RFC3550] and available under the RTP/AVP profile [RFC3551] are the minimum that can be assumed for a baseline circuit breaker mechanism that is suitable for all unicast applications of RTP. Accordingly, for an RTP circuit breaker to be useful, it needs to be able to detect that an RTP flow is causing excessive congestion using only basic RTCP features, without needing RTCP XR feedback or the RTP/AVPF profile for rapid RTCP reports.

Three potential congestion signals are available from the basic RTCP SR/RR packets and are reported for each synchronisation source (SSRC) in the RTP session:

1. The sender can estimate the network round-trip time once per RTCP reporting interval, based on the contents and timing of RTCP SR and RR packets.
2. Receivers report a jitter estimate (the statistical variance of the RTP data packet inter-arrival time) calculated over the RTCP reporting interval. Due to the nature of the jitter calculation ([RFC3550], section 6.4.4), the jitter is only meaningful for RTP flows that send a single data packet for each RTP timestamp value (i.e., audio flows, or video flows where each frame comprises one RTP packet).
3. Receivers report the fraction of RTP data packets lost during the RTCP reporting interval, and the cumulative number of RTP packets lost over the entire RTP session.

These congestion signals limit the possible circuit breakers, since they give only limited visibility into the behaviour of the network.

RTT estimates are widely used in congestion control algorithms, as a proxy for queuing delay measures in delay-based congestion control or

to determine connection timeouts. RTT estimates derived from RTCP SR and RR packets sent according to the RTP/AVP timing rules are far too infrequent to be useful though, and don't give enough information to distinguish a delay change due to routing updates from queuing delay caused by congestion. Accordingly, we cannot use the RTT estimate alone as an RTP circuit breaker.

Increased jitter can be a signal of transient network congestion, but in the highly aggregated form reported in RTCP RR packets, it offers insufficient information to estimate the extent or persistence of congestion. Jitter reports are a useful early warning of potential network congestion, but provide an insufficiently strong signal to be used as a circuit breaker.

The remaining congestion signals are the packet loss fraction and the cumulative number of packets lost. These are robust indicators of congestion in a network where packet loss is primarily due to queue overflows, although less accurate in networks where losses can be caused by non-congestive packet corruption. TCP uses packet loss as a congestion signal.

Two packet loss regimes can be observed: 1) RTCP RR packets show a non-zero packet loss fraction, while the extended highest sequence number received continues to increment; and 2) RR packets show a loss fraction of zero, but the extended highest sequence number received does not increment even though the sender has been transmitting RTP data packets. The former corresponds to the TCP congestion avoidance state, and indicates a congested path that is still delivering data; the latter corresponds to a TCP timeout, and is most likely due to a path failure. We derive circuit breaker conditions for these two loss regimes in the following.

4.1. RTP/AVP Circuit Breaker #1: Media Timeout

If RTP data packets are being sent while the corresponding RTCP RR packets report a non-increasing extended highest sequence number received, this is an indication that those RTP data packets are not reaching the receiver. This could be a short-term issue affecting only a few packets, perhaps caused by a slow-to-open firewall or a transient connectivity problem, but if the issue persists, it is a sign of a more ongoing and significant problem. Accordingly, if a sender of RTP data packets receives two or more consecutive RTCP RR packets from the same receiver that correspond to its transmission, and have a non-increasing extended highest sequence number received field (i.e., at least three RTCP RR packets that report the same value in the extended highest sequence number received field, when the sender has sent data packets that would have caused an increase in the reported value of the extended highest sequence number

received if they had reached the receiver), then that sender SHOULD cease transmission. What it means to cease transmission depends on the application, but the intention is that the application will stop sending RTP data packets until the user makes an explicit attempt to restart the call (RTP flows halted by the circuit breaker SHOULD NOT be restarted automatically unless the sender has received information that the congestion has dissipated).

Systems that usually send at a high data rate, but that can reduce their data rate significantly (i.e., by at least a factor of ten), MAY first reduce their sending rate to this lower value to see if this resolves the congestion, but MUST then cease transmission if the problem does not resolve itself within a further two RTCP reporting intervals. An example of this might be a video conferencing system that backs off to sending audio only, before completely dropping the call. If such a reduction in sending rate resolves the congestion problem, the sender MAY gradually increase the rate at which it sends data after a reasonable amount of time has passed, provided it takes care not to cause the problem to recur ("reasonable" is intentionally not defined here).

The choice of two RTCP reporting intervals is to give enough time for transient problems to resolve themselves, but to stop problem flows quickly enough to avoid causing serious ongoing network congestion. A single RTCP report showing no reception could be caused by numerous transient faults, and so will not cease transmission. Waiting for more than two RTCP reports before stopping a flow might avoid some false positives, but would lead to problematic flows running for a long time before being cut off.

4.2. RTP/AVP Circuit Breaker #2: RTCP Timeout

In addition to media timeouts, as were discussed in Section 4.1, an RTP session has the possibility of an RTCP timeout. This can occur when RTP data packets are being sent, but there are no RTCP reports returned from the receiver. This is either due to a failure of the receiver to send RTCP reports, or a failure of the return path that is preventing those RTCP reporting from being delivered.

According to RFC 3550 [RFC3550], any participant that has not sent an RTCP packet within the last two RTCP intervals is removed from the sender list. Therefore, an RTP sender SHOULD cease transmission if it does not receive a single RTCP RR packet and during this period has sent 3 RTCP SR packets to the RTP receiver. Similarly, the same circuit breaker rule applies to an RTCP receiver which has not received a single SR packet, and in the corresponding period it has sent 3 RTCP RR packets. What it means to cease transmission depends on the application, but the intention is that the application will

stop sending RTP data packets until the user makes an explicit attempt to restart the call (RTP flows halted by the circuit breaker SHOULD NOT be restarted automatically unless the sender has received information that the congestion has dissipated).

4.3. RTP/AVP Circuit Breaker #3: Congestion

If RTP data packets are being sent, and the corresponding RTCP RR packets show non-zero packet loss fraction and increasing extended highest sequence number received, then those RTP data packets are arriving at the receiver, but some degree of congestion is occurring. The RTP/AVP profile [RFC3551] states that:

If best-effort service is being used, RTP receivers SHOULD monitor packet loss to ensure that the packet loss rate is within acceptable parameters. Packet loss is considered acceptable if a TCP flow across the same network path and experiencing the same network conditions would achieve an average throughput, measured on a reasonable time scale, that is not less than the RTP flow is achieving. This condition can be satisfied by implementing congestion control mechanisms to adapt the transmission rate (or the number of layers subscribed for a layered multicast session), or by arranging for a receiver to leave the session if the loss rate is unacceptably high.

The comparison to TCP cannot be specified exactly, but is intended as an "order-of-magnitude" comparison in time scale and throughput. The time scale on which TCP throughput is measured is the round-trip time of the connection. In essence, this requirement states that it is not acceptable to deploy an application (using RTP or any other transport protocol) on the best-effort Internet which consumes bandwidth arbitrarily and does not compete fairly with TCP within an order of magnitude.

The phrase "order of magnitude" in the above means within a factor of ten, approximately. In order to implement this, it is necessary to estimate the throughput a TCP connection would achieve over the path. For a long-lived TCP Reno connection, Padhye et al. [Padhye] showed that the throughput can be estimated using the following equation:

$$X = \frac{S}{R \cdot \sqrt{2 \cdot b \cdot p / 3} + (t_{RTO} \cdot (3 \cdot \sqrt{3 \cdot b \cdot p / 8} \cdot p \cdot (1 + 32 \cdot p^2)))}$$

where:

X is the transmit rate in bytes/second.

s is the packet size in bytes. If data packets vary in size, then the average size is to be used.

R is the round trip time in seconds.

p is the loss event rate, between 0 and 1.0, of the number of loss events as a fraction of the number of packets transmitted.

t_RTO is the TCP retransmission timeout value in seconds, approximated by setting t_RTO = 4*R.

b is the number of packets acknowledged by a single TCP acknowledgement ([RFC3448] recommends the use of b=1 since many TCP implementations do not use delayed acknowledgements).

This is the same approach to estimated TCP throughput that is used in [RFC3448]. Under conditions of low packet loss, this formula can be approximated as follows with reasonable accuracy:

$$X = \frac{s}{R * \sqrt{p*2/3}}$$

It is RECOMMENDED that this simplified throughput equation be used, since the reduction in accuracy is small, and it is much simpler to calculate than the full equation.

Given this TCP equation, two parameters need to be estimated and reported to the sender in order to calculate the throughput: the round trip time, R, and the loss event rate, p (the packet size, s, is known to the sender). The round trip time can be estimated from RTCP SR and RR packets. This is done too infrequently for accurate statistics, but is the best that can be done with the standard RTCP mechanisms.

RTCP RR packets contain the packet loss fraction, rather than the loss event rate, so p cannot be reported (TCP typically treats the loss of multiple packets within a single RTT as one loss event, but RTCP RR packets report the overall fraction of packets lost, not caring about when the losses occurred). Using the loss fraction in place of the loss event rate can overestimate the loss. We believe that this overestimate will not be significant, given that we are only interested in order of magnitude comparison ([Floyd] section 3.2.1 shows that the difference is small for steady-state conditions and random loss, but using the loss fraction is more conservative in the case of bursty loss).

