AVT
Internet-Draft
Intended status: Standards Track
Expires: January 6, 2012

J. Lennox, Ed.
Vidyo
E. Ivov
Jitsi
E. Marocco
Telecom Italia

July 5, 2011

A Real-Time Transport Protocol (RTP) Header Extension for Client-to-Mixer Audio Level Indication
draft-ietf-avtext-client-to-mixer-audio-level-03

Abstract

This document defines a mechanism by which packets of Real-Time Transport Protocol (RTP) audio streams can indicate, in an RTP header extension, the audio level of the audio sample carried in the RTP packet. In large conferences, this can reduce the load on an audio mixer or other middlebox which wants to forward only a few of the loudest audio streams, without requiring it to decode and measure every stream that is received.

Status of this Memo

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

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

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

This Internet-Draft will expire on January 6, 2012.

Table of Contents

1. Introduction

In a centralized Real-Time Transport Protocol (RTP) [RFC3550] audio conference, an audio mixer or forwarder receives audio streams from many or all of the conference participants. It then selectively forwards some of them to other participants in the conference. In large conferences, it is possible that such a server might be receiving a large number of streams, of which only a few should be forwarded to the other conference participants.

In such a scenario, in order to pick the audio streams to forward, a centralized server needs to decode, measure audio levels, and possibly perform voice activity detection on audio data from a large number of streams. The need for such processing limits the size or number of conferences such a server can support.

As an alternative, this document defines an RTP header extension [RFC5285] through which senders of audio packets can indicate the audio level of the packets' payload, reducing the processing load for a server.

The header extension in this draft is different than, but complementary with, the one defined in [I-D.ietf-avtext-mixer-to-client-audio-level], which defines a mechanism by which audio mixers can indicate to clients the levels of the contributing sources that made up the mixed audio.

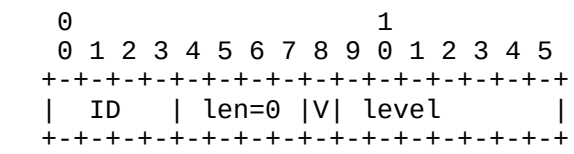
2. Terminology

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] and indicate requirement levels for compliant implementations.

3. Audio Levels

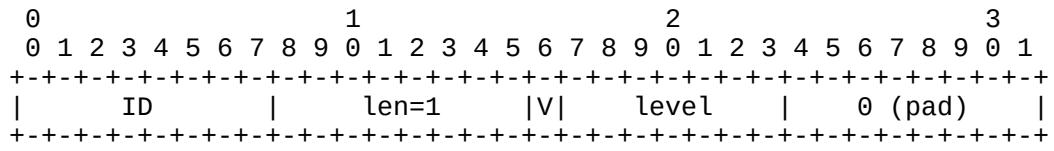
The audio level header extension carries the level of the audio in the RTP payload of the packet it is associated with. This information is carried in an RTP header extension element as defined by the "General Mechanism for RTP Header Extensions" [RFC5285].

The payload of the audio level header extension element can be encoded using the one or the two-byte header defined in [RFC5285]. Figure 1 and Figure 2 show sample audio level encodings with each of them.



Sample audio level encoding using the one-byte header format

Figure 1



Sample audio level encoding using the two-byte header format

Figure 2

Note that, as indicated in [RFC5285] length field in the one-byte header format takes the value 0 to indicate that 1 byte follows. In the two-byte header format on the other hand it takes the value of 1.

The magnitude of the audio level itself is packed into the seven least significant bits of the single byte of the header extension, shown in Figure 1 and Figure 2. The least significant bit of the audio level magnitude is packed into the least significant bit of the byte. The most significant bit of the byte is used as a separate flag bit "V", defined below.

The audio level is expressed in -dBov, with values from 0 to 127 representing 0 to -127 dBov. dBov is the level, in decibels, relative to the overload point of the system, i.e. the maximum-amplitude signal that can be handled by the system without clipping. (Note: Representation relative to the overload point of a system is particularly useful for digital implementations, since one does not need to know the relative calibration of the analog circuitry.) For example, in the case of u-law (audio/pcmu) audio [ITU.G711.1988], the 0 dBov reference would be a square wave with values +/- 8031. (This translates to 6.18 dBm0, relative to u-law’s dBm0 definition in Table 6 of G.711.)

The audio level for digital silence, for example for a muted audio

source, MUST be represented as 127 (-127 dBov), regardless of the dynamic range of the encoded audio format.

The audio level header extension only carries the level of the audio in the RTP payload of the packet it is associated with, with no long-term averaging or smoothing applied. That level is measured as a root mean square of all the samples in the measured range.

To simplify implementation of the encoding procedures described here, the reference implementation section in [I-D.ietf-avtext-mixer-to-client-audio-level] provides a sample Java implementation of an audio level calculator that helps obtain such values from raw linear PCM audio samples.

In addition, a flag bit (labeled V) indicates whether the encoder believes the audio packet contains voice activity (1) or does not (0). The voice activity detection algorithm is unspecified and left implementation-specific.

When this header extension is used with RTP data sent using the RTP Payload for Redundant Audio Data [RFC2198], the header's data describes the contents of the primary encoding.

Note: This audio level is defined in the same manner as is audio noise level in the RTP Payload Comfort Noise specification [RFC3389]. In the comfort noice specification, the overall magnitude of the noise level in comfort noise is encoded into the first byte of the payload, with spectral information about the noise in subsequent bytes. This specification's audio level parameter is defined so as to be identical to the comfort noise payload's noise-level byte.

4. Signaling (Setup) Information

The URI for declaring this header extension in an extmap attribute is "urn:ietf:params:rtp-hdrext:ssrc-audio-level". There is no additional setup information needed for this extension (i.e. no extensionattributes).

5. Considerations on Use

Mixers and forwarders generally should not base audio forwarding decisions directly on packet-by-packet audio level information, but rather should apply some analysis of the audio levels and trends. This general rule applies whether audio levels are provided by endpoints (as defined in this document), or are calculated at a server, as would be done in the absence of this information. This

section discusses several issues that mixers and forwarders may wish to take into account. (Note that this section provides design guidance only, and is not normative.)

First of all, audio levels should generally be measured over longer intervals than that of a single audio packet. In order to avoid false-positives for short bursts of sound (such as a cough or a dropped microphone), it is often useful to require that a participant's audio level be maintained for some period of time before considering it to be "real", i.e. some type of low-pass filter should be applied to the audio levels. Note, though, that such filtering must be balanced with the need to avoid clipping of the beginning of a speaker's speech.

Additionally, different participants may have their audio input set differently. It may be useful to apply some sort of automatic gain control to the audio levels. There are a number of possible approaches to acheiving this, e.g. by measuring peak audio levels, by average audio levels during speech, or by measuring background audio levels (average audio level levels during non-speech).

6. Security Considerations

A malicious endpoint could choose to set the values in this header extension falsely, so as to falsely claim that audio or voice is or is not present. It is not clear what could be gained by falsely claiming that audio is not present, but an endpoint falsely claiming that audio is present could perform a denial-of-service attack on an audio conference, so as to send silence to suppress other conference members' audio. Thus, a device relying on audio level data from untrusted endpoints SHOULD periodically audit the level information transmitted, taking appropriate corrective action if endpoints appear to be sending incorrect data. (Note that as it is valid for an endpoint to choose to measure audio levels prior to encoding, some degree of discrepancy SHOULD be tolerated.)

In the Secure Real-Time Transport Protocol (SRTP) [RFC3711], RTP header extensions are authenticated but not encrypted. When this header extension is used, audio levels are therefore visible on a packet-by-packet basis to an attacker passively observing the audio stream. As discussed in [I-D.perkins-avt-srtp-vbr-audio], such an attacker might be able to infer information about the conversation, possibly with phoneme-level resolution. In scenarios where this is a concern, additional mechanisms SHOULD be used to protect the confidentiality of the header extension. One solution is header extension encryption [I-D.lennox-avtcore-srtp-encrypted-header-ext].

7. IANA Considerations

This document defines a new extension URI to the RTP Compact Header Extensions subregistry of the Real-Time Transport Protocol (RTP) Parameters registry, according to the following data:

Extension URI: urn:ietf:params:rtp-hdrext:ssrc-audio-level
Description: Audio Level
Contact: jonathan@vidyo.com
Reference: RFC XXXX

Note to RFC Editor: please replace "RFC XXXX" with the number of this RFC.

8. References

8.1. Normative References

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

[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.

[RFC3550] Schulzrinne, H., Casner, S., Frederick, R., and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications", STD 64, RFC 3550, July 2003.

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

8.2. Informative References

[I-D.ietf-avtext-mixer-to-client-audio-level] Ivov, E., Marocco, E., and J. Lennox, "A Real-Time Transport Protocol (RTP) Header Extension for Mixer-to-Client Audio Level Indication", draft-ietf-avtext-mixer-to-client-audio-level-02 (work in progress), May 2011.

[I-D.lennox-avtcore-srtp-encrypted-header-ext] Lennox, J., "Encryption of Header Extensions in the Secure Real-Time Transport Protocol (SRTP)", draft-lennox-avtcore-srtp-encrypted-header-ext-00 (work in progress), March 2011.

[I-D.perkins-avt-srtp-vbr-audio]
Perkins, C. and J. Valin, "Guidelines for the use of Variable Bit Rate Audio with Secure RTP", draft-perkins-avt-srtp-vbr-audio-05 (work in progress), December 2010.

[ITU.G711.1988]
International Telecommunications Union, "Pulse Code Modulation (PCM) of Voice Frequencies", ITU-T Recommendation G.711, November 1988.

[RFC3389] Zopf, R., "Real-time Transport Protocol (RTP) Payload for Comfort Noise (CN)", RFC 3389, September 2002.

[RFC3711] Baugher, M., McGrew, D., Naslund, M., Carrara, E., and K. Norrman, "The Secure Real-time Transport Protocol (SRTP)", RFC 3711, March 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 -02

- Changed encoding related text so that it would cover both the one-byte and the two-byte header formats.
- Clarified use of root mean square for dBov calculation
- Other minor editorial changes.

A.2. Changes From Draft -01

- Changed the URI for declaring this header extension from "urn:ietf:params:rtp-hdrext:audio-level" to "urn:ietf:params:rtp-hdrext:ssrc-audio-level" for consistency with [I-D.ietf-avtext-mixer-to-client-audio-level].
- Removed the "Limitations" section; it was discussing a potential extension that consensus indicated was out of scope of this document.
- Closed the P.56 open issue. It was agreed on IETF 80 that P.56 is mostly about speech levels and the levels transported by the extension defined here should also be able to serve as an indication for noise.
- Closed the open issue about transmitting noise floor information. Noise floor is (loosely) inferrable by observing the per-packet level information over a period of time, so the additional complexity seemed unnecessary.

- Editorial changes for consistency with [I-D.ietf-avtext-mixer-to-client-audio-level].
- Moved several descriptions of normative items that previously had only been described in informative sections of the text.
- Other editorial clarifications.

A.3. Changes From Draft -00

- Added references to the sample level calculator in [I-D.ietf-avtext-mixer-to-client-audio-level].
- Changed affiliation for Emil Ivov.

A.4. Changes From Individual Submission Draft -01

- This version is primarily a document refresh.
- Emil Ivov and Enrico Marocco have been added as co-authors.
- Additional open issues listed.

A.5. Changes From Individual Submission Draft -00

- The draft name has been changed to clarify that this document defines Client-To-Mixer Audio Levels, to more clearly distinguish it from [I-D.ietf-avtext-mixer-to-client-audio-level].
- The header extension format has been changed from a two-byte to a one-byte payload, eliminating the 7 reserved bits and the one must-be-zero bit.
- The sections Considerations on Use (Section 5) and Limitations have been added.
- It has been noted that senders MAY indicate -127 dBov for digital silence, and that level measurement MAY be done prior to encoding audio.
- A reference to [I-D.lennox-avtcore-srtp-encrypted-header-ext] has been added to the security considerations.
- The term "header extension" is now used consistentenly throughout the document (as opposed to "extension header").

Authors' Addresses

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

Email: jonathan@vidyo.com

Emil Ivov
Jitsi
Strasbourg 67000
France

Email: emcho@jitsi.org

Enrico Marocco
Telecom Itialia
Via G. Reiss Romoli, 274
Turin 10148
Italy

Email: enrico.marocco@telecomitalia.it

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: January 6, 2012

E. Ivov, Ed.
Jitsi
E. Marocco, Ed.
Telecom Italia
J. Lennox
Vidyo, Inc.

July 5, 2011

A Real-Time Transport Protocol (RTP) Header Extension for Mixer-to-Client Audio Level Indication
draft-ietf-avtext-mixer-to-client-audio-level-03

Abstract

This document describes a mechanism for RTP-level mixers in audio conferences to deliver information about the audio level of individual participants. Such audio level indicators are transported in the same RTP packets as the audio data they pertain to.

Status of this Memo

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

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

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

This Internet-Draft will expire on January 6, 2012.

Copyright Notice

Table of Contents

1. Introduction

The Framework for Conferencing with the Session Initiation Protocol (SIP) defined in RFC 4353 [RFC4353] presents an overall architecture for multi-party conferencing. Among others, the framework borrows from RTP [RFC3550] and extends the concept of a mixer entity "responsible for combining the media streams that make up a conference, and generating one or more output streams that are delivered to recipients". Every participant would hence receive, in a flat single stream, media originating from all the others.

Using such centralized mixer-based architectures simplifies support for conference calls on the client side since they would hardly differ from one-to-one conversations. However, the method also introduces a few limitations. The flat nature of the streams that a mixer would output and send to participants makes it difficult for users to identify the original source of what they are hearing.

Mechanisms that allow the mixer to send to participants cues on current speakers (e.g. the CSRC fields in RTP [RFC3550]) only work for speaking/silent binary indications. There are, however, a number of use cases where one would require more detailed information. Possible examples include the presence of background chat/noise/ music/typing, someone breathing noisily in their microphone, or other cases where identifying the source of the disturbance would make it easy to remove it (e.g. by sending a private IM to the concerned party asking them to mute their microphone). A more advanced scenario could involve an intense discussion between multiple participants that the user does not personally know. Audio level information would help better recognize the speakers by associating with them complex (but still human readable) characteristics like loudness and speed for example.

One way of presenting such information in a user friendly manner would be for a conferencing client to attach audio level indicators to the corresponding participant related components in the user interface as displayed in Figure 1.

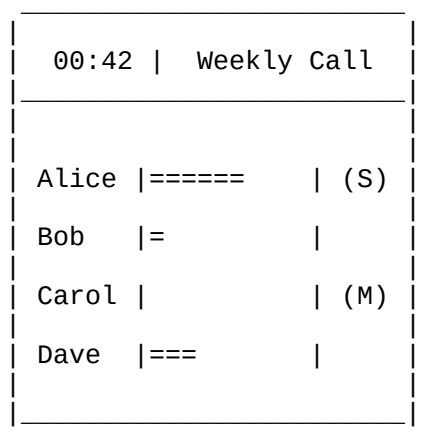


Figure 1: Displaying detailed speaker information to the user by including audio level for every participant.

Implementing a user interface like the above requires analysis of the media sent from other participants. In a conventional audio conference this is only possible for the mixer since all other conference participants are generally receiving a single, flat audio stream and have therefore no immediate way of determining individual audio levels.

This document specifies an RTP extension header that allows such mixers to deliver audio level information to conference participants by including it directly in the RTP packets transporting the corresponding audio data.

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].

