AVTCORE Working Group
INTERNET-DRAFT
Category: Informational
Expires: April 29, 2018

B. Aboba
Microsoft Corporation
P. Thatcher
Google
C. Perkins
University of Glasgow
29 October 2017

QUIC Multiplexing
draft-aboba-avtcore-quic-multiplexing-01.txt

Abstract

If QUIC is to be used in a peer-to-peer manner, with NAT traversal, then it is necessary to be able to demultiplex QUIC and STUN flows running on a single UDP port. This memo discusses options for how to perform such demultiplexing. It also considers demultiplexing of QUIC and WebRTC traffic (both media and data) when running on a single UDP port.

Status of This Memo

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

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

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

This Internet-Draft will expire on April 29, 2018.

Copyright Notice

Copyright (c) 2017 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (http://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction

QUIC [I-D.ietf-quic-transport] is a new network transport protocol. While it is initially intended as a replacement for TCP in order to better support HTTP/2 [RFC7540] it should eventually be useful as a general purpose transport. HTTP is an asymmetric client-server protocol, but other uses of QUIC might operate in a peer-to-peer manner and so will need effective NAT traversal using ICE [RFC5245], which which makes use of STUN [RFC5389] and TURN [RFC5766] to discover NAT bindings. This STUN and TURN traffic needs to run on the same UDP port as the QUIC traffic. Accordingly, if QUIC is to be used in a peer-to-peer manner, then it needs to be possible to demultiplex QUIC, STUN, and TURN traffic running on a single UDP port. This memo discusses how to do this.

In addition, there are a number of ways in which communication between WebRTC peers may utilize QUIC. One of these is transport of RTP over QUIC, described in [I-D.rtpfolks-quic-rtp-over-quic]. Another is use of QUIC for data exchange. A Javascript API for use of QUIC in WebRTC data exchange has been incorporated into the ORTC API [ORTC], under development within the W3C ORTC Community Group.

In a WebRTC scenario where ICE is utilized for NAT traversal, SRTP [RFC3711] is keyed using DTLS-SRTP [RFC5764] and QUIC is used for data exchange, RTP/RTCP [RFC3550], STUN, TURN, DTLS [RFC6347], ZRTP [RFC6189] and QUIC may all need to be multiplexed over a single ICE transport.

As noted in [RFC7983] Figure 3, protocol demultiplexing currently relies upon differentiation based on the first octet, as follows:

```
                 +----------------+
                 |        [0..3] -+--> forward to STUN
                 |                |
                 |      [16..19] -+--> forward to ZRTP
                 |                |
     packet -->  |      [20..63] -+--> forward to DTLS
                 |                |
                 |      [64..79] -+--> forward to TURN Channel
                 |                |
                 |    [128..191] -+--> forward to RTP/RTCP
                 +----------------+
```

Figure 1: DTLS-SRTP receiver's packet demultiplexing algorithm.

As noted by Colin Perkins and Lars Eggert in [QUIC-Issue] this creates a potential conflict with the current design of the QUIC headers described in [I-D.ietf-quic-transport], since the first octet of the QUIC header is either:

```
 +-+-+-+-+-+-+-+-+
 |1|   Type (7)  |  Long header packet
 +-+-+-+-+-+-+-+-+
```

which potentially produces values of the first octet in the range 129-134, conflicting with RTP/RTCP, or

```
 +-+-+-+-+-+-+-+-+
 |0|C|K| Type (5)|  Short header packet
 +-+-+-+-+-+-+-+-+
```

which produces values for the first octet in the ranges 1-3, 33-35, 65-67 or 97-99, potentially conflicting with STUN, DTLS and TURN.

1.1. Terminology

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

2. Solutions

This section presents potential solutions to the QUIC multiplexing problem, including changes to the QUIC headers, addition of a multiplexing octet and use of heuristics.

2.1. QUIC Header Changes

As noted in [QUIC-Issue], one potential solution involves changes to the QUIC headers, such as setting the top two bits of the first octet of a QUIC packet to 1. This would imply a reduction in the size of the type fields:

```
 +-+-+-+-+-+-+-+-+
 |1|1|1|Type (5) |  Long header packet
 +-+-+-+-+-+-+-+-+

 +-+-+-+-+-+-+-+-+
 |1|1|0|C|K|Type3|  Short header packet
 +-+-+-+-+-+-+-+-+
```

Note: [QUIC-Spin] proposes to add a spin bit to the type octet within the QUIC header, in order to allow for RTT calculation. This would leave 4 bits for the type field in the long header packet and 2 bits for the type field in the short header, which would accommodate the type field values allocated in [I-D.ietf-quic-transport].

The advantage to this approach is that it adds no additional overhead on-the-wire. However it does require a reduction in the size of the QUIC Type fields and could potentially require allocation of the following initial octet code points for QUIC: For the Long header, 225-230 (241-246 when the spin bit is set) and for the Short header, 193-195 (209-11 with spin bit set), 209-211 (225-227 with spin bit set) and 217-219 (233-235 with the spin bit set). Utilizing all of these code points for QUIC would leave limited code points available for future allocations.

2.2. Multiplexing Shim

In this approach, an initial octet not allocated within [RFC7983] would be prepended to each QUIC packet, allowing QUIC packets to be differentiated from RTP, RTCP, DTLS, STUN, TURN and ZRTP based on the first octet alone. As an example, an octet with decimal value 192 could be used:

```
+-+-+-+-+-+-+-+-+
|1|1|0|0|0|0|0|0|
+-+-+-+-+-+-+-+-+
```

Advantages of this approach include simplicity and the consumption of only a single initial octet code point for demultiplexing of QUIC. The disadvantage is the addition of a single octet of overhead to every QUIC packet, which could impact performance where small payloads are exchanged, such as in peer-to-peer gaming.

2.3. Heuristics

During the QUIC WG interim in Seattle, Martin Thomson suggested the following heuristics for differentiation of QUIC packets from RTP/RTCP/DTLS/STUN/TURN/ZRTP:

1. Demultiplex differently during the "QUIC handshake" and "steady state".
2. During handshake, we only need to worry about the QUIC Long header, which simplifies the logic.
   a. Force all handshake packets to utilize the QUIC Long header.
   b. The QUIC Long header (0x1XXXXXXX) (or 0x11XXXXXX with the spin bit set) does not conflict with STUN (0x000000XX), DTLS (0x000XXXXX), or TURN Channel (0x01XXXXXX).
   c. The QUIC Long header does conflict with RTP/RTCP (0x10XXXXXX), but those packets typically aren’t sent until the QUIC handshake is completed. Corner case: an application starts off with audio and video keyed with DTLS-SRTP without QUIC, then the application wishes to add QUIC data (e.g. the user

clicks on the "white-board" icon).
   i. Alternative: force the RTP padding bit to 1 using a one-byte pad if there isn't already padding (pad == 0x01). Then force QUIC to have a type < 64 (the current max is 8).
   ii. Alternative: Disallow QUIC in this case, use SCTP data exchange instead.

3. During "steady state", we only need to worry about the QUIC Short header.
   a. QUIC doesn't need the Long header after the handshake.
   b. The QUIC Short header (0x0XXXXXXX or 0x01XXXXXX with the spin bit set) does not conflict with RTP/RTCP (0x10XXXXXX), so we only need to worry about conflicts with STUN/TURN/DTLS/ZRTP.
   c. Disallow simultaneous use of DTLS and QUIC Short header packets.
      i. Alternative: when using DTLS and QUIC at the same time, only use the QUIC Long header. Not optimal, but isn't really needed.
   d. ICE can be demultiplexed using the magic cookie and checksum.
      i. Alternative: STUN can only conflict with 3 QUIC packet types: Version Negotiation, Client Initial, and Server Stateless Retry. Out of those, none should be needed during the steady state.
   e. We shouldn't need to demultiplex QUIC with TURN channel data or other STUN traffic. What about consent packets?

This approach has the advantage that it requires no changes to QUIC headers, nor does it add any overhead to QUIC packets. Disadvantages include additional complexity within the multiplexing algorithm, the consumption of additional multiplexing code points, and potential future difficulties in adapting the algorithm to support changes to the QUIC protocol or additional protocols to be multiplexed.

3. Security Considerations

The solutions discussed in this document could potentially introduce some additional security considerations beyond those detailed in [RFC7983].

Due to the additional logic required, if mis-implemented, heuristics have the potential to mis-classify packets.

When QUIC is used for only for data exchange, the TLS-within-QUIC exchange [I-D.ietf-quic-tls] derives keys used solely to protect the

QUIC data packets. If properly implemented, this should not affect the transport of SRTP nor the derivation of SRTP keys via DTLS-SRTP, but if badly implemented, both transport and key derivation could be adversely impacted.

4. IANA Considerations

This document does not require actions by IANA.

5. References

5.1. Informative References

[I-D.ietf-quic-tls]
Thomson, M. and S. Turner, "Using Transport Layer Security (TLS) to Secure QUIC", draft-ietf-quic-tls-07 (work in progress), October 13, 2017.

[I-D.ietf-quic-transport]
Iyengar, J. and M. Thomson, "QUIC: A UDP-Based Multiplexed and Secure Transport", draft-ietf-quic-transport-07 (work in progress), October 13, 2017.

[I-D.rtpfolks-quic-rtp-over-quic]
Ott, J., Even, R., Perkins, C. and V. Singh, "RTP over QUIC", draft-rtpfolks-quic-rtp-over-quic-01 (work in progress), September 1, 2017.

[ORTC] Raymond, R., Aboba, B. and J. Uberti, "Object RTC (ORTC) API for WebRTC", W3C, http://draft.ortc.org/, October 2017.

[QUIC-Issue] Perkins, C., "QUIC header format/demultiplexing", https://github.com/quicwg/base-drafts/issues/426, March, 2017.

[QUIC-Spin] Huitema, C., "QUIC Latency Spin Bit", https://github.com/quicwg/base-drafts/issues/609, June 2017.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <http://www.rfc-editor.org/info/rfc2119>.

[RFC3550] Schulzrinne, H., Casner, S., Frederick, R., and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications", STD 64, RFC 3550, DOI 10.17487/RFC3550, July 2003, <http://www.rfc-editor.org/info/rfc3550>.

[RFC3711] Baugher, M., McGrew, D., Naslund, M., Carrara, E., and K. Norrman, "The Secure Real-time Transport Protocol (SRTP)", RFC 3711, DOI 10.17487/RFC3711, March 2004, <http://www.rfc-editor.org/info/rfc3711>.

[RFC5245] Rosenberg, J., "Interactive Connectivity Establishment (ICE): A Protocol for Network Address Translator (NAT) Traversal for Offer/Answer Protocols", RFC 5245, DOI 10.17487/RFC5245, April 2010, <http://www.rfc-editor.org/info/rfc5245>.

[RFC5389] Rosenberg, J., Mahy, R., Matthews, P., and D. Wing, "Session Traversal Utilities for NAT (STUN)", RFC 5389, DOI 10.17487/RFC5389, October 2008, <http://www.rfc-editor.org/info/rfc5389>.

[RFC5764] McGrew, D. and E. Rescorla, "Datagram Transport Layer Security (DTLS) Extension to Establish Keys for the Secure Real-time Transport Protocol (SRTP)", RFC 5764, DOI 10.17487/RFC5764, May 2010, <http://www.rfc-editor.org/info/rfc5764>.

[RFC5766] Mahy, R., Matthews, P., and J. Rosenberg, "Traversal Using Relays around NAT (TURN): Relay Extensions to Session Traversal Utilities for NAT (STUN)", RFC 5766, DOI 10.17487/RFC5766, April 2010, <http://www.rfc-editor.org/info/rfc5766>.

[RFC6189] Zimmermann, P., Johnston, A., Ed., and J. Callas, "ZRTP: Media Path Key Agreement for Unicast Secure RTP", RFC 6189, DOI 10.17487/RFC6189, April 2011, <http://www.rfc-editor.org/info/rfc6189>.

[RFC6347] Rescorla, E. and N. Modadugu, "Datagram Transport Layer Security Version 1.2", RFC 6347, DOI 10.17487/RFC6347, January 2012, <http://www.rfc-editor.org/info/rfc6347>.

[RFC7540] Belshe, M., Peon, R., and M. Thomson, Ed., "Hypertext Transfer Protocol Version 2 (HTTP/2)", RFC 7540, DOI 10.17487/RFC7540, May 2015, <https://www.rfc-editor.org/info/rfc7540>.

[RFC7983] Petit-Huguenin, M. and G. Salgueiro, "Multiplexing Scheme Updates for Secure Real-time Transport Protocol (SRTP) Extension for Datagram Transport Layer Security (DTLS)", RFC 7983, DOI 10.17487/RFC7983, September 2016, <https://www.rfc-editor.org/info/rfc7983>.

Acknowledgments

We would like to thank Martin Thomson, Roni Even and other participants in the IETF QUIC and AVTCORE working groups for their discussion of the QUIC multiplexing issue, and their input relating to potential solutions.

Authors' Addresses

Bernard Aboba
Microsoft Corporation
One Microsoft Way
Redmond, WA 98052
USA

Email: bernard.aboba@gmail.com

Peter Thatcher
Google
747 6th St S
Kirkland, WA 98033
USA

Email: pthatcher@google.com

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

Email: csp@csperkins.org

AVTCore R. Even
Internet-Draft O. Idan
Intended status: Standards Track Huawei Technologies
Expires: February 28, 2018 August 27, 2017

Frame Priority Marking RTP Header Extension
draft-even-avtcore-priority-markings-00.txt

Abstract

This document updates the Frame Marking RTP header extension in draft-ietf-avtext-framemarking-14 used to convey information about video frames that is critical for error recovery and packet forwarding in RTP middle-boxes or network nodes. The flags for frame marking for non-scalable streams include the D bit to mark a frame that can be discarded, and still provide a decodable media stream. There is also the I bit for frames that can be decoded independent of prior frames, e.g. intra-frame.

This memo adds priority values for the non-scalable streams droppable frames

Status of This Memo

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

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

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

This Internet-Draft will expire on February 28, 2018.

Copyright Notice

Copyright (c) 2017 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust’s Legal Provisions Relating to IETF Documents (http://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction

Frame Marking RTP Header Extension [I-D.ietf-avtext-framemarking] provides a single bit for marking frames that may be discarded by a middle box for non-scalable streams. Having one bit for marking a discardable frame provides the same information to a middle box that need to drop few frames or many frames. If the encoder wants to mark multiple frames as droppable allowing the middle box to discard part or all the discardable frames. The middle box can use more information for deciding which frames to drop. A video stream is composed of Group of Pictures (GOP) where the GOP includes I,P and B frames. A GOP is typically bound by I frames and is 15-30,60 frames long but can vary with frame rate, content complexity and encoder implementation. There are a couple of use cases that can benefit if discard priority is available.

- o When there are contiguous non referenced B frames dropping all of them will reduce the actual frame rate. By providing different priority to each of these B frames the middle box can affect the actual frame rate. This information can be also deducted based on the number of contiguous frames but having priority will make it easier for the middle box for example when the frames are interleaved.

- o When there are referenced B frames, for example a non referenced B frame (B1) followed by a B frame (B2) referenced by B1 only. If B1 is dropped then B2 can be dropped too. By using priority B1 can have lower priority than B2.

- o Dropping a P frame that is close to the end of the GOP is also possible comparing to a P frame in the beginning of the GOP. The encoder can know when such P frame exist and mark is as droppable with lowest priority.

2. Requirements Notation

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

3. Frame Priority

This memo adds two P bits to the RTP header extension defined in [I-D.ietf-avtext-framemarking] section 3.1.

RTP Header Extension for non-scalable streams:

```
 0                   1
  0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |  ID=? |  L=0  |S|E|I|D|P P 0 0|
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

P: Priority bits (2 bits). If the D bit is set to zero these bits MUST be zero. If the D bit is set to 1 the values 00 is the highest drop priority (this will be the case when priority is not specified) and 11 is the lowest drop priority.

Based on the use cases from the introduction, the priority of the non referenced B frame will be 00, the priority of the referenced B frames will be 01 and the priority of the droppable P frame will be 10. If the middle box drops the frames marked with priority 00 it can now drop the frames marked with priority 01 since they are not needed for decoding the stream.

4. IANA Considerations

There are no IANA actions

5. Security considerations

This memo does not add any security information to the ones in [I-D.ietf-avtext-framemarking]

6. Normative References

[I-D.ietf-avtext-framemarking]
Berger, E., Nandakumar, S., and M. Zanaty, "Frame Marking RTP Header Extension", draft-ietf-avtext-framemarking-05 (work in progress), July 2017.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

Authors' Addresses

Roni Even
Huawei Technologies
Tel Aviv
Israel

Email: Roni.even@huawei.com

Ofer Idan
Huawei Technologies
Hod Hasharon
Israel

Email: ofer.idan@huawei.com

AVTCore
Internet-Draft
Intended status: Informational
Expires: February 9, 2018

W. Kim
J. Lee
J. Park
D. Kwon
NSRI
D. Kim
Kookmin Univ.
August 8, 2017

The ARIA Algorithm and Its Use with the Secure Real-time Transport Protocol(SRTP)

draft-ietf-avtcore-aria-srtp-11

Abstract

This document defines the use of the ARIA block cipher algorithm within the Secure Real-time Transport Protocol (SRTP). It details two modes of operation (CTR, GCM) and the SRTP Key Derivation Functions for ARIA. Additionally, this document defines DTLS-SRTP protection profiles and MIKEY parameter sets for the use with ARIA.

Status of This Memo

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

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

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

This Internet-Draft will expire on February 9, 2018.

Copyright Notice

Copyright (c) 2017 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (http://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction

This document defines the use of the ARIA [RFC5794] block cipher algorithm in the Secure Real-time Transport Protocol (SRTP) [RFC3711] for providing confidentiality for the Real-time Transport Protocol (RTP) [RFC3550] traffic and for the RTP Control Protocol (RTCP) [RFC3550] traffic.

1.1. ARIA

ARIA is a general-purpose block cipher algorithm developed by Korean cryptographers in 2003. It is an iterated block cipher with 128-, 192-, and 256-bit keys and encrypts 128-bit blocks in 12, 14, and 16 rounds, depending on the key size. It is secure and suitable for most software and hardware implementations on 32-bit and 8-bit processors. It was established as a Korean standard block cipher algorithm in 2004 [ARIAKS] and has been widely used in Korea, especially for government-to-public services. It was included in PKCS #11 in 2007 [ARIAPKCS]. The algorithm specification and object identifiers are described in [RFC5794].

1.2. Terminology

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

2. Cryptographic Transforms

Block ciphers ARIA and AES share common characteristics including mode, key size, and block size. ARIA does not have any restrictions for modes of operation that are used with this block cipher. We define two modes of running ARIA within the SRTP protocol, (1) ARIA in Counter Mode (ARIA-CTR) and (2) ARIA in Galois/Counter Mode (ARIA-GCM).

2.1. ARIA-CTR

Section 4.1.1 of [RFC3711] defines AES-128 counter mode encryption, which it refers to as "AES_CM". Section 2 of [RFC6188] defines "AES_256_CM" in SRTP. ARIA counter modes are defined in the same manner except that each invocation of AES is replaced by that of ARIA [RFC5794], and are denoted by ARIA_128_CTR and ARIA_256_CTR, respectively, according to the key lengths. The plaintext inputs to the block cipher are formed as in AES-CTR(AES_CM, AES_256_CM) and the block cipher outputs are processed as in AES-CTR. Note that, ARIA-CTR MUST be used only in conjunction with an authentication transform.

Section 3.2 of [RFC6904] defines AES-CTR for SRTP header extension keystream generation. When ARIA-CTR is used, the header extension keystream SHALL be generated in the same manner except that each invocation of AES is replaced by that of ARIA [RFC5794].

2.2. ARIA-GCM

GCM (Galois Counter Mode) [GCM][RFC5116] is an AEAD (Authenticated Encryption with Associated Data) block cipher mode. A detailed description of ARIA-GCM is defined similarly as AES-GCM found in [RFC5116][RFC5282].

The document [RFC7714] describes the use of AES-GCM with SRTP [RFC3711][RFC6904]. The use of ARIA-GCM with SRTP is defined the same as that of AES-GCM except that each invocation of AES is replaced by ARIA [RFC5794]. When encryption of header extensions [RFC6904] is in use, a separate keystream to encrypt selected RTP header extension elements MUST be generated in the same manner defined in [RFC7714] except that AES-CTR is replaced by ARIA-CTR.

3. Key Derivation Functions

Section 4.3.3 of [RFC3711] defines the AES-128 counter mode key derivation function, which it refers to as "AES-CM PRF". Section 3 of [RFC6188] defines the AES-256 counter mode key derivation function, which it refers to as "AES_256_CM_PRF". The ARIA-CTR PRF is defined in a same manner except that each invocation of AES is replaced by that of ARIA. According to the key lengths of underlying encryption algorithm, ARIA-CTR PRFs are denoted by "ARIA_128_CTR_PRF" and "ARIA_256_CTR_PRF". The usage requirements of [RFC6188][RFC7714] regarding the AES-CM PRF apply to the ARIA-CTR PRF as well.

4. Protection Profiles

This section defines SRTP Protection Profiles that use the ARIA transforms and key derivation functions defined in this document. The following list indicates the SRTP transform parameters for each protection profile. Those are described for use with DTLS-SRTP [RFC5764].

The parameters cipher_key_length, cipher_salt_length, auth_key_length, and auth_tag_length express the number of bits in the values to which they refer. The maximum_lifetime parameter indicates the maximum number of packets that can be protected with each single set of keys when the parameter profile is in use. All of these parameters apply to both RTP and RTCP, unless the RTCP parameters are separately specified.