The congestion circuit breaker is therefore: when RTCP RR packets are received, estimate the TCP throughput using the simplified equation above, and the measured R , p (approximated by the loss fraction), and s . Compare this with the actual sending rate. If the actual sending rate is more than ten times the estimated sending rate derived from the TCP throughput equation for two consecutive RTCP reporting intervals, the sender SHOULD cease transmission. What it means to cease transmission depends on the application, but the intention is that the application will stop sending RTP data packets until the user makes an explicit attempt to restart the call (RTP flows halted by the circuit breaker SHOULD NOT be restarted automatically unless the sender has received information that the congestion has dissipated).

Systems that usually send at a high data rate, but that can reduce their data rate significantly (i.e., by at least a factor of ten), MAY first reduce their sending rate to this lower value to see if this resolves the congestion, but MUST then cease transmission if the problem does not resolve itself within a further two RTCP reporting intervals. An example of this might be a video conferencing system that backs off to sending audio only, before completely dropping the call. If such a reduction in sending rate resolves the congestion problem, the sender MAY gradually increase the rate at which it sends data after a reasonable amount of time has passed, provided it takes care not to cause the problem to recur ("reasonable" is intentionally not defined here).

As in Section 4.1, we use two reporting intervals to avoid triggering the circuit breaker on transient failures. This circuit breaker is a worst-case condition, and congestion control needs to be performed to keep well within this bound. It is expected that the circuit breaker will only be triggered if the usual congestion control fails for some reason.

5. RTP Circuit Breakers for Systems Using the RTP/AVPF Profile

Use of the Extended RTP Profile for RTCP-based Feedback (RTP/AVPF) [RFC4585] allows receivers to send early RTCP reports in some cases, to inform the sender about particular events in the media stream. There are several use cases for such early RTCP reports, including providing rapid feedback to a sender about the onset of congestion.

Receiving rapid feedback about congestion events potentially allows congestion control algorithms to be more responsive, and to better adapt the media transmission to the limitations of the network. It is expected that many RTP congestion control algorithms will adopt the RTP/AVPF profile for this reason, defining new transport layer

feedback reports that suit their requirements. Since these reports are not yet defined, and likely very specific to the details of the congestion control algorithm chosen, they cannot be used as part of the generic RTP circuit breaker.

If the extension for Reduced-Size RTCP [RFC5506] is not used, early RTCP feedback packets sent according to the RTP/AVPF profile will be compound RTCP packets that include an RTCP SR/RR packet. That RTCP SR/RR packet MUST be processed as if it were sent as a regular RTCP report and counted towards the circuit breaker conditions specified in Section 4.1 and Section 4.3 of this memo. This will potentially make the RTP circuit breaker fire earlier than it would if the RTP/AVPF profile was not used.

Reduced-size RTCP reports sent under to the RTP/AVPF early feedback rules that do not contain an RTCP SR or RR packet MUST be ignored by the RTP circuit breaker (they do not contain the information used by the circuit breaker algorithm). In this case, the circuit breaker will only use the information contained in the periodic RTCP SR/RR packets. This allows the use of low-overhead early RTP/AVPF feedback without triggering the RTP circuit breaker, and so is suitable for RTP congestion control algorithms that need to quickly report loss events in between regular RTCP reports.

6. Impact of RTCP XR

RTCP Extended Report (XR) blocks provide additional reception quality metrics, but do not change the RTCP timing rules. Some of the RTCP XR blocks provide information that might be useful for congestion control purposes, others provided non-congestion-related metrics. The presence of RTCP XR blocks in a compound RTCP packet does not affect the RTP circuit breaker algorithm; for consistency and ease of implementation, only the reception report blocks contained in RTCP SR or RR packets are used by the RTP circuit breaker algorithm.

7. Impact of Explicit Congestion Notification (ECN)

ECN-CE marked packets SHOULD be treated as if it were lost for the purposes of congestion control, when determining the optimal media sending rate for an RTP flow. If an RTP sender has negotiated ECN support for an RTP session, and has successfully initiated ECN use on the path to the receiver [RFC6679], then ECN-CE marked packets SHOULD be treated as if they were lost when calculating if the congestion-based RTP circuit breaker (Section 4.3) has been met.

The use of ECN for RTP flows does not affect the media timeout RTP

circuit breaker (Section 4.1) or the RTCP timeout circuit breaker (Section 4.2), since these are both connectivity checks that simply determinate if any packets are being received.

8. Security Considerations

The security considerations of [RFC3550] apply.

If the RTP/AVPF profile is used to provide rapid RTCP feedback, the security considerations of [RFC4585] apply. If ECN feedback for RTP over UDP/IP is used, the security considerations of [RFC6679] apply.

If non-authenticated RTCP reports are used, an on-path attacker can trivially generate fake RTCP packets that indicate high packet loss rates, causing the circuit breaker to trigger and disrupting an RTP session. This is somewhat more difficult for an off-path attacker, due to the need to guess the randomly chosen RTP SSRC value and the RTP sequence number. This attack can be avoided if RTCP packets are authenticated, for example using the Secure RTP profile [RFC3711].

9. IANA Considerations

There are no actions for IANA.

10. Acknowledgements

The authors would like to thank Harald Alvestrand, Randell Jesup, Matt Mathis, and Abheek Saha for their valuable feedback.

11. References

11.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC3448] Handley, M., Floyd, S., Padhye, J., and J. Widmer, "TCP Friendly Rate Control (TFRC): Protocol Specification", RFC 3448, January 2003.
- [RFC3550] Schulzrinne, H., Casner, S., Frederick, R., and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications", STD 64, RFC 3550, July 2003.

- [RFC3551] Schulzrinne, H. and S. Casner, "RTP Profile for Audio and Video Conferences with Minimal Control", STD 65, RFC 3551, July 2003.
- [RFC3611] Friedman, T., Caceres, R., and A. Clark, "RTP Control Protocol Extended Reports (RTCP XR)", RFC 3611, November 2003.
- [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.

11.2. Informative References

- [Floyd] Floyd, S., Handley, M., Padhye, J., and J. Widmer, "Equation-Based Congestion Control for Unicast Applications", Proc. ACM SIGCOMM 2000, DOI 10.1145/347059.347397, August 2000.
- [I-D.ietf-xrblock-rtcp-xr-burst-gap-discard] Clark, A., Huang, R., and W. Wu, "RTP Control Protocol(RTCP) Extended Report (XR) Block for Discard Count metric Reporting", draft-ietf-xrblock-rtcp-xr-burst-gap-discard-06 (work in progress), October 2012.
- [I-D.ietf-xrblock-rtcp-xr-burst-gap-loss] Clark, A., Zhang, S., Zhao, J., and W. Wu, "RTP Control Protocol (RTCP) Extended Report (XR) Block for Burst/Gap Loss metric Reporting", draft-ietf-xrblock-rtcp-xr-burst-gap-loss-04 (work in progress), October 2012.
- [I-D.ietf-xrblock-rtcp-xr-delay] Clark, A., Gross, K., and W. Wu, "RTP Control Protocol (RTCP) Extended Report (XR) Block for Delay metric Reporting", draft-ietf-xrblock-rtcp-xr-delay-10 (work in progress), October 2012.
- [I-D.ietf-xrblock-rtcp-xr-discard] Clark, A., Zorn, G., and W. Wu, "RTP Control Protocol (RTCP) Extended Report (XR) Block for Discard Count metric Reporting", draft-ietf-xrblock-rtcp-xr-discard-09 (work in progress), October 2012.
- [I-D.ietf-xrblock-rtcp-xr-discard-rle-metrics] Ott, J., Singh, V., and I. Curcio, "RTP Control Protocol

(RTCP) Extended Reports (XR) for Run Length Encoding (RLE) of Discarded Packets",
draft-ietf-xrblock-rtcp-xr-discard-rle-metrics-04 (work in progress), July 2012.

[I-D.ietf-xrblock-rtcp-xr-pdv]

Clark, A. and W. Wu, "RTP Control Protocol (RTCP) Extended Report (XR) Block for Packet Delay Variation Metric Reporting", draft-ietf-xrblock-rtcp-xr-pdv-08 (work in progress), September 2012.

[Padhye]

Padhye, J., Firoiu, V., Towsley, D., and J. Kurose, "Modeling TCP Throughput: A Simple Model and its Empirical Validation", Proc. ACM SIGCOMM 1998, DOI 10.1145/285237.285291, August 1998.

[RFC3168]

Ramakrishnan, K., Floyd, S., and D. Black, "The Addition of Explicit Congestion Notification (ECN) to IP", RFC 3168, September 2001.

[RFC3711]

Baughner, M., McGrew, D., Naslund, M., Carrara, E., and K. Norrman, "The Secure Real-time Transport Protocol (SRTP)", RFC 3711, March 2004.

[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.

[RFC5450]

Singer, D. and H. Desineni, "Transmission Time Offsets in RTP Streams", RFC 5450, March 2009.

[RFC5506]

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

[RFC6051]

Perkins, C. and T. Schierl, "Rapid Synchronisation of RTP Flows", RFC 6051, November 2010.

[RFC6679]

Westerlund, M., Johansson, I., Perkins, C., O'Hanlon, P., and K. Carlberg, "Explicit Congestion Notification (ECN) for RTP over UDP", RFC 6679, August 2012.