3. Protocol Operation

According to RFC 3550 [RFC3550] a mixer is expected to include in outgoing RTP packets a list of identifiers (CSRC IDs) indicating the sources that contributed to the resulting stream. The presence of such CSRC IDs allows RTP clients to determine, in a binary way, the active speaker(s) in any given moment. RTCP also provides a basic mechanism to map the CSRC IDs to user identities through the CNAME

field. More advanced mechanisms, may exist depending on the signaling protocol used to establish and control a conference. In the case of the Session Initiation Protocol [RFC3261] for example, the Event Package for Conference State [RFC4575] defines a <src-id> tag which binds CSRC IDs to media streams and SIP URIs.

This document describes an RTP header extension that allows mixers to indicate the audio-level of every conference participant (CSRC) in addition to simply indicating their on/off status. This new header extension uses "General Mechanism for RTP Header Extensions" described in [RFC5285].

Each instance of this header contains a list of one-octet audio levels expressed in -dBov, with values from 0 to 127 representing 0 to -127 dBov(see Figure 2 and Figure 3). Appendix A provides a reference implementation indicating one way of obtaining such values from raw audio samples.

Every audio level value pertains to the CSRC identifier located at the corresponding position in the CSRC list. In other words, the first value would indicate the audio level of the conference participant represented by the first CSRC identifier in that packet and so forth. The number and order of these values MUST therefore match the number and order of the CSRC IDs present in the same packet.

When encoding audio level information, a mixer SHOULD include in a packet information that corresponds to the audio data being transported in that same packet. It is important that these values follow the actual stream as closely as possible. Therefore a mixer SHOULD also calculate the values after the original contributing stream has undergone possible processing such as level normalization, and noise reduction for example.

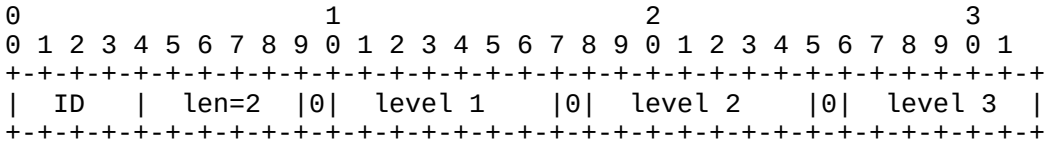
It may sometimes happen that a conference involves more than a single mixer. In such cases each of the mixers MAY choose to relay the CSRC list and audio-level information they receive from peer mixers (as long as the total CSRC count remains below 16). Given that the maximum audio level is not precisely defined by this specification, it is likely that in such situations average audio levels would be perceptibly different for the participants located behind the different mixers.

4. Audio Levels

The audio level header extension carries the level of the audio in the RTP payload of the packet it is associated with. This

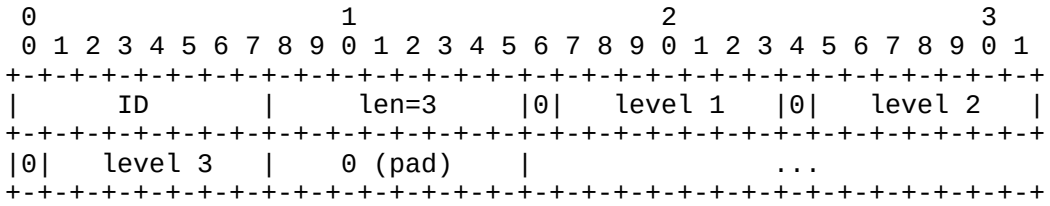
information is carried in an RTP header extension element as defined by the "General Mechanism for RTP Header Extensions" [RFC5285].

The payload of the audio level header extension element can be encoded using the one or the two-byte header defined in [RFC5285]. Figure 2 and Figure 3 show sample audio level encodings with each of them.



Sample audio level encoding using the one-byte header format

Figure 2



Sample audio level encoding using the two-byte header format

Figure 3

In the case of the one-byte header format, the 4-bit len field is the number minus one of data bytes (i.e. audio level values) transported in this header extension element following the one-byte header. Therefore, the value zero in this field indicates that one byte of data follows. In the case of the two-byte header format the 8-bit len field contains the exact number of audio levels carried in the extension. RFC 3550 [RFC3550] only allows RTP packets to carry a maximum of 15 CSRC IDs. Given that audio levels directly refer to CSRC IDs, implementations MUST NOT include more than 15 audio level values. The maximum value allowed in the len field is therefore 14 for one-byte header format adn 15 for two-byte header format.

Audio levels in this document are defined in the same manner as is

audio noise level in the RTP Payload Comfort Noise specification [RFC3389]. In the comfort noice specification, the overall magnitude of the noise level in comfort noise is encoded into the first byte of the payload, with spectral information about the noise in subsequent bytes. This specification's audio level parameter is defined so as to be identical to the comfort noise payload's noise-level byte.

The magnitude of the audio level itself is packed into the seven least significant bits of the single byte of the header extension, shown in Figure 2 and Figure 3. The least significant bit of the audio level magnitude is packed into the least significant bit of the byte. The most significant bit of the byte is unused and always set to 0.

The audio level is expressed in -dBov, with values from 0 to 127 representing 0 to -127 dBov. dBov is the level, in decibels, relative to the overload point of the system, i.e. the maximum-amplitude signal that can be handled by the system without clipping. (Note: Representation relative to the overload point of a system is particularly useful for digital implementations, since one does not need to know the relative calibration of the analog circuitry.) For example, in the case of u-law (audio/pcmu) audio [ITU.G.711], the 0 dBov reference would be a square wave with values +/- 8031. (This translates to 6.18 dBm0, relative to u-law's dBm0 definition in Table 6 of G.711.)

The audio level for digital silence, for example for a muted audio source, MUST be represented as 127 (-127 dBov), regardless of the dynamic range of the encoded audio format.

The audio level header extension only carries the level of the audio in the RTP payload of the packet it is associated with, with no long-term averaging or smoothing applied. That level is measured as a root mean square of all the samples in the measured range.

To simplify implementation of the encoding procedures described here, this specification provides a sample Java implementation (Appendix A) of an audio level calculator that helps obtain such values from raw linear PCM audio samples.

5. Signaling Information

The URI for declaring the audio level header extension in an SDP extmap attribute and mapping it to a local extension header identifier is "urn:ietf:params:rtp-hdrext:csrc-audio-level". There is no additional setup information needed for this extension (i.e. no extensionattributes).

An example attribute line in the SDP, for a conference might be:

```
a=extmap:7 urn:ietf:params:rtp-hdrext:csrc-audio-level
```

The above mapping will most often be provided per media stream (in the media-level section(s) of SDP, i.e., after an "m=" line) or globally if there is more than one stream containing audio level indicators in a session.

Presence of the above attribute in the SDP description of a media stream indicates that RTP packets in that stream, which contain the level extension defined in this document, will be carrying them with an ID of 7.

Conferencing clients that support audio level indicators and have no mixing capabilities would not be able to content for this audio level extension and would hence have to always include the direction parameter in the "extmap" attribute with a value of "recvonly". Conference focus entities with mixing capabilities can omit the direction or set it to "sendrecv" in SDP offers. Such entities would need to set it to "sendonly" in SDP answers to offers with a "recvonly" parameter and to "sendrecv" when answering other "sendrecv" offers.

This specification only defines use of the audio level extensions in audio streams. They MUST NOT be advertised with other media types such as video or text for example.

The following Figure 4 and Figure 5 show two example offer/answer exchanges between a conferencing client and a focus, and between two conference focus entities.

```
v=0
o=alice 2890844526 2890844526 IN IP6 host.example.com
c=IN IP6 host.example.com
t=0 0
m=audio 49170 RTP/AVP 0 4
a=rtpmap:0 PCMU/8000
a=rtpmap:4 G723/8000
a=extmap:1/recvonly urn:ietf:params:rtp-hdrext:csrc-audio-level

v=0
i=A Seminar on the session description protocol
o=conf-focus 2890844730 2890844730 IN IP6 focus.example.net
c=IN IP6 focus.example.net
t=0 0
m=audio 52543 RTP/AVP 0
a=rtpmap:0 PCMU/8000
a=extmap:1/sendonly urn:ietf:params:rtp-hdrext:csrc-audio-level
```

A client-initiated example SDP offer/answer exchange negotiating an audio stream with one-way flow of of audio level information.

Figure 4

```
v=0
i=Un seminaire sur le protocole de description des sessions
o=fr-focus 2890844730 2890844730 IN IP6 focus.fr.example.net
c=IN IP6 focus.fr.example.net
t=0 0
m=audio 49170 RTP/AVP 0
a=rtpmap:0 PCMU/8000
a=extmap:1/sendrecv urn:ietf:params:rtp-hdrext:csrc-audio-level

v=0
i=A Seminar on the session description protocol
o=us-focus 2890844526 2890844526 IN IP6 focus.us.example.net
c=IN IP6 focus.us.example.net
t=0 0
m=audio 52543 RTP/AVP 0
a=rtpmap:0 PCMU/8000
a=extmap:1/sendrecv urn:ietf:params:rtp-hdrext:csrc-audio-level
```

An example SDP offer/answer exchange between two conference focus entities with mixing capabilities negotiating an audio stream with bidirectional flow of audio level information.

Figure 5

6. Security Considerations

1. This document defines a means of attributing audio level to a particular participant in a conference. An attacker may try to modify the content of RTP packets in a way that would make audio activity from one participant appear as coming from another.
2. Furthermore, the fact that audio level values would not be protected even in an SRTP session might be of concern in some cases where the activity of a particular participant in a conference is confidential. Also, as discussed in [I-D.perkins-avt-srtp-vbr-audio], an attacker might be able to infer information about the conversation, possibly with phoneme-level resolution.
3. Both of the above are concerns that stem from the design of the RTP protocol itself and they would probably also apply when using CSRC identifiers the way they were specified in RFC 3550 [RFC3550]. It is therefore important that according to the needs of a particular scenario, implementors and deployers consider use of header extension encryption [I-D.lennox-avtcore-srtp-encrypted-header-ext] or a lower level security and authentication mechanism.

7. IANA Considerations

This document defines a new extension URI that, if approved, would need to be added to the RTP Compact Header Extensions sub-registry of the Real-Time Transport Protocol (RTP) Parameters registry, according to the following data:

```
Extension URI: urn:ietf:params:rtp-hdrext:csrc-audio-level
Description:   Mixer-to-client audio level indicators
Contact:       emcho@jitsi.org
Reference:     RFC XXXX
```

Note to the RFC-Editor: please replace "RFC XXXX" by the number of this RFC.

8. Acknowledgments

Lyubomir Marinov contributed level measurement and rendering code.

Keith Drage, Roni Even, Ingemar Johansson, Michael Ramalho, Magnus Westerlund and several others provided helpful feedback over the dispatch mailing list.

Jitsi’s participation in this specification is funded by the NLnet

Foundation.

9. Changes From Earlier Versions

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

9.1. Changes From Draft -02

- Removed the no-data use case that allowed sending levels in RTP packets. Choosing the right RTP payload type for this use case would have incurred complexity without bringing any real value.
- Merged the "Header Format" and the "Audio level encoding" sections into a single "Audio Levels" section.
- Changed encoding related text so that it would cover both the one-byte and the two-byte header formats.
- Clarified use of root mean square for dBov calculation
- Added a reference to [I-D.perkins-avt-srtp-vbr-audio] to better explain some "Security Considerations" .
- Other minor editorial changes.

9.2. Changes From Draft -01

- Removed code related the AudioLevelRenderer from "APPENDIX A. Reference Implementation" as it was considered an implementation matter by the working group.
- Modified the AudioLevelCalculator in "APPENDIX A. Reference Implementation" to take overload as a parameter.
- Clarified non-use of audio levels in video streams
- Closed the P.56 open issue. It was agreed on IETF 80 that P.56 is mostly about speech levels and the levels transported by the extension defined here should also be able to serve as an indication for noise.
- The Open Issues section has been removed as all issues that were in there are now resolved or clarified.
- Editorial changes for consistency with [I-D.ietf-avtext-client-to-mixer-audio-level].

9.3. Changes From Draft -00

- Added code for sound pressure calculation and measurement in "APPENDIX A. Reference Implementation".
- Changed affiliation for Emil Ivov.
- Removed "Appendix: Design choices".

10. References

10.1. 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.

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

10.2. Informative References

[I-D.ietf-avtext-client-to-mixer-audio-level]
Lennox, J., Ivov, E., and E. Marocco, "A Real-Time Transport Protocol (RTP) Header Extension for Client-to-Mixer Audio Level Indication", draft-ietf-avtext-client-to-mixer-audio-level-02 (work in progress), June 2011.

[I-D.lennox-avtcore-srtp-encrypted-header-ext]
Lennox, J., "Encryption of Header Extensions in the Secure Real-Time Transport Protocol (SRTP)", draft-lennox-avtcore-srtp-encrypted-header-ext-00 (work in progress), March 2011.

[I-D.perkins-avt-srtp-vbr-audio]
Perkins, C. and J. Valin, "Guidelines for the use of Variable Bit Rate Audio with Secure RTP", draft-perkins-avt-srtp-vbr-audio-05 (work in progress), December 2010.

[ITU.G.711]
International Telecommunications Union, "Pulse Code Modulation (PCM) of Voice Frequencies", ITU-T Recommendation G.711, November 1988.

[ITU.P56.1993]
International Telecommunications Union, "Objective Measurement of Active Speech Level", ITU-T Recommendation P.56, March 1988.

[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.

[RFC3389] Zopf, R., "Real-time Transport Protocol (RTP) Payload for Comfort Noise (CN)", RFC 3389, September 2002.

[RFC4353] Rosenberg, J., "A Framework for Conferencing with the Session Initiation Protocol (SIP)", RFC 4353, February 2006.

[RFC4575] Rosenberg, J., Schulzrinne, H., and O. Levin, "A Session Initiation Protocol (SIP) Event Package for Conference State", RFC 4575, August 2006.

Appendix A. Reference Implementation

This appendix contains Java code for a reference implementation of the level calculation and rendering methods.The code is not normative and by no means the only possible implementation. Its purpose is to help implementors add audio level support to mixers and clients.

The Java code contains an AudioLevelCalculator class that calculates the sound pressure level of a signal with specific samples. It can be used in mixers to generate values suitable for the level extension headers.

The implementation is provided in Java but does not rely on any of the language specific and can be easily ported to another.