```
SRTP_ARIA_128_CTR_HMAC_SHA1_80
        cipher:                   ARIA_128_CTR
        cipher_key_length:        128 bits
        cipher_salt_length:       112 bits
        key derivation function:  ARIA_128_CTR_PRF
        auth_function:            HMAC-SHA1
        auth_key_length:          160 bits
        auth_tag_length:          80 bits
        maximum_lifetime:         at most 2^31 SRTCP packets and
                                  at most 2^48 SRTP packets

SRTP_ARIA_128_CTR_HMAC_SHA1_32
```

```
        cipher:                    ARIA_128_CTR
        cipher_key_length:         128 bits
        cipher_salt_length:        112 bits
        key derivation function:   ARIA_128_CTR_PRF
        auth_function:             HMAC-SHA1
        auth_key_length:           160 bits
        SRTP auth_tag_length:      32 bits
        SRTCP auth_tag_length:     80 bits
        maximum_lifetime:          at most 2^31 SRTCP packets and
                                   at most 2^48 SRTP packets

   SRTP_ARIA_256_CTR_HMAC_SHA1_80
        cipher:                    ARIA_256_CTR
        cipher_key_length:         256 bits
        cipher_salt_length:        112 bits
        key derivation function:   ARIA_256_CTR_PRF
        auth_function:             HMAC-SHA1
        auth_key_length:           160 bits
        auth_tag_length:           80 bits
        maximum_lifetime:          at most 2^31 SRTCP packets and
                                   at most 2^48 SRTP packets

   SRTP_ARIA_256_CTR_HMAC_SHA1_32
        cipher:                    ARIA_256_CTR
        cipher_key_length:         256 bits
        cipher_salt_length:        112 bits
        key derivation function:   ARIA_256_CTR_PRF
        auth_function:             HMAC-SHA1
        auth_key_length:           160 bits
        SRTP auth_tag_length:      32 bits
        SRTCP auth_tag_length:     80 bits
        maximum_lifetime:          at most 2^31 SRTCP packets and
                                   at most 2^48 SRTP packets

   SRTP_AEAD_ARIA_128_GCM
        cipher:                    ARIA_128_GCM
        cipher_key_length:         128 bits
        cipher_salt_length:        96 bits
        aead_auth_tag_length:      128 bits
        auth_function:             NULL
        auth_key_length:           N/A
        auth_tag_length:           N/A
        key derivation function:   ARIA_128_CTR_PRF
        maximum_lifetime:          at most 2^31 SRTCP packets and
                                   at most 2^48 SRTP packets

   SRTP_AEAD_ARIA_256_GCM
        cipher:                    ARIA_256_GCM
```

```
cipher_key_length:        256 bits
cipher_salt_length:       96 bits
aead_auth_tag_length:     128 bits
auth_function:            NULL
auth_key_length:          N/A
auth_tag_length:          N/A
key derivation function:  ARIA_256_CTR_PRF
maximum_lifetime:         at most 2^31 SRTCP packets and
                          at most 2^48 SRTP packets
```

The ARIA-CTR protection profiles use the same authentication transform that is mandatory to implement in SRTP, HMAC-SHA1 with a 160-bit key.

Note that SRTP Protection Profiles that use AEAD algorithms do not specify an auth_function, auth_key_length, or auth_tag_length, since they do not use a separate auth_function, auth_key, or auth_tag. The term aead_auth_tag_length is used to emphasize that this refers to the authentication tag provided by the AEAD algorithm and that this tag is not located in the authentication tag field provided by SRTP/SRTCP.

The PRFs for ARIA protection profiles are defined by ARIA-CTR PRF of the equal key length with the encryption algorithm (see Section 2). SRTP_ARIA_128_CTR_HMAC and SRTP_AEAD_ARIA_128_GCM MUST use the ARIA_128_CTR_PRF Key Derivation Function. And SRTP_ARIA_256_CTR_HMAC and SRTP_AEAD_ARIA_256_GCM MUST use the ARIA_256_CTR_PRF Key Derivation Function.

MIKEY specifies the SRTP protection profile definition separately from the key length (which is specified by the Session Encryption key length) and the authentication tag length. The DTLS-SRTP [RFC5764] protection profiles are mapped to MIKEY parameter sets as shown below.

	Encryption Algorithm	Encryption Key Length	Auth. Tag Length
SRTP_ARIA_128_CTR_HMAC_80	ARIA-CTR	16 octets	10 octets
SRTP_ARIA_128_CTR_HMAC_32	ARIA-CTR	16 octets	4 octets
SRTP_ARIA_256_CTR_HMAC_80	ARIA-CTR	32 octets	10 octets
SRTP_ARIA_256_CTR_HMAC_32	ARIA-CTR	32 octets	4 octets

Figure 1: Mapping MIKEY parameters to ARIA-CTR with HMAC algorithm

	Encryption Algorithm	Encryption Key Length	AEAD Auth. Tag Length
SRTP_AEAD_ARIA_128_GCM	ARIA-GCM	16 octets	16 octets
SRTP_AEAD_ARIA_256_GCM	ARIA-GCM	32 octets	16 octets

Figure 2: Mapping MIKEY parameters to AEAD algorithm

5. Security Considerations

At the time of publication of this document no security problem has been found on ARIA. Previous security analysis results are summarized in [ATY].

The security considerations in [GCM] [RFC3711] [RFC5116] [RFC6188] [RFC6904] [RFC7714] apply to this document as well. This document includes crypto suites with authentication tags of length less than 80 bits. These suites MAY be used for certain application contexts where longer authentication tags may be undesirable, for example, those mentioned in [RFC3711] section 7.5. Otherwise, short authentication tags SHOULD NOT be used, since may reduce authentication strength. See [RFC3711] section 9.5 for a discussion of risks related to weak authentication in SRTP.

At the time of publication of this document, SRTP recommends HMAC-SHA1 as the default and mandatory-to-implement MAC algorithm. All currently registered SRTP crypto suites except the GCM based ones use HMAC-SHA1 as their HMAC algorithm to provide message authentication. Due to security concerns with SHA-1 [RFC6194], the IETF is gradually moving away from SHA-1 and towards stronger hash algorithms such as SHA-2 or SHA-3 families. For SRTP, however, SHA-1 is only used in the calculation of an HMAC, and no security issue is known for this usage at the time of this publication.

6. IANA Considerations

6.1. DTLS-SRTP

DTLS-SRTP [RFC5764] defines a DTLS-SRTP "SRTP Protection Profile". In order to allow the use of the algorithms defined in this document in DTLS-SRTP, IANA is requested to add the protection profiles below to the "DTLS-SRTP Protection Profiles" created by [RFC5764], located on the following IANA page at time of writing: http://www.iana.org/assignments/srtp-protection/.

    SRTP_ARIA_128_CTR_HMAC_SHA1_80 = {TBD,TBD}

```
SRTP_ARIA_128_CTR_HMAC_SHA1_32 = {TBD,TBD}
SRTP_ARIA_256_CTR_HMAC_SHA1_80 = {TBD,TBD}
SRTP_ARIA_256_CTR_HMAC_SHA1_32 = {TBD,TBD}
SRTP_AEAD_ARIA_128_GCM = {TBD,TBD}
SRTP_AEAD_ARIA_256_GCM = {TBD,TBD}
```

6.2. MIKEY

[RFC3830] and [RFC5748] define encryption algorithms and PRFs for the SRTP policy in MIKEY. In order to allow the use of the algorithms defined in this document in MIKEY, IANA is requested to add the two encryption algorithms below to the "MIKEY Security Protocol Parameters SRTP Type 0 (Encryption algorithm)" and to add the PRF below to the "MIKEY Security Protocol Parameters SRTP Type 5 (Pseudo Random Function)" created by [RFC3830], located on the following IANA page at time of writing: http://www.iana.org/assignments/mikey-payloads/.

SRTP Enc. alg	Value
ARIA-CTR	TBD
ARIA-GCM	TBD

Default session encryption key length is 16 octets.

SRTP PRF	Value
ARIA-CTR	TBD

7. References

7.1. Normative References

[GCM] Dworkin, M., "Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC", NIST SP 800-38D, November 2007.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <http://www.rfc-editor.org/info/rfc2119>.

[RFC3550] Schulzrinne, H., Casner, S., Frederick, R., and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications", STD 64, RFC 3550, DOI 10.17487/RFC3550, July 2003, <http://www.rfc-editor.org/info/rfc3550>.

[RFC3711] Baugher, M., McGrew, D., Naslund, M., Carrara, E., and K. Norrman, "The Secure Real-time Transport Protocol (SRTP)", RFC 3711, DOI 10.17487/RFC3711, March 2004, <http://www.rfc-editor.org/info/rfc3711>.

[RFC3830] Arkko, J., Carrara, E., Lindholm, F., Naslund, M., and K. Norrman, "MIKEY: Multimedia Internet KEYing", RFC 3830, DOI 10.17487/RFC3830, August 2004, <http://www.rfc-editor.org/info/rfc3830>.

[RFC5116] McGrew, D., "An Interface and Algorithms for Authenticated Encryption", RFC 5116, DOI 10.17487/RFC5116, January 2008, <http://www.rfc-editor.org/info/rfc5116>.

[RFC5282] Black, D. and D. McGrew, "Using Authenticated Encryption Algorithms with the Encrypted Payload of the Internet Key Exchange version 2 (IKEv2) Protocol", RFC 5282, DOI 10.17487/RFC5282, August 2008, <http://www.rfc-editor.org/info/rfc5282>.

[RFC5764] McGrew, D. and E. Rescorla, "Datagram Transport Layer Security (DTLS) Extension to Establish Keys for the Secure Real-time Transport Protocol (SRTP)", RFC 5764, DOI 10.17487/RFC5764, May 2010, <http://www.rfc-editor.org/info/rfc5764>.

[RFC5794] Lee, J., Lee, J., Kim, J., Kwon, D., and C. Kim, "A Description of the ARIA Encryption Algorithm", RFC 5794, DOI 10.17487/RFC5794, March 2010, <http://www.rfc-editor.org/info/rfc5794>.

[RFC6188] McGrew, D., "The Use of AES-192 and AES-256 in Secure RTP", RFC 6188, DOI 10.17487/RFC6188, March 2011, <http://www.rfc-editor.org/info/rfc6188>.

[RFC6904] Lennox, J., "Encryption of Header Extensions in the Secure Real-time Transport Protocol (SRTP)", RFC 6904, DOI 10.17487/RFC6904, April 2013, <http://www.rfc-editor.org/info/rfc6904>.

[RFC7714] McGrew, D. and K. Igoe, "AES-GCM Authenticated Encryption in the Secure Real-time Transport Protocol (SRTP)", RFC 7714, DOI 10.17487/RFC7714, December 2015, <http://www.rfc-editor.org/info/rfc7714>.

7.2. Informative References

[ARIAKS] Korean Agency for Technology and Standards, "128 bit block encryption algorithm ARIA - Part 1: General (in Korean)", KS X 1213-1:2009, December 2009.

[ARIAPKCS] RSA Laboratories, "Additional PKCS #11 Mechanisms", PKCS #11 v2.20 Amendment 3 Revision 1, January 2007.

[ATY] Abdelkhalek, A., Tolba, M., and A. Youssef, "Improved linear cryptanalysis of round-reduced ARIA", Information Security - ISC 2016, Lecture Notes in Computer Science (LNCS) Vol. 9866, pp. 18-34, September 2016.

[RFC5748] Yoon, S., Jeong, J., Kim, H., Jeong, H., and Y. Won, "IANA Registry Update for Support of the SEED Cipher Algorithm in Multimedia Internet KEYing (MIKEY)", RFC 5748, DOI 10.17487/RFC5748, August 2010, <http://www.rfc-editor.org/info/rfc5748>.

[RFC6194] Polk, T., Chen, L., Turner, S., and P. Hoffman, "Security Considerations for the SHA-0 and SHA-1 Message-Digest Algorithms", RFC 6194, DOI 10.17487/RFC6194, March 2011, <http://www.rfc-editor.org/info/rfc6194>.

Appendix A. Test Vectors

All values are in hexadecimal and represented by the network order (called big endian).

A.1. ARIA-CTR Test Vectors

Common values are organized as follows:

```
Rollover Counter:           00000000
Sequence Number:            315e
SSRC:                       20e8f5eb
Authentication Key:         f93563311b354748c978913795530631
                            16452309
Session Salt:               cd3a7c42c671e0067a2a2639b43a
Initialization Vector:      cd3a7c42e69915ed7a2a263985640000
RTP header:                 8008315ebf2e6fe020e8f5eb
RTP Payload:                f57af5fd4ae19562976ec57a5a7ad55a
                            5af5c5e5c5fdf5c55ad57a4a7272d572
                            62e9729566ed66e97ac54a4a5a7ad5e1
                            5ae5fdd5fd5ac5d56ae56ad5c572d54a
                            e54ac55a956afd6aed5a4ac562957a95
                            16991691d572fd14e97ae962ed7a9f4a
                            955af572e162f57a956666e17ae1f54a
                            95f566d54a66e16e4afd6a9f7ae1c5c5
                            5ae5d56afde916c5e94a6ec56695e14a
                            fde1148416e94ad57ac5146ed59d1cc5
```

A.1.1. SRTP_ARIA_128_CTR_HMAC_SHA1_80

```
      Session Key:                0c5ffd37a11edc42c325287fc0604f2e

      Encrypted RTP Payload:      1bf753f412e6f35058cc398dc851aae3
                                  a6ccdcb463fbed9cfb3de2fb76fdffa9
                                  e481f5efb64c92487f59dabbc7cc72da
                                  092485f3fbad87888820b86037311fa4
                                  4330e18a59a1e1338ba2c21458493a57
                                  463475c54691f91cec785429119e0dfc
                                  d9048f90e07fecd50b528e8c62ee6e71
                                  445de5d7f659405135aff3604c2ca4ff
                                  4aaca40809cb9eee42cc4ad232307570
                                  81ca289f2851d3315e9568b501fdce6d

      Authenticated portion || Rollover Counter:
                                  8008315ebf2e6fe020e8f5eb1bf753f4
                                  12e6f35058cc398dc851aae3a6ccdcb4
                                  63fbed9cfb3de2fb76fdffa9e481f5ef
                                  b64c92487f59dabbc7cc72da092485f3
                                  fbad87888820b86037311fa44330e18a
                                  59a1e1338ba2c21458493a57463475c5
                                  4691f91cec785429119e0dfcd9048f90
                                  e07fecd50b528e8c62ee6e71445de5d7
                                  f659405135aff3604c2ca4ff4aaca408
                                  09cb9eee42cc4ad23230757081ca289f
                                  2851d3315e9568b501fdce6d00000000

      Authentication Tag:         f9de4e729054672b0e35
```

A.1.2. SRTP_ARIA_256_CTR_HMAC_SHA1_80

```
      Session Key:                 0c5ffd37a11edc42c325287fc0604f2e
                                   3e8cd5671a00fe3216aa5eb105783b54

      Encrypted RTP Payload:       c424c59fd5696305e5b13d8e8ca76566
                                   17ccd7471088af9debf07b55c750f804
                                   a5ac2b737be48140958a9b420524112a
                                   e72e4da5bca59d2b1019ddd7dbdc30b4
                                   3d5f046152ced40947d62d2c93e7b8e5
                                   0f02db2b6b61b010e4c1566884de1fa9
                                   702cdf8157e8aedfe3dd77c76bb50c25
                                   ae4d624615c15acfdeeb5f79482aaa01
                                   d3e4c05eb601eca2bd10518e9d46b021
                                   16359232e9eac0fabd05235dd09e6dea

      Authenticated portion || Rollover Counter:
                                   8008315ebf2e6fe020e8f5ebc424c59f
                                   d5696305e5b13d8e8ca7656617ccd747
                                   1088af9debf07b55c750f804a5ac2b73
                                   7be48140958a9b420524112ae72e4da5
                                   bca59d2b1019ddd7dbdc30b43d5f0461
                                   52ced40947d62d2c93e7b8e50f02db2b
                                   6b61b010e4c1566884de1fa9702cdf81
                                   57e8aedfe3dd77c76bb50c25ae4d6246
                                   15c15acfdeeb5f79482aaa01d3e4c05e
                                   b601eca2bd10518e9d46b02116359232
                                   e9eac0fabd05235dd09e6dea00000000

      Authentication Tag:          192f515fab04bbb4e62c
```

A.2. ARIA-GCM Test Vectors

Common values are organized as follows:

```
      Rollover Counter:           00000000
      Sequence Number:            315e
      SSRC:                       20e8f5eb
      Encryption Salt:            000000000000000000000000