Authors' Addresses

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

Email: csp@csp Perkins.org

Varun Singh
Aalto University
School of Electrical Engineering
Otakaari 5 A
Espoo, FIN 02150
Finland

Email: varun@comnet.tkk.fi
URI: <http://www.netlab.tkk.fi/~varun/>

AVTCORE
Internet-Draft
Updates: 3550 (if approved)
Intended status: Standards Track
Expires: April 25, 2013

J. Lennox
Vidyo
M. Westerlund
Ericsson
October 22, 2012

Real-Time Transport Protocol (RTP) Considerations for Endpoints Sending
Multiple Media Streams
draft-lennox-avtcore-rtp-multi-stream-01

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 April 25, 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. Issue Cases	4
4.1. Cascaded Multi-party Conference with Source Projecting Mixers	5
5. Multi-Stream Endpoint RTP Media Recommendations	5
6. Multi-Stream Endpoint RTCP Recommendations	5
6.1. RTCP Reporting Requirement	6
6.2. Initial Reporting Interval	6
6.3. Compound RTCP Packets	6
7. RTCP Bandwidth Considerations When Sources have Greatly-Differing Bandwidths	7
8. Grouping of RTCP Reception Statistics and Other Feedback	7
8.1. Semantics and Behavior of Reporting Groups	8
8.2. RTCP Source Description (SDES) item for Reporting Groups	9
8.3. SDP signaling for Reporting Groups	9
8.4. Bandwidth Benefits of RTCP Reporting Groups	9
8.5. Consequences of RTCP Reporting Groups	10
9. Security Considerations	10
10. Open Issues	11
11. IANA Considerations	11
12. References	11
12.1. Normative References	11
12.2. Informative References	12
Appendix A. Changes From Earlier Versions	13
A.1. Changes From Draft -00	13
Authors' Addresses	13

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.

The document starts with terminology and some use cases where multiple sources will occur. This is followed by some case studies to try to identify issues that exist and need considerations. This is followed by RTP and RTCP recommendations to resolve issues. Next are security considerations and remaining open issues.

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.ietf-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, see Section 3.6 [I-D.westerlund-avtcore-rtp-topologies-update], 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. Issue Cases

This section tries to illustrate a few cases that have been determined to cause issues.

4.1. Cascaded Multi-party Conference with Source Projecting Mixers

This issue case tries to illustrate the effect of having multiple SSRCs sent by an endpoint, by considering the traffic between two source-projecting mixers in a large multi-party conference.

For concreteness, consider a 200-person conference, where 16 sources are viewed at any given time. Assuming participants are distributed evenly among the mixers, each mixer would have 100 sources "behind" (projected through) it, of which at any given time eight are active senders. Thus, the RTP session between the mixers consists of two endpoints, but 200 sources.

The RTCP bandwidth implications of this scenario are discussed further in Section 8.4.

(TBD: Other examples?)

5. 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.

6. Multi-Stream Endpoint RTCP Recommendations

This section contains a number of different RTCP clarifications or recommendations that enables more efficient and simpler behavior without loss of functionality.

The Real-Time Transport Control Protocol (RTCP) is defined in Section 6 of [RFC3550], but it is largely documented in terms of "participants". In many cases, the specification's recommendations for "participants" should be interpreted as applying to individual media streams, rather than to endpoints. This section describes several concrete cases where this applies.

6.1. RTCP Reporting Requirement

For each of an endpoint's media streams, whether or not it is currently sending media, 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 8.)

6.2. Initial Reporting Interval

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." This applies to individual media sources as well. Thus, endpoints MAY send an initial RTCP packet for an SSRC immediately upon adding it to a unicast session.

This allowance also applies, as written, when initially joining a unicast session. However, in this case some caution should be exercised if the end-point or mixer has a large number of sources (SSRCs) as this can create a significant burst. How big an issue this depends on the number of source to send initial SR or RR and Session Description CNAME items for in relation to the RTCP bandwidth. TBD: Maybe some recommendation here?

6.3. Compound RTCP Packets

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 a 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 an 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.)

Due to RTCP's randomization of reporting times, there is a fair bit of tolerance in precisely when an endpoint schedules RTCP to be sent. Thus, one potential way of implementing this recommendation would be to randomize all of an endpoint's sources together, with a single randomization schedule, so an MTU's worth of RTCP all comes out simultaneously.

TBD: Multiplexing RTCP packets from multiple different sources may require some adjustment to the calculation of RTCP's `avg_rtcp_size`, as the RTCP group interval is proportional to `avg_rtcp_size` times the group size.

7. RTCP Bandwidth Considerations When Sources have Greatly-Differing Bandwidths

it is possible for an RTP session to carry sources of greatly differing bandwidths. One example is the scenario of [I-D.ietf-avtcore-multi-media-rtp-session], when audio and video are sent in the same session. However, this can occur even within a single media type, for example a video session carrying both 5 fps QCIF and 60 fps 1080p HD video, or an audio session carrying both G.729 and L24/48000/6 audio.

TBD: recommend how RTCP bandwidths should be chosen in these scenarios. Likely, these recommendations will be different for sessions using AVPF-based profiles (where the `trr-int` parameter is available) than for those using AVP.

8. Grouping of RTCP Reception Statistics and Other Feedback

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 has S senders in each reporting interval, there will either be $N \times S$ 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, this document defines a new RTCP mechanism, Reporting Groups, to indicate sources which originate from the same endpoint, and which therefore would have identical reception reports.

8.1. Semantics and Behavior of Reporting Groups

An RTCP Reporting Group indicates that a set of sources originate from a single entity in an RTP session, and therefore all the sources in the group's view of the network is identical. Typically, a Reporting Group corresponds to a physical entity in the network.

If reporting groups are in use, an endpoint **MUST NOT** send reception reports from one source in a reporting group about another one in the same group ("self-reports"). Similarly, an endpoint **MUST NOT** send reception reports about a remote media source from more than one sources in a reporting group ("cross-reports"). Instead, it **MUST** 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 **MUST 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 a local audio source, and reports about remote video sources from a local video source), or it **MAY** choose a single local source for all its reports. This reporting source **MUST** also be the source for any AVPF Feedback [RFC4585] or Extended Report (XR) [RFC3611] packets about the corresponding remote sources as well. If a reporting source leaves the session (i.e., if it sends a BYE, or leaves the group without sending BYE under the rules of [RFC3550] section 6.3.7), another reporting source **MUST** be chosen, if the sources it was reporting on are still in the session.

If AVPF feedback is in use, a reporting source MAY send immediate or early feedback at any point when any member of the reporting group could validly do so.

An endpoint SHOULD NOT create single-source reporting groups, unless it is anticipated that the group might have additional sources added to it in the future.

8.2. RTCP Source Description (SDES) item for Reporting Groups

A new Source Description (SDES) item, "RGRP", indicates that a source is a member of a specified reporting group. Syntactically, its format is the same as the RTCP CNAME [RFC6222], and MUST be chosen with the same global-uniqueness and privacy considerations as CNAME.

Every source which belongs to a reporting group MUST include an RGRP SDES item in an SDES packet, alongside its CNAME, in every compound RTCP packet in which it sends an RR or SR packet. (I.e., in every RTCP packet it sends, unless Reduced-Size RTCP [RFC5506] is in use.)

8.3. SDP signaling for Reporting Groups

TBD

8.4. Bandwidth Benefits of RTCP Reporting Groups

To understand the benefits of RTCP reporting groups, consider the scenario described in Section 4.1. This scenario describes an environment in which the two endpoints in a session each have a hundred sources, of which eight each are sending within any given reporting interval.

For ease of analysis, we can make the simplifying approximation that the duration of the RTCP reporting interval is equal to the total size of the RTCP packets sent during an RTCP interval, divided by the RTCP bandwidth. (This will be approximately true in scenarios where the bandwidth is not so high that the minimum RTCP interval is reached.) For further simplification, we can assume RTCP senders are following the recommendations of Section 6.3; thus, the per-packet transport-layer overhead will be small relative to the RTCP data. Thus, only the actual RTCP data itself need be considered.

In a report interval in this scenario, there will, as a baseline, be 200 SDES packets, 184 RR packets, and 16 SR packets. This amounts to approximately 6.5 kB of RTCP per report interval, assuming 16-byte CNAMEs and no other SDES information.

Using naive everyone-reports-on-every-sender feedback rules, each of the 184 receivers will send 16 report blocks, and each of the 16 senders will send 15. This amounts to approximately 76 kB of report block traffic per interval; 92% of RTCP traffic consists of report blocks.

If reporting groups are used, however, there is only 0.4 kB of reports per interval, with no loss of useful information. Additionally, there will be (assuming 16-byte RGRPs as well) an additional 3.2 kB per cycle of RGRP SDES items. Put another way, the naive case's reporting interval is approximately 7.5 times longer than if reporting groups are in use.

8.5. Consequences of RTCP Reporting Groups

The RTCP traffic generated by receivers using RTCP Reporting Groups might appear, to observers unaware of these semantics, to be generated by receivers who are experiencing a network disconnection, as the non-reporting sources appear not to be receiving a given sender at all.

This could be a potentially critical problem for such a sender using RTCP for congestion control, as such a sender might think that it is sending so much traffic that it is causing complete congestion collapse.

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 such backward compatibility issues would actually cause trouble in practice.

9. 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.

10. Open Issues

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

1. Further clarifications on how to handle the RTCP scheduler when sending multiple sources in one compound packet.
2. How should the use of reporting groups be signaled in SDP?
3. How should the RTCP avg_rtcp_size be calculated when RTCP packets are routinely multiplexed among multiple RTCP senders?
4. Do we need to provide a recommendation for unicast session joiners with many sources to not use 0 initial minimal interval from bit-rate burst perspective?

11. IANA Considerations

This document adds an additional SDES type to the IANA "RTCP SDES Item Types" Registry, as follows:

Value	Abbrev	Name	Reference
TBD	RGRP	Reporting Group	[RFCXXXX]

Figure 1: Initial Contents of IANA Source Attribute Registry

(Note to the RFC-Editor: please replace "TBD" with the IANA-assigned value, and "XXXX" with the number of this document, prior to publication as an RFC.)