A.1. AudioLevelCalculator.java

```
/**
 * Calculates the audio level of specific samples of a singal based on
 * sound pressure level.
 */
public class AudioLevelCalculator
{

    /**
     * Calculates the sound pressure level of a signal with specific
     * <tt>samples</tt>.
     *
     * @param samples the samples of the signal to calculate the sound
     * pressure level of. The samples are specified as an <tt>int</tt>
     * array starting at <tt>offset</tt>, extending <tt>length</tt>
     * number of elements and each <tt>int</tt> element in the specified
     * range representing a sample of the signal to calculate the sound
     * pressure level of. Though a sample is provided in the form of an
     * <tt>int</tt> value, the sample size in bits is determined by the
```

```
     * caller via <tt>overload</tt>.
     *
     * @param offset the offset in <tt>samples</tt> at which the samples
     * start
     *
     * @param length the length of the signal specified in
     * <tt>samples<tt> starting at <tt>offset</tt>
     *
     * @param overload the overload (point) of <tt>signal</tt>.
     * For example, <tt>overload</tt> may be {@link Byte#MAX_VALUE}
     * for 8-bit signed samples or {@link Short#MAX_VALUE} for
     * 16-bit signed samples.
     *
     * @return the sound pressure level of the specified signal
     */
    public static int calculateSoundPressureLevel(
        int[] samples, int offset, int length,
        int overload)
    {
        /*
         * Calcuate the root mean square of the signal i.e. the
         * effective sound pressure.
         */
        double rms = 0;

        for (; offset < length; offset++)
        {
            double sample = samples[offset];

            sample /= overload;
            rms += sample * sample;
        }
        rms = (length == 0) ? 0 : Math.sqrt(rms / length);

        /*
         * The sound pressure level is a logarithmic measure of the
         * effectivesound pressure of a sound relative to a reference
         * value and is measured in decibels.
         */
        double db;

        /*
         * The minimum sound pressure level which matches the maximum
         * of the sound meter.
         */
        final double MIN_SOUND_PRESSURE_LEVEL = 0;
        /*
         * The maximum sound pressure level which matches the maximum
```

```
         * of the sound meter.
         */
        final double MAX_SOUND_PRESSURE_LEVEL
            = 127 /* HUMAN TINNITUS (RINGING IN THE EARS) BEGINS */;

        if (rms > 0)
        {
            /*
             * The commonly used "zero" reference sound pressure in air
             * is 20 uPa RMS, which is usually considered the threshold
             * of human hearing.
             */
            final double REF_SOUND_PRESSURE = 0.00002;

            db = 20 * Math.log10(rms / REF_SOUND_PRESSURE);

            /*
             * Ensure that the calculated level is within the minimum
             * and maximum sound pressure level.
             */
            if (db < MIN_SOUND_PRESSURE_LEVEL)
                db = MIN_SOUND_PRESSURE_LEVEL;
            else if (db > MAX_SOUND_PRESSURE_LEVEL)
                db = MAX_SOUND_PRESSURE_LEVEL;
        }
        else
        {
            db = MIN_SOUND_PRESSURE_LEVEL;
        }

        return (int) db;
    }
}
```

AudioLevelCalculator.java

Authors' Addresses

Emil Ivov (editor)
Jitsi
Strasbourg 67000
France

Email: emcho@jitsi.org

Enrico Marocco (editor)
Telecom Italia
Via G. Reiss Romoli, 274
Turin 10148
Italy

Email: enrico.marocco@telecomitalia.it

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

Email: jonathan@vidyo.com

Network Working Group
Internet-Draft
Updates: 3550 (if approved)
Intended status: Standards Track
Expires: January 12, 2012

M. Petit-Huguenin
Stonyfish, Inc.

July 11, 2011

Support for multiple clock rates in an RTP session
draft-ietf-avtext-multiple-clock-rates-01

Abstract

This document clarifies the RTP specification when different clock rates are used in an RTP session. It also provides guidance on how to interoperate with legacy RTP implementations that use multiple clock rates.

Status of this Memo

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

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

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

This Internet-Draft will expire on January 12, 2012.

Copyright Notice

Table of Contents

1. Introduction

The clock rate is a parameter of the payload format. It is often defined as been the same as the sampling rate but it is not always the case (see e.g. the G722 and MPA audio codecs in [RFC3551]).

An RTP sender can switch between different payloads during the lifetime of an RTP session and because clock rates are defined by payload types, it is possible that the clock rate also varies during an RTP session. RTP [RFC3550] lists using multiple clock rates as one of the reasons to not use different payloads on the same SSRC but unfortunately this advice was not always followed and some RTP implementations change the payload in the same SSRC even if the different payloads use different clock rates.

This creates three problems:

- The method used to calculate the RTP timestamp field in an RTP packet is underspecified.
- When the same SSRC is used for different clock rates, it is difficult to know what clock rate was used for the RTP timestamp field in an RTCP SR packet.
- When the same SSRC is used for different clock rates, it is difficult to know what clock rate was used for the interarrival jitter field in an RTCP RR packet.

Table 1 contains a non-exhaustive list of fields in RTCP packets that uses a clock rate as unit:

Field name	RTCP packet type	Reference
RTP timestamp	SR	[RFC3550]
Interarrival jitter	RR	[RFC3550]
min_jitter	XR Summary Block	[RFC3611]
max_jitter	XR Summary Block	[RFC3611]
mean_jitter	XR Summary Block	[RFC3611]
dev_jitter	XR Summary Block	[RFC3611]
Interarrival jitter	IJ	[RFC5450]
RTP timestamp	SMPTETC	[RFC5484]
Jitter	RSI Jitter Block	[RFC5760]
Median jitter	RSI Stats Block	[RFC5760]

Table 1

This document first tries to list in Section 2 and subsections all known algorithms used in existing RTP implementations at the time of writing. This sections are not normative.

Section 4 and subsections then recommend a unique algorithm that modifies [RFC3550]. This sections are normative.

Section 5 and subsections then analyze what happen when the legacy algorithms listed in Section 2 are used with the new algorithm listed in Section 4. This sections are not normative.

2. Legacy RTP

The following sections describe the various ways legacy RTP implementations behave when multiple clock rates are used. Legacy RTP refers to RFC 3550 without the modifications introduced by this document.

[[We need to list here all the methods used in the field. Please send them to the author. NDA can be arranged if needed]]

2.1. Different SSRC

One way of managing multiple clock rates is to use a different SSRC for each different clock rate, as in this case there is no ambiguity on the clock rate used by fields in the RTCP packets. This method also seems to be the original intent of RTP as can be deduced from points 2 and 3 of section 5.2 of RFC 3550.

On the other hand changing the SSRC can be a problem for some implementations designed to work only with unicast IP addresses, where having multiple SSRCs is considered a corner case. Lip synchronization can also be a problem in the interval between the beginning of the new stream and the first RTCP SR packet. This is not different than what happen at the beginning of the RTP session but it can be more annoying for the end-user.

2.2. Same SSRC

The simplest way of managing multiple clock rates is to use the same SSRC for all the payload types regardless of the clock rates.

Unfortunately there is no clear definition on how the RTP timestamp should be calculated in this case. The following subsection presents one algorithm used in the field.

2.2.1. Monotonic timestamps

The most common method of calculating the RTP timestamp ensures that the value increases monotonically. The formula used by this method is as follow:

timestamp = previous_timestamp + (current_capture_time - previous_capture_time) * current_clock_rate

The problem with this method is that the jitter calculation on the receiving side gives invalid result during the transition between two clock rates, as shown in Table 2. The capture and arrival time are in seconds, starting at the beginning of the capture of the first packet; clock rate is in Hz; the RTP timestamp does not include the random offset; the transit, jitter, and average jitter use the clock rate as unit.

Capt. time	Clock rate	RTP timestamp	Arrival time	Transit	Jitter	Average jitter
0	8000	0	0.1	800		
0.02	8000	160	0.12	800	0	0
0.04	8000	320	0.14	800	0	0
0.06	8000	480	0.16	800	0	0
0.08	16000	800	0.18	2080	480	30
0.1	16000	1120	0.2	2080	0	28
0.12	16000	1440	0.22	2080	0	26
0.14	8000	1600	0.24	320	720	70
0.16	8000	1760	0.26	320	0	65

Table 2

Calculating the correct transit time on the receiving side can be done by using the following formulas:

(1) current_time_capture = current_timestamp - previous_timestamp) / current_clock_rate + previous_time_capture
(2) transit = current_clock_rate * (time_arrival - current_time_capture)
(3) previous_time_capture = current_time_capture

The main problem with this method, in addition to the fact that the jitter calculation described in RFC 3550 cannot be used, is that is it dependent on the previous RTP packets, packets that can be reordered or lost in the network. But it seems that this is what most implementations are using.

3. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this

document are to be interpreted as described in [RFC2119].

Clock rate: The multiplier used to convert from a wallclock value in seconds to an equivalent RTP timestamp value (without the fixed random offset). Note that RFC 3550 uses various terms like "clock frequency", "media clock rate", "timestamp unit", "timestamp frequency", and "RTP timestamp clock rate" as synonymous to clock rate.

RTP Sender: A logical network element that sends RTP packets, sends RTCP SR packets, and receives RTCP RR packets.

RTP Receiver: A logical network element that receives RTP packets, receives RTCP SR packets, and sends RTCP RR packets.

4. Recommendations

4.1. RTP Sender

An RTP Sender with RTCP turned off (i.e. by setting the RS and RR bandwidth modifiers defined in [RFC3556] to 0) SHOULD use a different SSRC for each different clock rate but MAY use different clock rates on the same SSRC as long as the RTP timestamp without the random offset is calculated as explained below:

[[This was designed to help VoIP implementations who anyway never cared about RTCP. Do we want to keep this?]]

Each time the clock rate changes, the start_offset and capture_start values are calculated with the following formulas:

```
start_offset = (capture_time - capture_start) * previous_clock_rate
capture_start = capture_time
```

For the first RTP packet, the values are initialized with the following formulas:

```
start_offset = 0
capture_start = capture_time
```

After eventually updating this values, the RTP timestamp is calculated with the following formula:

```
timestamp = (capture_time - capture_start) * clock_rate +
start_offset
```

Note that in all the formulas, capture_time is the first instant the new timestamp rate is used.

An RTP Sender with RTCP turned on MUST use a different SSRC for each different clock rate. An RTCP BYE MUST be sent and a new SSRC MUST be used if the clock rate switches back to a value already seen in the RTP stream.

To accelerate lip synchronization, the next compound RTCP packet sent by the RTP sender MUST contain multiple SR packets, the first one containing the mapping for the current clock rate and the next SR packets containing the mapping for the other clock rates seen during the last period.

[[Some legacy implementations may dislike receiving multiple SR packets. What should we do?]]

The RTP extension defined in [RFC6051] MAY be used to accelerate the synchronization.

4.2. RTP Receiver

An RTP Receiver MUST calculate the jitter using the following formula:

D(i,j) = (arrival_time_j * clock_rate_i - timestamp_j) - (arrival_time_i * clock_rate_i - timestamp_i)

An RTP Receiver MUST be able to handle a compound RTCP packet with multiple SR packets.

For interoperability with legacy RTP implementations, an RTP receiver MAY use the information in two consecutive SR packets to calculate the clock rate used, i.e. if Ni is the NTP timestamp for the SR packet i, Ri the RTP timestamp for the SR packet i and Nj and Rj the NTP timestamp and RTP timestamp for the previous SR packet j, then the clock rate can be guessed as the closest to (Ri - Rj) / (Ni - Nj).

5. Interoperability Analysis

The next subsections analyze the various combinations between legacy RTP implementations and RTP implementations that follow this document specifications.

TBD

6. Security Considerations

TBD

7. IANA Considerations

No IANA considerations.

8. Acknowledgements

Thanks to Colin Perkins, Ali C. Begen and Magnus Westerlund for their comments, suggestions and questions that helped to improve this document.

Thanks to Robert Sparks and the attendees of SIPit 26 for the survey on multiple clock rates interoperability.

This document was written with the xml2rfc tool described in [RFC2629].

9. References

9.1. Normative References

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

[RFC3550] Schulzrinne, H., Casner, S., Frederick, R., and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications", STD 64, RFC 3550, July 2003.

9.2. Informative References

[RFC2629] Rose, M., "Writing I-Ds and RFCs using XML", RFC 2629, June 1999.

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

[RFC3556] Casner, S., "Session Description Protocol (SDP) Bandwidth Modifiers for RTP Control Protocol (RTCP) Bandwidth", RFC 3556, July 2003.

[RFC3611] Friedman, T., Caceres, R., and A. Clark, "RTP Control

Protocol Extended Reports (RTCP XR)", RFC 3611, November 2003.

[RFC5450] Singer, D. and H. Desineni, "Transmission Time Offsets in RTP Streams", RFC 5450, March 2009.

[RFC5484] Singer, D., "Associating Time-Codes with RTP Streams", RFC 5484, March 2009.

[RFC5760] Ott, J., Chesterfield, J., and E. Schooler, "RTP Control Protocol (RTCP) Extensions for Single-Source Multicast Sessions with Unicast Feedback", RFC 5760, February 2010.

[RFC6051] Perkins, C. and T. Schierl, "Rapid Synchronisation of RTP Flows", RFC 6051, November 2010.

[uRTR] Wenger, S. and C. Perkins, "RTP Timestamp Frequency for Variable Rate Audio Codecs", draft-ietf-avt-variable-rate-audio-00 (work in progress), October 2004.

Appendix A. Using a fixed clock rate

An alternate way of fixing the multiple clock rates issue was proposed in [uRTR]. This document proposed to define a unified clock rate, but the proposal was rejected at IETF 61.

Appendix B. Release notes

This section must be removed before publication as an RFC.

B.1. Modifications between draft-ietf-avtext-multiple-clock-rates-01 and draft-ietf-avtext-multiple-clock-rates-00

- New text says that the algorithms listed are the one currently known.
- Explains capture_time.
- Nits.

B.2. Modifications between draft-ietf-avtext-multiple-clock-rates-00 and draft-petithuguenin-avtext-multiple-clock-rates-01

- Changed capture_state to capture_start.

B.3. Modifications between draft-petithuguenin-avtext-multiple-clock-rates-01 and draft-petithuguenin-avtext-multiple-clock-rates-00

- Clarified the goals for this documents
- Removed the non-monotonic method (replaced by Magnus formula).
- Moved the "RTP Sender and RTP Receiver section inside a new "Recommendations" section.
- Inserted the new Sender formula inside the Recommendation section.
- Inserted the new jitter formula in the RTP Receiver section.
- Emptied the Analysis sections.

B.4. Modifications between draft-petithuguenin-avtext-multiple-clock-rates-00 and draft-petithuguenin-avt-multiple-clock-rates-03

- Initial release for avtext WG.

B.5. Modifications between draft-petithuguenin-avt-multiple-clock-rates-03 and draft-petithuguenin-avt-multiple-clock-rates-02

- Updated RFC reference.

B.6. Modifications between draft-petithuguenin-avt-multiple-clock-rates-02 and draft-petithuguenin-avt-multiple-clock-rates-01

- Having multiple SRs in a compound RTCP packet is OK.
- If RTCP is used, must send a BYE and not reuse the SSRC.
- Removed resolved notes.
- Acknowledged SIPit 26 survey.
- Fixed some nits.

B.7. Modifications between draft-petithuguenin-avt-multiple-clock-rates-01 and draft-petithuguenin-avt-multiple-clock-rates-00

- Complete rewrite as a Standard Track I-D modifying RFC 3550.

Author’s Address

Marc Petit-Huguenin
Stonyfish, Inc.

Email: petithug@acm.org

AVTEXT
Internet-Draft
Intended status: Informational
Expires: December 12, 2011

A. Begen
Cisco

June 10, 2011

Considerations and Guidelines for Deploying the Rapid Acquisition of Multicast RTP Sessions (RAMS) Method draft-ietf-avtext-rams-scenarios-00

Abstract

The Rapid Acquisition of Multicast RTP Sessions (RAMS) solution is a method based on RTP and RTP Control Protocol (RTCP) that enables an RTP receiver to rapidly acquire and start consuming the RTP multicast data. Upon a request from the RTP receiver, an auxiliary unicast RTP retransmission session is set up between a retransmission server and the RTP receiver, over which the reference information about the new multicast stream the RTP receiver is about to join is transmitted at an accelerated rate. This often precedes, but may also accompany, the multicast stream itself. When there is only one multicast stream to be acquired, the RAMS solution works in a straightforward manner. However, when there are two or more multicast streams to be acquired from the same or different multicast RTP sessions, care should be taken to configure each RAMS session appropriately. This document provides example scenarios and offers guidelines.

Status of this Memo

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

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

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

This Internet-Draft will expire on December 12, 2011.

Copyright Notice

Table of Contents

1. Introduction

The Rapid Acquisition of Multicast RTP Sessions (RAMS) solution is a method based on RTP and RTP Control Protocol (RTCP) that enables an RTP receiver to rapidly acquire and start consuming the RTP multicast data. Through an auxiliary unicast RTP retransmission session [RFC4588], the RTP receiver receives a reference information about the new multicast stream it is about to join. This often precedes, but may also accompany, the multicast stream itself. The RAMS solution is documented in detail in [I-D.ietf-avt-rapid-acquisition-for-rtp].