      Initialization Vector:      000020e8f5eb00000000315e
      RTP Payload:                f57af5fd4ae19562976ec57a5a7ad55a
                                  5af5c5e5c5fdf5c55ad57a4a7272d572
                                  62e9729566ed66e97ac54a4a5a7ad5e1
                                  5ae5fdd5fd5ac5d56ae56ad5c572d54a
                                  e54ac55a956afd6aed5a4ac562957a95
                                  16991691d572fd14e97ae962ed7a9f4a
                                  955af572e162f57a956666e17ae1f54a
                                  95f566d54a66e16e4afd6a9f7ae1c5c5
                                  5ae5d56afde916c5e94a6ec56695e14a
                                  fde1148416e94ad57ac5146ed59d1cc5
      Associated Data:            8008315ebf2e6fe020e8f5eb
```

The length of encrypted payload is larger than that of payload by 16 octets that is the length of the tag from GCM.

A.2.1. SRTP_AEAD_ARIA_128_GCM

```
      Key:                        e91e5e75da65554a48181f3846349562

      Encrypted RTP Payload:      4d8a9a0675550c704b17d8c9ddc81a5c
                                  d6f7da34f2fe1b3db7cb3dfb9697102e
                                  a0f3c1fc2dbc873d44bceeae8e444297
                                  4ba21ff6789d3272613fb9631a7cf3f1
                                  4bacbeb421633a90ffbe58c2fa6bdca5
                                  34f10d0de0502ce1d531b6336e588782
                                  78531e5c22bc6c85bbd784d78d9e680a
                                  a19031aaf89101d669d7a3965c1f7e16
                                  229d7463e0535f4e253f5d18187d40b8
                                  ae0f564bd970b5e7e2adfb211e89a953
                                  5abace3f37f5a736f4be984bbffbedc1
```

A.2.2. SRTP_AEAD_ARIA_256_GCM

```
      Key:                       0c5ffd37a11edc42c325287fc0604f2e
                                 3e8cd5671a00fe3216aa5eb105783b54

      Encrypted RTP Payload:     6f9e4bcbc8c85fc0128fb1e4a0a20cb9
                                 932ff74581f54fc013dd054b19f99371
                                 425b352d97d3f337b90b63d1b082adee
                                 ea9d2d7391897d591b985e55fb50cb53
                                 50cf7d38dc27dda127c078a149c8eb98
                                 083d66363a46e3726af217d3a00275ad
                                 5bf772c7610ea4c23006878f0ee69a83
                                 97703169a419303f40b72e4573714d19
                                 e2697df61e7c7252e5abc6bade876ac4
                                 961bfac4d5e867afca351a48aed52822
                                 e210d6ced2cf430ff841472915e7ef48
```

A.3. Key Derivation Test Vector

This section provides test vectors for the default key derivation function that uses ARIA in Counter Mode. In the following, we walk through the initial key derivation for the ARIA Counter Mode cipher that requires a 16/24/32 octet session encryption key according to the session encryption key length and a 14 octet session salt, and an authentication function that requires a 94 octet session authentication key. These values are called the cipher key, the cipher salt, and the auth key in the following. The test vectors are generated in the same way with the test vectors of key derivation functions in [RFC3711] and [RFC6188] but with each invocation of AES replaced with an invocation of ARIA.

A.3.1. ARIA_128_CTR_PRF

The inputs to the key derivation function are the 16 octet master key and the 14 octet master salt:

```
     master key:  e1f97a0d3e018be0d64fa32c06de4139
     master salt: 0ec675ad498afeebb6960b3aabe6

     index DIV kdr:                 000000000000
     label:                       00
     master salt:   0ec675ad498afeebb6960b3aabe6
     -----------------------------------------------
     xor:           0ec675ad498afeebb6960b3aabe6     (x, PRF input)

     x*2^16:        0ec675ad498afeebb6960b3aabe60000 (ARIA-CTR input)

     cipher key:    dbd85a3c4d9219b3e81f7d942e299de4 (ARIA-CTR output)
```

ARIA-CTR protection profile requires a 14 octet cipher salt while ARIA-GCM protection profile requires a 12 octet cipher salt.

```
  index DIV kdr:                000000000000
  label:                      02
  master salt:   0ec675ad498afeebb6960b3aabe6
  ----------------------------------------------
  xor:           0ec675ad498afee9b6960b3aabe6     (x, PRF input)

  x*2^16:        0ec675ad498afee9b6960b3aabe60000 (ARIA-CTR input)

                 9700657f5f34161830d7d85f5dc8be7f (ARIA-CTR output)

  cipher salt:   9700657f5f34161830d7d85f5dc8     (ARIA-CTR profile)
                 9700657f5f34161830d7d85f         (ARIA-GCM profile)
  index DIV kdr:                000000000000
  label:                      01
  master salt:   0ec675ad498afeebb6960b3aabe6
  -----------------------------------------------
  xor:           0ec675ad498afeeab6960b3aabe6     (x, PRF input)

  x*2^16:        0ec675ad498afeeab6960b3aabe60000 (ARIA-CTR input)
```

Below, the auth key is shown on the left, while the corresponding ARIA input blocks are shown on the right.

```
  auth key                            ARIA input blocks

  d021877bd3eaf92d581ed70ddc050e03  0ec675ad498afeeab6960b3aabe60000
  f11257032676f2a29f57b21abd3a1423  0ec675ad498afeeab6960b3aabe60001
  769749bdc5dd9ca5b43ca6b6c1f3a7de  0ec675ad498afeeab6960b3aabe60002
  4047904bcf811f601cc03eaa5d7af6db  0ec675ad498afeeab6960b3aabe60003
  9f88efa2e51ca832fc2a15b126fa7be2  0ec675ad498afeeab6960b3aabe60004
  469af896acb1852c31d822c45799      0ec675ad498afeeab6960b3aabe60005
```

A.3.2. ARIA_256_CTR_PRF

The inputs to the key derivation function are the 32 octet master key and the 14 octet master salt:

```
     master key:  0c5ffd37a11edc42c325287fc0604f2e
                  3e8cd5671a00fe3216aa5eb105783b54
     master salt: 0ec675ad498afeebb6960b3aabe6

     index DIV kdr:                 000000000000
     label:                       00
     master salt: 0ec675ad498afeebb6960b3aabe6
     -----------------------------------------------
     xor:         0ec675ad498afeebb6960b3aabe6     (x, PRF input)

     x*2^16:      0ec675ad498afeebb6960b3aabe60000 (ARIA-CTR input)

     cipher key:  0649a09d93755fe9c2b2efba1cce930a (ARIA-CTR 1st output)
                  f2e76ce8b77e4b175950321aa94b0cf4 (ARIA-CTR 2nd output)
```

ARIA-CTR protection profile requires a 14 octet cipher salt while ARIA-GCM protection profile requires a 12 octet cipher salt.

```
     index DIV kdr:                 000000000000
     label:                       02
     master salt:  0ec675ad498afeebb6960b3aabe6
     ----------------------------------------------
     xor:          0ec675ad498afee9b6960b3aabe6     (x, PRF input)

     x*2^16:       0ec675ad498afee9b6960b3aabe60000 (ARIA-CTR input)

                   194abaa8553a8eba8a413a340fc80a3d (ARIA-CTR output)

     cipher salt:  194abaa8553a8eba8a413a340fc8     (ARIA-CTR profile)
                   194abaa8553a8eba8a413a34         (ARIA-GCM profile)

     index DIV kdr:                 000000000000
     label:                       01
     master salt:  0ec675ad498afeebb6960b3aabe6
     -----------------------------------------------
     xor:          0ec675ad498afeeab6960b3aabe6     (x, PRF input)

     x*2^16:       0ec675ad498afeeab6960b3aabe60000 (ARIA-CTR input)
```

Below, the auth key is shown on the left, while the corresponding ARIA input blocks are shown on the right.