12. References

12.1. 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.
- [RFC6222] Begen, A., Perkins, C., and D. Wing, "Guidelines for Choosing RTP Control Protocol (RTCP) Canonical Names (CNAMEs)", RFC 6222, April 2011.

12.2. Informative References

- [I-D.ietf-avtcore-multi-media-rtp-session]
Westerlund, M., Perkins, C., and J. Lennox, "Multiple Media Types in an RTP Session",
draft-ietf-avtcore-multi-media-rtp-session-01 (work in progress), October 2012.
- [I-D.ietf-clue-framework]
Romanow, A., Duckworth, M., Pepperell, A., and B. Baldino, "Framework for Telepresence Multi-Streams",
draft-ietf-clue-framework-07 (work in progress), October 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-01 (work in progress), August 2012.
- [I-D.westerlund-avtcore-rtp-topologies-update]
Westerlund, M. and S. Wenger, "RTP Topologies",
draft-westerlund-avtcore-rtp-topologies-update-01 (work in progress), October 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.

Appendix A. Changes From Earlier Versions

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

A.1. Changes From Draft -00

- o Added the Reporting Group semantic to explicitly indicate which sources come from a single endpoint, rather than leaving it implicit.
- o Specified that Reporting Group semantics (as they now are) apply to AVPF and XR, as well as to RR/SR report blocks.
- o Added a description of the cascaded source-projecting mixer, along with a calculation of its RTCP overhead if reporting groups are not in use.
- o Gave some guidance on how the flexibility of RTCP randomization allows some freedom in RTCP multiplexing.
- o Clarified the language of several of the recommendations.
- o Added an open issue discussing how `avg_rtcp_size` should be calculated for multiplexed RTCP.
- o Added an open issue discussing RTCP bandwidths should be chosen for sessions where source bandwidths greatly differ.

Authors' Addresses

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

Email: jonathan@vidyo.com

Magnus Westerlund
Ericsson
Farogatan 6
SE-164 80 Kista
Sweden

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

Network Working Group
Internet-Draft
Obsoletes: 6222 (if approved)
Intended status: Standards Track
Expires: April 17, 2013

E. Rescorla
RTFM, Inc.
A. Begen
Cisco
October 14, 2012

Guidelines for Choosing RTP Control Protocol (RTCP)
Canonical Names (CNAMEs)
draft-rescorla-avtcore-6222bis-00

Abstract

The RTP Control Protocol (RTCP) Canonical Name (CNAME) is a persistent transport-level identifier for an RTP endpoint. While the Synchronization Source (SSRC) identifier of an RTP endpoint may change if a collision is detected or when the RTP application is restarted, its RTCP CNAME is meant to stay unchanged, so that RTP endpoints can be uniquely identified and associated with their RTP media streams.

For proper functionality, RTCP CNAMEs should be unique within the participants of an RTP session. However, the existing guidelines for choosing the RTCP CNAME provided in the RTP standard are insufficient to achieve this uniqueness. RFC 6222 was published to update those guidelines to allow endpoints to choose unique RTCP CNAMEs. Unfortunately, later investigations showed that some parts of the new algorithms were unnecessarily complicated and/or ineffective. This document specifies a replacement for those parts.

Status of This Memo

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

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

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

This Internet-Draft will expire on April 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	3
2. Requirements Notation	3
3. Deficiencies with Earlier Guidelines for Choosing an RTCP CNAME	3
4. Choosing an RTCP CNAME	4
4.1. Persistent RTCP CNAMEs versus Per-Session RTCP CNAMEs	4
4.2. Requirements	5
5. Procedure to Generate a Unique Identifier	6
6. Security Considerations	7
6.1. Considerations on Uniqueness of RTCP CNAMEs	7
6.2. Session Correlation Based on RTCP CNAMEs	7
7. Acknowledgments	8
8. References	8
8.1. Normative References	8
8.2. Informative References	8

1. Introduction

In Section 6.5.1 of the RTP specification, [RFC3550], there are a number of recommendations for choosing a unique RTCP CNAME for an RTP endpoint. However, in practice, some of these methods are not guaranteed to produce a unique RTCP CNAME. [RFC6222] updated the guidelines for choosing RTCP CNAMEs, superseding those presented in Section 6.5.1 of [RFC3550]. Unfortunately, some parts of the new algorithms are rather complicated and also produce RTCP CNAMEs which in some cases are potentially linkable over multiple RTCP sessions even if a new RTCP CNAME is generated for each session. This document specifies a replacement for the algorithm in Section 5 of [RFC6222], which does not have this limitation and is also simpler to implement.

For a discussion on the linkability of RTCP CNAMEs produced by [RFC6222], refer to [I-D.rescorla-avtcore-random-cname].

2. Requirements Notation

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 [RFC2119].

3. Deficiencies with Earlier Guidelines for Choosing an RTCP CNAME

The recommendation in [RFC3550] is to generate an RTCP CNAME of the form "user@host" for multiuser systems, or "host" if the username is not available. The "host" part is specified to be the fully qualified domain name (FQDN) of the host from which the real-time data originates. While this guidance was appropriate at the time [RFC3550] was written, FQDNs are no longer necessarily unique and can sometimes be common across several endpoints in large service provider networks. This document replaces the use of FQDN as an RTCP CNAME by alternative mechanisms.

IPv4 addresses are also suggested for use in RTCP CNAMEs in [RFC3550], where the "host" part of the RTCP CNAME is the numeric representation of the IPv4 address of the interface from which the RTP data originates. As noted in [RFC3550], the use of private network address space [RFC1918] can result in hosts having network addresses that are not globally unique. Additionally, this shared use of the same IPv4 address can also occur with public IPv4 addresses if multiple hosts are assigned the same public IPv4 address and connected to a Network Address Translation (NAT) device [RFC3022]. When multiple hosts share the same IPv4 address, whether private or public, using the IPv4 address as the RTCP CNAME leads to

RTCP CNAMEs that are not necessarily unique.

It is also noted in [RFC3550] that if hosts with private addresses and no direct IP connectivity to the public Internet have their RTP packets forwarded to the public Internet through an RTP-level translator, they could end up having non-unique RTCP CNAMEs. The suggestion in [RFC3550] is that such applications provide a configuration option to allow the user to choose a unique RTCP CNAME; this technique puts the burden on the translator to translate RTCP CNAMEs from private addresses to public addresses if necessary to keep private addresses from being exposed. Experience has shown that this does not work well in practice.

4. Choosing an RTCP CNAME

It is difficult, and in some cases impossible, for a host to determine if there is a NAT between itself and its RTP peer. Furthermore, even some public IPv4 addresses can be shared by multiple hosts in the Internet. Using the numeric representation of the IPv4 address as the "host" part of the RTCP CNAME is NOT RECOMMENDED.

4.1. Persistent RTCP CNAMEs versus Per-Session RTCP CNAMEs

The RTCP CNAME can be either persistent across different RTP sessions for an RTP endpoint or unique per session, meaning that an RTP endpoint chooses a different RTCP CNAME for each RTP session.

An RTP endpoint that is emitting multiple related RTP streams that require synchronization at the other endpoint(s) MUST use the same RTCP CNAME for all streams that are to be synchronized. This requires a short-term persistent RTCP CNAME that is common across several RTP streams, and potentially across several related RTP sessions. A common example of such use occurs when lip-syncing audio and video streams in a multimedia session, where a single participant has to use the same RTCP CNAME for its audio RTP session and for its video RTP session. Another example might be to synchronize the layers of a layered audio codec, where the same RTCP CNAME has to be used for each layer.

A longer-term persistent RTCP CNAME is sometimes useful to facilitate third-party monitoring, consistent with [RFC3550]. One such use might be to allow network management tools to correlate the ongoing quality of service for a participant across multiple RTP sessions for fault diagnosis, and to understand long-term network performance statistics. An implementation that wishes to discourage this type of third-party monitoring can generate a unique RTCP CNAME for each RTP session, or group of related RTP sessions, that it joins. Such a

per-session RTCP CNAME cannot be used for traffic analysis, and so provides some limited form of privacy (note that there are non-RTP means that can be used by a third party to correlate RTP sessions, so the use of per-session RTCP CNAMEs will not prevent a determined traffic analyst from monitoring such sessions).

This memo defines several different ways by which an implementation can choose an RTCP CNAME. It is possible, and legitimate, for independent implementations to make different choices of RTCP CNAME when running on the same host. This can hinder third-party monitoring, unless some external means is provided to configure a persistent choice of RTCP CNAME for those implementations.

Note that there is no backwards compatibility issue (with [RFC3550]-compatible implementations) introduced in this memo, since the RTCP CNAMEs are opaque strings to remote peers.

4.2. Requirements

RTP endpoints will choose to generate RTCP CNAMEs that are persistent or per-session. An RTP endpoint that wishes to generate a persistent RTCP CNAME MUST use one of the following two methods:

- o To produce a long-term persistent RTCP CNAME, an RTP endpoint MUST generate and store a Universally Unique Identifier (UUID) [RFC4122] for use as the "host" part of its RTCP CNAME. The UUID MUST be version 1, 2, or 4, as described in [RFC4122], with the "urn:uuid:" stripped, resulting in a 36-octet printable string representation.
- o To produce a short-term persistent RTCP CNAME, an RTP endpoint MUST either (a) use the numeric representation of the layer-2 (Media Access Control (MAC)) address of the interface that is used to initiate the RTP session as the "host" part of its RTCP CNAME or (b) generate and use an identifier by following the procedure described in Section 5. In either case, the procedure is performed once per initialization of the software. After obtaining an identifier by doing (a) or (b), the least significant 48 bits are converted to the standard colon-separated hexadecimal format [RFC5342], e.g., "00:23:32:af:9b:aa", resulting in a 17-octet printable string representation.