The RAMS specification [I-D.ietf-avt-rapid-acquisition-for-rtp] has provisions for concurrently acquiring multiple streams inside a multicast RTP session. However, the specification has mostly focused on the simplest case, which is when the RTP receiver acquires only one multicast stream at a time, to explain the protocol details.

There are certain deployment models where a multicast RTP session may have two or more multicast streams associated with it. There are also cases, where an RTP receiver may be interested in acquiring one or more multicast streams from several multicast RTP sessions. In scenarios where multiple RAMS sessions will be simultaneously run by an RTP receiver, they need to be coordinated. In this document, we present scenarios from real-life deployments and provide guidelines.

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].

Editor’s note: I am inclined not use any 2119 keyword in this document and remove this section altogether.

3. Background

In the following discussion, we assume that there are two RTP streams (1 and 2) that are somehow associated with each other. These could be audio and video elementary streams for the same TV channel, or they could be an MPEG2-TS stream (that has audio and video multiplexed together) and its Forward Error Correction (FEC) stream.

It is important to note that a source-specific multicast (SSM) session is defined by its (distribution) source address and

(destination) multicast group and there can be only one feedback target per SSM session [RFC5760]. So, if the RTP streams are distributed by different sources or over different multicast groups, they are considered different SSM sessions. While different SSM sessions can normally share the same feedback target address and/or port, RAMS requires each unique feedback target (i.e., the combination of address and port) to be associated with at most one RTP session (See Section 6.2 of [I-D.ietf-avt-rapid-acquisition-for-rtp]).

Two or more multicast RTP streams can be transmitted in the same RTP session (i.e., in a single UDP flow). This is called Synchronization Source (SSRC) multiplexing. In this case, (de)multiplexing is done at the SSRC level. Alternatively, the multicast RTP streams can be transmitted in different RTP sessions (i.e., in different UDP flows), which is called session multiplexing. In practice, there are different deployment models for each multiplexing scheme.

Generally, two different media streams with different clock rates are suggested to use different SSRCs or to be carried in different RTP sessions to avoid complications in RTCP reports. Some of the fields in RAMS messages might depend on the clock rate. Thus, in a single RTP session, RTP streams carrying payloads with different clock rates need to have different SSRCs. Since RTP streams in the same RTP session but with different SSRCs do not share the sequence numbering, each stream needs to be acquired individually.

In the remaining sections, only the relevant portions of the SDP descriptions [RFC4566] will be provided. For an example of a full SDP description, refer to Section 8.3 of [I-D.ietf-avt-rapid-acquisition-for-rtp].

4. Example Scenarios

4.1. Scenario #1: Two Multicast Groups

This is the scenario for session multiplexing where RTP streams 1 and 2 are transmitted over different multicast groups. A practical use case is where the first and second SSM sessions carry the primary video stream and its associated FEC stream, respectively.

We run an individual RAMS session for each of these RTP streams that we want to rapidly acquire. These RAMS sessions can be run in parallel. If they are, the RTP receiver needs to pay attention to using the shared bandwidth appropriately among the two unicast bursts. As explained earlier, there has to be a different feedback target for these two SSM sessions.

```
a=group:FEC-FR Channel1_Video Channel1_FEC
m=video 40000 RTP/AVPF 96
c=IN IP4 233.252.0.1/127
a=source-filter:incl IN IP4 233.252.0.1 198.51.100.1
a=rtcp:41000 IN IP4 192.0.2.1
a=ssrc:1 cname:ch1_video@example.com
a=mid:Channel1_Video
m=application 40000 RTP/AVPF 97
c=IN IP4 233.252.0.2/127
a=source-filter:incl IN IP4 233.252.0.2 198.51.100.1
a=rtcp:42000 IN IP4 192.0.2.1
a=ssrc:2 cname:ch1_fec@example.com
a=mid:Channel1_FEC
```

Note that the multicast destination ports in the above SDP do not matter, and they could be the same or different. The "FEC-FR" grouping semantics are defined in [RFC5956].

4.2. Scenario #2: One Multicast Group

This is the scenario for session multiplexing where RTP streams 1 and 2 are transmitted over the same multicast group with different destination ports. A practical use case is where the SSM session carries the primary video and audio streams, each destined to a different port.

Similar to scenario #1, we run individual RAMS sessions for each RTP stream that we want to rapidly acquire (Note that the RAMS request sent by an RTP receiver could indicate the desire to acquire all or a subset or one of the available RTP streams in an SSM session). Compared to the previous scenario, the only difference is that in this case the join times for both streams need to be coordinated as they are on the same multicast session.

```
m=video 40000 RTP/AVPF 96
c=IN IP4 233.252.0.1/127
a=source-filter:incl IN IP4 233.252.0.1 198.51.100.1
a=rtcp:41000 IN IP4 192.0.2.1
a=ssrc:1 cname:ch1_video@example.com
a=mid:Channel1_Video
m=audio 40001 RTP/AVPF 97
c=IN IP4 233.252.0.1/127
a=source-filter:incl IN IP4 233.252.0.1 198.51.100.1
a=rtcp:41000 IN IP4 192.0.2.1
a=ssrc:2 cname:ch1_audio@example.com
a=mid:Channel1_Audio
```

Note that the destination ports in the above SDP need to be distinct per [RFC5888].

If RTP streams 1 and 2 share the same distribution source, then there is only one SSM session, which means that there can be only one feedback target (as shown in the SDP description above). This requires RTP streams 1 and 2 to have their own unique SSRC values (also as shown in the SDP description above). If RTP streams 1 and 2 do not share the same distribution source, meaning that their respective SSM sessions can use different feedback target transport addresses, then their SSRC values do not have to be different from each other.

4.3. Scenario #3: SSRC Multiplexing

This is the scenario for SSRC multiplexing where both RTP streams are transmitted over the same multicast group to the same destination port. This is a less practical scenario but it could be used where the SSM session carries both the primary video and audio stream, destined to the same port.

Similar to scenario #2, we run individual RAMS sessions and the join time needs to be coordinated. In this case, there is only one distribution source and the destination multicast address is shared. Thus, there is always one SSM session and one feedback target.

```
m=video 40000 RTP/AVPF 96 97
c=IN IP4 233.252.0.1/127
a=source-filter:incl IN IP4 233.252.0.1 198.51.100.1
a=rtcp:41000 IN IP4 192.0.2.1
a=ssrc:1 cname:ch1_video@example.com
a=ssrc:2 cname:ch1_audio@example.com
a=mid:Channel1
```

4.4. Scenario #4: Payload-Type Multiplexing

This is the scenario for payload-type multiplexing.

In this case, instead of two, we have only one RTP stream (and one RTP session) carrying both payload types (e.g., media payload and its FEC data). While this scheme is permissible per [RFC3550], it has several drawbacks. For example, RTP packets carrying different payload formats will share the same sequence numbering space, and the retransmission and RAMS operations will not be able to be applied based on the payload type. For other drawbacks and details, see Section 5.2 of [RFC3550].

5. Feedback Target and SSRC Signaling Issues

The RAMS protocol uses the common packet format from [RFC4585], which has a field to signal the media sender SSRC. The SSRCs for the RTP streams can be signaled out-of-band in the SDP, or could be learned from the RTP packets once the transmission starts. In RAMS, the latter cannot be used.

Signaling the media sender SSRC value helps the feedback target correctly identify the RTP stream to be acquired. If a feedback target is serving multiple SSM sessions on a particular port, all the RTP streams in these SSM sessions are supposed to have a unique SSRC value. However, since this is not an easy requirement to satisfy, RAMS specification forbids to have more than one RTP session to be associated with a specific feedback target.

6. FEC during RAMS and Bandwidth Issues

Suppose that RTP stream 1 denotes the primary video stream that has a bitrate of 10 Mbps and RTP stream 2 denotes the FEC stream that has a bitrate of 1 Mbps. Also assume that the RTP receiver knows that it can receive data at a maximum bitrate of 22 Mbps. SDP can specify the bitrate ("b=" line in Kbps) of each media session (per "m" line).

6.1. Scenario #1

This is the scenario for session multiplexing where RTP streams 1 and 2 are transmitted over different multicast groups.

This is the preferred deployment model for FEC. Having FEC in a different multicast group provides flexibility for not only the RTP receivers that are not FEC capable but also the ones that are FEC capable but are not willing to receive FEC during the rapid acquisition.

```
a=group:FEC-FR Channel1_Video Channel1_FEC
m=video 40000 RTP/AVPF 96
c=IN IP4 233.252.0.1/127
a=source-filter:incl IN IP4 233.252.0.1 198.51.100.1
a=rtcp:41000 IN IP4 192.0.2.1
a=rtpmap:96 MP2T/90000
b=TIAS:10000
a=ssrc:1 cname:ch1_video@example.com
a=mid:Channel1_Video
m=application 40000 RTP/AVPF 97
c=IN IP4 233.252.0.2/127
a=source-filter:incl IN IP4 233.252.0.2 198.51.100.1
a=rtcp:42000 IN IP4 192.0.2.1
a=rtpmap:97 1d-interleaved-parityfec/90000
b=TIAS:1000
a=ssrc:2 cname:ch1_fec@example.com
a=mid:Channel1_FEC
```

If the RTP receiver does not want to receive FEC until the acquisition of the primary video stream is completed, the total available bandwidth can be used for faster acquisition of the primary video stream. In this case, the RTP receiver can request a Max Receive Bitrate of 22 Mbps in the RAMS Request message. Once RAMS has been completed, the RTP receiver can join the FEC multicast session, if desired.

If the RTP receiver wants to rapidly acquire both primary and FEC streams, it needs to allocate the total bandwidth among the two RAMS sessions and indicate individual Max Receive Bitrate values in each respective RAMS Request message. Since less bandwidth will be used to acquire the primary video stream, the acquisition of the primary video session will take a longer time on the average.

While the RTP receiver can update the Max Receive Bitrate values during the course of the RAMS session, this approach is more error-prone due to the possibility of losing the update messages.

6.2. Scenario #2

This is the scenario for session multiplexing where RTP streams 1 and 2 are transmitted over the same multicast group with different destination ports.

```
a=group:FEC-FR Channel1_Video Channel1_FEC
m=video 40000 RTP/AVPF 96
c=IN IP4 233.252.0.1/127
a=source-filter:incl IN IP4 233.252.0.1 198.51.100.1
a=rtcp:41000 IN IP4 192.0.2.1
a=rtpmap:96 MP2T/90000
b=TIAS:10000
a=ssrc:1 cname:ch1_video@example.com
a=mid:Channel1_Video
m=application 40001 RTP/AVPF 97
c=IN IP4 233.252.0.1/127
a=source-filter:incl IN IP4 233.252.0.1 198.51.100.1
a=rtcp:41000 IN IP4 192.0.2.1
a=rtpmap:97 1d-interleaved-parityfec/90000
b=TIAS:1000
a=ssrc:2 cname:ch1_fec@example.com
a=mid:Channel1_FEC
```

Similar to scenario #1, the RTP receiver can first ask for RAMS for the primary video stream at the full receive bitrate. But, upon the multicast join, the available bandwidth for the burst drops to 11 Mbps instead of 12 Mbps. Regardless of whether FEC is desired or not by the RTP receiver, its bitrate needs to be taken into account once the RTP receiver joins the SSM session.

If the RTP receiver wants to rapidly acquire both primary and FEC streams, the same conditions explained for scenario #1 apply. The only difference from scenario #1 is that here the join time is the same for both the primary video and FEC streams.

6.3. Scenario #3

This is the scenario for SSRC multiplexing where both RTP streams are transmitted over the same multicast group to the same destination port.

```
m=video 40000 RTP/AVPF 96 97
c=IN IP4 233.252.0.1/127
a=source-filter:incl IN IP4 233.252.0.1 198.51.100.1
a=rtcp:41000 IN IP4 192.0.2.1
a=rtpmap:96 MP2T/90000
a=rtpmap:97 1d-interleaved-parityfec/90000
a=fmtp:97 L=10; D=10; repair-window=200000
a=ssrc:1 cname:ch1_video@example.com
a=ssrc:2 cname:ch1_fec@example.com
a=mid:Channel1
b=TIAS:11000
a=mid:Channel1
```

This is similar to scenario #2. However, since we cannot explicitly specify the bitrates for the primary and FEC streams, the RTP receiver can request to rapidly acquire both streams in parallel. In this case, two separate RAMS Request messages have to be sent by the RTP receiver to the feedback target.

Note that based on the "a=fmtp" line for the FEC stream, it could be possible to infer the bitrate of this FEC stream and set the Max Receive Bitrate value accordingly.

7. Security Considerations

There are no security considerations in this document.

8. IANA Considerations

There are no IANA considerations in this document.

9. Acknowledgments

I would like to thank various individuals in the AVTEXT and MMUSIC WGs, and my friends at Cisco for their comments and feedback.

10. References

10.1. Normative References

[I-D.ietf-avt-rapid-acquisition-for-rtp]
Steeg, B., Begen, A., Caenegem, T., and Z. Vax, "Unicast-Based Rapid Acquisition of Multicast RTP Sessions", draft-ietf-avt-rapid-acquisition-for-rtp-17 (work in

progress), November 2010.

[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.

[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.

[RFC5760] Ott, J., Chesterfield, J., and E. Schooler, "RTP Control Protocol (RTCP) Extensions for Single-Source Multicast Sessions with Unicast Feedback", RFC 5760, February 2010.

10.2. Informative References

[RFC5888] Camarillo, G. and H. Schulzrinne, "The Session Description Protocol (SDP) Grouping Framework", RFC 5888, June 2010.

[RFC5956] Begen, A., "Forward Error Correction Grouping Semantics in the Session Description Protocol", RFC 5956, September 2010.

Author’s Address

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

Email: abegen@cisco.com

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: December 11, 2011

M. Ramalho, Ed.
P. Jones
Cisco Systems
N. Harada
NTT
M. Perumal
Cisco Systems
L. Miao
Huawei Technologies
June 9, 2011

RTP Payload Format for G.711.0
draft-ramalho-payload-g7110-00

Abstract

This document specifies the Real-Time Transport Protocol (RTP) payload format for ITU-T Recommendation G.711.0. ITU-T Rec. G.711.0 defines a lossless and stateless compression for G.711 packet payloads typically used in IP networks. This document also defines two storage mode formats for G.711.0. A media type registration for this RTP payload format is also included.

Status of this Memo

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

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

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

This Internet-Draft will expire on December 11, 2011.

Copyright Notice

Table of Contents

1. Introduction

The International Telecommunication Union (ITU-T) Recommendation G.711.0 [G.711.0] specifies a stateless and lossless compression for G.711 packet payloads typically used in VoIP networks. This document specifies the Real-Time Transport Protocol (RTP) RFC 3550 [RFC3550] payload format and storage modes for this compression.

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. G.711.0 Codec Background

ITU-T Recommendation G.711.0 [G.711.0] is a lossless and stateless compression mechanism for ITU-T Recommendation G.711 [G.711] and thus is not a "codec" in the sense of "lossy" codecs typically carried by RTP. When negotiated end-to-end ITU-T Rec. G.711.0 is negotiated as if it were a codec, with the understanding that ITU-T Rec. G.711.0 losslessly encoded the underlying (lossy) G.711 pulse code modulation (PCM) sample representation of an audio signal. For this reason ITU-T Rec. G.711.0 will be interchangeably referred to in this document as a "lossless data compression algorithm" or a "codec", depending on context. Within this document, individual G.711 PCM samples will be referred to as "G.711 symbols" or just "symbols" for brevity.