```
auth key                            ARIA input blocks

e58d42915873b71899234807334658f2    0ec675ad498afeeab6960b3aabe60000
0bc460181d06e02b7a9e60f02ff10bfc    0ec675ad498afeeab6960b3aabe60001
9ade3795cf78f3e0f2556d9d913470c4    0ec675ad498afeeab6960b3aabe60002
e82e45d254bfb8e2933851a3930ffe7d    0ec675ad498afeeab6960b3aabe60003
fca751c03ec1e77e35e28dac4f17d1a5    0ec675ad498afeeab6960b3aabe60004
80bdac028766d3b1e8f5a41faa3c        0ec675ad498afeeab6960b3aabe60005
```

Authors' Addresses

Woo-Hwan Kim
National Security Research Institute
P.O.Box 1, Yuseong
Daejeon 34188
Korea

EMail: whkim5@nsr.re.kr

Jungkeun Lee
National Security Research Institute
P.O.Box 1, Yuseong
Daejeon 34188
Korea

EMail: jklee@nsr.re.kr

Je-Hong Park
National Security Research Institute
P.O.Box 1, Yuseong
Daejeon 34188
Korea

EMail: jhpark@nsr.re.kr

Daesung Kwon
National Security Research Institute
P.O.Box 1, Yuseong
Daejeon 34188
Korea

EMail: ds_kwon@nsr.re.kr

Dong-Chan Kim
Kookmin University
77 Jeongneung-ro, Seongbuk-gu
Seoul 02707
Korea

EMail: dckim@kookmin.ac.kr

IETF RMCAT Working Group Z. Sarker
Internet-Draft Ericsson AB
Intended status: Standards Track C. Perkins
Expires: May 3, 2018 University of Glasgow
V. Singh
callstats.io
M. Ramalho
Cisco Systems
October 30, 2017

RTP Control Protocol (RTCP) Feedback for Congestion Control
draft-ietf-avtcore-cc-feedback-message-00

Abstract

This document describes a feedback message intended to enable congestion control for interactive real-time traffic. The RTP Media Congestion Avoidance Techniques (RMCAT) Working Group formed a design team to analyze feedback requirements from various congestion control algorithms and to design a generic feedback message to help ensure interoperability across those algorithms. The feedback message is designed for a sender-based congestion control, which means the receiver of the media will send necessary feedback to the sender of the media to perform the congestion control at the sender.

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 https://datatracker.ietf.org/drafts/current/.

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

This Internet-Draft will expire on May 3, 2018.

Copyright Notice

Copyright (c) 2017 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (https://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction

For interactive real-time traffic the typical protocol choice is Realtime Transport Protocol (RTP) over User Datagram Protocol (UDP). RTP does not provide any guarantee of Quality of Service (QoS), reliable or timely delivery and expects the underlying transport protocol to do so. UDP alone certainly does not meet that expectation. However, RTP Control Protocol (RTCP) provides a mechanism to periodically send transport and media metrics to the media sender which can be utilized and extended for the purposes of RMCAT congestion control. For a congestion control algorithm which operates at the media sender, RTCP messages can be transmitted from the media receiver back to the media sender to enable congestion control. In the absence of standardized messages for this purpose, the congestion control algorithm designers have designed proprietary RTCP messages that convey only those parameters required for their respective designs. As a direct result, the different congestion control (a.k.a. rate adaptation) designs are not interoperable. To enable algorithm evolution as well as interoperability across designs (e.g., different rate adaptation algorithms), it is highly desirable to have generic congestion control feedback format.

To help achieve interoperability for unicast RTP congestion control, this memo proposes a common RTCP feedback format that can be used by NADA [I-D.ietf-rmcat-nada], SCReAM [I-D.ietf-rmcat-scream-cc], Google Congestion Control [I-D.ietf-rmcat-gcc] and Shared Bottleneck Detection [I-D.ietf-rmcat-sbd], and hopefully future RTP congestion control algorithms as well.

2. Terminology

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

In addition the terminology defined in [RFC3550], [RFC3551], [RFC3611], [RFC4585], and [RFC5506] applies.

3. Feedback Message

The design team analyzed the feedback requirements from the different proposed candidate in RMCAT WG. The analysis showed some commonalities between the proposed solution candidate and some can be derived from other information. The design team has agreed to have following packet information block in the feedback message to satisfy different requirement analyzed.

- Packet Identifier : RTP sequence number. The RTP packet header includes an incremental packet sequence number that the sender needs to correlate packets sent at the sender with packets received at the receiver.

- Packet Arrival Time : Arrival time stamp at the receiver of the media. The sender requires the arrival time stamp of the respective packet to determine delay and jitter the packet had experienced during transmission. In a sender based congestion control solution the sender requires to keep track of the sent packets - usually packet sequence number, packet size and packet send time. With the packet arrival time the sender can detect the delay and jitter information. Along with packet loss and delay information the sender can estimate the available bandwidth and thus adapt to the situation.

- Packet Explicit Congestion Notification (ECN) Marking : If ECN [RFC3168] is used, it is necessary to report on the 2-bit ECN mark in received packets, indicating for each packet whether it is marked not-ECT, ECT(0), ECT(1), or ECN-CE. If the path on which the media traffic traversing is ECN capable then the sender can use the Congestion Experienced (ECN-CE) marking information for congestion control. It is important that the receiver sends the

ECN-CE marking information of the packet back to the sender to take the advantages of ECN marking. Note that how the receiver gets the ECN marking information at application layer is out of the scope of this design team. Additional information for ECN use with RTP can be found at [RFC6679].

The feedback messages can have one or more of the above information blocks. For RTCP based feedback message the packet information block will be grouped by Synchronization Source (SSRC) identifier.

As a practical matter, we note that host Operating System (OS) process interruptions can occur at inopportune times. Thus, the recording of the sent times at the sender and arrival times at the receiver should be made with deliberate care. This is because the time duration of host OS interruptions can be significant relative to the precision desired in the one-way delay estimates. Specifically, the send time should be recorded at the latest opportunity prior to outputting the media packet at the sender (e.g., socket or RTP API) and the arrival time at the receiver (e.g., socket or RTP API) should be recorded at the earliest opportunity available to the receiver.

3.1. RTCP Congestion Control Feedback Report

Congestion control feedback can be sent as part of a regular scheduled RTCP report, or in an RTP/AVPF early feedback packet. If sent as early feedback, congestion control feedback MAY be sent in a non-compound RTCP packet [RFC5506] if the RTP/AVPF profile [RFC4585] or the RTP/SAVPF profile [RFC5124] is used.

Irrespective of how it is transported, the congestion control feedback is sent as a Transport Layer Feedback Message (RTCP packet type 205). The format of this RTCP packet is as follows:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|V=2|P| FMT=CCFB |   PT = 205   |          length               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                  SSRC of packet sender                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                  SSRC of 1st media source                     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|          begin_seq            |             end_seq           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|L|ECN|  Arrival time offset    | ...                           .
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
.                                                               .
.                                                               .
.                                                               .
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                  SSRC of nth media source                     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|          begin_seq            |             end_seq           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|L|ECN|  Arrival time offset    | ...                           |
.                                                               .
.                                                               .
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                 Report Timestamp (32bits)                     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The first 8 octets are the RTCP header, with PT=205 and FMT=CCFB specifying the remainder is a congestion control feedback packet, and including the SSRC of the packet sender. (NOTE TO RFC EDITOR: please replace CCFB here and in the above diagram with the IANA assigned RTCP feedback packet type)

Section 6.1 of [RFC4585] requires the RTCP header to be followed by the SSRC of the media source being reported upon. Accordingly, the RTCP header is followed by a report for each SSRC received, followed by the Report Timestamp.

The report for each SSRC received starts with the SSRC of that media source. Then, each sequence number between the begin_seq and end_seq (both inclusive) is represented by a packet metric block of 16-bits that contains the L, ECN, and ATO fields. If an odd number of reports are included, i.e., end_seq - begin_seq is odd then 16 bits of zero padding MUST be added after the last report, to align the RTCP packet to a four (4) bytes boundary. The L, ECN, and ATO fields are as follows:

- L (1 bit): is a boolean to indicate if the packet was received. 0 represents that the packet was not yet received and all the subsequent bits (ECN and ATO) are also set to 0. 1 represent the packet was received and the subsequent bits in the block need to be parsed.

- ECN (2 bits): is the echoed ECN mark of the packet. These are set to 00 if not received, or if ECN is not used.

- Arrival time offset (ATO, 13 bits): is the relative arrival time of the RTP packets at the receiver before this feedback report was generated measured in milliseconds. It is calculated by subtracting the reception timestamp of the RTP packet denoted by this 16bit block and the timestamp (RTS) of this report. If the measured value is greater than 8.189 seconds (the value that would be coded as 0x1FFD), the value 0x1FFE MUST be reported to indicate an over-range positive measurement. If the measurement is unavailable, the value 0x1FFF MUST be reported.

Report Timestamp (RTS, 32 bits): represents the timestamp when this report was generated. The sender of the feedback message decides on the wall-clock. Usually, it should be derived from the same wall-clock that is used for timestamping RTP packets arrival . Consistency in the unit and resolution (10th of millisecond should be good enough) is important here. In addition, the media sender can ask for a specific resolution it wants.

4. Feedback Frequency and Overhead

There is a trade-off between speed and accuracy of reporting, and the overhead of the reports. [I-D.ietf-rmcat-rtp-cc-feedback] discusses this trade-off, and the possible rates of feedback.

It is a general understanding that the congestion control algorithms will work better with more frequent feedback - per packet feedback. However, RTCP bandwidth and transmission rules put some upper limits on how frequently the RTCP feedback messages can be send from the media receiver to the media sender. It has been shown [I-D.ietf-rmcat-rtp-cc-feedback] that in most cases a per frame feedback is a reasonable assumption on how frequent the RTCP feedback messages can be transmitted. The design team also have noted that even if a higher frequency of feedback is desired it is not viable if the feedback messages starts to compete against the media traffic on the feedback path during congestion period. Analyzing the feedback interval requirement [feedback-requirements] it can be seen that the candidate algorithms can perform with a feedback interval range of 50-200ms. A value within this range need to be negotiated at session setup.

5. Design Rationale

The primary function of RTCP Sender Report (SR) / Receiver Report (RR) is to report the reception quality of media. The regular SR / RR reports contain information about observed jitter, fractional packet loss and cumulative packet loss. The original intent of this information was to assist flow and congestion control mechanisms. Even though it is possible to do congestion control based on information provided in the SR/RR reports it is not sufficient to design an efficient congestion control algorithm for interactive real-time communication. An efficient congestion control algorithm requires more fine grain information on per packet (see Section 3) to react to the congestion or to avoid funder congestion on the path.

Codec Control Message for AVPF [RFC5104] defines Temporary Maximum Media Bit Rate (TMMBR) message which conveys a temporary maximum bitrate limitation from the receiver of the media to the sender of the media. Even though it is not designed to replace congestion control, TMMBR has been used as a means to do receiver based congestion control where the session bandwidth is high enough to send frequent TMMBR messages especially with reduced sized reports [RFC5506]. This requires the receiver of the media to analyze the data reception, detect congestion level and recommend a maximum bitrate suitable for current available bandwidth on the path with an assumption that the sender of the media always honors the TMMBR message. This requirement is completely opposite of the sender based congestion control approach. Hence, TMMBR cannot be as a signaling means for a sender based congestion control mechanism. However, TMMBR should be viewed a complimentary signaling mechanism to establish receiver’s current view of acceptable maximum bitrate which a sender based congestion control should honor.

There are number of RTCP eXtended Report (XR) blocks have been defined for reporting of delay, loss and ECN marking. It is possible to combine several XR blocks to report the loss and ECN marking at the cost of overhead and complexity. However, there is no existing RTCP XR block to report packet arrival time.

Considering the issues discussed here it is rational to design a new congestion control feedback signaling mechanism for sender based congestion control algorithm.

6. Acknowledgements

This document is an outcome of RMCAT design team discussion. We would like to thank all participants specially Xiaoquing Zhu, Stefan Holmer, David, Ingemar Johansson and Randell Jesup for their valuable contribution to the discussions and to the document.

7. IANA Considerations

IANA is requested to assign a new value in the "FMT Values for RTPFB Payload Types" registry for the CCFB transport layer feedback packet described in Section 3.1.

8. Security Considerations

There is a risk of causing congestion if an on-path attacker modifies the feedback messages in such a manner to make available bandwidth greater than it is in reality. [More on security consideration TBD.]

9. References

9.1. Normative References

[I-D.ietf-rmcat-rtp-cc-feedback]
Perkins, C., "RTP Control Protocol (RTCP) Feedback for Congestion Control in Interactive Multimedia Conferences", draft-ietf-rmcat-rtp-cc-feedback-03 (work in progress), November 2016.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC3168] Ramakrishnan, K., Floyd, S., and D. Black, "The Addition of Explicit Congestion Notification (ECN) to IP", RFC 3168, DOI 10.17487/RFC3168, September 2001, <https://www.rfc-editor.org/info/rfc3168>.

[RFC3550] Schulzrinne, H., Casner, S., Frederick, R., and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications", STD 64, RFC 3550, DOI 10.17487/RFC3550, July 2003, <https://www.rfc-editor.org/info/rfc3550>.

[RFC3551] Schulzrinne, H. and S. Casner, "RTP Profile for Audio and Video Conferences with Minimal Control", STD 65, RFC 3551, DOI 10.17487/RFC3551, July 2003, <https://www.rfc-editor.org/info/rfc3551>.

[RFC3611] Friedman, T., Ed., Caceres, R., Ed., and A. Clark, Ed., "RTP Control Protocol Extended Reports (RTCP XR)", RFC 3611, DOI 10.17487/RFC3611, November 2003, <https://www.rfc-editor.org/info/rfc3611>.

[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, DOI 10.17487/RFC4585, July 2006, <https://www.rfc-editor.org/info/rfc4585>.

[RFC5124] Ott, J. and E. Carrara, "Extended Secure RTP Profile for Real-time Transport Control Protocol (RTCP)-Based Feedback (RTP/SAVPF)", RFC 5124, DOI 10.17487/RFC5124, February 2008, <https://www.rfc-editor.org/info/rfc5124>.

[RFC5506] Johansson, I. and M. Westerlund, "Support for Reduced-Size Real-Time Transport Control Protocol (RTCP): Opportunities and Consequences", RFC 5506, DOI 10.17487/RFC5506, April 2009, <https://www.rfc-editor.org/info/rfc5506>.

[RFC6679] Westerlund, M., Johansson, I., Perkins, C., O'Hanlon, P., and K. Carlberg, "Explicit Congestion Notification (ECN) for RTP over UDP", RFC 6679, DOI 10.17487/RFC6679, August 2012, <https://www.rfc-editor.org/info/rfc6679>.

9.2. Informative References

[feedback-requirements]
"RMCAT Feedback Requirements", <://www.ietf.org/proceedings/95/slides/slides-95-rmcat-1.pdf>.

[I-D.ietf-rmcat-gcc]
Holmer, S., Lundin, H., Carlucci, G., Cicco, L., and S. Mascolo, "A Google Congestion Control Algorithm for Real-Time Communication", draft-ietf-rmcat-gcc-02 (work in progress), July 2016.

[I-D.ietf-rmcat-nada]
Zhu, X., Pan, R., Ramalho, M., Cruz, S., Jones, P., Fu, J., and S. D'Aronco, "NADA: A Unified Congestion Control Scheme for Real-Time Media", draft-ietf-rmcat-nada-05 (work in progress), September 2017.

[I-D.ietf-rmcat-sbd]
Hayes, D., Ferlin, S., Welzl, M., and K. Hiorth, "Shared Bottleneck Detection for Coupled Congestion Control for RTP Media.", draft-ietf-rmcat-sbd-08 (work in progress), July 2017.

[I-D.ietf-rmcat-scream-cc]
Johansson, I. and Z. Sarker, "Self-Clocked Rate Adaptation for Multimedia", draft-ietf-rmcat-scream-cc-13 (work in progress), October 2017.

[RFC5104] Wenger, S., Chandra, U., Westerlund, M., and B. Burman, "Codec Control Messages in the RTP Audio-Visual Profile with Feedback (AVPF)", RFC 5104, DOI 10.17487/RFC5104, February 2008, <https://www.rfc-editor.org/info/rfc5104>.

Authors' Addresses

Zaheduzzaman Sarker
Ericsson AB
Luleae
Sweden

Phone: +46107173743
Email: zaheduzzaman.sarker@ericsson.com

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

Email: csp@csperkins.org

Varun Singh
Nemu Dialogue Systems Oy
Runeberginkatu 4c A 4
Helsinki 00100
Finland

Email: varun.singh@iki.fi
URI: http://www.callstats.io/

Michael A. Ramalho
Cisco Systems, Inc.
6310 Watercrest Way Unit 203
Lakewood Ranch, FL 34202
USA

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

AVTCORE WG M. Westerlund
Internet-Draft Ericsson
Updates: 3550, 3551 (if approved) C. Perkins
Intended status: Standards Track University of Glasgow
Expires: June 20, 2016 J. Lennox
Vidyo
December 18, 2015

Sending Multiple Types of Media in a Single RTP Session
draft-ietf-avtcore-multi-media-rtp-session-13

Abstract

This document specifies how an RTP session can contain RTP Streams with media from multiple media types such as audio, video, and text. This has been restricted by the RTP Specification, and thus this document updates RFC 3550 and RFC 3551 to enable this behaviour for applications that satisfy the applicability for using multiple media types in a single RTP session.

Status of This Memo

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

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

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

This Internet-Draft will expire on June 20, 2016.

Copyright Notice

Copyright (c) 2015 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (http://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction

The Real-time Transport Protocol [RFC3550] was designed to use separate RTP sessions to transport different types of media. This implies that different transport layer flows are used for different RTP streams. For example, a video conferencing application might send audio and video traffic RTP flows on separate UDP ports. With increased use of network address/port translation, firewalls, and other middleboxes it is, however, becoming difficult to establish multiple transport layer flows between endpoints. Hence, there is pressure to reduce the number of concurrent transport flows used by RTP applications.

This memo updates [RFC3550] and [RFC3551] to allow multiple media types to be sent in a single RTP session in certain cases, thereby reducing the number of transport layer flows that are needed. It makes no changes to RTP behaviour when using multiple RTP streams containing media of the same type (e.g., multiple audio streams or multiple video streams) in a single RTP session. However

[I-D.ietf-avtcore-rtp-multi-stream] provides important clarifications to RTP behaviour in that case.

This memo is structured as follows. Section 2 defines terminology. Section 3 further describes the background to, and motivation for, this memo and Section 4 describes the scenarios where this memo is applicable. Section 5 discusses issues arising from the base RTP and RTCP specification when using multiple types of media in a single RTP session, while Section 6 considers the impact of RTP extensions. We discuss signalling in Section 7. Finally, security considerations are discussed in Section 8.

2. Terminology

The terms Encoded Stream, Endpoint, Media Source, RTP Session, and RTP Stream are used as defined in [RFC7656]. We also define the following terms:

Media Type: The general type of media data used by a real-time application. The media type corresponds to the value used in the <media> field of an SDP m= line. The media types defined at the time of this writing are "audio", "video", "text", "image", "application", and "message". [RFC4566] [RFC6466]

Quality of Service (QoS): Network mechanisms that are intended to ensure that the packets within a flow or with a specific marking are transported with certain properties.

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

3. Background and Motivation

RTP was designed to support multimedia sessions, containing multiple types of media sent simultaneously, by using multiple transport layer flows. The existence of network address translators, firewalls, and other middleboxes complicates this, however, since a mechanism is needed to ensure that all the transport layer flows needed by the application can be established. This has three consequences:

1. increased delay to establish a complete session, since each of the transport layer flows needs to be negotiated and established;

2. increased state and resource consumption in the middleboxes that can lead to unexpected behaviour when middlebox resource limits are reached; and

3. increased risk that a subset of the transport layer flows will fail to be established, thus preventing the application from communicating.

Using fewer transport layer flows can hence be seen to reduce the risk of communication failure, and can lead to improved reliability and performance.

One of the benefits of using multiple transport layer flows is that it makes it easy to use network layer quality of service (QoS) mechanisms to give differentiated performance for different flows. However, we note that many RTP-using application don’t use network QoS features, and don’t expect or desire any separation in network treatment of their media packets, independent of whether they are audio, video or text. When an application has no such desire, it doesn’t need to provide a transport flow structure that simplifies flow based QoS.

Given the above issues, it might seem appropriate for RTP-based applications to send all their RTP streams bundled into one RTP session, running over a single transport layer flow. However, this is prohibited by the RTP specification, because the design of RTP makes certain assumptions that can be incompatible with sending multiple media types in a single RTP session. Specifically, the RTP control protocol (RTCP) timing rules assume that all RTP media flows in a single RTP session have broadly similar RTCP reporting and feedback requirements, which can be problematic when different types of media are multiplexed together. Various RTP extensions also make assumptions about SSRC use and RTCP reporting that are incompatible with sending different media types in a single RTP session.

This memo updates [RFC3550] and [RFC3551] to allow RTP sessions to contain more than one media type in certain circumstances, and gives guidance on when it is safe to send multiple media types in a single RTP session.

4. Applicability

This specification has limited applicability, and anyone intending to use it needs to ensure that their application and use case meets the following criteria:

Equal treatment of media: The use of a single RTP session normally results in similar network treatment for all types of media used within the session. Applications that require significantly different network quality of service (QoS) or RTCP configuration for different RTP streams are better suited by sending those RTP streams in separate RTP session, using separate transport layer flows for each, since that gives greater flexibility. Further guidance on how to provide differential treatment for some media is given in [I-D.ietf-avtcore-multiplex-guidelines] and [RFC7657].

Compatible RTCP Behaviour: The RTCP timing rules enforce a single RTCP reporting interval for all participants in an RTP session. Flows with very different media sending rate or RTCP feedback requirements cannot be multiplexed together, since this leads to either excessive or insufficient RTCP for some flows, depending on how the RTCP session bandwidth, and hence reporting interval, is configured. For example, it is likely infeasible to find a single RTCP configuration that simultaneously suits both a low-rate audio flow with no feedback, and a high-quality video flow with sophisticated RTCP-based feedback. Thus, combining these into a single RTP session is difficult and/or inadvisable.

Signalled Support: The extensions defined in this memo are not compatible with unmodified [RFC3550]-compatible endpoints. Their use requires signalling and mutual agreement by all participants within an RTP session. This requirement can be a problem for signalling solutions that can't negotiate with all participants. For declarative signalling solutions, mandating that the session is using multiple media types in one RTP session can be a way of attempting to ensure that all participants in the RTP session follow the requirement. However, for signalling solutions that lack methods for enforcing that a receiver supports a specific feature, this can still cause issues.

Consistent support for multiparty RTP sessions: If it is desired to send multiple types of media in a multiparty RTP session, then all participants in that session need to support sending multiple type of media in a single RTP session. It is not possible, in the general case, to implement a gateway that can interconnect an endpoint using multiple types of media sent using separate RTP sessions, with one or more endpoints that send multiple types of media in a single RTP session.

One reason for this is that the same SSRC value can safely be used for different streams in multiple RTP sessions, but when collapsed to a single RTP session there is an SSRC collision. This would not be an issue, since SSRC collision detection will resolve the conflict, except that some RTP payload formats and extensions use matching SSRCs to identify related flows, and break when a single RTP session is used.

A middlebox that remaps SSRC values when combining multiple RTP sessions into one also needs to be aware of all possible RTCP packet types that might be used, so that it can remap the SSRC values in those packets. This is impossible to do without restricting the set of RTCP packet types that can be used to those that are known by the middlebox. Such a middlebox might also have difficulty due to differences in configured RTCP bandwidth and other parameters between the RTP sessions.

Finally, the use of a middlebox that translates SSRC values can negatively impact the possibility for loop detection, as SSRC/CSRC can’t be used to detect the loops; instead some other RTP stream or media source identity name space that is common across all interconnect parts is needed.

Ability to operate with limited payload type space: An RTP session has only a single 7-bit payload type space for all its payload type numbers. Some applications might find this space limiting when using different media types and RTP payload formats within a single RTP session.

Avoids incompatible Extensions: Some RTP and RTCP extensions rely on the existence of multiple RTP sessions and relate RTP streams between sessions. Others report on particular media types, and cannot be used with other media types. Applications that send multiple types of media into a single RTP session need to avoid such extensions.

5. Using Multiple Media Types in a Single RTP Session

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

5.1. Allowing Multiple Media Types in an RTP Session

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

> For example, in a teleconference composed of audio and video media encoded separately, each medium SHOULD be carried in a separate RTP session with its own destination transport address.
>
> Separate audio and video streams SHOULD NOT be carried in a single RTP session and demultiplexed based on the payload type or SSRC fields.

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

> For example, in a teleconference composed of audio and video media encoded separately, each medium SHOULD be carried in a separate

> RTP session with its own destination transport address, unless specification [RFCXXXX] is followed and the application meets the applicability constraints.

The second sentence is changed to:

> Separate audio and video media sources SHOULD NOT be carried in a single RTP session, unless the guidelines specified in [RFCXXXX] are followed.

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

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

This specification’s purpose is to override that existing SHALL NOT under certain conditions. Thus this sentence also has to be changed to allow for multiple media type’s payload types in the same session. The sentence containing "SHALL NOT" in the above paragraph is changed to:

> Payload types of different media types SHALL NOT be interleaved or multiplexed within a single RTP session unless [RFCXXXX] is used, and the application conforms to the applicability constraints. Multiple RTP sessions MAY be used in parallel to send multiple media types.

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

5.2. Demultiplexing media types within an RTP session

When receiving packets from a transport layer flow, an endpoint will first separate the RTP and RTCP packets from the non-RTP packets, and pass them to the RTP/RTCP protocol handler. The RTP and RTCP packets are then demultiplexed based on their SSRC into the different RTP streams. For each RTP stream, incoming RTCP packets are processed, and the RTP payload type is used to select the appropriate media decoder. This process remains the same irrespective of whether multiple media types are sent in a single RTP session or not.

As explained below, it is important to note that the RTP payload type is never used to distinguish RTP streams. The RTP packets are demultiplexed into RTP streams based on their SSRC, then the RTP payload type is used to select the correct media decoding pathway for each RTP stream.

5.3. Per-SSRC Media Type Restrictions

An SSRC in an RTP session can change between media formats of the same type, subject to certain restrictions [RFC7160], but MUST NOT change media type during its lifetime. For example, an SSRC can change between different audio formats, but cannot start sending audio then change to sending video. The lifetime of an SSRC ends when an RTCP BYE packet for that SSRC is sent, or when it ceases transmission for long enough that it times out for the other participants in the session.

The main motivation is that a given SSRC has its own RTP timestamp and sequence number spaces. The same way that you can’t send two encoded streams of audio with the same SSRC, you can’t send one encoded audio and one encoded video stream with the same SSRC. Each encoded stream when made into an RTP stream needs to have the sole control over the sequence number and timestamp space. If not, one would not be able to detect packet loss for that particular encoded stream. Nor can one easily determine which clock rate a particular SSRCs timestamp will increase with. For additional arguments why RTP payload type based multiplexing of multiple media sources doesn’t work, see [I-D.ietf-avtcore-multiplex-guidelines].

Within an RTP session where multiple media types have been configured for use, an SSRC can only send one type of media during its lifetime (i.e., it can switch between different audio codecs, since those are both the same type of media, but cannot switch between audio and video). Different SSRCs MUST be used for the different media sources, the same way multiple media sources of the same media type already have to do. The payload type will inform a receiver which media type the SSRC is being used for. Thus the payload type MUST be unique across all of the payload configurations independent of media type that is used in the RTP session.

5.4. RTCP Considerations

When sending multiple types of media that have different rates in a single RTP session, endpoints MUST follow the guidelines for handling RTCP described in Section 7 of [I-D.ietf-avtcore-rtp-multi-stream].

6. Extension Considerations

This section outlines known issues and incompatibilities with RTP and RTCP extensions when multiple media types are used in a single RTP sessions. Future extensions to RTP and RTCP need to consider, and document, any potential incompatibility.

6.1. RTP Retransmission Payload Format

The RTP Retransmission Payload Format [RFC4588] can operate in either SSRC-multiplexed mode or session-multiplex mode.

In SSRC-multiplexed mode, retransmitted RTP packets are sent in the same RTP session as the original packets, but use a different SSRC with the same RTCP SDES CNAME. If each endpoint sends only a single original RTP stream and a single retransmission RTP stream in the session, this is sufficient. If an endpoint sends multiple original and retransmission RTP streams, as would occur when sending multiple media types in a single RTP session, then each original RTP stream and the retransmission RTP stream have to be associated using heuristics. By having retransmission requests outstanding for only one SSRC not yet mapped, a receiver can determine the binding between original and retransmission RTP stream. Another alternative is the use of different RTP payload types, allowing the signalled "apt" (associated payload type) parameter of the RTP retransmission payload format to be used to associate retransmitted and original packets.

Session-multiplexed mode sends the retransmission RTP stream in a separate RTP session to the original RTP stream, but using the same SSRC for each, with association being done by matching SSRCs between the two sessions. This is unaffected by the use of multiple media types in a single RTP session, since each media type will be sent using a different SSRC in the original RTP session, and the same SSRCs can be used in the retransmission session, allowing the streams to be associated. This can be signalled using SDP with the BUNDLE [I-D.ietf-mmusic-sdp-bundle-negotiation] and FID grouping [RFC5888] extensions. These SDP extensions require each "m=" line to only be included in a single FID group, but the RTP retransmission payload format uses FID groups to indicate the m= lines that form an original and retransmission pair. Accordingly, when using the BUNDLE extension to allow multiple media types to be sent in a single RTP session, each original media source (m= line) that is retransmitted needs a corresponding m= line in the retransmission RTP session. In case there are multiple media lines for retransmission, these media lines will form an independent BUNDLE group from the BUNDLE group with the source streams.

An example SDP fragment showing the grouping structures is provided in Figure 1. This example is not legal SDP and only the most important attributes have been left in place. Note that this SDP is not an initial BUNDLE offer. As can be seen there are two bundle groups, one for the source RTP session and one for the retransmissions. Then each of the media sources are grouped with its retransmission flow using FID, resulting in three more groupings.