In the two cases above, the "user@" part of the RTCP CNAME MAY be omitted on single-user systems and MAY be replaced by an opaque token on multi-user systems, to preserve privacy.

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 uniqueness), 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.

It is believed that obtaining uniqueness (with a high probability) is an important property that requires careful evaluation of the method. This document provides a number of methods, at least one of which would be suitable for all deployment scenarios. This document therefore does not provide for the implementor to define and select an alternative method.

A future specification might define an alternative method for generating RTCP CNAMEs, as long as the proposed method has appropriate uniqueness and there is consistency between the methods used for multiple RTP sessions that are to be correlated. However, such a specification needs to be reviewed and approved before deployment.

The mechanisms described in this document are to be used to generate RTCP CNAMEs, and they are not to be used for generating general-purpose unique identifiers.

5. Procedure to Generate a Unique Identifier

The algorithm described below is intended to be used for locally generated unique identifiers. It is based on simply generating a cryptographically pseudorandom value [RFC4086]. This value **MUST** be at least 96 bits but **MAY** be longer.

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 than the algorithm described in Section 5 of [RFC6222]. 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.

6. Security Considerations

The security considerations of [RFC3550] apply to this memo.

6.1. Considerations on Uniqueness of RTCP CNAMEs

The considerations in this section apply to random RTCP CNAMEs.

The recommendations given in this document for RTCP CNAME generation ensure that a set of cooperating participants in an RTP session will, with very high probability, have unique RTCP CNAMEs. However, neither [RFC3550] nor this document provides any way to ensure that participants will choose RTCP CNAMEs appropriately, and thus implementations MUST NOT rely on the uniqueness of CNAMEs for any essential security services. This is consistent with [RFC3550], which does not require that RTCP CNAMEs are unique within a session but instead says that condition SHOULD hold. As described in the Security Considerations section of [RFC3550], because each participant in a session is free to choose its own RTCP CNAME, they can do so in such a way as to impersonate another participant. That is, participants are trusted to not impersonate each other. No recommendation for generating RTCP CNAMEs can prevent this impersonation, because an attacker can neglect the stipulation. Secure RTP (SRTP) [RFC3711] keeps unauthorized entities out of an RTP session, but it does not aim to prevent impersonation attacks from unauthorized entities.

Because of the properties of the PRNG, there is no significant privacy/linkability difference between long and short RTCP CNAMEs. However, the requirement to generate unique RTCP CNAMEs implies a certain minimum length. A length of 96 bits allows on the order of 2^{40} RTCP CNAMEs globally before there is a large chance of collision (there is about a 50% chance of one collision after 2^{48} RTCP CNAMEs).

6.2. Session Correlation Based on RTCP CNAMEs

In some environments, notably telephony, a fixed RTCP CNAME value allows separate RTP sessions to be correlated and eliminates the obfuscation provided by IPv6 privacy addresses [RFC4941] or IPv4 Network Address Port Translation (NAPT) [RFC3022]. SRTP [RFC3711] can help prevent such correlation by encrypting Secure RTCP (SRTCP), but it should be noted that SRTP only mandates SRTCP integrity protection (not encryption). Thus, RTP applications used in such environments should consider encrypting their SRTCP or generate a per-session RTCP CNAME as discussed in Section 4.1.

7. Acknowledgments

Thanks to Marc Petit-Huguenin, who suggested using UUIDs in generating RTCP CNAMEs. Also, thanks to David McGrew for providing text for the Security Considerations section.

8. References

8.1. 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.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC4122] Leach, P., Mealling, M., and R. Salz, "A Universally Unique IDentifier (UUID) URN Namespace", RFC 4122, July 2005.
- [RFC4648] Josefsson, S., "The Base16, Base32, and Base64 Data Encodings", RFC 4648, October 2006.
- [RFC5342] Eastlake, D., "IANA Considerations and IETF Protocol Usage for IEEE 802 Parameters", BCP 141, RFC 5342, September 2008.
- [RFC4086] Eastlake, D., Schiller, J., and S. Crocker, "Randomness Requirements for Security", BCP 106, RFC 4086, June 2005.

8.2. Informative References

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

RFC 6222, April 2011.

- [RFC1918] Rekhter, Y., Moskowitz, R.,
 Karrenberg, D., Groot, G., and
 E. Lear, "Address Allocation for
 Private Internets", BCP 5,
 RFC 1918, February 1996.
- [RFC3022] Srisuresh, P. and K. Egevang,
 "Traditional IP Network Address
 Translator (Traditional NAT)",
 RFC 3022, January 2001.
- [RFC3711] Baugher, M., McGrew, D.,
 Naslund, M., Carrara, E., and K.
 Norrman, "The Secure Real-time
 Transport Protocol (SRTP)",
 RFC 3711, March 2004.
- [RFC4941] Narten, T., Draves, R., and S.
 Krishnan, "Privacy Extensions
 for Stateless Address
 Autoconfiguration in IPv6",
 RFC 4941, September 2007.
- [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.
- [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.
- [I-D.ietf-rtcweb-security-arch] Rescorla, E., "RTCWEB Security

Architecture", draft-ietf-rtcweb-security-arch-03 (work in progress), July 2012.

[I-D.rescorla-avtcore-random-cname] Rescorla, E., "Random algorithm for RTP CNAME generation", draft-rescorla-avtcore-random-cname-00 (work in progress), July 2012.

Authors' Addresses

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

Phone: +1 650 678 2350
EMail: ekr@rtfm.com

Ali Begen
Cisco
181 Bay Street
Toronto, ON M5J 2T3
CANADA

EMail: abegen@cisco.com

avtcore
Internet-Draft
Updates: 3551 (if approved)
Intended status: Standards Track
Expires: February 8, 2013

T. Terriberry
Mozilla Corporation
August 7, 2012

Update to Recommended Codecs for the AVP RTP Profile
draft-terriberry-avp-codecs-00

Abstract

[RFC3551] defines the AVP RTP profile, which is the basis for many other profiles, such as SAVP [RFC3711], AVPF [RFC4585], and SAVPF [RFC5124]. This document updates [RFC3551] (and by extension, the profiles that build upon it) to reflect changes in audio codec usage since the document was originally published.

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 February 8, 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	4
3. Updates to RFC 3551	5
3.1. Updates to Section 6	5
4. Security Considerations	6
5. IANA Considerations	7
6. Acknowledgments	8
7. References	9
7.1. Normative References	9
7.2. Informative References	9
Author's Address	10

1. Introduction

[RFC3551] says that audio applications operating under the AVP profile SHOULD be able to send and receive PCMU and DVI4. However, in practice, many RTP deployments do not support DVI4, and its utility is limited in the presence of much more modern codecs. This document updates the recommended audio codec selection for the AVP profile to remove the SHOULD for DVI4.

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].

3. Updates to RFC 3551

The text of [RFC3551] is hereby updated as set forth below.

3.1. Updates to Section 6

In the final paragraph of Section 6, replace, "payload types 0 (PCMU) and 5 (DVI4)," with "payload type 0 (PCMU)." Also, add a final sentence to this paragraph that states, "Some environments may make support for PCMU mandatory."

4. Security Considerations

This document does not introduce any new security considerations for [RFC3551].

5. IANA Considerations

This document has no actions for IANA.

6. Acknowledgments

Thanks to Colin Perkins for suggesting this update.

7. References

7.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC3551] Schulzrinne, H. and S. Casner, "RTP Profile for Audio and Video Conferences with Minimal Control", STD 65, RFC 3551, July 2003.

7.2. Informative References

- [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.

Author's Address

Timothy B. Terriberry
Mozilla Corporation
650 Castro Street
Mountain View, CA 94041
USA

Phone: +1 650 903-0800
Email: tterribe@xiph.org

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: April 25, 2013

M. Westerlund
Ericsson
C. Perkins
University of Glasgow
October 22, 2012

Multiple RTP Sessions on a Single Lower-Layer Transport
draft-westerlund-avtcore-transport-multiplexing-04

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 April 25, 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. ICE and DTLS-SRTP Integration	12
5.3. Signalling Fallback	12
6. Specification	13
6.1. Shim Layer	14
6.2. Signalling	17
6.3. SRTP Key Management	18
6.3.1. Security Description	19
6.3.2. DTLS-SRTP	19
6.3.3. MIKEY	19
6.4. Examples	20
6.4.1. RTP Packet with Transport Header	20
6.4.2. SDP Offer/Answer example	21
7. Open Issues	25
8. IANA Considerations	26
9. Security Considerations	26
10. Acknowledgements	26
11. References	27
11.1. Normative References	27
11.2. Informational References	27
Appendix A. Possible Solutions	29

A.1.	Header Extension	29
A.2.	Multiplexing Shim	30
A.3.	Single Session	31
A.4.	Use the SRTP MKI field	32
A.5.	Use an Octet in the Padding	33
A.6.	Redefine the SSRC field	33
Appendix B.	Comparison	34
B.1.	Support of Multiple RTP Sessions Over Single Transport . .	34
B.2.	Enable Same SSRC Value in Multiple RTP Sessions	34
B.2.1.	Avoid SSRC Translation in Gateways/Translation	34
B.2.2.	Support Existing Extensions	35
B.3.	Ensure SRTP Functions	35
B.4.	Don't Redefine Used Bits	36
B.5.	Firewall Friendly	37
B.6.	Monitoring and Reporting	38
B.7.	Usable over Multicast	39
B.8.	Incremental Deployment	39
B.9.	Summary and Conclusion	40
Authors' Addresses	41

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.ietf-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 had 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 recommended a shim layer variant as single solution, which specified in detail including signalling solution and examples. The other considered 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.ietf-avtcore-multi-media-rtp-session].