This section describes the ITU-T Recommendation G.711 [G.711] codec, its properties, typical uses cases and its key design properties.

3.1. General Information and Use of the ITU-T G.711.0 Codec

ITU-T Recommendation G.711 is the benchmark standard for narrowband telephony. It has been successful for many decades because of its proven voice quality, ubiquity and utility. A new ITU-T recommendation, G.711.0, has been established for defining a stateless and lossless compression for G.711 packet payloads typically used in VoIP networks. ITU-T Rec. G.711.0 is also known as ITU-T Rec. G.711 Annex A [G.711-A1], as ITU-T Rec. G.711 Annex A is effectively a pointer ITU-T Rec. G.711.0. Henceforth in this document, ITU-T Rec. G.711.0 will simply be referred to as "G.711.0" and ITU-T Rec. G.711 simply as "G.711".

G.711.0 may be employed end to end; in which case the RTP payload format specification and use will be nearly identical to the G.711 RTP specification found in RFC 3550 [RFC3550]. The only significant difference other than the payload type (which will be a dynamically assigned payload type) will be the recommendation not to use Voice Activity Detection (as G.711.0 achieves its greatest compression during "VAD silence intervals"). SDP signaling elements are proposed for this use case of G.711.0 herein.

G.711.0, being both lossless and stateless, may also be employed as a LOSSLESS compression mechanism somewhere in between end systems which have negotiated use of G.711. For this case, the G.711 payloads and the corresponding G.711 RTP headers should appear to the end systems as having been transported transparently. This use case will be referred to as "G.711.0 in the Middle" and will be described in detail in Section 6 (Section 6). G.711.0, being both lossless and stateless, can be employed multiple times (e.g., on multiple,

individual hops or series of hops) of a given flow with no degradation of quality relative to end-to-end G.711. Stated another way, multiple "lossless transcodes" from/to G.711.0/G.711 do not negatively affect voice quality as may occur with lossy transcodes to/from dissimilar codecs. Since the use of G.711.0 as a compression mechanism can be used on any hop or hops of an end-to-end G.711 flow, neither Session Description Protocol (SDP) signaling elements nor G.711.0 negotiation mechanisms will be mandated in this document for this particular use case (although SDP descriptions in this document MAY be used for such G.711.0 negotiation).

Lastly, it is expected that G.711.0 will be used as an archival format for recorded G.711 streams. Therefore, a G.711.0 Storage Mode Format is also included in this document.

3.2. Key Properties of G.711.0 Design

The fundamental design of G.711.0 resulted from the desire losslessly encode and compress frames of G.711 symbols independent of what types of signals those G.711 frames contained. The primary G.711.0 use case is for G.711 encoded, zero-mean, acoustic signals (such as speech and music).

G.711.0 attributes are below:

A1 Compression for zero-mean acoustic signals: G.711.0 was designed as its primary use case for the compression of G.711 payloads which contained "speech" or other zero-mean acoustic signals. G.711.0 obtains greater than 50% average compression in service provider environments [ICASSP].

A2 Lossless for any G.711 payload: G.711.0 was designed to be lossless for any valid G.711 payload - even if the payload consisted of apparently random G.711 symbols (e.g., a modem or FAX payload). G.711.0 could be used for "aggregate 64 kbps G.711 channels" carried over IP without explicit concern if a subset of these channels happened to be carrying something other than voice or general audio. To the extent that a particular channel carried something than voice or general audio, G.711.0 ensured that it was carried losslessly, if not significantly compressed.

A3 Stateless: Compression of a frame of G.711 symbols was only to be dependent on that frame and not on any prior frame. Although greater compression is usually available by observing a longer history of past G.711 symbols, it was decided to for the compression design would be stateless to completely eliminate error propagation common in many lossy codec designs

(e.g., ITU-T Rec. G.729 [G.729], ITU-T Rec. G.722 [G.722]). That is, the decoding process need not be concerned about lost prior packets because the decompression of a given G.711.0 frame is not dependent on potentially lost prior frames. Owing to this stateless property, the frames input to the G.711.0 encoder may be changed "on-the-fly" (a 5 ms encoding could be followed by a 20 ms encoding).

A4 Self-describing: This property is defined as the ability to determine how many source G.711 samples are contained within the G.711.0 frame solely by information contained within the G.711.0 frame. Generally, the number of source G.711 symbols can be determined by decoding the initial octets of the compressed G.711.0 frame (these octets are called "prefix codes" in the standard)[ICASSP]. A G.711.0 decoder need not know what ptime is, as it is able to decompress the G.711.0 frame presented to it without signaling knowledge.

A5 Accommodate G.711 payload sizes typically used in IP: G.711 input frames of length typically found in VoIP applications represent SDP ptimes (see RFC 4566 [RFC4566]) of 5 ms, 10 ms, 20 ms, 30 ms or 40 ms. Since the dominant sampling frequency for G.711 is 8000 samples per second, G.711.0 was designed to compress G.711 input frames of 40, 80, 160, 240 or 320 samples.

A6 Bounded expansion: Since attribute A2 above requires G.711.0 to be lossless for any payload, by definition there exists at least one potential G.711 payload which must be "uncompressible". Since the quantum of compression is an octet, the minimum expansion of such an uncompressible payload was designed to be the minimum possible of one octet. Thus G.711.0 "compressed" frames can be of length one octet to X+1 octets, where X is the size of the input G.711 frame in octets. G.711.0 can therefore be viewed as a Variable Bit Rate (VBR) encoding in which the size of the G.711.0 output frame is a function of the G.711 symbols input to it.

A7 Algorithmic delay: G.711.0 was designed to have the algorithmic delay equal to the time represented by the number of samples in the G.711 input frame (i.e., no "look-ahead").

A8 Low Complexity: Less than 1.0 WMOPS average and low memory footprint (˜5k octets RAM, ˜5.7k octets ROM and ˜3.6 basic operations) [ICASSP] [G.711].

A9 Both A-law and Mu-law supported: G.711 has two operating laws, A-law and Mu-law. These two laws are also known as PCMA and PCMU in RFC 3550 [RFC3550]. The use of A-law or Mu-law should be signaled in SDP for IP applications.

These attributes generally make it trivial to compress a G.711 input frame consisting of 40, 80, 160, 240 or 320 samples. After the input frame is presented to a G.711.0 encoder, a G.711.0 "self-describing" output frame is produced. The number of samples contained within this frame is easily determined at the G.711.0 decoder by virtue of attribute A4. The G.711.0 decoder can decode the G.711.0 frame back to a G.711 frame by using only data within the G.711.0 frame.

Lastly we note that losing a G.711.0 encoded packet is identical in effect of losing a G.711 packet (when using RTP); this is because a G.711.0 payload, like the corresponding G.711 payload, is stateless. Thus, it is anticipated that existing G.711 PLC mechanisms will be employed when a G.711.0 packet is lost and an identical MOS degradation relative to G.711 loss will be achieved.

3.3. G.711 Input Frames to G.711.0 Output Frames

G.711.0 is a lossless and stateless compression of G.711 frames. The following figure depicts this where "A" is the process of G.711.0 encoding and "B" is the process of G.711.0 decoding.

1:1 Mapping from G.711 Input Frame to G.711.0 Output Frame

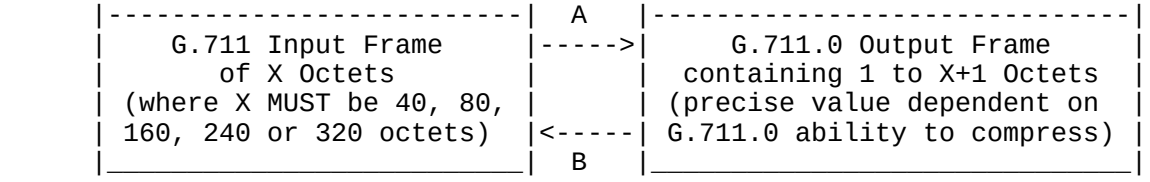


Figure 1

Note that the mapping is 1:1 (lossless) in both directions, subject to two constraints. The first constraint is that the input frame provided to the G.711.0 encoder (process "A") has a specific number of input G.711 symbols consistent with attribute A5 (40, 80, 160, 240 or 320 octets). The second constraint is that the compression law used to create the G.711 input frame (A-law or Mu-law) must be known, consistent with attribute A9.

Subject to these two constraints, the input G.711 frame is processed

by the G.711.0 encoder ("A") and produces a "self-describing" G.711.0 output frame, consistent with attribute A4. Depending on the source G.711 symbols, the G.711.0 output frame can contain anywhere from 1 to X+1 octets, where X is the number of input G.711 symbols. For virtually every use of G.711.0 resulting from zero-mean acoustic signal capture, we expect compression.

Since the G.711.0 output frame is "self-describing", a G.711.0 decoder (process "B") can losslessly reproduce the original G.711 input frame with only the knowledge of which companding law was used (A-law or Mu-law). The G.711.0 frame, being "self-describing", allows for the G.711.0 decoder ("B") to know precisely how many G.711 symbols to create.

Since G.711.0 was designed with typical G.711 payload lengths as a design constraint (attribute A5), this lossless encoding can be performed only with knowledge of the companding law being used. This information is anticipated to be signaled in SDP and will be described later in this document.

If the original inputs were known to be from a zero-mean acoustic signal coded by G.711, an intelligent G.711.0 encoder could infer the G.711 companding law in use (via G.711 input signal histogram). Likewise, an intelligent G.711.0 decoder producing G.711 from the G.711.0 frames could also infer the encoding law in use. Thus G.711.0 could be designed for use in applications that have limited stream signaling between the G.711 endpoints (i.e., they only know "G.711 at 8k sampling is being used", but nothing more). Such usage is not further described in this document. Additionally, if the original inputs were known to come from zero-mean acoustic signals, an intelligent G.711.0 encoder could tell if the G.711.0 payload had been encrypted - as the symbols would not have the distribution expected in either companding law and would appear random. Such determination is also not further discussed in this document.

It is easily seen that this process is 1:1 and that G.711.0 based lossless compression can be employed multiple times, as the original G.711 input symbols are always reproduced with 100% fidelity.

G.711.0 frames containing more source G.711 symbols compress more as a general rule, but there are exceptions. For example, an intelligent G.711.0 encoder may choose to encode 20 ms of G.711 as two individual 10 ms G.711.0 frames if a higher overall compression will result (this might occur if the first 10 ms was "silence" and two, 10 ms G.711.0 frames contained fewer octets than one 20 ms G.711.0 frame). For this reason, we will explicitly allow multiple G.711.0 encoded frames in the G.711.0 RTP payload in Section 4.2.2 (Section 3.3) below even though the usual case is anticipated to be

only one G.711.0 frame per RTP payload.

4. RTP Header and Payload

In this section we describe the precise format for G.711.0 frames carried via RTP. We begin with RTP header description relative to G.711, then provide two G.711.0 payload examples.

4.1. G.711.0 RTP Header

Relative to G.711 RTP headers, the utilization of G.711.0 does not create any special requirements with respect to the contents of the RTP packet header. The only significant difference is that the payload type (PT) RTP header field will have a value corresponding to the dynamic payload type assigned to the flow (whereas G.711 PCMU has a static PT = 0 and G.711 PCMA has a static PT = 8 [RFC3551]).

Voice Activity Detection (VAD) SHOULD NOT be used when G.711.0 is negotiated because G.711.0 obtains high compression during "VAD silence intervals" and one of the advantages of G.711.0 over G.711 with VAD is the lack of any VAD-inducing artifacts in the received signal. However, if VAD is employed, the Marker bit (M) MUST be set in the first packet of a talkspurt, that is, the first packet after a silence period which packets have not been transmitted contiguously as per rules specified in [RFC3550] for G.711 payloads.

With this introduction, the RTP packet header fields are defined as follows:

V - As per [RFC3550]

P - As per [RFC3550]

X - As per [RFC3550]

CC - As per [RFC3550]

M - As per [RFC3550]

PT- Dynamic PT assigned, consistent with MIME allocation for G711.0 defined in Media Type Definition (Section 5.1 (Section 5.1)).

SN - As per [RFC3550]

timestamp - As per [RFC3550]

SSRC - As per [RFC3550]

CSRC - As per [RFC3550]

Where V (version bits), P (padding bit), X (extension bit), CC (CSRC count), M (marker bit), PT (payload type), SN (sequence number), timestamp, SSRC (synchronizing source) and CSRC (contributing sources) are as defined in [RFC3550] and as typically used with G.711. PT (payload type) is as defined in [RFC3550].

4.2. G.711.0 RTP Payload

In this section we provide two examples for carrying G.711.0 frames in RTP payloads. The first example is used when it is desired to carry only one G.711.0 frame is in the payload. This example is a subset of the second and shown separately for clarity.

4.2.1. Single G.711.0 Frame per RTP Payload Example

This example depicts a single G.711.0 frame in the RTP payload. This is expected to be the dominant RTP payload case for G.711.0, as the G.711.0 encoding process supports the SDP packet times (ptime and maxptime, see [RFC4566]) commonly used when G.711 is transported in RTP. Additionally, as mentioned previously, larger G.711.0 frames generally compress more effectively than a multiplicity of smaller G.711.0 frames.

The following Figure illustrates the single G.711.0 frame per RTP payload case.

Single G.711.0 Frame in RTP Payload Case

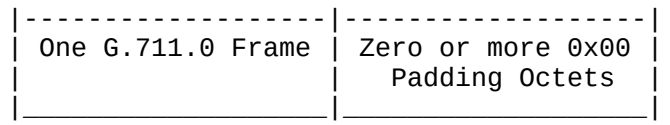


Figure 2

Encoding Process: A single G.711.0 frame is inserted into the RTP payload. The amount of time represented by the G.711 symbols compressed in the G.711.0 frame MUST correspond to the ptime signaled for applications using SDP. Although generally not desired, padding desired in the RTP payload after the G.711.0 frame MAY be created by placing one or more 0x00 octets after the G.711.0 frame. Such padding may be desired based on security considerations (see Section 11 (Section 11)).

Decoding Process: Passing the entire RTP payload to the G.711.0 decoder is sufficient for the G.711.0 decoder to create the source G.711 symbols. Any padding inserted after the G.711.0 frame (i.e., the 0x00 octets) present in the RTP payload is silently ignored by the G.711.0 decoding process. The decoding process is fully described in Section Section 4.2.3 below.

4.2.2. Multiple G.711.0 Frames per RTP Payload Example

This example depicts the case where multiple G.711.0 frames are desired in the RTP payload.

As described in Section 3.3 (Section 3.3), an "intelligent G.711.0 encoder" can decide to encode, let's say, 20 ms of G.711 symbols as two, 10 ms G.711.0 frames because a greater compression is attained for that particular 20 ms segment. Thus such "smart encoding" of such inputs is accommodated by the ability to have multiple G.711.0 frames in the RTP payload.

Note that since each G.711.0 frame is self-describing (see Attribute A4 in Section 3.2 (Section 3.2)), the individual G.711.0 frames in the RTP payload need not represent the same duration of time (i.e., a 5 ms G.711.0 frame could be followed by a 20 ms G.711.0 frame). Owing to this, the amount of time represented in the RTP payload MAY be any integer multiple of 5 ms (as 5 ms is the smallest interval of time that can be represented in a G.711.0 frame).

The following Figure illustrates the multiple G.711.0 frame per RTP payload case where the number of G.711.0 frames placed in the RTP payload is N.