```
a=group:BUNDLE foo bar fiz
a=group:BUNDLE zoo kelp glo
a=group:FID foo zoo
a=group:FID bar kelp
a=group:FID fiz glo
m=audio 10000 RTP/AVP 0
a=mid:foo
a=rtpmap:0 PCMU/8000
m=video 10000 RTP/AVP 31
a=mid:bar
a=rtpmap:31 H261/90000
m=video 10000 RTP/AVP 31
a=mid:fiz
a=rtpmap:31 H261/90000
m=audio 40000 RTP/AVPF 99
a=rtpmap:99 rtx/90000
a=fmtp:99 apt=0;rtx-time=3000
a=mid:zoo
m=video 40000 RTP/AVPF 100
a=rtpmap:100 rtx/90000
a=fmtp:199 apt=31;rtx-time=3000
a=mid:kelp
m=video 40000 RTP/AVPF 100
a=rtpmap:100 rtx/90000
a=fmtp:199 apt=31;rtx-time=3000
a=mid:glo
```

Figure 1: SDP example of Session Multiplexed RTP Retransmission

6.2. RTP Payload Format for Generic FEC

The RTP Payload Format for Generic Forward Error Correction (FEC) [RFC5109] (and its predecessor [RFC2733]) can either send the FEC stream as a separate RTP stream, or it can send the FEC combined with the original RTP stream as a redundant encoding [RFC2198].

When sending FEC as a separate stream, the RTP Payload Format for generic FEC requires that FEC stream to be sent in a separate RTP session to the original stream, using the same SSRC, with the FEC stream being associated by matching the SSRC between sessions. The

RTP session used for the original streams can include multiple RTP streams, and those RTP streams can use multiple media types. The repair session only needs one RTP Payload type to indicate FEC data, irrespective of the number of FEC streams sent, since the SSRC is used to associate the FEC streams with the original streams. Hence, it is RECOMMENDED that the FEC stream use the "application/ulpfec" media type for [RFC5109], and the "application/parityfec" media type for [RFC2733]. It is legal, but NOT RECOMMENDED, to send FEC streams using media specific payload format names (e.g., using both the "audio/ulpfec" and "video/ulpfec" payload formats for a single RTP session containing both audio and video flows), since this unnecessarily uses up RTP payload type values, and adds no value for demultiplexing since there might be multiple streams of the same media type).

The combination of an original RTP session using multiple media types with an associated generic FEC session can be signalled using SDP with the BUNDLE extension [I-D.ietf-mmusic-sdp-bundle-negotiation]. In this case, the RTP session carrying the FEC streams will be its own BUNDLE group. The m= line for each original stream and the m= line for the corresponding FEC stream are grouped using the SDP grouping framework using either the FEC-FR [RFC5956] grouping or, for backwards compatibility, the FEC [RFC4756] grouping. This is similar to the situation that arises for RTP retransmission with session multiplexing discussed in Section 6.1.

The Source-Specific Media Attributes [RFC5576] specification defines an SDP extension (the "FEC" semantic of the "ssrc-group" attribute) to signal FEC relationships between multiple RTP streams within a single RTP session. This cannot be used with generic FEC, since the FEC repair packets need to have the same SSRC value as the source packets being protected. There was work on an Unequal Layer Protection (ULP) extension to allow it be use FEC RTP streams within the same RTP Session as the source stream [I-D.lennox-payload-ulp-ssrc-mux].

When the FEC is sent as a redundant encoding, the considerations in Section 6.3 apply.

6.3. RTP Payload Format for Redundant Audio

The RTP Payload Format for Redundant Audio [RFC2198] can be used to protect audio streams. It can also be used along with the generic FEC payload format to send original and repair data in the same RTP packets. Both are compatible with RTP sessions containing multiple media types.

This payload format requires each different redundant encoding use a different RTP payload type number. When used with generic FEC in sessions that contain multiple media types, this requires each media type to use a different payload type for the FEC stream. For example, if audio and text are sent in a single RTP session with generic ULP FEC sent as a redundant encoding for each, then payload types need to be assigned for FEC using the audio/ulpfec and text/ulpfec payload formats. If multiple original payload types are used in the session, different redundant payload types need to be allocated for each one. This has potential to rapidly exhaust the available RTP payload type numbers.

7. Signalling

Establishing a single RTP session using multiple media types requires signalling. This signalling has to:

1. ensure that any participant in the RTP session is aware that this is an RTP session with multiple media types;

2. ensure that the payload types in use in the RTP session are using unique values, with no overlap between the media types;

3. ensure RTP session level parameters, for example the RTCP RR and RS bandwidth modifiers, the RTP/AVPF trr-int parameter, transport protocol, RTCP extensions in use, and any security parameters, are consistent across the session; and

4. ensure that RTP and RTCP functions that can be bound to a particular media type are reused where possible, rather than configuring multiple code-points for the same thing.

When using SDP signalling, the BUNDLE extension [I-D.ietf-mmusic-sdp-bundle-negotiation] is used to signal RTP sessions containing multiple media types.

8. Security Considerations

RTP provides a range of strong security mechanisms that can be used to secure sessions [RFC7201], [RFC7202]. The majority of these are independent of the type of media sent in the RTP session; however it is important to check that the security mechanism chosen is compatible with all types of media sent within the session.

Sending multiple media types in a single RTP session will generally require that all use the same security mechanism, whereas media sent using different RTP sessions can be secured in different ways. When different media types have different security requirements, it might be necessary to send them using separate RTP sessions to meet those different requirements. This can have significant costs in terms of resource usage, session set-up time, etc.

9. IANA Considerations

This memo makes no request of IANA.

10. Acknowledgements

The authors would like to thank Christer Holmberg, Gunnar Hellstroem, Charles Eckel, Tolga Asveren, Warren Kumari, and Meral Shirazipour for their feedback on the document.

11. References

11.1. Normative References

[I-D.ietf-avtcore-rtp-multi-stream]
Lennox, J., Westerlund, M., Wu, Q., and C. Perkins, "Sending Multiple RTP Streams in a Single RTP Session", draft-ietf-avtcore-rtp-multi-stream-11 (work in progress), December 2015.

[I-D.ietf-mmusic-sdp-bundle-negotiation]
Holmberg, C., Alvestrand, H., and C. Jennings, "Negotiating Media Multiplexing Using the Session Description Protocol (SDP)", draft-ietf-mmusic-sdp-bundle-negotiation-23 (work in progress), July 2015.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <http://www.rfc-editor.org/info/rfc2119>.

[RFC3550] Schulzrinne, H., Casner, S., Frederick, R., and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications", STD 64, RFC 3550, DOI 10.17487/RFC3550, July 2003, <http://www.rfc-editor.org/info/rfc3550>.

[RFC3551] Schulzrinne, H. and S. Casner, "RTP Profile for Audio and Video Conferences with Minimal Control", STD 65, RFC 3551, DOI 10.17487/RFC3551, July 2003, <http://www.rfc-editor.org/info/rfc3551>.

11.2. Informative References

[I-D.ietf-avtcore-multiplex-guidelines]
Westerlund, M., Perkins, C., and H. Alvestrand, "Guidelines for using the Multiplexing Features of RTP to Support Multiple Media Streams", draft-ietf-avtcore-multiplex-guidelines-03 (work in progress), October 2014.

[I-D.lennox-payload-ulp-ssrc-mux]
Lennox, J., "Supporting Source-Multiplexing of the Real-Time Transport Protocol (RTP) Payload for Generic Forward Error Correction", draft-lennox-payload-ulp-ssrc-mux-00 (work in progress), February 2013.

[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, DOI 10.17487/RFC2198, September 1997, <http://www.rfc-editor.org/info/rfc2198>.

[RFC2733] Rosenberg, J. and H. Schulzrinne, "An RTP Payload Format for Generic Forward Error Correction", RFC 2733, DOI 10.17487/RFC2733, December 1999, <http://www.rfc-editor.org/info/rfc2733>.

[RFC4566] Handley, M., Jacobson, V., and C. Perkins, "SDP: Session Description Protocol", RFC 4566, DOI 10.17487/RFC4566, July 2006, <http://www.rfc-editor.org/info/rfc4566>.

[RFC4588] Rey, J., Leon, D., Miyazaki, A., Varsa, V., and R. Hakenberg, "RTP Retransmission Payload Format", RFC 4588, DOI 10.17487/RFC4588, July 2006, <http://www.rfc-editor.org/info/rfc4588>.

[RFC4756] Li, A., "Forward Error Correction Grouping Semantics in Session Description Protocol", RFC 4756, DOI 10.17487/RFC4756, November 2006, <http://www.rfc-editor.org/info/rfc4756>.

[RFC5109] Li, A., Ed., "RTP Payload Format for Generic Forward Error Correction", RFC 5109, DOI 10.17487/RFC5109, December 2007, <http://www.rfc-editor.org/info/rfc5109>.

[RFC5576] Lennox, J., Ott, J., and T. Schierl, "Source-Specific Media Attributes in the Session Description Protocol (SDP)", RFC 5576, DOI 10.17487/RFC5576, June 2009, <http://www.rfc-editor.org/info/rfc5576>.

[RFC5888] Camarillo, G. and H. Schulzrinne, "The Session Description Protocol (SDP) Grouping Framework", RFC 5888, DOI 10.17487/RFC5888, June 2010, <http://www.rfc-editor.org/info/rfc5888>.

[RFC5956] Begen, A., "Forward Error Correction Grouping Semantics in the Session Description Protocol", RFC 5956, DOI 10.17487/RFC5956, September 2010, <http://www.rfc-editor.org/info/rfc5956>.

[RFC6466] Salgueiro, G., "IANA Registration of the 'image' Media Type for the Session Description Protocol (SDP)", RFC 6466, DOI 10.17487/RFC6466, December 2011, <http://www.rfc-editor.org/info/rfc6466>.

[RFC7160] Petit-Huguenin, M. and G. Zorn, Ed., "Support for Multiple Clock Rates in an RTP Session", RFC 7160, DOI 10.17487/RFC7160, April 2014, <http://www.rfc-editor.org/info/rfc7160>.

[RFC7201] Westerlund, M. and C. Perkins, "Options for Securing RTP Sessions", RFC 7201, DOI 10.17487/RFC7201, April 2014, <http://www.rfc-editor.org/info/rfc7201>.

[RFC7202] Perkins, C. and M. Westerlund, "Securing the RTP Framework: Why RTP Does Not Mandate a Single Media Security Solution", RFC 7202, DOI 10.17487/RFC7202, April 2014, <http://www.rfc-editor.org/info/rfc7202>.

[RFC7656] Lennox, J., Gross, K., Nandakumar, S., Salgueiro, G., and B. Burman, Ed., "A Taxonomy of Semantics and Mechanisms for Real-Time Transport Protocol (RTP) Sources", RFC 7656, DOI 10.17487/RFC7656, November 2015, <http://www.rfc-editor.org/info/rfc7656>.

[RFC7657] Black, D., Ed. and P. Jones, "Differentiated Services (Diffserv) and Real-Time Communication", RFC 7657, DOI 10.17487/RFC7657, November 2015, <http://www.rfc-editor.org/info/rfc7657>.

Authors' Addresses

Magnus Westerlund
Ericsson
Farogatan 6
SE-164 80 Kista
Sweden

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

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

Email: csp@csperkins.org

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

Email: jonathan@vidyo.com

Network Working Group
Internet-Draft
Intended status: Informational
Expires: June 17, 2019

M. Westerlund
B. Burman
Ericsson
C. Perkins
University of Glasgow
H. Alvestrand
Google
R. Even
Huawei

December 14, 2018

Guidelines for using the Multiplexing Features of RTP to Support Multiple Media Streams
draft-ietf-avtcore-multiplex-guidelines-08

Abstract

The Real-time Transport Protocol (RTP) is a flexible protocol that can be used in a wide range of applications, networks, and system topologies. That flexibility makes for wide applicability, but can complicate the application design process. One particular design question that has received much attention is how to support multiple media streams in RTP. This memo discusses the available options and design trade-offs, and provides guidelines on how to use the multiplexing features of RTP to support multiple media streams.

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 https://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 June 17, 2019.

Copyright Notice

Copyright (c) 2018 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (https://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction

The Real-time Transport Protocol (RTP) [RFC3550] is a commonly used protocol for real-time media transport. It is a protocol that provides great flexibility and can support a large set of different applications. RTP was from the beginning designed for multiple participants in a communication session. It supports many topology paradigms and usages, as defined in [RFC7667]. RTP has several multiplexing points designed for different purposes. These enable support of multiple RTP streams and switching between different encoding or packetization of the media. By using multiple RTP sessions, sets of RTP streams can be structured for efficient processing or identification. Thus, the question for any RTP application designer is how to best use the RTP session, the RTP stream identifier (SSRC), and the RTP payload type to meet the application's needs.

There have been increased interest in more advanced usage of RTP. For example, multiple RTP streams can be used when a single endpoint has multiple media sources (like multiple cameras or microphones) that need to be sent simultaneously. Consequently, questions are raised regarding the most appropriate RTP usage. The limitations in some implementations, RTP/RTCP extensions, and signalling has also been exposed. The authors also hope that clarification on the usefulness of some functionalities in RTP will result in more complete implementations in the future.

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

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

2. Definitions

2.1. Terminology

The definitions in Section 3 of [RFC3550] are referenced normatively.

The taxonomy defined in [RFC7656] is referenced normatively.

The following terms and abbreviations are used in this document:

Multiparty: A communication situation including multiple endpoints. In this document, it will be used to refer to situations where more than two endpoints communicate.

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

RTP Receiver: An Endpoint or Middlebox receiving RTP streams and RTCP messages. It uses at least one SSRC to send RTCP messages. An RTP Receiver may also be an RTP sender.

RTP Sender: An Endpoint sending one or more RTP streams, but also sending RTCP messages.

RTP Session Group: One or more RTP sessions that are used together to perform some function. Examples are multiple RTP sessions used to carry different layers of a layered encoding. In an RTP Session Group, CNAMEs are assumed to be valid across all RTP sessions, and designate synchronisation contexts that can cross RTP sessions; i.e. SSRCs that map to a common CNAME can be assumed to have RTCP SR timing information derived from a common clock such that they can be synchronised for playout.

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

Note: The above definitions of RTP Receiver and RTP sender are intended to be consistent with the usage in [RFC3550].

2.2. Subjects Out of Scope

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

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

3. RTP Multiplexing Overview

3.1. Reasons for Multiplexing and Grouping RTP Streams

There are several reasons why an endpoint might choose to send multiple media streams. In the below discussion, please keep in mind that the reasons for having multiple RTP streams vary and include but are not limited to the following:

- o Multiple media sources
- o Multiple RTP streams might be needed to represent one media source (for instance when using layered encodings)
- o A retransmission stream might repeat some parts of the content of another RTP stream
- o An FEC stream might provide material that can be used to repair another RTP stream
- o Alternative encodings, for instance using different codecs for the same audio stream

- o Alternative formats, for instance multiple resolutions of the same video stream

For each of these reasons, it is necessary to decide if each additional RTP stream is sent within the same RTP session as the other RTP streams, or if it is necessary to use additional RTP sessions to group the RTP streams. The choice suitable for one reason, might not be the choice suitable for another reason. The clearest understanding is associated with multiplexing multiple media sources of the same media type. However, all reasons warrant discussion and clarification on how to deal with them. As the discussion below will show, in reality we cannot choose a single one of SSRC or RTP session multiplexing solutions. To utilise RTP well and as efficiently as possible, both are needed. The real issue is finding the right guidance on when to create additional RTP sessions and when additional RTP streams in the same RTP session is the right choice.

3.2. RTP Multiplexing Points

This section describes the multiplexing points present in the RTP protocol that can be used to distinguish RTP streams and groups of RTP streams. Figure 1 outlines the process of demultiplexing incoming RTP streams:

```
                           |
                           | packets
           +--             v
           |        +------------+
           |        |   Socket   |   Transport Protocol Demultiplexing
           |        +------------+
           |            ||  ||
      RTP  |       RTP/ ||  |+-----> SCTP ( ...and any other protocols)
   Session |       RTCP ||  +------> STUN (multiplexed using same port)
           +--          ||
           +--          ||
           |      (split by SSRC)
           |      ||    ||    ||
           |      ||    ||    ||
     RTP   |     +--+  +--+  +--+
   Streams |     |PB|  |PB|  |PB| Jitter buffer, process RTCP, etc.
           |     +--+  +--+  +--+
           +--      |   |      |
           (select decoder based on PT)
           +--      |  /       |
           |        +----+     |
           |         /   |     |
   Payload |     +---+ +---+ +---+
   Formats |     |Dec| |Dec| |Dec| Decoders
           |     +---+ +---+ +---+
           +--
```

Figure 1: RTP Demultiplexing Process

3.2.1. RTP Session

An RTP Session is the highest semantic layer in the RTP protocol, and represents an association between a group of communicating endpoints. RTP does not contain a session identifier, yet RTP sessions must be possible to separate both across different endpoints and within a single endpoint.

For RTP session separation across endpoints, the set of participants that form an RTP session is defined as those that share a single synchronisation source space [RFC3550]. That is, if a group of participants are each aware of the synchronisation source identifiers belonging to the other participants, then those participants are in a single RTP session. A participant can become aware of a synchronisation source identifier by receiving an RTP packet containing it in the SSRC field or CSRC list, by receiving an RTCP packet mentioning it in an SSRC field, or through signalling (e.g., the Session Description Protocol (SDP) [RFC4566] "a=ssrc:" attribute

[RFC5576]). Thus, the scope of an RTP session is determined by the participants' network interconnection topology, in combination with RTP and RTCP forwarding strategies deployed by the endpoints and any middleboxes, and by the signalling.

For RTP session separation within a single endpoint, RTP relies on the underlying transport layer, and on the signalling to identify RTP sessions in a manner that is meaningful to the application. A single endpoint can have one or more transport flows for the same RTP session, and a single RTP session can span multiple transport layer flows. The signalling layer might give RTP sessions an explicit identifier, or the identification might be implicit based on the addresses and ports used. Accordingly, a single RTP session can have multiple associated identifiers, explicit and implicit, belonging to different contexts. For example, when running RTP on top of UDP/IP, an endpoint can identify and delimit an RTP session from other RTP sessions by receiving the multiple UDP flows used as identified based on their UDP source and destination IP addresses and UDP port numbers. Another example is SDP media descriptions (the "m=" line and the following associated lines) signals the transport flow and RTP session configuration for the endpoints part of the RTP session. SDP grouping framework [RFC5888] allows labeling of the media descriptions, for example used so that RTP Session Groups can be created. With Negotiating Media Multiplexing Using the Session Description Protocol (SDP)[I-D.ietf-mmusic-sdp-bundle-negotiation], multiple media descriptions where each represents the RTP streams sent or received for a media source are part of a common RTP session.

The RTP protocol makes no normative statements about the relationship between different RTP sessions, however the applications that use more than one RTP session will have some higher layer understanding of the relationship between the sessions they create.

3.2.2. Synchronisation Source (SSRC)

A synchronisation source (SSRC) identifies an source of an RTP stream or an RTP receiver when sending RTCP. Every endpoint has at least one SSRC identifier, even if it does not send RTP packets. RTP endpoints that are only RTP receivers still send RTCP and use their SSRC identifiers in the RTCP packets they send. An endpoint can have multiple SSRC identifiers if it sends multiple RTP streams. Endpoints that are both RTP sender and RTP receiver use the same SSRC in both roles.

The SSRC is a 32-bit identifier. It is present in every RTP and RTCP packet header, and in the payload of some RTCP packet types. It can also be present in SDP signalling. Unless pre-signalled, e.g. using the SDP "a=ssrc:" attribute [RFC5576], the SSRC is chosen at random.

It is not dependent on the network address of the endpoint, and is intended to be unique within an RTP session. SSRC collisions can occur, and are handled as specified in [RFC3550] and [RFC5576], resulting in the SSRC of the colliding RTP streams or receivers changing. An endpoint that changes its network transport address during a session have to choose a new SSRC identifier to avoid being interpreted as looped source, unless the transport layer mechanism, e.g ICE [RFC8445], handles such changes.

SSRC identifiers that belong to the same synchronisation context (i.e., that represent RTP streams that can be synchronised using information in RTCP SR packets) use identical CNAME chunks in corresponding RTCP SDES packets. SDP signalling can also be used to provide explicit SSRC grouping [RFC5576].

In some cases, the same SSRC identifier value is used to relate streams in two different RTP sessions, such as in RTP retransmission [RFC4588]. This is to be avoided since there is no guarantee that SSRC values are unique across RTP sessions. For the RTP retransmission [RFC4588] case it is recommended to use explicit binding of the source RTP stream and the redundancy stream, e.g. using the RepairedRtpStreamId RTCP SDES item [I-D.ietf-avtext-rid].

Note that RTP sequence number and RTP timestamp are scoped by the SSRC and thus specific per RTP stream.

Different types of entities use a SSRC to identify themselves, as follows:

A real media source: Uses the SSRC to identify a "physical" media source.

A conceptual media source: Uses the SSRC to identify the result of applying some filtering function in a network node, for example a filtering function in an RTP mixer that provides the most active speaker based on some criteria, or a mix representing a set of other sources.

An RTP receiver: Uses the SSRC to identify itself as the source of its RTCP reports.

Note that an endpoint that generates more than one media type, e.g. a conference participant sending both audio and video, need not (and should not) use the same SSRC value across RTP sessions. RTCP compound packets containing the CNAME SDES item is the designated method to bind an SSRC to a CNAME, effectively cross-correlating SSRCs within and between RTP Sessions as coming from the same endpoint. The main property attributed to SSRCs associated with the same CNAME is that they are from a particular synchronisation context and can be synchronised at playback.

An RTP receiver receiving a previously unseen SSRC value will interpret it as a new source. It might in fact be a previously existing source that had to change SSRC number due to an SSRC conflict. However, the originator of the previous SSRC ought to have ended the conflicting source by sending an RTCP BYE for it prior to starting to send with the new SSRC, so the new SSRC is anyway effectively a new source.

3.2.3. Contributing Source (CSRC)

The Contributing Source (CSRC) is not a separate identifier. Rather an SSRC identifier is listed as a CSRC in the RTP header of a packet generated by an RTP mixer, if the corresponding SSRC was in the header of one of the packets that contributed to the mix.

It is not possible, in general, to extract media represented by an individual CSRC since it is typically the result of a media mixing (merge) operation by an RTP mixer on the individual media streams corresponding to the CSRC identifiers. The exception is the case when only a single CSRC is indicated as this represent forwarding of an RTP stream, possibly modified. The RTP header extension for Mixer-to-Client Audio Level Indication [RFC6465] expands on the receiver's information about a packet with a CSRC list. Due to these restrictions, CSRC will not be considered a fully qualified multiplexing point and will be disregarded in the rest of this document.

3.2.4. RTP Payload Type

Each RTP stream utilises one or more RTP payload formats. An RTP payload format describes how the output of a particular media codec is framed and encoded into RTP packets. The payload format used is identified by the payload type (PT) field in the RTP packet header. The combination of SSRC and PT therefore identifies a specific RTP stream encoding format. The format definition can be taken from [RFC3551] for statically allocated payload types, but ought to be explicitly defined in signalling, such as SDP, both for static and dynamic payload types. The term "format" here includes whatever can be described by out-of-band signalling means. In SDP, the term "format" includes media type, RTP timestamp sampling rate, codec, codec configuration, payload format configurations, and various robustness mechanisms such as redundant encodings [RFC2198].

The RTP payload type is scoped by the sending endpoint within an RTP session. PT has the same meaning across all RTP streams in an RTP session. All SSRCs sent from a single endpoint share the same payload type definitions. The RTP payload type is designed such that only a single payload type is valid at any time instant in the RTP stream's timestamp time line, effectively time-multiplexing different payload types if any change occurs. The payload type used can change on a per-packet basis for an SSRC, for example a speech codec making use of generic comfort noise [RFC3389]. If there is a true need to send multiple payload types for the same SSRC that are valid for the same instant, then redundant encodings [RFC2198] can be used. Several additional constraints than the ones mentioned above need to be met to enable this use, one of which is that the combined payload sizes of the different payload types ought not exceed the transport MTU. If it is acceptable to send multiple formats of the same media source as separate RTP streams (with separate SSRC), simulcast [I-D.ietf-mmusic-sdp-simulcast] can be used.

Other aspects of RTP payload format use are described in How to Write an RTP Payload Format [RFC8088].

The payload type is not a multiplexing point at the RTP layer (see Appendix A for a detailed discussion of why using the payload type as an RTP multiplexing point does not work). The RTP payload type is, however, used to determine how to consume and decode an RTP stream. The RTP payload type number is sometimes used to associate an RTP stream with the signalling; this is not recommended since a specific payload type value can be used in multiple bundled "m=" sections [I-D.ietf-mmusic-sdp-bundle-negotiation]. This association is only possible if unique RTP payload type numbers are used in each context.

3.3. Issues Related to RTP Topologies

The impact of how RTP multiplexing is performed will in general vary with how the RTP session participants are interconnected, described by RTP Topology [RFC7667].

Even the most basic use case, denoted Topo-Point-to-Point in [RFC7667], raises a number of considerations that are discussed in detail in following sections. They range over such aspects as:

- Does my communication peer support RTP as defined with multiple SSRCs per RTP session?

- Do I need network differentiation in form of QoS?

- Can the application more easily process and handle the media streams if they are in different RTP sessions?

- Do I need to use additional RTP streams for RTP retransmission or FEC?

For some point to multi-point topologies (e.g. Topo-ASM and Topo-SSM in [RFC7667]), multicast is used to interconnect the session participants. Special considerations (documented in Section 4.2.3) are then needed as multicast is a one-to-many distribution system.

Sometimes an RTP communication can end up in a situation when the communicating peers are not compatible for various reasons:

- No common media codec for a media type thus requiring transcoding.
- Different support for multiple RTP streams and RTP sessions.
- Usage of different media transport protocols, i.e., RTP or other.
- Usage of different transport protocols, e.g., UDP, DCCP, or TCP.
- Different security solutions, e.g., IPsec, TLS, DTLS, or SRTP with different keying mechanisms.

In many situations this is resolved by the inclusion of a translator between the two peers, as described by Topo-PtP-Translator in [RFC7667]. The translator's main purpose is to make the peers look compatible to each other. There can also be other reasons than compatibility to insert a translator in the form of a middlebox or gateway, for example a need to monitor the RTP streams. If the stream transport characteristics are changed by the translator, appropriate media handling can require thorough understanding of the application logic, specifically any congestion control or media adaptation.

The point to point topology can contain one to many RTP sessions with one to many media sources per session, each having one or more RTP streams per media source.

3.4. Issues Related to RTP and RTCP Protocol

Using multiple RTP streams is a well-supported feature of RTP. However, for most implementers or people writing RTP/RTCP applications or extensions attempting to apply multiple streams, it can be unclear when it is most appropriate to add an additional RTP stream in an existing RTP session and when it is better to use multiple RTP sessions. This section discusses the various considerations needed.

3.4.1. The RTP Specification

RFC 3550 contains some recommendations and a bullet list with 5 arguments for different aspects of RTP multiplexing. Let's review Section 5.2 of [RFC3550], reproduced below:

"For efficient protocol processing, the number of multiplexing points should be minimised, as described in the integrated layer processing design principle [ALF]. In RTP, multiplexing is provided by the destination transport address (network address and port number) which is different for each RTP session. For example, in a teleconference composed of audio and video media encoded separately, each medium SHOULD be carried in a separate RTP session with its own destination transport address.

Separate audio and video streams SHOULD NOT be carried in a single RTP session and demultiplexed based on the payload type or SSRC fields. Interleaving packets with different RTP media types but using the same SSRC would introduce several problems:

1. If, say, two audio streams shared the same RTP session and the same SSRC value, and one were to change encodings and thus acquire a different RTP payload type, there would be no general way of identifying which stream had changed encodings.

2. An SSRC is defined to identify a single timing and sequence number space. Interleaving multiple payload types would require different timing spaces if the media clock rates differ and would require different sequence number spaces to tell which payload type suffered packet loss.

3. The RTCP sender and receiver reports (see Section 6.4) can only describe one timing and sequence number space per SSRC and do not carry a payload type field.

4. An RTP mixer would not be able to combine interleaved streams of incompatible media into one stream.

5. Carrying multiple media in one RTP session precludes: the use of different network paths or network resource allocations if appropriate; reception of a subset of the media if desired, for example just audio if video would exceed the available bandwidth; and receiver implementations that use separate processes for the different media, whereas using separate RTP sessions permits either single- or multiple-process implementations.

Using a different SSRC for each medium but sending them in the same RTP session would avoid the first three problems but not the last two.

On the other hand, multiplexing multiple related sources of the same medium in one RTP session using different SSRC values is the norm for multicast sessions. The problems listed above don't apply: an RTP mixer can combine multiple audio sources, for example, and the same treatment is applicable for all of them. It might also be appropriate to multiplex streams of the same medium using different SSRC values in other scenarios where the last two problems do not apply."

Let's consider one argument at a time. The first argument is for using different SSRC for each individual RTP stream, which is fundamental to RTP operation.

The second argument is advocating against demultiplexing RTP streams within a session based on their RTP payload type numbers, which still stands as can been seen by the extensive list of issues found in Appendix A.

The third argument is yet another argument against payload type multiplexing.

The fourth argument is against multiplexing RTP packets that require different handling into the same session. As we saw in the discussion of RTP mixers, the RTP mixer must embed application logic to handle streams anyway; the separation of streams according to stream type is just another piece of application logic, which might or might not be appropriate for a particular application. One type of application that can mix different media sources "blindly" is the audio-only "telephone" bridge; most other types of applications need application-specific logic to perform the mix correctly.

The fifth argument discusses network aspects that we will discuss more below in Section 4.2. It also goes into aspects of implementation, like Split Component Terminal (see Section 3.10 of [RFC7667]) endpoints where different processes or inter-connected devices handle different aspects of the whole multi-media session.

A summary of RFC 3550's view on multiplexing is to use unique SSRCs for anything that is its own media/packet stream, and to use different RTP sessions for media streams that don't share a media type. This document supports the first point; it is very valid. The latter needs further discussion, as imposing a single solution on all usages of RTP is inappropriate. Multiple Media Types in an RTP Session specification [I-D.ietf-avtcore-multi-media-rtp-session] provides a detailed analysis of the potential issues in having multiple media types in the same RTP session. This document provides a wider scope for an RTP session and considers multiple media types in one RTP session as a possible choice for the RTP application designer.

3.4.2. Multiple SSRCs in a Session

Using multiple SSRCs at one endpoint in an RTP session requires resolving some unclear aspects of the RTP specification. These could potentially lead to some interoperability issues as well as some potential significant inefficiencies, as further discussed in "RTP Considerations for Endpoints Sending Multiple Media Streams" [RFC8108]. An RTP application designer should consider these issues and the possible application impact from lack of appropriate RTP handling or optimization in the peer endpoints.

Using multiple RTP sessions can potentially mitigate application issues caused by multiple SSRCs in an RTP session.

3.4.3. Binding Related Sources

A common problem in a number of various RTP extensions has been how to bind related RTP streams together. This issue is common to both using additional SSRCs and multiple RTP sessions.

The solutions can be divided into a few groups:

- RTP/RTCP based
- Signalling based (SDP)
- Grouping related RTP sessions
- Grouping SSRCs within an RTP session

Most solutions are explicit, but some implicit methods have also been applied to the problem.

The SDP-based signalling solutions are:

SDP Media Description Grouping: The SDP Grouping Framework [RFC5888] uses various semantics to group any number of media descriptions. These has previously been considered primarily as grouping RTP sessions, [I-D.ietf-mmusic-sdp-bundle-negotiation] groups multiple media descriptions as a single RTP session.

SDP SSRC grouping: Source-Specific Media Attributes in SDP [RFC5576] includes a solution for grouping SSRCs the same way as the Grouping framework groups Media Descriptions.

This supports a lot of use cases. All these solutions have shortcomings in cases where the session's dynamic properties are such that it is difficult or resource consuming to keep the list of related SSRCs up to date.

An RTP/RTCP-based solution is to use the RTCP SDES CNAME to bind the RTP streams to an endpoint or synchronization context. For applications with a single RTP stream per type (Media, Source or Redundancy) this is sufficient independent if one or more RTP sessions are used. However, some applications choose not to use it because of perceived complexity or a desire not to implement RTCP and instead use the same SSRC value to bind related RTP streams across multiple RTP sessions. RTP Retransmission [RFC4588] in multiple RTP session mode and Generic FEC [RFC5109] both use this method. This method may work but might have some downsides in RTP sessions with many participating SSRCs. When an SSRC collision occurs, this will force one to change SSRC in all RTP sessions and thus resynchronize all of them instead of only the single media stream having the collision. Therefore, it is not recommended to use identical SSRC values to relate RTP streams.

Another solution to bind SSRCs is an implicit method used by RTP Retransmission [RFC4588] when doing retransmissions in the same RTP session as the source RTP stream. The receiver missing a packet issues an RTP retransmission request, and then awaits a new SSRC carrying the RTP retransmission payload and where that SSRC is from the same CNAME. This limits a requester to having only one outstanding request on any new source SSRCs per endpoint.

RTP Payload Format Restrictions [I-D.ietf-mmusic-rid] provides an RTP/RTCP based mechanism to unambiguously identify the RTP streams within an RTP session and restrict the streams' payload format parameters in a codec-agnostic way beyond what is provided with the regular Payload Types. The mapping is done by specifying an "a=rid" value in the SDP offer/answer signalling and having the corresponding "rtp-stream-id" value as an SDES item and an RTP header extension. The RID solution also includes a solution for binding redundancy RTP streams to their original source RTP streams, given that those use RID identifiers.

It can be noted that Section 8.3 of the RTP Specification [RFC3550] recommends using a single SSRC space across all RTP sessions for layered coding. Based on the experience so far however, we recommend to use a solution doing explicit binding between the RTP streams so what the used SSRC values are do not matter. That way solutions using multiple RTP streams in a single RTP session and multiple RTP sessions uses the same solution.

3.4.4. Forward Error Correction

There exist a number of Forward Error Correction (FEC) based schemes for how to reduce the packet loss of the original streams. Most of the FEC schemes will protect a single source flow. The protection is achieved by transmitting a certain amount of redundant information that is encoded such that it can repair one or more packet losses over the set of packets the redundant information protects. This sequence of redundant information also needs to be transmitted as its own media stream, or in some cases, instead of the original media stream. Thus, many of these schemes create a need for binding related flows as discussed above. Looking at the history of these schemes, there are schemes using multiple SSRCs and schemes using multiple RTP sessions, and some schemes that support both modes of operation.

Using multiple RTP sessions supports the case where some set of receivers might not be able to utilise the FEC information. By placing it in a separate RTP session and if separating RTP sessions on transport level, FEC can easily be ignored already on transport level.

In usages involving multicast, having the FEC information on its own multicast group allows for similar flexibility. This is especially useful when receivers see very heterogeneous packet loss rates. Those receivers that are not seeing packet loss don’t need to join the multicast group with the FEC data, and so avoid the overhead of receiving unnecessary FEC packets, for example.

4. Considerations for RTP Multiplexing

4.1. Interworking Considerations

There are several different kinds of interworking, and this section discusses two; interworking between different applications including the implications of potentially different RTP multiplexing point choices and limitations that have to be considered when working with some legacy applications.

4.1.1. Application Interworking