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

Using fewer transport 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 receive. 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 presentation 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.ietf-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 the solution to be used on a single leg without affecting any other leg, nor require advanced translation 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 an absolute 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 gateway 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 defined by IETF that is integrated into RTP. This integration has several aspects that need to be considered when designing a solution for multiplexing RTP sessions on the same lower layer transport.

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

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

It is here important to contrast SRTP against a set of other possible protection mechanisms. DTLS, TLS, and IPsec are all protecting and encapsulating the entire RTP and RTCP packets. They don't perform any partial operations on the RTP and RTCP packets. Any change that is considered to be part of the RTP and RTCP packet is transparent to them, but possibly not to SRTP. Thus the impact on SRTP operations 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 a 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 may not work or poorly. Instead only IP/UDP header compression is likely to be applied. Thus a prefix will loose some compression efficiency until compression profiles for IP/UDP/SHIM/RTP has been developed, implemented and deployed. Postfix don't have that issue, but nor can it ever gain anything from header compression which an prefixed solution could once an updated profile is deployed. Postfix also will have reduced efficiency compressing sessions when the same SSRC is used in two different RTP sessions as the RTP header fields like sequence number etc will not behave as expected and need frequent explicit updates.

The question of a prefixed or a 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.

After discussion in the WG it has been determined that prefixed is the preferred solution.

5.2. ICE and DTLS-SRTP Integration

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

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

Note: This limits a single byte SHIM to only allow a maximum of 64 RTP sessions over a single transport flow.

5.3. 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.ietf-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. This is to support decomposed end-points or cases where certain media traffic is required to go to a central processing node while others goes directly to a peer.

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.

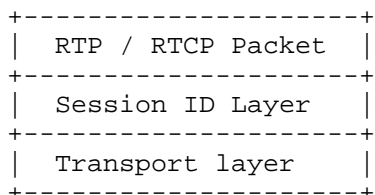
Note: Signalling solution is awaiting resolution of design path for bundle and will then consider that solution and issues raised.

6. Specification

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

6.1. Shim Layer

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



Stack View with Session ID SHIM

The above stack is in fact a layered one as it does allow multiple RTP Sessions to be multiplexed on top of the Session ID shim layer. This enables the example presented in Figure 1 where four sessions, S1-S4 is sent over the same Transport layer and where the Session ID layer will combine and encapsulate them with the session ID on transmission and separate and decapsulate them on reception.

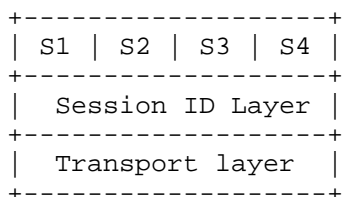


Figure 1: Multiple RTP Session On Top of Session ID Layer

The Session ID layer encapsulates one RTP or RTCP packet from a given RTP session and prefixes the 2-byte Session ID layer to the packet. The Session ID layer is depicted below (Figure 2) and consists of first 2 fixed bit values (10b) followed by a 14 bits unsigned integer field with the Session ID (SID) value.

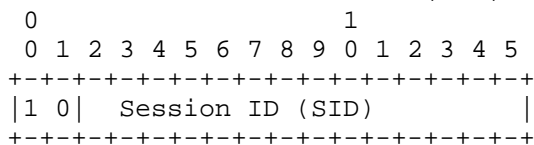


Figure 2: Session ID layer

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 0-16383. 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 [RFC0768], DCCP [RFC4340], TCP [RFC0793], and SCTP [RFC4960] can all be scoped by one or more 5-tuple (Transport protocol, source address and port, destination address and port). The case of multiple 5-tuples occur in the case of multi-unicast topologies, also called meshed multiparty RTP sessions or in case any application would need more than 8192 RTP sessions.

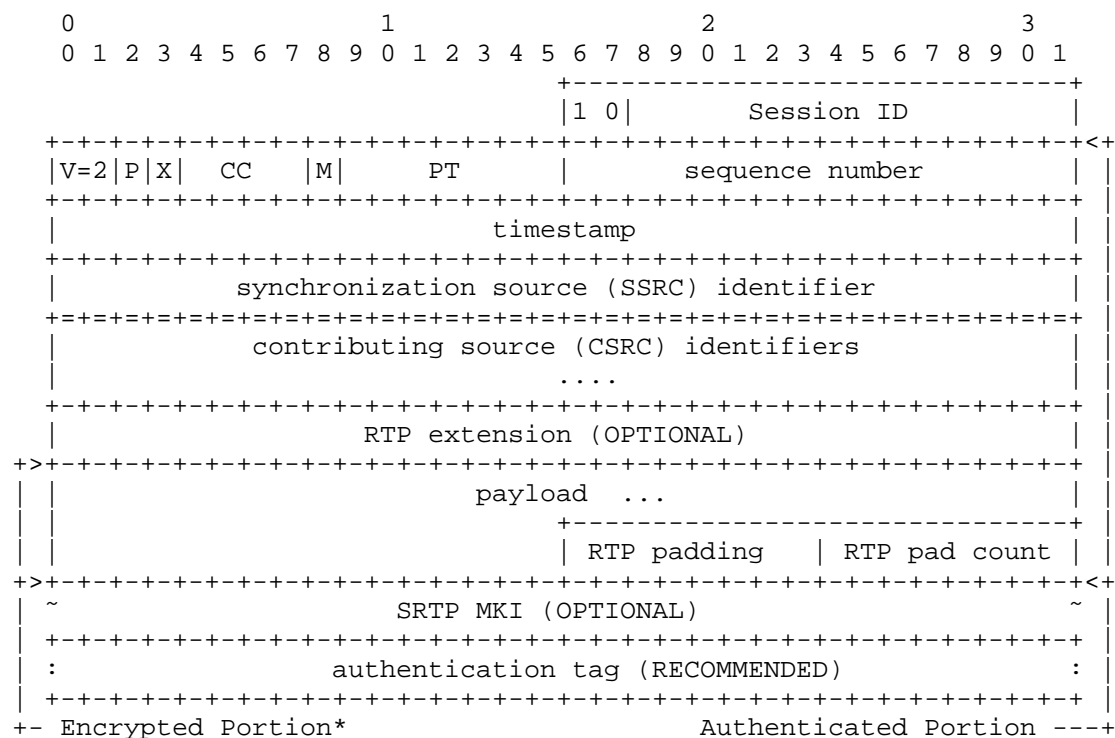


Figure 3: SRTP Packet encapsulated by Session ID Layer

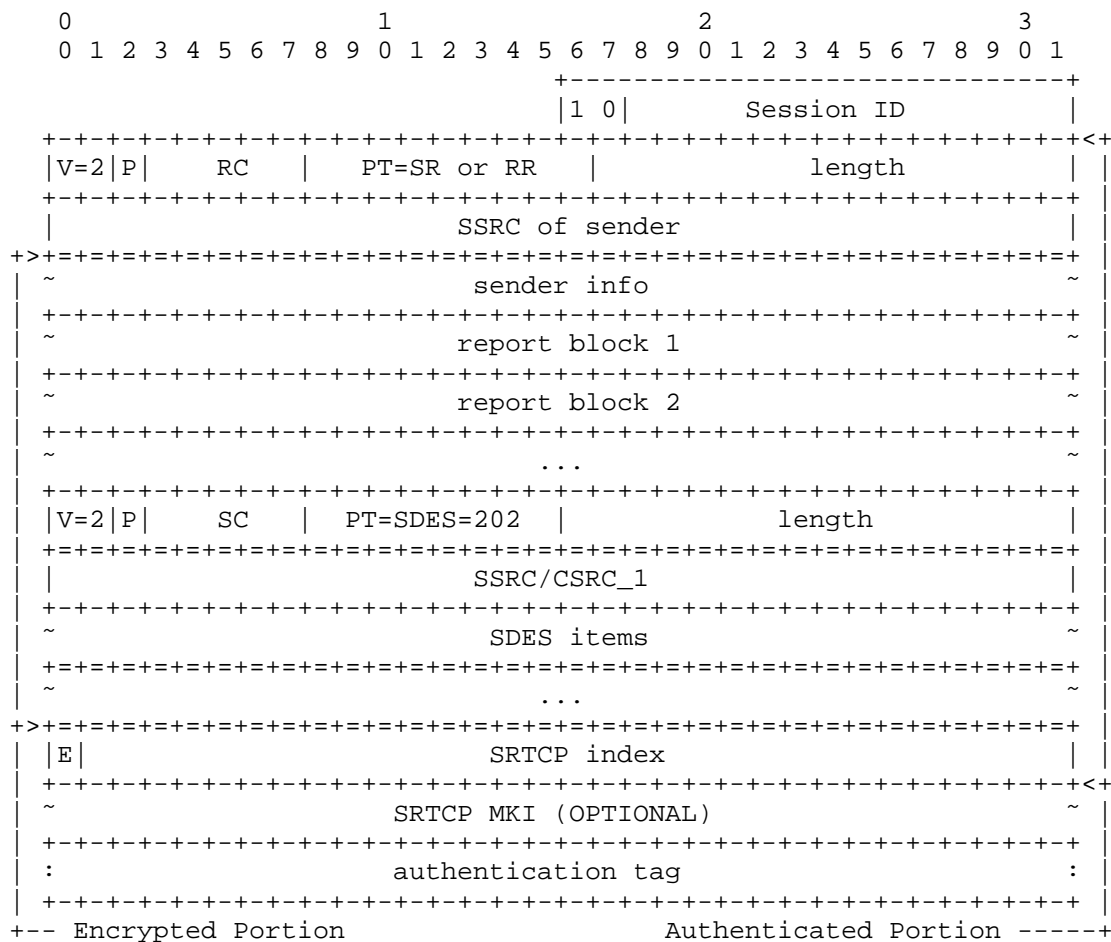


Figure 4: SRTCP packet encapsulated by Session ID layer

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

Note: This section may need updating as the direction of the solution for Bundle has settled and the impact of the raised issues has been analyzed.