Multiple G.711.0 Frames in RTP Payload Case

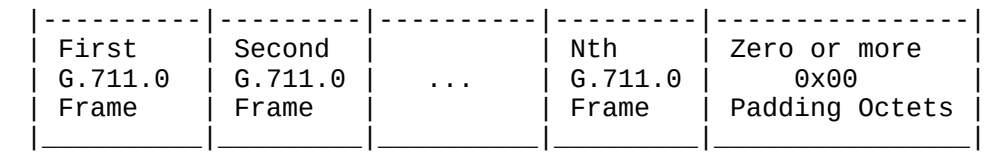


Figure 3

We note here that the individual G.711.0 frames can be, and generally are, of different lengths. The decoding process in the following section is used to determine the frame boundaries.

Encoding Process: One or more G.711.0 frames are placed in the RTP

payload simply by concatenating the G.711.0 frames together. The amount of time represented by the G.711 symbols compressed in all the G.711.0 frames in the RTP payload MUST correspond to the ptime signaled for applications using SDP. Although not generally desired, padding desired in the RTP payload SHOULD be placed after the last G.711.0 frame in the payload and MAY be created by placing one or more 0x00 octets after the last G.711.0 frame. Such padding may be desired based on security considerations (see Section 11 (Section 11)).

Decoding Process: As G.711.0 frames can be of varying length, the payload decoding process described in the following section is used to determine where the individual G.711.0 frame boundaries are.

4.2.3. G.711.0 RTP Payload Decoding Process

This decoding process is a standard part of G.711.0 bit stream decoding and is implemented in the ITU-T Rec. G.711.0 reference code.

Before describing the decoding, we note here that the largest possible G.711.0 frame is created whenever the largest number of G.711 symbols is encoded (320 from Section 3.2 (Section 3.2), property A5) and these 320 symbols are "uncompressible" by the G.711.0 encoder. In this case (via property A6 in Section 3.2 (Section 3.2)) the G.711.0 output frame will be 321 octets long. We also note that the value 0x00 chosen for the optional padding cannot be the first octet of a valid G.711.0 frame (see [G.711.0]). We also note that whenever more than one G.711.0 frame is contained in the RTP payload, the decoding of the individual G.711.0 frames will occur multiple times.

For the decoding heuristic below, let N be the number of octets in the RTP payload (i.e., excluding any RTP padding, but including any RTP payload padding), let P equal the number of RTP payload octets processed by the G.711.0 decoding process, let J represent the present G.711.0 frame being decoded, let K be the number of G.711 symbols in the output buffer, let Q be the number of octets contained in the present G.711.0 frame being processed and let "!=" represent not equal to. The keyword "STOP" is used below to indicate the end of the processing of G.711.0 frames in the RTP payload. The heuristic below assumes an output buffer for the decoded G.711 source symbols of length sufficient to accommodate the expected number of G.711 symbols and an input buffer of length 321 octets.

G.711.0 RTP Decoding Heuristic:

H1 Initialize the number of processed octets to zero (P = 0). Initialize the G.711.0 frame counter J to zero (J = 0). Initialize the counter for how many G.711 symbols are in the output buffer to zero (K = 0).

H2 Read min{320+1, (N-P)} octets into the internal buffer from the (P+1) octet of the RTP payload. We note at this point, N-P octets have yet to be processed and that 320+1 octets is the largest possible G.711.0 frame.

H3 Analyze the first octet in the internal buffer. If this octet is other than 0x00 (a padding octet), increment the G.711.0 frame counter (set J = J + 1) and continue to Step H4. Otherwise increment the processed packets counter by one (set P = P + 1). If the result of this increment results in P = N then STOP (as all RTP Payload octet have been processed), otherwise go to Step H2.

H4 Pass the internal buffer to the G.711.0 decoder. The G.711.0 decoder will read the first octet (called the "prefix code" octet in [G.711.0]) to determine the number of source G.711 samples M are contained in this G.711.0 frame.

H5 The G.711.0 decoder will produce exactly M G.711 source symbols. If J = 1, these M symbols will be the first in the output buffer and are placed at the beginning of the output buffer. If J != 1, these M symbols are concatenated with the prior symbols in the output buffer. Set K = K + M (as there are now this many G.711 source symbols in the output buffer).

H6 In process H5, the G.711.0 decoder will have consumed some number of packets, Q, in the internal buffer to produce the M G.711 symbols. Increment the number of processed octets by this quantity; that is set P = P + Q.

H7 If P < N there are more octets in the RTP payload left to process, go to Step H2. If P >= N, STOP (as all RTP payload octets have been processed).

At this point, the output buffer will contain precisely K G.711 source symbols which should correspond to the ptime signaled if SDP was used and the encoding process was without error. We also note, as an aside, that the heuristic above (and the ITU-T G.711.0 reference code) accommodates padding octets (0x00) placed anywhere in the RTP payload.

If the decoder is at a playout endpoint location, this G.711 buffer SHOULD be used in the same manner as a received G.711 payload would

have been used (passed to a playout buffer, to a PLC implementation. etc.). If not, then the instructions in Section 6 (Section 6) (G.711.0 "In The Middle") should be followed.

5. Payload Format Parameters

This section defines the parameters that may be used to configure optional features in the G.711.0 RTP transmission.

The parameters defined here as a part of the media subtype registration for the G.711.0 codec. Mapping of the parameters into Session Description Protocol (SDP) RFC 4566 [RFC4566] is also provided for those applications that use SDP.

5.1. Media Type Registration

Type name: audio

Subtype name: G7110

Required Parameters:

rate: The RTP timestamp clock rate, which is equal to the sampling rate. The typical rate is 8000, but other rates may be specified.

complaw: Indicates the companding law (A-law or mu-law) employed. The case-insensitive values are "a" or "mu".

Optional parameters:

channels: how many audio streams are represented in the G.711.0 payload - defaults to 1; stereo would be 2, etc.

[Editor's Note: We are considering specifying more than one channel for multiplexing or conference switching applications. One option for the delimiting the channels with the RTP payload would be to use one of the few not allowed G.711.0 frame prefix codes to delineate the channel data and appropriate modification to the RTP decoding heuristic in Section 4.2.3 (Section 4.2.3). Note that the channel order is already well specified in RFC 3551 [RFC3551].]

ptime, maxptime: see RFC 4566 [RFC4566]

Encoding considerations:

This media type is framed binary data (see Section 4.8 in RFC 4288 [RFC4288]) compressed as per ITU-T Rec. G.711.0.

Security considerations:

This media type does not carry active content. It does transfer compressed data. See Section 4 of RFC 4856 [RFC4856].

Interoperability considerations: none

Published specification:

ITU-T Rec. G.711.0 and RFC QQQQ.

[RFC Editor: please replace QQQQ with a reference to this RFC]

Applications that use this media type:

Audio and video streaming and conferencing tools.

Additional information: none

Person & email address to contact for further information:

Michael Ramalho <mramalho@cisco.com> or <mar42@cornell.edu>

Intended usage: COMMON

Restrictions on usage:

This media type depends on RTP framing, and hence is only defined for transfer via RTP [RFC3550]. Transport within other framing protocols is not defined at this time.

Author: Michael Ramalho

Change controller:

IETF Audio/Video Transport working group delegated from the IESG.

5.2. Mapping to SDP Parameters

The information carried in the media type specification has a specific mapping to fields in the Session Description Protocol (SDP), which is commonly used to describe RTP sessions. When SDP is used to specify sessions employing G.711.0, the mapping is as follows:

- The media type ("audio") goes in SDP "m=" as the media name.
- The media subtype ("G7110") goes in SDP "a=rtpmap" as the encoding name.

- The required parameter "rate" also goes in "a=rtpmap" as the clock rate.
- The parameters "ptime" and "maxptime" go in the SDP "a=ptime" and "a=maxptime" attributes, respectively.
- Remaining parameters go in the SDP "a=fmtp" attribute by copying them directly from the media type string as a semicolon-separated list of parameter=value pairs.

5.3. Offer/Answer Considerations

There are no special considerations when using the SDP offer/answer RFC 3264 [RFC3264] as all the SDP parameters are declaritive.

[EDITOR'S NOTE: This may change in a future revision if the channel parameter is negotiated. This could happen when the offer desires channels = N and the answerer can only support a number less than N. We would then insert an offer/answer example in a future revision of this draft.]

5.4. SDP Example

The following examples illustrate how to signal G.711.0 via SDP:

```
m=audio RTP/AVP 98

a=rtpmap:  98 G7110/8000

a=prime:  20

a=fmtp:98 complaw = mu
```

In the above example, the dynamic payload type 98 is mapped to G.711.0 via the "a=rtpmap" parameter. The packetization time (ptime) is indicated to be 20 ms of audio. The mandatory "complaw" is on the "a=fmtp" parameter line.

6. G.711.0 "In The Middle"

When G.711 has been negotiated end-to-end, G.711.0 compression can be employed by entities in the middle of the end-to-end G.711 flow as a compression mechanism. When used in this manner, it can be used with or without compression of the RTP header. In either case, the G.711 payloads AND the corresponding G.711 RTP headers MUST appear to the end systems as having been transported transparently.

6.1. G.711.0 "In The Middle" - No RTP Header Compression

This figure below illustrates how the compression could be accomplished without the RTP header compression.

G.711.0 Compression "In The Middle" - No RTP Header Compression Case

Figure 4

Figure 4 depicts G.711.0 compression in Box C and G.711.0 decompression back to G.711 in Box E. It depicts the case where only compression of the G.711 payload is desired; the RTP header (including any extensions) is simply copied with the exception that the G.711 payload type (the usual static PT of 0 or 8 is shown) is

replaced by a PT negotiated between Box C and Box E (depicted as PT = Q).

Note that if there are no hops between Box C and E (i.e, no Box D), this is equivalent to compression over a single link. The compression segment represented by Box C, Box E and Box F is labeled a "G.711.0 compression segment" in the above figure.

Since G.711.0 is a lossless and stateless compression, there can be multiple such segments between the sending and receiving endpoints (not shown).

The G.711.0 compression and decompression (Box C and E) may reside in a variety of network elements such as, but not limited to, switches, routers, middleboxes (NATs/PATs, firewalls, session border controllers, transport acceleration devices) and is purposely not specified here.

There may be many "potential G.711.0 compression/decompression points" along the end-to-end G.711 flow; the mechanisms by which certain entities determine that they should perform G.711.0-based compression and decompression are outside the scope of this document.

The method by which G.711.0 compression segment endpoints negotiate which RTP payload type (Q shown above) is to be used is outside the scope of this document, although the SDP elements described herein MAY be used.

Firewalls, NATs, SBCs, etc. that may exist in the path of the G.711.0 packets (Box D) and who may drop packets of unexpected payload types may need additional configuration and/or intelligence to let the compressed G.711.0 packets through. Mechanisms to do this are also outside the scope of this document.

[EDITOR'S NOTE: Because many boxes of this type inspect signaling to determine which RTP packets are allowed to progress, we are considering a hint to be place in the G.711 (not G.711.0) SDP that says, in essence, if you do compress this packet, please use PT = Q. In this way middleboxes may also know to pass PT = Q packets as well as PT = [0 | 8]. Such a G.711 SDP entry may look like:

Example of G.711 SDP with hint for G.711.0 PT

```
m=audio RTP/AVP 0
a=rtpmap: 0 PCMU
a=fmtp:0 G7110 = Q   <<< the G.711 SDP hint
```

Here, the last line provides the hint that this G.711 Mu-law flow may be represented anywhere on the end-to-end G.711 flow as PT = 0 (uncompressed) or PT = Q (compressed) RTP packets. Then candidate compression entities MAY choose to honor that hint in their respective dynamic payload type negotiation for G.711.0. Of course, this hint would require the registration of an optional parameter for the PCMU and PCMA media registrations. We hereby solicit feedback on this concept.]

6.2. G.711.0 "In The Middle" - With RTP Header Compression

When it is desired to compress the G.711 header as well, the G.711.0 compression segment endpoints of the previous section have further functionality by which they also compress the headers. However, this functionality is outside of the scope of this document.

We simply note here that if such functionality is employed, that the G.711 payloads AND the corresponding G.711 RTP headers MUST appear to the end systems as having been transported transparently. Such RTP header compression functionality SHOULD be stateless so as to minimize error propagation for lost packets to be consistent with G.711.0 design goal attribute A3.

6.3. G.711.0 "In The Middle" - Implications for Voice Quality and Added Delay

As described in the sections immediately previous, G.711.0 can be employed multiple times (e.g., on multiple, individual hops or series of hops) of a given G.711 flow. Owing to the stateless design of G.711.0 and any RTP header compression scheme recommended above, there is no error propagation owing to loss of a G.711.0 packet. Thus the impact of an individual packet drop of a G.711.0 RTP packet is identical to the impact of the corresponding equivalent G.711 RTP packet.

Stated another way, multiple "lossless transcodes" from/to G.711.0/ G.711 do not negatively affect voice quality as may occur with lossy transcodes to/from dissimilar codecs.

G.711.0 provides over 50% reduction in average payload size with exactly 0.0000% quality loss relative to G.711 [ICASSP].

For completeness, we note that a G.711.0 encode/decode average complexity is 1 WMOPS (see Section 3.2 (Section 3.2), attribute A8). Given such low complexity, less than 1 ms of compression/ decompression delay per each G.711.0 compression segment is expected in most implementations.

6.4. G.711.0 "In The Middle" - Multiplexing Multiple G.711 Flows

It may also be desired to multiplex the payloads of many G.711 channels into one "G.711.0 payload". As with the previous section, functionality for the mechanism used is outside the scope of this document.

[EDITOR'S NOTE: If we allow the channels parameter to be >1, then a mechanism for multiplexing the G.711.0 payloads into one RTP payload would be specified in this document. However, we don't know of a standardized use case for multiplexing a multiplicity of media streams representing multiple endpoint flows in one RTP media flow. This would be advantageous if, for example, multiple G.711 flows traverse two known endpoints and where the endpoints could "add and delete" G.711 flows "on-the-fly" (perhaps by association of the RTP CSRCs) and then fake out the endpoints with simulated RTP headers in a manner similar to WAN optimization product does today for TCP. Comments are welcome on this point.]

If such RTP multiplexing compression functionality is designed, the RTP header compression used with the G.711.0 multiplexing SHOULD be stateless so as to minimize error propagation for lost packets to be consistent with G.711.0 design goal attribute A3.

7. G.711.0 Storage Mode

There are two storage modes defined for the G.711.0; one for short recordings and one for long recordings.

For short recordings, the recommendation will be similar to many other IETF codecs (e.g., iLBC, EVRC-NW) and will fundamentally be a concatenation of received G.711.0 frames and erasure frames.

However, since G.711.0 has variable length frames, it is prudent for long recordings (over many minutes long) to use an indexing methodology that enables playout far into the recording without decoding all the G.711.0 frames since the recording began.

For either storage mode, a "G.711.0 erasure frame" is defined in the following section because ITU-T Rec. G.711.0 [G.711.0] does not define one.

7.1. G.711.0 Erasure Frame

A G.711.0 erasure frame is a representation of the amount of time in lost or not received G.711.0 frames. Lost frames are typically determined by unexpected discontinuities observed in the RTP send timestamps at the receiver.

A G.711.0 compressed frame can represent 40, 80, 160, 240 or 320 G.711 symbols (Attribute A5 of Section 3.2 (Section 3.2)). As G.711.0 also does not use the value of 0x01 for the first octet of any valid G.711.0 frame, it can be used as an identifier for an erasure frame. Thus an erasure frame is the two octet quantity shown below.