It is not uncommon that applications or services of similar but not identical usage, especially the ones intended for interactive communication, encounter a situation where one want to interconnect two or more of these applications.

In these cases, one ends up in a situation where one might use a gateway to interconnect applications. This gateway must then either change the multiplexing structure or adhere to the respective limitations in each application.

There are two fundamental approaches to building a gateway: using an RTP Translator interworking (RTP bridging), where the gateway acts as an RTP Translator, with the two applications being members of the same RTP session; or Gateway Interworking with RTP termination, where there are independent RTP sessions running from each interconnected application to the gateway.

4.1.2. RTP Translator Interworking

From an RTP perspective, the RTP Translator approach could work if all the applications are using the same codecs with the same payload types, have made the same multiplexing choices, and have the same capabilities in number of simultaneous RTP streams combined with the same set of RTP/RTCP extensions being supported. Unfortunately, this might not always be true.

When a gateway is implemented via an RTP Translator, an important consideration is if the two applications being interconnected need to use the same approach to multiplexing. If one side is using RTP session multiplexing and the other is using SSRC multiplexing with bundle, it is possible for the RTP translator to map the RTP streams between both sides if the order of SDP "m=" lines between both sides are the same. There are also challenges with SSRC collision handling since there may be a collision on the SSRC multiplexing side but the RTP session multiplexing side will not be aware of any collision unless SSRC translation is applied on the RTP translator. Furthermore, if one of the applications is capable of working in several modes (such as being able to use additional RTP streams in one RTP session or multiple RTP sessions at will), and the other one is not, successful interconnection depends on locking the more flexible application into the operating mode where interconnection can be successful, even if no participants are using the less flexible application when the RTP sessions are being created.

4.1.3. Gateway Interworking

When one terminates RTP sessions at the gateway, there are certain tasks that the gateway has to carry out:

- o Generating appropriate RTCP reports for all RTP streams (possibly based on incoming RTCP reports), originating from SSRCs controlled by the gateway.

- o Handling SSRC collision resolution in each application's RTP sessions.

- o Signalling, choosing and policing appropriate bit-rates for each session.

For applications that uses any security mechanism, e.g., in the form of SRTP, the gateway needs to be able to decrypt incoming packets and re-encrypt them in the other application's security context. This is necessary even if all that's needed is a simple remapping of SSRC numbers. If this is done, the gateway also needs to be a member of the security contexts of both sides, of course.

Other tasks a gateway might need to apply include transcoding (for incompatible codec types), media-level adaptations that cannot be solved through media negotiation (such as rescaling for incompatible video size requirements), suppression of content that is known not to be handled in the destination application, or the addition or removal of redundancy coding or scalability layers to fit the needs of the destination domain.

From the above, we can see that the gateway needs to have an intimate knowledge of the application requirements; a gateway is by its nature application specific, not a commodity product.

This fact reveals the potential for these gateways to block application evolution by blocking RTP and RTCP extensions that the applications have been extended with but that are unknown to the gateway.

If one uses security functions, like SRTP, and as can be seen from above, they incur both additional risk due to the requirement to have the gateway in the security association between the endpoints (unless the gateway is on the transport level), and additional complexities in form of the decrypt-encrypt cycles needed for each forwarded packet. SRTP, due to its keying structure, also requires that each RTP session needs different master keys, as use of the same key in two RTP sessions can for some ciphers result in two-time pads that completely breaks the confidentiality of the packets.

4.1.4. Multiple SSRC Legacy Considerations

Historically, the most common RTP use cases have been point to point Voice over IP (VoIP) or streaming applications, commonly with no more than one media source per endpoint and media type (typically audio or video). Even in conferencing applications, especially voice-only, the conference focus or bridge has provided a single stream with a mix of the other participants to each participant. It is also common to have individual RTP sessions between each endpoint and the RTP mixer, meaning that the mixer functions as an RTP-terminating gateway.

When establishing RTP sessions that can contain endpoints that aren't updated to handle multiple streams following these recommendations, a particular application can have issues with multiple SSRCs within a single session. These issues include:

1. Need to handle more than one stream simultaneously rather than replacing an already existing stream with a new one.

2. Be capable of decoding multiple streams simultaneously.

3. Be capable of rendering multiple streams simultaneously.

This indicates that gateways attempting to interconnect to this class of devices has to make sure that only one RTP stream of each type gets delivered to the endpoint if it's expecting only one, and that the multiplexing format is what the device expects. It is highly unlikely that RTP translator-based interworking can be made to function successfully in such a context.

4.2. Network Considerations

The RTP multiplexing choice has impact on network level mechanisms that need to be considered by the implementer.

4.2.1. Quality of Service

When it comes to Quality of Service mechanisms, they are either flow based or packet marking based. RSVP [RFC2205] is an example of a flow based mechanism, while Diff-Serv [RFC2474] is an example of a packet marking based one. For a packet marking based scheme, the method of multiplexing will not affect the possibility to use QoS.

However, for a flow based scheme there is a clear difference between the multiplexing methods. Additional SSRC will result in all RTP streams being part of the same 5-tuple (protocol, source address, destination address, source port, destination port) which is the most common selector for flow based QoS.

It must also be noted that packet marking based QoS mechanisms can have limitations. A general observation is that different Differentiated Services Code Points (DSCP) can be assigned to different packets within a flow as well as within an RTP stream. However, care must be taken when considering which forwarding behaviours that are applied on path due to these DSCPs. In some cases the forwarding behaviour can result in packet reordering. For more discussion of this see [RFC7657].

The method for assigning marking to packets can impact what number of RTP sessions to choose. If this marking is done using a network ingress function, it can have issues discriminating the different RTP streams. The network API on the endpoint also needs to be capable of setting the marking on a per-packet basis to reach the full functionality.

4.2.2. NAT and Firewall Traversal

In today's network there exist a large number of middleboxes. The ones that normally have most impact on RTP are Network Address Translators (NAT) and Firewalls (FW).

Below we analyse and comment on the impact of requiring more underlying transport flows in the presence of NATs and Firewalls:

End-Point Port Consumption: A given IP address only has 65536 available local ports per transport protocol for all consumers of ports that exist on the machine. This is normally never an issue for an end-user machine. It can become an issue for servers that handle large number of simultaneous streams. However, if the application uses ICE to authenticate STUN requests, a server can serve multiple endpoints from the same local port, and use the whole 5-tuple (source and destination address, source and destination port, protocol) as identifier of flows after having securely bound them to the remote endpoint address using the STUN request. In theory the minimum number of media server ports needed are the maximum number of simultaneous RTP Sessions a single endpoint can use. In practice, implementation will probably benefit from using more server ports to simplify implementation or avoid performance bottlenecks.

NAT State: If an endpoint sits behind a NAT, each flow it generates to an external address will result in a state that has to be kept in the NAT. That state is a limited resource. In home or Small Office/Home Office (SOHO) NATs, memory or processing are usually the most limited resources. For large scale NATs serving many internal endpoints, available external ports are likely the scarce resource. Port limitations is primarily a problem for larger centralised NATs where endpoint independent mapping requires each flow to use one port for the external IP address. This affects the maximum number of internal users per external IP address. However, it is worth pointing out that a real-time video conference session with audio and video is likely using less than

10 UDP flows, compared to certain web applications that can use 100+ TCP flows to various servers from a single browser instance.

NAT Traversal Extra Delay: Performing the NAT/FW traversal takes a certain amount of time for each flow. It also takes time in a phase of communication between accepting to communicate and the media path being established which is fairly critical. The best case scenario for how much extra time it takes after finding the first valid candidate pair following the specified ICE procedures are: 1.5*RTT + Ta*(Additional_Flows-1), where Ta is the pacing timer. That assumes a message in one direction, and then an immediate triggered check back. The reason it isn't more, is that ICE first finds one candidate pair that works prior to attempting to establish multiple flows. Thus, there is no extra time until one has found a working candidate pair. Based on that working pair the needed extra time is to in parallel establish the, in most cases 2-3, additional flows. However, packet loss causes extra delays, at least 100 ms, which is the minimal retransmission timer for ICE.

NAT Traversal Failure Rate: Due to the need to establish more than a single flow through the NAT, there is some risk that establishing the first flow succeeds but that one or more of the additional flows fail. The risk that this happens is hard to quantify, but ought to be fairly low as one flow from the same interfaces has just been successfully established. Thus only rare events such as NAT resource overload, or selecting particular port numbers that are filtered etc., ought to be reasons for failure.

Deep Packet Inspection and Multiple Streams: Firewalls differ in how deeply they inspect packets. There exist some potential that deeply inspecting firewalls will have similar legacy issues with multiple SSRCs as some stack implementations.

Using additional RTP streams in the same RTP session and transport flow does not introduce any additional NAT traversal complexities per RTP stream. This can be compared with normally one or two additional transport flows per RTP session when using multiple RTP sessions. Additional lower layer transport flows will be needed, unless an explicit de-multiplexing layer is added between RTP and the transport protocol. At time of writing no such mechanism was defined.

4.2.3. Multicast

Multicast groups provides a powerful tool for a number of real-time applications, especially the ones that desire broadcast-like behaviours with one endpoint transmitting to a large number of receivers, like in IPTV. There are also the RTP/RTCP extension to better support Source Specific Multicast (SSM) [RFC5760]. Another application is the Many to Many communication, which RTP [RFC3550] was originally built to support, but the multicast semantics do result in a certain number of limitations.

One limitation is that for any group, sender side adaptation to the actual receiver properties causes degradation for all participants to what is supported by the receiver with the worst conditions among the group participants. For broadcast type of applications this is not acceptable. Instead, various receiver-based solutions are employed to ensure that the receivers achieve best possible performance. By using scalable encoding and placing each scalability layer in a different multicast group, the receiver can control the amount of traffic it receives. To have each scalability layer on a different multicast group, one RTP session per multicast group is used.

In addition, the transport flow considerations in multicast are a bit different from unicast; NATs with port translation are not useful in the multicast environment, meaning that the entire port range of each multicast address is available for distinguishing between RTP sessions.

Thus, when using broadcast applications it appears easiest and most straightforward to use multiple RTP sessions for sending different media flows used for adapting to network conditions. It is also common that streams that improve transport robustness are sent in their own multicast group to allow for interworking with legacy or to support different levels of protection.

For many to many applications there are different needs. Here, the most appropriate choice will depend on how the actual application is realized. With sender side congestion control there might not exist any benefit with using multiple RTP sessions.

The properties of a broadcast application using RTP multicast:

1. Uses a group of RTP sessions, not one. Each endpoint will need to be a member of a number of RTP sessions in order to perform well.

2. Within each RTP session, the number of RTP receivers is likely to be much larger than the number of RTP senders.

3. The applications need signalling functions to identify the relationships between RTP sessions.

4. The applications need signalling or RTP/RTCP functions to identify the relationships between SSRCs in different RTP sessions when needs beyond CNAME exist.

Both broadcast and many to many multicast applications do share a signalling requirement; all of the participants will need to have the same RTP and payload type configuration. Otherwise, A could for example be using payload type 97 as the video codec H.264 while B thinks it is MPEG-2. It is to be noted that SDP offer/answer [RFC3264] is not appropriate for ensuring this property in broadcast/ multicast context. The signalling aspects of broadcast/multicast are not explored further in this memo.

Security solutions for this type of group communications are also challenging. First, the key-management and the security protocol need to support group communication. Second, source authentication requires special solutions. For more discussion on this please review Options for Securing RTP Sessions [RFC7201].

4.3. Security and Key Management Considerations

When dealing with point-to-point, 2-member RTP sessions only, there are few security issues that are relevant to the choice of having one RTP session or multiple RTP sessions. However, there are a few aspects of multiparty sessions that might warrant consideration. For general information of possible methods of securing RTP, please review RTP Security Options [RFC7201].

4.3.1. Security Context Scope

When using SRTP [RFC3711] the security context scope is important and can be a necessary differentiation in some applications. As SRTP's crypto suites are (so far) built around symmetric keys, the receiver will need to have the same key as the sender. This results in that no one in a multi-party session can be certain that a received packet really was sent by the claimed sender and not by another party having access to the key. At least unless TESLA source authentication [RFC4383], which adds delay to achieve source authentication. In most cases symmetric ciphers provide sufficient security properties, but there are a few cases where this does create issues.

The first case is when someone leaves a multi-party session and one wants to ensure that the party that left can no longer access the RTP streams. This requires that everyone re-keys without disclosing the keys to the excluded party.

A second case is when using security as an enforcing mechanism for differentiation. Take for example a scalable layer or a high quality simulcast version that only premium users are allowed to access. The mechanism preventing a receiver from getting the high quality stream can be based on the stream being encrypted with a key that user can't access without paying premium, having the key-management limit access to the key.

SRTP [RFC3711] has no special functions for dealing with different sets of master keys for different SSRCs. The key-management functions have different capabilities to establish different sets of keys, normally on a per endpoint basis. For example, DTLS-SRTP [RFC5764] and Security Descriptions [RFC4568] establish different keys for outgoing and incoming traffic from an endpoint. This key usage has to be written into the cryptographic context, possibly associated with different SSRCs.

4.3.2. Key Management for Multi-party sessions

Performing key-management for multi-party sessions can be a challenge. This section considers some of the issues.

Multi-party sessions, such as transport translator based sessions and multicast sessions, can neither use Security Description [RFC4568] nor DTLS-SRTP [RFC5764] without an extension as each endpoint provides its set of keys. In centralised conferences, the signalling counterpart is a conference server and the media plane unicast counterpart (to which DTLS messages would be sent) is the transport translator. Thus, an extension like Encrypted Key Transport [I-D.ietf-perc-srtp-ekt-diet] or a MIKEY [RFC3830] based solution that allows for keying all session participants with the same master key is needed.

4.3.3. Complexity Implications

The usage of security functions can surface complexity implications from the choice of multiplexing and topology. This becomes especially evident in RTP topologies having any type of middlebox that processes or modifies RTP/RTCP packets. Where there is very small overhead for an RTP translator or mixer to rewrite an SSRC value in the RTP packet of an unencrypted session, the cost is higher when using cryptographic security functions. For example, if using SRTP [RFC3711], the actual security context and exact crypto key are determined by the SSRC field value. If one changes SSRC, the encryption and authentication must use another key. Thus, changing the SSRC value implies a decryption using the old SSRC and its security context, followed by an encryption using the new one.

5. RTP Multiplexing Design Choices

This section discusses how some RTP multiplexing design choices can be used in applications to achieve certain goals, and a summary of the implications of such choices. For each design there is discussion of benefits and downsides.

5.1. Multiple Media Types in one Session

This design uses a single RTP session for multiple different media types, like audio and video, and possibly also transport robustness mechanisms like FEC or Retransmission. An endpoint can have zero, one or more media sources per media type, resulting in a number of RTP streams of various media types and both source and redundancy type.

The Pros:

1. Single RTP session which implies:

    * Minimal NAT/FW state.

    * Minimal NAT/FW Traversal Cost.

    * Fate-sharing for all media flows.

2. Can handle dynamic allocations of RTP streams well on an RTP level. Depends on the application's needs for explicit indication of the stream usage and how timely that can be signalled.

3. Minimal overhead for security association establishment.

The Cons:

a. Less suitable for interworking with other applications that uses individual RTP sessions per media type or multiple sessions for a single media type, due to the potential need of SSRC translation.

b. Negotiation of bandwidth for the different media types is currently only possible using RID [I-D.ietf-mmusic-rid] in SDP.

c. Not suitable for Split Component Terminal (see Section 3.10 of [RFC7667]).

d. Flow-based QoS cannot provide separate treatment of RTP streams compared to others in the single RTP session.

e. If there is significant asymmetry between the RTP streams' RTCP reporting needs, there are some challenges in configuration and usage to avoid wasting RTCP reporting on the RTP stream that does not need that frequent reporting.

f. Not suitable for applications where some receivers like to receive only a subset of the RTP streams, especially if multicast or transport translator is being used.

g. Additional concern with legacy implementations that do not support the RTP specification fully when it comes to handling multiple SSRC per endpoint, as also multiple simultaneous media types need to be handled.

h. If the applications need finer control over which session participants that are included in different sets of security associations, most key-management will have difficulties establishing such a session.

5.2. Multiple SSRCs of the Same Media Type

In this design, each RTP session serves only a single media type. The RTP session can contain multiple RTP streams, either from a single endpoint or from multiple endpoints. This commonly creates a low number of RTP sessions, typically only one for audio and one for video, with a corresponding need for two listening ports when using RTP/RTCP multiplexing.

The Pros:

1. Works well with Split Component Terminal (see Section 3.10 of [RFC7667]) where the split is per media type.

2. Enables Flow-based QoS with different prioritisation between media types.

3. For applications with dynamic usage of RTP streams, i.e. frequently added and removed, having much of the state associated with the RTP session rather than per individual SSRC can avoid the need for in-session signalling of meta-information about each SSRC.

4. Low overhead for security association establishment.

The Cons:

a. Slightly higher number of RTP sessions needed compared to Multiple Media Types in one Session Section 5.1. This implies:

* More NAT/FW state

* Increased NAT/FW Traversal Cost in both processing and delay.

b. Some potential for concern with legacy implementations that don't support the RTP specification fully when it comes to handling multiple SSRC per endpoint.

c. Not possible to control security association for sets of RTP streams within the same media type with today's key- management mechanisms, unless these are split into different RTP sessions.

For RTP applications where all RTP streams of the same media type share same usage, this structure provides efficiency gains in amount of network state used and provides more fate sharing with other media flows of the same type. At the same time, it is still maintaining almost all functionalities when it comes to negotiation in the signalling of the properties for the individual media type, and also enables flow based QoS prioritisation between media types. It handles multi-party session well, independently of multicast or centralised transport distribution, as additional sources can dynamically enter and leave the session.

5.3. Multiple Sessions for one Media type

This design goes one step further than above (Section 5.2) by using multiple RTP sessions also for a single media type. The main reason for going in this direction is that the RTP application needs separation of the RTP streams due to their usage. Some typical reasons for going to this design are scalability over multicast, simulcast, need for extended QoS prioritisation of RTP streams due to their usage in the application, or the need for fine- grained signalling using today's tools.

The Pros:

1. More suitable for multicast usage where receivers can individually select which RTP sessions they want to participate in, assuming each RTP session has its own multicast group.

2. The application can indicate its usage of the RTP streams on RTP session level, in case multiple different usages exist.

3. Less need for SSRC specific explicit signalling for each media stream and thus reduced need for explicit and timely signalling.

4. Enables detailed QoS prioritisation for flow-based mechanisms.

5. Works well with Split Component Terminal (see Section 3.10 of [RFC7667]).

6. The scope for who is included in a security association can be structured around the different RTP sessions, thus enabling such functionality with existing key-management.