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.

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

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

Details to be written up.

6.3.3. MIKEY

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

Lets start with the later, i.e. the SDP transport, which shares the properties with Security Description in that is can be associated

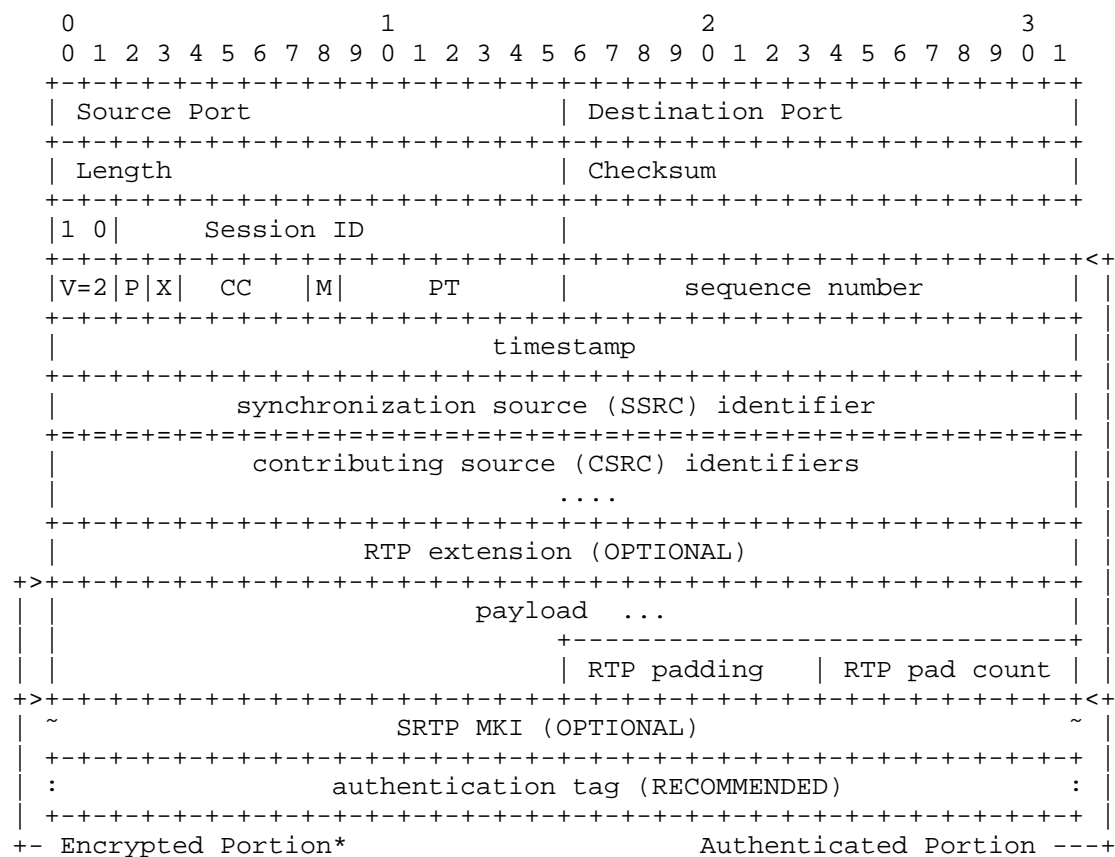
with a particular media description in a SDP. As long as one avoids using the session level attribute one can be certain to correctly associate the key exchange with a given SRTP/SRTCP context.

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

6.4. Examples

6.4.1. 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. The details for how to do key-derivation, preferably in such a way that it can be reused by multiple key-management solutions like MIKEY and DTLS-SRTP
4. The signalling solution will be revisited when the BUNDLE solution discussion has yielded some result.

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]. Eric Rescorla contributed the basic idea of optimizing the DTLS-SRTP key-management by modifying the key derivation process.

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

11. References

11.1. Normative References

- [I-D.ietf-mmusic-sdp-bundle-negotiation]
Holmberg, C. and H. Alvestrand, "Multiplexing Negotiation Using Session Description Protocol (SDP) Port Numbers", draft-ietf-mmusic-sdp-bundle-negotiation-01 (work in progress), August 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.ietf-avtcore-multi-media-rtp-session]
Westerlund, M., Perkins, C., and J. Lennox, "Multiple Media Types in an RTP Session", draft-ietf-avtcore-multi-media-rtp-session-00 (work in progress), October 2012.
- [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-multiplex-architecture]
Westerlund, M., Burman, B., Perkins, C., and H.

Alvestrand, "Guidelines for using the Multiplexing Features of RTP",
draft-westerlund-avtcore-multiplex-architecture-02 (work
in progress), July 2012.

- [RFC0768] Postel, J., "User Datagram Protocol", STD 6, RFC 768, August 1980.
- [RFC0793] Postel, J., "Transmission Control Protocol", STD 7, RFC 793, September 1981.
- [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.
- [RFC4340] Kohler, E., Handley, M., and S. Floyd, "Datagram Congestion Control Protocol (DCCP)", RFC 4340, March 2006.
- [RFC4567] Arkko, J., Lindholm, F., Naslund, M., Norrman, K., and E. Carrara, "Key Management Extensions for Session Description Protocol (SDP) and Real Time Streaming Protocol (RTSP)", RFC 4567, July 2006.
- [RFC4568] Andreasen, F., Baugher, M., and D. Wing, "Session Description Protocol (SDP) Security Descriptions for Media Streams", RFC 4568, July 2006.
- [RFC4960] Stewart, R., "Stream Control Transmission Protocol", RFC 4960, September 2007.
- [RFC5109] Li, A., "RTP Payload Format for Generic Forward Error Correction", RFC 5109, December 2007.
- [RFC5245] Rosenberg, J., "Interactive Connectivity Establishment (ICE): A Protocol for Network Address Translator (NAT) Traversal for Offer/Answer Protocols", RFC 5245, April 2010.
- [RFC5285] Singer, D. and H. Desineni, "A General Mechanism for RTP Header Extensions", RFC 5285, July 2008.
- [RFC5389] Rosenberg, J., Mahy, R., Matthews, P., and D. Wing, "Session Traversal Utilities for NAT (STUN)", RFC 5389, October 2008.

- [RFC5506] Johansson, I. and M. Westerlund, "Support for Reduced-Size Real-Time Transport Control Protocol (RTCP): Opportunities and Consequences", RFC 5506, April 2009.
- [RFC5761] Perkins, C. and M. Westerlund, "Multiplexing RTP Data and Control Packets on a Single Port", RFC 5761, April 2010.
- [RFC5764] McGrew, D. and E. Rescorla, "Datagram Transport Layer Security (DTLS) Extension to Establish Keys for the Secure Real-time Transport Protocol (SRTP)", RFC 5764, May 2010.

Appendix A. Possible Solutions

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

A.1. Header Extension

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

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

1. to be part of the security context to verify the field
2. to be part of the security context to verify the field and remove it before forwarding the packet
3. to be outside of the security context and leave the header extension in the packet. However, that requires successful negotiation of the header extension, but not of the functionality, with the receiving end-points.

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

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

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

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

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 SRTCP/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 SRTCP/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 demultiplex several sessions on a single transport, but at the cost of losing the RTP session as a separator for different type of streams. Lacking different RTP sessions to demultiplex incoming packets, a receiver will have to dig deeper into the packet before determining what to do with it. Care must be taken in that inspection. For example, you must be careful to ensure that each real media source uses its own SSRC in the session and that this SSRC doesn't change media type.

The loss of the RTP session as a separator for different usages or purpose would be a 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 placed on this important aspect. In multi-party situations using central nodes there exist some difficulties in having a legacy implementation using multiple RTP

sessions interworking with an end-point having only a single RTP session across the central node. The main reason is the fact that the one using single session with multiple media types has only one SSRC space, while the other end-points have multiple spaces. Thus translation 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 5) 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 5: Summary Table of Evaluation (Successfully (S), Partially (P) or Fails (F) to meet requirement)

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

Authors' Addresses

Magnus Westerlund
Ericsson
Farogatan 6
SE-164 80 Kista
Sweden

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

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

Email: csp@csperkins.org

Audio/Video Transport Working Group
Internet-Draft
Intended status: Standards Track
Expires: April 25, 2013

Q. Wu
R. Even
Huawei
October 22, 2012

Advertisement for multi-source endpoint multiplexing multiple media type
in the same RTP session
draft-wu-avtcore-multisrc-endpoint-adver-02.txt

Abstract

When two endpoints with multiple media sources are in communication, each media source or each receiver within either endpoint may send or receive reception report independently. This may incur a lot of duplicated reception report to and from the endpoint with multiple sources. This document discusses how to tackle these problems and propose three phases for suppressing reception reports.