The Two Octet G.711.0 Erasure Frame Definition

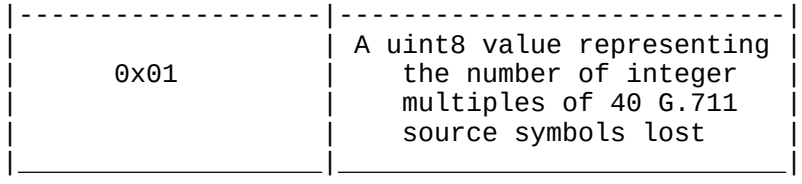


Figure 5

The value of the second octet is representative of how many G.711 source samples were lost. For example, a value of 4 implies 160 samples were lost independently of how many G.711.0 frames those 160

samples were represented by. Erasure frames themselves may be concatenated if it is desired to create one per G.711.0 frame, or one per G.711.0 RTP payload, etc. This erasure frame format can therefore represent 255*40 = 10,200 missing G.711 symbols (i.e., about 1.2 seconds of G.711 at 8000 samples per second).

The erasure frame has been designed to represent precisely what was observed at the receiver for true archival purposes (including potentially law enforcement). Therefore, the use of erasure frames for missing G.711.0 frames is RECOMMENDED for archival purposes.

If such an archival needs are not required, a storage implementation MAY provide packet loss concealment (PLC) or simply "silence" (G.711.0 analog zero representation) for the missing source G.711 data and choose not to use an erasure frame. Such PLC/silence insertion MUST represent the precise amount of time represented by the missing data to maintain synchronization with the original media.

7.2. G.711.0 Storage Mode - Short Recordings

This short recording storage format is used for storing G.711.0 encoded frames. The file begins with a magic number to identify the coder that is used. The magic number for G.711.0 A-law corresponds to the ASCII character string "#!G7110A\n", i.e., "0x23 0x21 0x47 0x37 0x31 0x31 0x30 0x41 0x0A". Likewise, the magic number for G.711.0 MU-law corresponds to the ASCII character string "#!G7110M\n", i.e., "0x23 0x21 0x47 0x37 0x31 0x31 0x4E 0x4D 0x0A". The codec data frames including any necessary erasure frames are stored in consecutive order concatenated together as shown in Section 4.2.2 (Section 4.2.2).

To decode the individual G.711.0 frames, a heuristic similar to the one presented in Section 4.2.2 (Section 4.2.2) modified appropriately to recognize and process erasure frames as legitimate G.711.0 frames in the recording format.

7.3. G.711.0 Storage Mode - Long Recordings

[EDITOR'S NOTE: The long recordings storage mode format is TBD. This storage mode will likely have indexing capability and metadata capabilities.]

8. Acknowledgements

There have been many people contributing to G.711.0 in the course of its development. The people listed here deserve special mention: Takehiro Moriya, Claude Lamblin, Herve Taddei, Simao Campos, Yusuke Hiwasaki, Jacek Stachurski, Lorin Netsch, Paul Coverdale, Patrick Luthi, Paul Barrett, Jari Hagqvist, Pengjun (Jeff) Huang, and Jon Gibbs.

9. Contributors

The authors thank everyone who have comtributed to this document. The people listed here deserve special mention: Ali Begen and Roni Even.

10. IANA Considerations

One media type (audio/G7110) has been defined and requires IANA registration in the media types registry. See Section 5.1 (Section 5.1)

11. Security Considerations

RTP packets using the payload format defined in this specification are subject to the security considerations discussed in the RTP specification [RFC3550], and in any appropriate RTP profile (for example RFC 3551 [RFC3551] or [RFC4585]. This implies that confidentiality of the media streams is achieved by encryption; for example, through the application of SRTP [RFC3711]. Because the data compression used with this payload format is applied end-to-end, any encryption needs to be performed after compression.

Note that the appropriate mechanism to ensure confidentiality and integrity of RTP packets and their payloads is very dependent on the application and on the transport and signaling protocols employed. Thus, although SRTP is given as an example above, other possible choices exist.

Note that end-to-end security with either authentication, integrity or confidentiality protection will prevent a network element not within the security context from performing media-aware operations other than discarding complete packets. To allow any (media-aware) intermediate network element to perform its operations, it is required to be a trusted entity which is included in the security context establishment.

G.711.0 has no known denial-of-service attacks due to decoding, as data posing as a desired G711.0 payload will be decoded into something (as per the decoding algorithm) with a finite amount of computation. This is due to the decompression algorithm having a finite worst-case processing path (no infinite computational loops are possible).

G.711.0 is a variable bit rate (VBR) audio codec. There have been recent concerns with VBR speech codecs where a passive observer can identify phrases from a standard speech corpus by means of the lengths produced by the encoder even when the payload is encrypted [IEEE]. In this paper, it was determined that some code excited linear prediction (CELP) codecs would produce discrete packet lengths for some phonemes. And furthermore with the use of appropriately designed Hidden Markov Models (HMMs) that such a system could predict phrases with unexpected accuracy. One CELP codec studied, SPEEX, had the property that it produced 21 different packet lengths in its wideband mode and that these packet lengths probabilistically mapped to phonemes that a HMM system could be trained on. In this paper it was determined that a mitigation technique would be to pad the output of the encoder with random padding lengths to the effect: 1) that more discrete payload sizes would result, and 2) that the probabilistic mapping to phonemes would become less clear. As G.711

is not a speech model based codec, neither is G.711.0. A G.711.0 encoding, during talking periods, produces frames of varying frame lengths which are not likely to have a strong mapping to phonemes. Thus G.711.0 is not expected to have this same vulnerability. It should be noted that "silence" (only one value of G.711 in the entire G.711 input frame)" or "near silence" (only a few G.711 values) is easily detectable as G.711.0 frame lengths or one or a few octets. If one desires to mitigate for silence/non-silence detection, statistically variable padding should be added to G.711.0 frames that resulted in very small G.711.0 frames (less than about 20% of the symbols of the corresponding G.711 input frame). Methods of introducing padding in the G.711.0 payloads have been provided in the G.711.0 RTP payload definitions in Sections 4.2.1 (Section 4.2.1) and 4.2.2 (Section 4.2.2).

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.

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

[RFC4288] Freed, N. and J. Klensin, "Media Type Specifications and Registration Procedures", BCP 13, RFC 4288, December 2005.

[RFC4855] Casner, S., "Media Type Registration of RTP Payload Formats", RFC 4855, February 2007.

[RFC4856] Casner, S., "Media Type Registration of Payload Formats in the RTP Profile for Audio and Video Conferences", RFC 4856, February 2007.

[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.

[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.

[RFC3711] Baugher, M., McGrew, D., Naslund, M., Carrara, E., and K. Norrman, "The Secure Real-time Transport Protocol (SRTP)", RFC 3711, March 2004.

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

[G.711.0] ITU-T G.711.0, "Recommendation ITU-T G.711.0 - Lossless Compression of G.711 Pulse Code Modulation", September 2009.

[G.711] ITU-T G.711.0, "Recommendation ITU-T G.711 - Pulse Code Modulation (PCM) of Voice Frequencies", November 1988.

[G.711-A1] ITU-T G.711 Amendment 1, "Recommendation ITU-T G.711 Amendment 1 - Amendment 1: New Annex A on Lossless Encoding of PCM Frames", September 2009.

12.2. Informative References

[RFC2629] Rose, M., "Writing I-Ds and RFCs using XML", RFC 2629, June 1999.

[G.729] ITU-T G.729, "Recommendation ITU-T G.729 - Coding of speech at 8 kbit/s using conjugate-structure algebraic-code-excited linear prediction (CS-ACELP)", January 2007.

[G.722] ITU-T G.722, "Recommendation ITU-T G.722 - 7 kHz audio-coding within 64 kbit/s", November 1988.

[ICASSP] N. Harada, Y. Yamamoto, T. Moriya, Y. Hiwasaki, M. A. Ramalho, L. Netsch, Y. Stachurski, Miao Lei, H. Taddei, and Q. Fengyan, "Emerging ITU-T Standard G.711.0 - Lossless Compression of G.711 Pulse Code Modulation, International Conference on Acoustics Speech and Signal Processing (ICASSP), 2010, ISBN 978-1-4244-4244-4295-9", March 2010.

[IEEE] C.V. Wright, L. Ballard, S.E. Coull, F. Monrose, and G.M. Masson, "Spot Me if You Can: Uncovering Spoken Phrases in Encrypted VoIP Conversations, IEEE Symposium on Security and Privacy, 2008, ISBN: 978-0-7695-3168-7", May 2008.

Authors' Addresses

Michael A. Ramalho (editor)
Cisco Systems, Inc.
4563 Tuscana Drive
Sarasota, FL 34241
USA

Phone: +1 919 476 2038
Email: mramalho@cisco.com

Paul E. Jones
Cisco Systems, Inc.
7025 Kit Creek Rd.
Research Triangle Park, NC 27709
USA

Phone: +1 919 476 2048
Email: paulej@packetizer.com

Noboru Harada
NTT Communications Science Labs
3-1 Morinosato-Wakamiya
Atsugi, Kanagawa 243-0198
JAPAN

Email: harada.noboru@lab.ntt.co.jp

Muthu Arul Mozhi Perumal
Cisco Systems, Inc.
Cessna Business Park
Sarjapur-Marathahalli Outer Ring Road
Bangalore, Karnataka 560103
India

Phone: +91 9449288768
Email: mpermumal@cisco.com

Miao Lei
Huawei Technologies Co. Ltd
No. 3 Xinxi Rd.
ShangDi, HaiDian District
Beijing, Beijing 100085
Beijing

Phone: +86 1082882759
Email: lei.miao@huawei.com

Audio/Video Transport Extensions
Internet-Draft
Intended status: Standards Track
Expires: September 12, 2011

A. Williams
Audinate

March 11, 2011

IEEE 1588/802.1AS Synchronisation for RTP Streams
draft-williams-avtext-avbsync-01

Abstract

Specification of an RTP header extension for carrying in-band synchronization metadata provided by the IEEE1588/802.1AS Precision Time Protocols.

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 September 12, 2011.

Table of Contents

1. Introduction

Synchronisation between RTP flows and between devices rendering RTP flows is currently facilitated by means of NTP format timestamps taken with respect to a shared reference clock. In many applications (e.g. professional, commercial and automotive AV), the NTP clock synchronisation protocol does not meet the necessary time alignment and synchronisation speed requirements.

Like NTP, the IEEE1588 family of clock synchronisation protocols provide a shared reference clock in an network - typically a LAN. IEEE1588 provides sub-microsecond synchronisation between devices on a LAN and typically locks within seconds at startup rather than minutes. With support from Ethernet switches, IEEE1588 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.

When using IEEE1588 clock synchronisation, networked AV systems can achieve sub 1 microsecond time alignment accuracy when rendering AV signals and can support latencies less than 1ms through a gigabit LAN.

Three flavours of IEEE1588 are in use today:

- IEEE 1588-2002 [3]: the original "Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems". This is often called IEEE1588v1 or PTPv1.

- IEEE 1588-2008 [4]: 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 often called IEEE1588v2 or PTPv2.

- IEEE 802.1AS [5]: "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.

By using an IEEE 1588 derived reference clock, synchronisation of RTP streams and devices in LANs can be considerably improved.

2. Timestamp formats

A global IEEE 1588/802.1AS timestamp is 80 bits in total, divided into two parts:

AS_sec: 48 bits seconds since epoch

AS_nsec: 32 bits nanoseconds

A shorter 32 bit timestamp is defined for use in streaming media protocols in the following way:

as_timestamp = (AS_sec * 10^9 + AS_nsec) modulo 2^32

The shorter as_timestamp field covers just over 4 seconds of time.

3. Header Extension

Figure 1 shows the fields of the AVB sync header extension. It uses the standard RTP header extension mechanism defined in RFC 5285 [2].

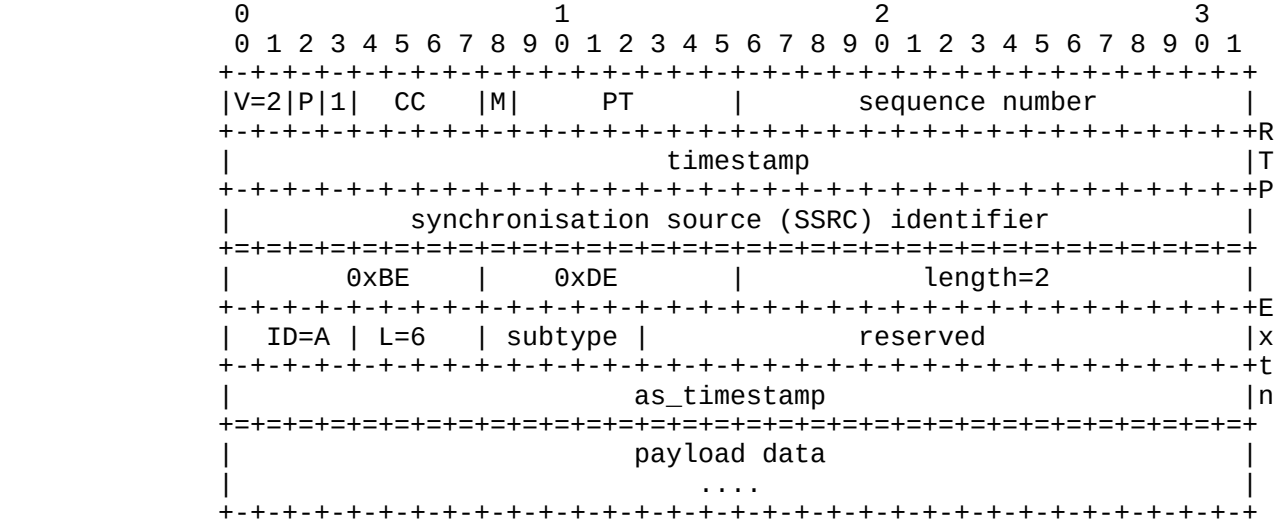


Figure 1: IEEE 1588/802.1AS Synchronisation Header Extension

The fields are defined as follows:

subtype: RTCP AVB packet subtype field, see Section 4.

as_timestamp: a 32 bit IEEE 1588/802.1AS timestamp as defined in Section 2.

reserved: as this specification evolves, additional fields are expected to be included in this header.

The as_timestamp MUST correspond to the same instant as the RTP timestamp in the packet's header, and MUST be derived from the same clock used to generate the as_timestamps in the RTCP AVB packets. Provided that it has knowledge of the SSRC to CNAME mapping, either from prior receipt of an RTCP CNAME packet or via out of band signalling [RFC5576], the receiver can use the information provided as input to the synchronization algorithm, in exactly the same way as if an additional RTCP AVB packet had been received for the flow.

4. IEEE 1733 / RTCP AVB Packet

IEEE 1733 [6] defines the "AVB RTCP packet" type reproduced in Figure 2. RTCP AVB packets contain a mapping between RTP timestamp and an 802.1AS timestamp as well as additional clock and QoS information.

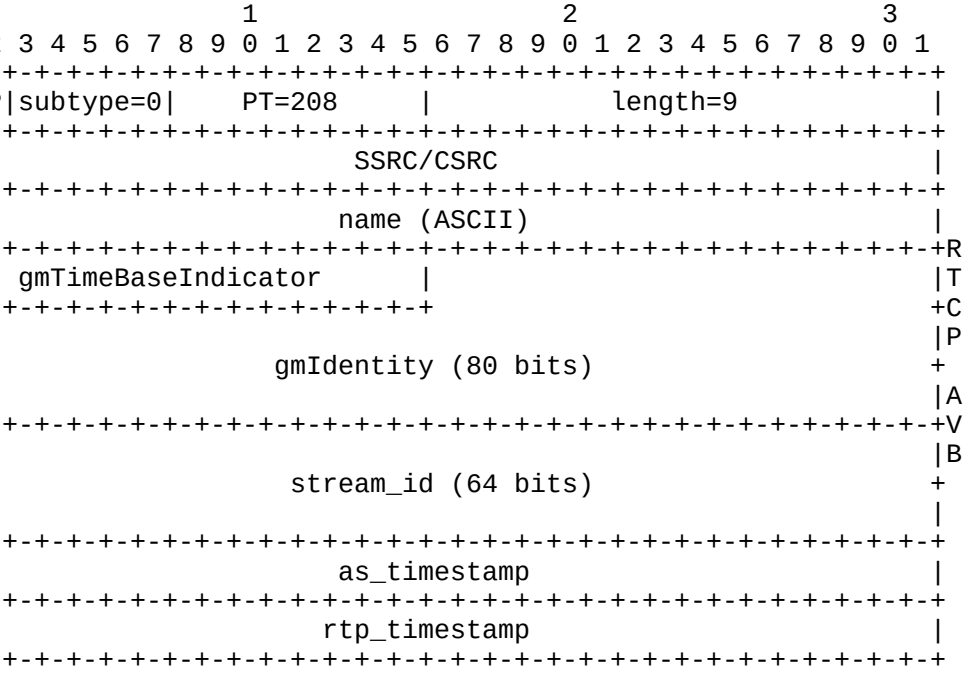


Figure 2: IEEE 1733/RTCP AVB packet format

A brief description of the major fields follows:

gmIdentity an 80 bit field uniquely identifying the current 802.1AS grand master clock used by the source to generate as_timestamps for this flow

stream_id a 64 bit number identifying the 802.1Qat [7] stream associated with this RTP flow

as_timestamp the 32 bit 802.1AS timestamp (Section 2) associated with the RTP timestamp carried in this packet

rtp_timestamp the RTP timestamp of a media packet

Please consult the IEEE 1733 specification [6] for more details.

5. IANA Considerations

TBD: A URN will be required to signal the presence of this header extension, such as:

urn:ietf:params:rtp-hdrext:avb-sync

6. Acknowledgements

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

7.2. Informative References

[3] Institute of Electrical and Electronics Engineers, "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>.

[4] Institute of Electrical and Electronics Engineers, "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>.

[5] Institute of Electrical and Electronics Engineers, "IEEE Standard for Local and Metropolitan Area Networks - IEEE Draft Standard for Local and Metropolitan Area Networks - Timing and Synchronization for Time-Sensitive Applications in Bridged Local Area Networks", IEEE Std 802.1AS-2011, 2011, <http://standards.ieee.org/findstds/standard/802.1AS-2011.html>.

[6] Institute of Electrical and Electronics Engineers, "IEEE P1733/ D7.0 Draft Standard for Layer 3 Transport Protocol for Time Sensitive Applications in Local Area Networks", IEEE Draft Std 1733/D7.0, February 2011, <http://grouper.ieee.org/groups/1733/>.

[7] Institute of Electrical and Electronics Engineers, "IEEE Standard for Local and Metropolitan Area Networks---Virtual Bridged Local Area Networks Amendment 14: Stream Reservation Protocol (SRP)", IEEE Std 802.1Qat-2010 (Revision of IEEE Std 802.1Q-2005), 2010, <http://standards.ieee.org/about/get/>.

Appendix A. An Appendix

Author’s Address

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

AVTEXT Working Group
Internet-Draft
Intended status: Informational
Expires: August 30, 2011

J. Xia
Huawei

February 26, 2011

Content Splicing for RTP Sessions
draft-xia-avtext-splicing-for-rtp-00

Abstract

This memo outlines RTP splicing. Splicing is a process that replaces the content of the main multimedia stream with other multimedia content, and delivers the substitutive multimedia content to receiver for a period of time. This memo discusses the requirements of splicing process on RTP layer, thus provides guideline for how a RTP Translator works to satisfy these requirements.

Status of this Memo

This Internet-Draft is submitted to IETF 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 August 30, 2011.

Copyright Notice

Table of Contents

1. Introduction

This document outlines how splicing can be used for RTP sessions. Splicing is a process that replaces the content of the main multimedia stream with other multimedia content, and delivers the substitutive multimedia content to receiver for a period of time. The substitutive content can be provided for example via another RTP stream or local media file storage.

One representative use case for splicing is targeted advertisements insertion, which allows operators to replace a national advertising slot with its own regional advertising content prior to providing to receiver. So far [SCTE30] and [SCTE35] have standardized MPEG2-TS splicing running over cable, but to date there is no guidelines for how to use the RTP Translator functionality defined in [RFC3550] to implement an RTP Splicer.

In this document, we describe the basic requirements of RTP splicing, then analyze how an RTP Translator can be used to implement RTP splicing to satisfy these requirements from the aspect of feasibility, implementation complexity and backward compatibility.

2. Terminology

The keywords "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].

Current RTP Stream

The RTP stream that the RTP receiver is currently receiving. The content of current RTP stream can be either main content or substitutive content.

Main RTP Stream

The RTP stream that the Splicer is receiving. The content of main RTP stream can be replaced by substitutive content for a period of time.

Main Content

The multimedia content that are conveyed in main RTP stream. main content will be replaced by the substitutive content during splicing period.

Substitutive Content

The multimedia content that replaces the content of main RTP stream during splicing period. The substitutive content can for example be contained in an RTP stream from a media source or fetched from local media file storage.

Insertion RTP Stream

A RTP stream that may provide substitutive content. If the substitutive content is provided from insertion RTP stream, the insertion RTP Stream must be terminated on Splicer.

Splicer

An intermediary node that inserts substitutive content into main RTP stream. Splicer sends substitutive content to RTP receiver as the payload of the current RTP stream.

3. RTP Splicing Discussion and Requirements

In this document, we assume an intermediary network element, which is referred to as Splicer, to play the key role to handle RTP splicing. When RTP splicing begins, Splicer replaces the main content in the main RTP stream with substitutive content and conveys the substituted RTP stream to RTP receiver for a period of time, then resumes the main content when RTP splicing finishes. The RTP splicing may happen more than once if substitutive content will be inserted in multiple time segments during the lifetime of the main RTP session.

To realize a seamless splicing on RTP receiver, Splicer must not cause any perceptible media clipping at the splicing joint. Therefore, main media source must reserve specific time slot in the main RTP stream for splicing and notify Splicer this specific time slot information. The substitutive content time length SHOULD be the same as the reserved time slot length. The details about how the specific time slot information is conveyed to Splicer are outside the scope of this memo.

Besides the media clipping avoidance, there are also a set of concrete requirements that must be satisfied on Splicer:

REQ-1:

Splicer MUST operate in either unicast or multicast session environment.

REQ-2:

Splicer MUST guarantee content substitution process is invisible to RTP receiver to prevent the RTP receiver from easily identifying the substitutive content.

The RTP packets whose payloads are replaced by substitutive content are required to keep their RTP header information to be consistent with those of main RTP packets whose payloads are unaltered:

The value of SSRC field in RTP header MUST be same as the value of corresponding field in main content RTP header, while the value of payload type field SHOULD be same as the value of corresponding field in main content RTP header. Note that in some cases, it may be necessary to transcode the substitutive content to ensure the payload type is the same, and that this may be prohibitively expensive, so it might be acceptable to leave the payload untranscoded.

The value of sequence number field in RTP header MUST be contiguous regardless of whether the substitutive content is inserted or not. It should be noted that the number of packets in the substitutive sequence number range may be different from the number of packets in the overridden main sequence number range due to the different characteristics or entropy coding.

REQ-3:

Splicer SHOULD ensure that the main media source can learn the real performance of the path between media source and downstream receiver for adaptation purpose.

REQ-4:

Splicer MUST be backward compatible with basic characteristics of [RFC3550], e.g., SSRC identifier collision resolution and loop detection.

REQ-5:

Splicer MUST have the capability to correctly handle any packet loss events regardless of whether the lost content is main content or substitutive content.

REQ-6:

If substitutive content comes from an insertion RTP stream, Splicer MUST terminate this stream, in such case, Splicer should generate RTCP reports upstream towards the insertion media source using its own SSRC.

4. Using RTP Translator for RTP Splicing

An RTP Translator that acts as media transcoder can replace the payloads of the main RTP packets with the substitutive content, and can assign new sequence numbers to the substituted packets with main RTP packets SSRC identifier intact. Note that the new sequence numbers of substituted RTP packets must seamlessly follow the sequence numbers of the previous main RTP packets. When splicing ends, Translator must switch back to the main RTP stream and output it to RTP receiver until next splicing occurs.

With respect to RTCP flow, Translator acting as a media transcoder will need to interpose itself into the RTCP flow as specified in section 7.2 of [RFC3550]. However, the substitutive content might have different characteristics compared to the main content it replaces. As a result, the translated RTCP Receiver Reports received by the main media source might be somewhat misleading for adaptation purposes, since they relate to other content that the main media source cannot control during the splicing period. Fortunately Translator has the capability to act as quality monitor and generate RTCP reports upstream towards main media source with its own SSRC thus reflecting the real characteristics of the main RTP stream and the reception quality on the Translator. These RTCP reports, as well as the translated RTCP reports sent from the downstream receiver, can provide main media source the general performance of the different segments of the path between main media source and RTP receiver. If the substitutive content is fetched from the insertion RTP stream, Translator acting as RTP receiver should generate RTCP receiver reports upstream towards the insertion media source to reflect the reception quality of the insertion RTP stream on the Translator.

When RTP receiver detects any packet loss during splicing, it may generate RTCP NACK message for packet loss recovery [RFC4585]. If Translator receives any RTCP NACK message from RTP receiver, Translator first needs to determine the sequence number range of lost packets requested by RTP receiver. Since parts of lost packets may contain substitutive content which does not relate to the main RTP stream, the translated RTCP NACK message might be somewhat misleading for packet loss recovery. Thus, when Translator is intercepting RTCP NACK packets, it should only pass those RTCP NACK packets that relate

to the relevant content upstream.

5. Processing Splicing on RTP Translator

Once Translator has learnt when to process splicing from main RTP source, it must get ready for the coming splicing in advance, e.g., fetches the substitutive content either from local media file storage or via insertion RTP stream. If the substitutive content comes from the insertion RTP stream, Translator must act as a RTP receiver and terminate this insertion RTP stream.

First the Translator should guarantee the media encoding of substitutive content to be same as the media encoding of main content. If the substitutive content uses other media encoding, Translator should perform media transcoding procedure for substitutive content.

When splicing begins, Translator replaces the main content with the substitutive content as if the substitutive content were sent from the main media source.

When splicing ends, Translator must switch back to main RTP stream, but since the number of packets in the substitutive sequence number may be different from the number of packets in the overridden main sequence number range, Translator still needs to modify the sequence numbers of subsequent main RTP packets prior to outputting them to RTP receiver even if no media transcoding occurs on main RTP stream. For subsequent current RTP packets, its start sequence number should be determined as being one after the end sequence number of previous substitutive RTP packets.

This is perhaps best explained by a pseudo code example:

```
when (splicing begin) {

the sequence number of the ith substitutive packet = the sequence
number of last main packet + i, until the splicing end;

}

when (splicing end) {

the sequence number of the following kth current packet = the
sequence number of last substitutive packet + k, until the next
splicing begin;

}
```

With regard to RTCP packets, Translator needs to rewrite RTCP Report. The "sender's byte count" field and the "sender's packet count" field in RTCP Sender Report should be changed. Translator then determines the proportion, P, of the number of packets Translator receives from main media source (numRev) to the number of packets Translator sends to RTP receiver (numSend) during a regular RTCP Reporting interval such that P = numRev / numSend. It should be noted that the value of P may be not fixed specially at the splicing joint, where the RTCP Reports may reflects the characteristics of the combination of main RTP packets and adjacent substitutive RTP packets. Once the proportion, P, is counted, the values of packet loss fields and the "extended last sequence number" field in RTCP Sender Report are scaled by dividing each of them by P, and reverse rewriting is made to the values of the corresponding fields in RTCP Receiver Report, i.e., multiplying each them by P. Meanwhile, Translator must generate RTCP Receiver Report upstream towards main media source using its own SSRC, reflecting the reception quality of the main RTP stream on the Translator. If the substitutive content comes from the insertion RTP stream, Translator acting as receiver must generate RTCP Receiver Report upstream towards insertion media source using its own SSRC.

If Translator receives any RTCP NACK message from RTP receiver for packet loss recovery, it first determines if the lost packets contain substitutive content. It is somewhat more complex that the lost packets requested in a single RTCP NACK message contain not only main content but also substitutive content. To address this, Translator must rewrite the translated RTCP NACK message into two separate RTCP NACK messages: one only requests the lost packets that contain main content, and is forwarded using the SSRC of that RTP receiver; and another only requests the lost packets that contain substitutive content, and is sent using the SSRC of the Translator if the substitutive content is delivered via insertion RTP stream, or fetching the lost substitutive content directly from corresponding local media file storage.

6. Implementation considerations

When Translator is used to handle RTP splicing, RTP receiver does not need any RTP/RTCP extension for splicing. As a trade-off, additional overhead could be induced on Translator compared to just translating the main RTP stream, since Translator must coordinate the switch between main content and substitutive content, and generate its own RTCP reports. Even if no new substitutive content will be inserted, the Translator still needs to modify the main RTP packets, and to recalculate the UDP/IP checksum until the main RTP session ends. If Translator serves multiple main RTP streams simultaneously, this may lead to large overhead on the Translator.

7. Security Considerations

The security considerations of the RTP specification [RFC3550], the Extended RTP profile for RTCP-Based Feedback [RFC4585], and the Secure Real-time Transport Protocol [RFC3711] apply. Translator must be trusted by main media source and insertion media source, and must be included in the security context.

8. IANA Considerations

No IANA actions are required.

9. Acknowledgments

The following individuals have reviewed the earlier versions of this specification and provided very valuable comments: Colin Perkins, Magnus Westerlund, Roni Even, Tom Van Caenegem, Joerg Ott, David R Oran, Cullen Jennings, Ali C Begen, and Ning Zong.

10. Change Log

10.1. draft-xia-avtext-splicing-for-rtp-00

The following are the major changes compared to previous version 00:

- Change primary RTP stream to main RTP stream, add current RTP stream as the streaming received by RTP receiver.
- Eliminate the ambiguity of inserted content with substitutive content which replaces the main content rather than pause it.
- Clarify the signaling requirements.
- Delete the description on Mixer and MCU in section 4, mainly focus on the direction whether a Translator can act as a Splicer.
- Add section 5 to describe the exact guidance on how an RTP Translator is used to handle splicing.
- Modify the security considerations section and add acknowledges section.

11. Normative References

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

[RFC2250] Hoffman, D., Fernando, G., Goyal, V., and M. Civanlar, "RTP Payload Format for MPEG1/MPEG2 Video", RFC 2250, January 1998.

[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.

[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.

[RFC5117] Westerlund, M. and S. Wenger, "RTP Topologies", RFC 5117, January 2008.

[RFC5760] Ott, J., Chesterfield, J., and E. Schooler, "RTP Control Protocol (RTCP) Extensions for Single-Source Multicast Sessions with Unicast Feedback", RFC 5760, February 2010.

[I-D.ietf-avt-ecn-for-rtp] Westerlund, M., "Explicit Congestion Notification (ECN) for RTP over UDP", draft-ietf-avt-ecn-for-rtp-03 (work in progress), October 2010.

[SCTE30] Society of Cable Telecommunications Engineers (SCTE), "Digital Program Insertion Splicing API", 2001.

[SCTE35] Society of Cable Telecommunications Engineers (SCTE), "Digital Program Insertion Cueing Message for Cable", 2004.

[H.323] ITU-T Recommendation H.323, "Packet-based multimedia communications systems", June 2006.

Author's Address

Jinwei Xia
Huawei
Hui Hong Mansion
Nanjing, Baixia District 210001
China

Phone: +86-025-86622310
Email: xiajinwei@huawei.com