Status of this Memo

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

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

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

This Internet-Draft will expire on April 25, 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	4
2.1. Standards Language	4
3. Protocol Overview	5
3.1. Report Source Grouping	5
3.2. Report Source Election	5
3.3. Report Source Advertisement	5
4. Protocol formats	7
4.1. SDES item for Reception Report Sending	7
4.1.1. RRS: Reception Report Sending SDES Item	7
4.2. SDES item for Reception Report Monitoring	7
4.2.1. RRM: Reception Report Monitoring SDES Item	8
5. Security Considerations	9
6. IANA Considerations	10
6.1. New RTCP SDES Type values	10
7. References	11
7.1. Normative References	11
7.2. Informative References	11
Authors' Addresses	12

1. Introduction

For some applications that use unicast transport, e.g., in RTCWeb application, an endpoint with multiple media sources (i.e., multiple-source hosts, e.g., a client with several cameras) may use a different SSRC for each medium but sending them in the same RTP session, which reduces communication failure due to NAT and firewall when using multiple RTP sessions or transport flows.

However when two endpoints with multiple media sources are in communication, each media source within either endpoint may send reception report independently and the receiving endpoint may not know the reception reports received from different media source are from the same sending endpoint. This creates the following three problems:

- o An endpoint with multiple media sources involved in one RTP session may send the reception report with duplicated information about the same remote media source from each of local media sources.
- o An endpoint with multiple media sources involved in one RTP session may receive the reception report with duplicated information about the same local media source from each of remote media sources.
- o An endpoint with multiple media sources involved in one RTP session also may send reception reports about one of its own media sources from another of its own (This is also referred to as RTCP self-reporting). Such reception reports cause additional redundant traffic travelling over the link between sending endpoints and receiving endpoint, which changes the report interval and may affect the media qualities.

This document discusses how to tackle these problems. Three phases for suppressing reception reports are proposed.

- o Grouping of media sources originate from a single endpoint and classifying these media sources into multiple groups.
- o Electing one single SSRC for reception report sending and one single SSRC for reception report monitoring based on local policy.
- o Advertising the SSRC that is used either for reception report sending or reception report monitoring.

2. Terminology

2.1. Standards 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. Protocol Overview

In order to suppress unnecessary reception reports to and from multi-source endpoint, multi-source endpoint should group media sources originating from a single endpoint and elect one or a set of specified SSRCs for reception report processing (e.g., reception report sending, reception report receiving and reception report monitoring).

3.1. Report Source Grouping

When an endpoint with multiple sources multiplexes multiple media types in the same RTP session, the media source originating from the same endpoint should be grouped together. Similarly the endpoint with multiple sources may have multiple one or more than one report source for monitoring. These report sources for monitoring should also be grouped together. Therefore an endpoint with multiple sources should at least split all its own media sources or receivers into two groups(i.e., split SSRCs into two groups): One is sending group, the other is monitoring group. Each group may have one or several group members. Each group member is identified by a different SSRC.

3.2. Report Source Election

When grouping for each endpoint with multiple sources is available, in order to prevent group members receiving duplicated data, one or more than one group member MUST be elected from sending group as report source for reception report sending. When one report source leaves the session or is down, another candidate report source can replace instead. If the monitoring is used, each endpoint with multiple sources MUST have at least one report source for monitoring purpose. One or more than one reporting sources MUST be selected from monitoring group for monitoring use.

Each endpoint with multiple sources at least has one SSRC for reception report sending. If the monitoring is used, the endpoint with multiple sources should choose another SSRC from monitoring group as monitoring report source.

3.3. Report Source Advertisement

When report sources are elected for reception report sending, and monitoring purpose respectively, it is necessary to signal which report source is used for which purpose.

In this document we define two new SDES items to advertise the reporting sources from endpoint with multiple sources that are used

for reception report sending and monitoring respectively (See section 5.1 and section 5.2 for protocol format details).

When those report sources are advertised, the selected report source can send reception report about the same remote media source on behalf of the media sources of its own, which prevent duplicated reception report sent from all the local media sources of its own for the same event.

If the other media source which is not selected as report source knows or understands the advertised report source, it should suppress the reception report for the same event.

If the other media source which is not selected as report source does not know or understand the advertised report source and detects the need to send reception report, this media source should wait for a (short) random dithering interval to check whether it sees a corresponding reception report message from any other receiver or other media sources of its own reporting the same event. If a corresponding reception report for the same event is received from other media sources or any other receivers, it should refrain from sending the reception report message.

In order to prevent duplicated reception report sent from one of its own media sources to another of its own within the same endpoint with multiple sources(i.e., self-reporting), the report source should know the other media sources of its own and suppress sending the reception report on the remote source to its own media sources.

Editor's note: Current draft does not discuss topologies like source projection mixer where the mixer keeps the original ssrc so even though they arrive from the mixer they have different CNAMEs. It will be useful to discuss those topologies in the future version.

4. Protocol formats

Editor's note: It is not clear if you need to signal which one is used or what is the group but if needed it may be better to allow the sender to indicate which SSRCs are part of a group. It will also require an negotiation in the offer/answer to verify that this mode is supported by the stream senders.

4.1. SDES item for Reception Report Sending

This sub-section defines the format of the Reception Report Sending SDES item. The SDES item is carried in the RTCP SDES packet. The packet format for the RTCP SDES is defined in Section 6.5 of [RFC3550]. Each SDES packet is composed of a header with fixed-length fields for version, source count, packet type (PT), and length, followed by zero or more chunks. Each chunk consists of an SSRC/CSRC identifier followed by zero or more SDES items. If this SDES item is carried, the CNAME SDES item should also be carried together with this SDES item in the same chunk. In the SDES packet, the PT field is set to SDES(202).

4.1.1. RRS: Reception Report Sending SDES Item

```

      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
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|   RRS=TBD   |   length   | Candidate Report Source SSRC |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|   ....   |
+---+---+---+---+---+

```

The Reception Report Sending Item is not mandatory item and intended for indicating the elected report source for an endpoint with multiple sources, which is responsible for reception report sending. The SSRC/CSRC identifier included in the same chunk (at the beginning of this chunk) as the item is the SSRC of elected sending report source. Additionally, this item may carry one or more than one Candidate Report Source SSRCs. The candidate Report Source SSRC follows the same format as SSRC/CSRC identifier defined in RFC3550. Its length is described by the length field. The value of the length field does not include the two octet SDES item header. This item MUST be ignored by applications that are not configured to make use of it.

4.2. SDES item for Reception Report Monitoring

This sub-section defines the format of the Reception Report Monitoring SDES item. The SDES item is carried in the RTCP SDES

packet. The packet format for the RTCP SDES is defined in Section 6.5 of [RFC3550]. Each SDES packet is composed of a header with fixed-length fields for version, source count, packet type (PT), and length, followed by zero or more chunks. Each chunk consists of an SSRC/CSRC identifier followed by zero or more SDES items. If this SDES item is carried, the CNAME SDES item should also be carried together with this SDES item in the same chunk. In the SDES packet, the PT field is set to SDES(202).

4.2.1. RRM: Reception Report Monitoring SDES Item

```

      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
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|   RRM=TBD   |   length   | Candidate Report Source SSRC
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|   ....
+---+---+---+---+---+

```

The Reception Report Sending Item is not mandatory item and intended for indicating the report source for an endpoint, which is responsible for reception report monitoring. The SSRC/CSRC identifier included in the same chunk as the item (at the beginning of this chunk) is the SSRC of elected sending report source. Additionally, this item may carry one or more than one Candidate Report Source SSRCs. The candidate Report Source SSRC follows the same format as SSRC/CSRC identifier defined in RFC3550. Its length is described by the length field. The value of the length field does not include the two octet SDES item header. This item MUST be ignored by applications that are not configured to make use of it.

5. Security Considerations

RTCP reports can contain sensitive information, including information about report source grouping for endpoint with multiple source and member of a session established between two or more endpoints. Therefore, the use of security mechanisms with RTP, as documented in Section 9 of [RFC3550] applies.

6. IANA Considerations

New SDES types for RTCP SDES are subject to IANA registration. For general guidelines on IANA considerations for RTCP SDES, refer to[RFC3550].

6.1. New RTCP SDES Type values

This document assigns two additional SDES type in the IANA "RTCP SDES Item Types Registry" to the new SDES items as follow:

abbrev.	name	value
RRSS:	Reception Report Sending Source	TBD
RRRS:	Reception Report Monitoring Source	TBD

[Note to RFC Editor: please replace RRSS and RRMS with the IANA provided RTCP SDES Item Types.]

7. References

7.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", March 1997.
- [RFC3550] Schulzrinne, H., "RTP: A Transport Protocol for Real-Time Applications", RFC 3550, July 2003.
- [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 4595, July 2006.
- [RFC5285] Singer, D. and H. Desineni, "A General Mechanism for RTP Header Extensions", RFC 5285, July 2008.

7.2. Informative References

- [I-D.ietf-avtcore-multi-media-rtp-session-00]
Westerlund, M., "Multiple Media Types in an RTP Session",
ID draft-ietf-avtcore-multi-media-rtp-session-00,
October 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",
ID draft-lennox-avtcore-rtp-multi-stream-00, July 2012.
- [I-D.wu-avtcore-multiplex-multisource-endpoint]
Wu, Q., "Bandwidth and RTCP timing issues for multi-source
endpoint",
ID draft-wu-avtcore-multiplex-multisource-endpoint-01,
October 2012.

Authors' Addresses

Qin Wu
Huawei
101 Software Avenue, Yuhua District
Nanjing, Jiangsu 210012
China

Email: sunseawq@huawei.com

Roni Even
Huawei
14 David Hamelech
Tel, Aviv 64953
Israel

Email: ron.even.tlv@gmail.com