The Cons:

a. Increases the amount of RTP sessions compared to Multiple SSRCs of the Same Media Type.

b. Increased amount of session configuration state.

c. For RTP streams that are part of scalability, simulcast or transport robustness, a method to bind sources across multiple RTP sessions is needed.

d. Some potential for concern with legacy implementations that does not support the RTP specification fully when it comes to handling multiple SSRC per endpoint.

e. Higher overhead for security association establishment due to the increased number of RTP sessions.

f. If the applications need finer control than on RTP session level over which participants that are included in different sets of security associations, most of today’s key-management will have difficulties establishing such a session.

For more complex RTP applications that have several different usages for RTP streams of the same media type, or uses scalability or simulcast, this solution can enable those functions at the cost of increased overhead associated with the additional sessions. This type of structure is suitable for more advanced applications as well as multicast-based applications requiring differentiation to different participants.

5.4. Single SSRC per Endpoint

In this design each endpoint in a point-to-point session has only a single SSRC, thus the RTP session contains only two SSRCs, one local and one remote. This session can be used both unidirectional, i.e. only a single RTP stream or bi-directional, i.e. both endpoints have one RTP stream each. If the application needs additional media flows between the endpoints, they will have to establish additional RTP sessions.

The Pros:

1. This design has great legacy interoperability potential as it will not tax any RTP stack implementations.

2. The signalling has good possibilities to negotiate and describe the exact formats and bit-rates for each RTP stream, especially using today's tools in SDP.

3. It is possible to control security association per RTP stream with current key-management, since each RTP stream is directly related to an RTP session, and the most used keying mechanisms operates on a per-session basis.

The Cons:

a. The number of RTP sessions grows directly in proportion with the number of RTP streams, which has the implications:

   * Linear growth of the amount of NAT/FW state with number of RTP streams.

   * Increased delay and resource consumption from NAT/FW traversal.

   * Likely larger signalling message and signalling processing requirement due to the amount of session related information.

   * Higher potential for a single RTP stream to fail during transport between the endpoints.

b. When the number of RTP sessions grows, the amount of explicit state for relating RTP streams also grows, depending on how the application needs to relate RTP streams.

c. The port consumption might become a problem for centralised services, where the central node's port or 5-tuple filter consumption grows rapidly with the number of sessions.

d. For applications where the RTP stream usage is highly dynamic, i.e. entering and leaving, the amount of signalling can grow high. Issues can also arise from the timely establishment of additional RTP sessions.

e. If, against the recommendation, the same SSRC value is reused in multiple RTP sessions rather than being randomly chosen, interworking with applications that use a different multiplexing structure will require SSRC translation.

RTP applications that need to interwork with legacy RTP applications can potentially benefit from this structure. However, a large number of media descriptions in SDP can also run into issues with existing implementations. For any application needing a larger number of media flows, the overhead can become very significant. This structure is also not suitable for multi-party sessions, as any given RTP stream from each participant, although having same usage in the application, needs its own RTP session. In addition, the dynamic behaviour that can arise in multi-party applications can tax the signalling system and make timely media establishment more difficult.

5.5. Summary

There are some clear similarities between these designs. Both the "Single SSRC per Endpoint" and the "Multiple Media Types in one Session" are cases that require full explicit signalling of the media stream relations. However, they operate on two different levels where the first primarily enables session level binding, and the second needs SSRC level binding. From another perspective, the two solutions are the two extreme points when it comes to number of RTP sessions needed.

The two other designs "Multiple SSRCs of the Same Media Type" and "Multiple Sessions for one Media Type" are two examples that primarily allows for some implicit mapping of the role or usage of the RTP streams based on which RTP session they appear in. It thus potentially allows for less signalling and in particular reduces the need for real-time signalling in dynamic sessions. They also represent points in between the first two designs when it comes to amount of RTP sessions established, i.e. representing an attempt to balance the amount of RTP sessions with the functionality the communication session provides both on network level and on signalling level.

6. Guidelines

This section contains a number of multi-stream guidelines for implementers or specification writers.

Do not require use of the same SSRC value across RTP sessions:
As discussed in Section 3.4.3 there exist drawbacks in using the same SSRC in multiple RTP sessions as a mechanism to bind related RTP streams together. It is instead recommended to use a mechanism to explicitly signal the relation, either in RTP/RTCP or in the signalling mechanism used to establish the RTP session(s).

Use additional RTP streams for additional media sources: In the cases where an RTP endpoint needs to transmit additional RTP streams of the same media type in the application, with the same processing requirements at the network and RTP layers, it is suggested to send them in the same RTP session. For example a telepresence room where there are three cameras, and each camera captures 2 persons sitting at the table, sending each camera as its own RTP stream within a single RTP session is suggested.

Use additional RTP sessions for streams with different requirements:

When RTP streams have different processing requirements from the network or the RTP layer at the endpoints, it is suggested that the different types of streams are put in different RTP sessions. This includes the case where different participants want different subsets of the set of RTP streams.

When using multiple RTP Sessions, use grouping: When using Multiple RTP session solutions, it is suggested to explicitly group the involved RTP sessions when needed using a signalling mechanism, for example The Session Description Protocol (SDP) Grouping Framework [RFC5888], using some appropriate grouping semantics.

RTP/RTCP Extensions Support Multiple RTP Streams as well as Multiple RTP sessions:
When defining an RTP or RTCP extension, the creator needs to consider if this extension is applicable to use with additional SSRCs and multiple RTP sessions. Any extension intended to be generic must support both. Extensions that are not as generally applicable will have to consider if interoperability is better served by defining a single solution or providing both options.

Transport Support Extensions: When defining new RTP/RTCP extensions intended for transport support, like the retransmission or FEC mechanisms, they must include support for both multiple RTP streams in the same RTP sessions and multiple RTP sessions, such that application developers can choose freely from the set of mechanisms without concerning themselves with which of the multiplexing choices a particular solution supports.

7. IANA Considerations

This document makes no request of IANA.

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

8. Security Considerations

The security considerations of the RTP specification [RFC3550] and any applicable RTP profile [RFC3551],[RFC4585],[RFC3711], the extensions for sending multiple media types in a single RTP session [I-D.ietf-avtcore-multi-media-rtp-session], RID [I-D.ietf-mmusic-rid], BUNDLE [I-D.ietf-mmusic-sdp-bundle-negotiation], [RFC5760], [RFC5761], apply if selected and thus needs to be considered in the evaluation.

There is discussion of the security implications of choosing multiple SSRC vs multiple RTP sessions in Section 4.3.

9. Contributors

Hui Zheng (Marvin) from Huawei contributed to WG draft versions -04 and -05 of the document.

10. References

10.1. Normative References

[RFC3550] Schulzrinne, H., Casner, S., Frederick, R., and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications", STD 64, RFC 3550, DOI 10.17487/RFC3550, July 2003, <https://www.rfc-editor.org/info/rfc3550>.

[RFC7656] Lennox, J., Gross, K., Nandakumar, S., Salgueiro, G., and B. Burman, Ed., "A Taxonomy of Semantics and Mechanisms for Real-Time Transport Protocol (RTP) Sources", RFC 7656, DOI 10.17487/RFC7656, November 2015, <https://www.rfc-editor.org/info/rfc7656>.

10.2. Informative References

[ALF] Clark, D. and D. Tennenhouse, "Architectural Considerations for a New Generation of Protocols", SIGCOMM Symposium on Communications Architectures and Protocols (Philadelphia, Pennsylvania), pp. 200--208, IEEE Computer Communications Review, Vol. 20(4), September 1990.

[I-D.ietf-avtcore-multi-media-rtp-session]
Westerlund, M., Perkins, C., and J. Lennox, "Sending Multiple Types of Media in a Single RTP Session", draft-ietf-avtcore-multi-media-rtp-session-13 (work in progress), December 2015.

[I-D.ietf-avtext-rid]
Roach, A., Nandakumar, S., and P. Thatcher, "RTP Stream Identifier Source Description (SDES)", draft-ietf-avtext-rid-09 (work in progress), October 2016.

[I-D.ietf-mmusic-rid]
Roach, A., "RTP Payload Format Restrictions", draft-ietf-mmusic-rid-15 (work in progress), May 2018.

[I-D.ietf-mmusic-sdp-bundle-negotiation]
Holmberg, C., Alvestrand, H., and C. Jennings, "Negotiating Media Multiplexing Using the Session Description Protocol (SDP)", draft-ietf-mmusic-sdp-bundle-negotiation-53 (work in progress), September 2018.

[I-D.ietf-mmusic-sdp-simulcast]
Burman, B., Westerlund, M., Nandakumar, S., and M. Zanaty, "Using Simulcast in SDP and RTP Sessions", draft-ietf-mmusic-sdp-simulcast-13 (work in progress), June 2018.

[I-D.ietf-perc-srtp-ekt-diet]
Jennings, C., Mattsson, J., McGrew, D., Wing, D., and F. Andreasen, "Encrypted Key Transport for DTLS and Secure RTP", draft-ietf-perc-srtp-ekt-diet-09 (work in progress), October 2018.

[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, DOI 10.17487/RFC2198, September 1997, <https://www.rfc-editor.org/info/rfc2198>.

[RFC2205] Braden, R., Ed., Zhang, L., Berson, S., Herzog, S., and S. Jamin, "Resource ReSerVation Protocol (RSVP) -- Version 1 Functional Specification", RFC 2205, DOI 10.17487/RFC2205, September 1997, <https://www.rfc-editor.org/info/rfc2205>.

[RFC2474] Nichols, K., Blake, S., Baker, F., and D. Black, "Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers", RFC 2474, DOI 10.17487/RFC2474, December 1998, <https://www.rfc-editor.org/info/rfc2474>.

[RFC2974] Handley, M., Perkins, C., and E. Whelan, "Session Announcement Protocol", RFC 2974, DOI 10.17487/RFC2974, October 2000, <https://www.rfc-editor.org/info/rfc2974>.

[RFC3261] Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston, A., Peterson, J., Sparks, R., Handley, M., and E. Schooler, "SIP: Session Initiation Protocol", RFC 3261, DOI 10.17487/RFC3261, June 2002, <https://www.rfc-editor.org/info/rfc3261>.

[RFC3264] Rosenberg, J. and H. Schulzrinne, "An Offer/Answer Model with Session Description Protocol (SDP)", RFC 3264, DOI 10.17487/RFC3264, June 2002, <https://www.rfc-editor.org/info/rfc3264>.

[RFC3389] Zopf, R., "Real-time Transport Protocol (RTP) Payload for Comfort Noise (CN)", RFC 3389, DOI 10.17487/RFC3389, September 2002, <https://www.rfc-editor.org/info/rfc3389>.

[RFC3551] Schulzrinne, H. and S. Casner, "RTP Profile for Audio and Video Conferences with Minimal Control", STD 65, RFC 3551, DOI 10.17487/RFC3551, July 2003, <https://www.rfc-editor.org/info/rfc3551>.

[RFC3711] Baugher, M., McGrew, D., Naslund, M., Carrara, E., and K. Norrman, "The Secure Real-time Transport Protocol (SRTP)", RFC 3711, DOI 10.17487/RFC3711, March 2004, <https://www.rfc-editor.org/info/rfc3711>.

[RFC3830] Arkko, J., Carrara, E., Lindholm, F., Naslund, M., and K. Norrman, "MIKEY: Multimedia Internet KEYing", RFC 3830, DOI 10.17487/RFC3830, August 2004, <https://www.rfc-editor.org/info/rfc3830>.

[RFC4103] Hellstrom, G. and P. Jones, "RTP Payload for Text Conversation", RFC 4103, DOI 10.17487/RFC4103, June 2005, <https://www.rfc-editor.org/info/rfc4103>.

[RFC4383] Baugher, M. and E. Carrara, "The Use of Timed Efficient Stream Loss-Tolerant Authentication (TESLA) in the Secure Real-time Transport Protocol (SRTP)", RFC 4383, DOI 10.17487/RFC4383, February 2006, <https://www.rfc-editor.org/info/rfc4383>.

[RFC4566] Handley, M., Jacobson, V., and C. Perkins, "SDP: Session Description Protocol", RFC 4566, DOI 10.17487/RFC4566, July 2006, <https://www.rfc-editor.org/info/rfc4566>.

[RFC4568] Andreasen, F., Baugher, M., and D. Wing, "Session Description Protocol (SDP) Security Descriptions for Media Streams", RFC 4568, DOI 10.17487/RFC4568, July 2006, <https://www.rfc-editor.org/info/rfc4568>.

[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, DOI 10.17487/RFC4585, July 2006, <https://www.rfc-editor.org/info/rfc4585>.

[RFC4588] Rey, J., Leon, D., Miyazaki, A., Varsa, V., and R. Hakenberg, "RTP Retransmission Payload Format", RFC 4588, DOI 10.17487/RFC4588, July 2006, <https://www.rfc-editor.org/info/rfc4588>.

[RFC5104] Wenger, S., Chandra, U., Westerlund, M., and B. Burman, "Codec Control Messages in the RTP Audio-Visual Profile with Feedback (AVPF)", RFC 5104, DOI 10.17487/RFC5104, February 2008, <https://www.rfc-editor.org/info/rfc5104>.

[RFC5109] Li, A., Ed., "RTP Payload Format for Generic Forward Error Correction", RFC 5109, DOI 10.17487/RFC5109, December 2007, <https://www.rfc-editor.org/info/rfc5109>.

[RFC5576] Lennox, J., Ott, J., and T. Schierl, "Source-Specific Media Attributes in the Session Description Protocol (SDP)", RFC 5576, DOI 10.17487/RFC5576, June 2009, <https://www.rfc-editor.org/info/rfc5576>.

[RFC5760] Ott, J., Chesterfield, J., and E. Schooler, "RTP Control Protocol (RTCP) Extensions for Single-Source Multicast Sessions with Unicast Feedback", RFC 5760, DOI 10.17487/RFC5760, February 2010, <https://www.rfc-editor.org/info/rfc5760>.

[RFC5761] Perkins, C. and M. Westerlund, "Multiplexing RTP Data and Control Packets on a Single Port", RFC 5761, DOI 10.17487/RFC5761, April 2010, <https://www.rfc-editor.org/info/rfc5761>.

[RFC5764] McGrew, D. and E. Rescorla, "Datagram Transport Layer Security (DTLS) Extension to Establish Keys for the Secure Real-time Transport Protocol (SRTP)", RFC 5764, DOI 10.17487/RFC5764, May 2010, <https://www.rfc-editor.org/info/rfc5764>.

[RFC5888] Camarillo, G. and H. Schulzrinne, "The Session Description Protocol (SDP) Grouping Framework", RFC 5888, DOI 10.17487/RFC5888, June 2010, <https://www.rfc-editor.org/info/rfc5888>.

[RFC6465] Ivov, E., Ed., Marocco, E., Ed., and J. Lennox, "A Real-time Transport Protocol (RTP) Header Extension for Mixer-to-Client Audio Level Indication", RFC 6465, DOI 10.17487/RFC6465, December 2011, <https://www.rfc-editor.org/info/rfc6465>.

[RFC7201] Westerlund, M. and C. Perkins, "Options for Securing RTP Sessions", RFC 7201, DOI 10.17487/RFC7201, April 2014, <https://www.rfc-editor.org/info/rfc7201>.

[RFC7657] Black, D., Ed. and P. Jones, "Differentiated Services (Diffserv) and Real-Time Communication", RFC 7657, DOI 10.17487/RFC7657, November 2015, <https://www.rfc-editor.org/info/rfc7657>.

[RFC7667] Westerlund, M. and S. Wenger, "RTP Topologies", RFC 7667, DOI 10.17487/RFC7667, November 2015, <https://www.rfc-editor.org/info/rfc7667>.

[RFC7826] Schulzrinne, H., Rao, A., Lanphier, R., Westerlund, M., and M. Stiemerling, Ed., "Real-Time Streaming Protocol Version 2.0", RFC 7826, DOI 10.17487/RFC7826, December 2016, <https://www.rfc-editor.org/info/rfc7826>.

[RFC8088] Westerlund, M., "How to Write an RTP Payload Format", RFC 8088, DOI 10.17487/RFC8088, May 2017, <https://www.rfc-editor.org/info/rfc8088>.

[RFC8108] Lennox, J., Westerlund, M., Wu, Q., and C. Perkins, "Sending Multiple RTP Streams in a Single RTP Session", RFC 8108, DOI 10.17487/RFC8108, March 2017, <https://www.rfc-editor.org/info/rfc8108>.

[RFC8445] Keranen, A., Holmberg, C., and J. Rosenberg, "Interactive Connectivity Establishment (ICE): A Protocol for Network Address Translator (NAT) Traversal", RFC 8445, DOI 10.17487/RFC8445, July 2018, <https://www.rfc-editor.org/info/rfc8445>.

Appendix A. Dismissing Payload Type Multiplexing

This section documents a number of reasons why using the payload type as a multiplexing point is unsuitable for most things related to multiple RTP streams. If one attempts to use Payload type multiplexing beyond its defined usage, that has well known negative effects on RTP. To use payload type as the single discriminator for multiple streams implies that all the different RTP streams are being sent with the same SSRC, thus using the same timestamp and sequence number space. This has many effects:

1. Putting restraint on RTP timestamp rate for the multiplexed media. For example, RTP streams that use different RTP timestamp rates cannot be combined, as the timestamp values need to be consistent across all multiplexed media frames. Thus streams are forced to use the same RTP timestamp rate. When this is not possible, payload type multiplexing cannot be used.

2. Many RTP payload formats can fragment a media object over multiple RTP packets, like parts of a video frame. These payload formats need to determine the order of the fragments to correctly decode them. Thus, it is important to ensure that all fragments related to a frame or a similar media object are transmitted in sequence and without interruptions within the object. This can relatively simple be solved on the sender side by ensuring that the fragments of each RTP stream are sent in sequence.

3. Some media formats require uninterrupted sequence number space between media parts. These are media formats where any missing RTP sequence number will result in decoding failure or invoking a repair mechanism within a single media context. The text/T140 payload format [RFC4103] is an example of such a format. These formats will need a sequence numbering abstraction function between RTP and the individual RTP stream before being used with payload type multiplexing.

4. Sending multiple streams in the same sequence number space makes it impossible to determine which payload type, which stream a packet loss relates to, and thus to which stream to potentially apply packet loss concealment or other stream-specific loss mitigation mechanisms.

5. If RTP Retransmission [RFC4588] is used and there is a loss, it is possible to ask for the missing packet(s) by SSRC and sequence number, not by payload type. If only some of the payload type multiplexed streams are of interest, there is no way of telling which missing packet(s) belong to the interesting stream(s) and all lost packets need be requested, wasting bandwidth.

6. The current RTCP feedback mechanisms are built around providing feedback on RTP streams based on stream ID (SSRC), packet (sequence numbers) and time interval (RTP Timestamps). There is almost never a field to indicate which payload type is reported,

so sending feedback for a specific RTP payload type is difficult without extending existing RTCP reporting.

7. The current RTCP media control messages [RFC5104] specification is oriented around controlling particular media flows, i.e. requests are done addressing a particular SSRC. Such mechanisms would need to be redefined to support payload type multiplexing.

8. The number of payload types are inherently limited. Accordingly, using payload type multiplexing limits the number of streams that can be multiplexed and does not scale. This limitation is exacerbated if one uses solutions like RTP and RTCP multiplexing [RFC5761] where a number of payload types are blocked due to the overlap between RTP and RTCP.

9. At times, there is a need to group multiplexed streams and this is currently possible for RTP sessions and for SSRC, but there is no defined way to group payload types.

10. It is currently not possible to signal bandwidth requirements per RTP stream when using payload type multiplexing.

11. Most existing SDP media level attributes cannot be applied on a per payload type level and would require re-definition in that context.

12. A legacy endpoint that does not understand the indication that different RTP payload types are different RTP streams might be slightly confused by the large amount of possibly overlapping or identically defined RTP payload types.

Appendix B. Signalling Considerations

Signalling is not an architectural consideration for RTP itself, so this discussion has been moved to an appendix. However, it is hugely important for anyone building complete applications, so it is deserving of discussion.

The issues raised here need to be addressed in the WGs that deal with signalling; they cannot be addressed by tweaking, extending or profiling RTP.

There exist various signalling solutions for establishing RTP sessions. Many are SDP [RFC4566] based, however SDP functionality is also dependent on the signalling protocols carrying the SDP. RTSP [RFC7826] and SAP [RFC2974] both use SDP in a declarative fashion, while SIP [RFC3261] uses SDP with the additional definition of Offer/Answer [RFC3264]. The impact on signalling and especially SDP needs to be considered as it can greatly affect how to deploy a certain multiplexing point choice.

B.1. Session Oriented Properties

One aspect of the existing signalling is that it is focused around RTP sessions, or at least in the case of SDP the media description. There are a number of things that are signalled on media description level but those are not necessarily strictly bound to an RTP session and could be of interest to signal specifically for a particular RTP stream (SSRC) within the session. The following properties have been identified as being potentially useful to signal not only on RTP session level:

- Bitrate/Bandwidth exist today only at aggregate or as a common "any RTP stream" limit, unless either codec-specific bandwidth limiting or RTCP signalling using TMMBR is used.

- Which SSRC that will use which RTP payload types (this will be visible from the first media packet, but is sometimes useful to know before packet arrival).

Some of these issues are clearly SDP's problem rather than RTP limitations. However, if the aim is to deploy an solution using additional SSRCs that contains several sets of RTP streams with different properties (encoding/packetization parameter, bit-rate, etc.), putting each set in a different RTP session would directly enable negotiation of the parameters for each set. If insisting on additional SSRC only, a number of signalling extensions are needed to clarify that there are multiple sets of RTP streams with different properties and that they need in fact be kept different, since a single set will not satisfy the application's requirements.

For some parameters, such as RTP payload type, resolution and framerate, a SSRC-linked mechanism has been proposed in [I-D.ietf-mmusic-rid]

B.2. SDP Prevents Multiple Media Types

SDP chose to use the m= line both to delineate an RTP session and to specify the top level of the MIME media type; audio, video, text, image, application. This media type is used as the top-level media type for identifying the actual payload format and is bound to a particular payload type using the rtpmap attribute. This binding has to be loosened in order to use SDP to describe RTP sessions containing multiple MIME top level types.

[I-D.ietf-mmusic-sdp-bundle-negotiation] describes how to let multiple SDP media descriptions use a single underlying transport in SDP, which allows to define one RTP session with media types having different MIME top level types.

B.3. Signalling RTP stream Usage

RTP streams being transported in RTP has some particular usage in an RTP application. This usage of the RTP stream is in many applications so far implicitly signalled. For example, an application might choose to take all incoming audio RTP streams, mix them and play them out. However, in more advanced applications that use multiple RTP streams there will be more than a single usage or purpose among the set of RTP streams being sent or received. RTP applications will need to signal this usage somehow. The signalling used will have to identify the RTP streams affected by their RTP-level identifiers, which means that they have to be identified either by their session or by their SSRC + session.

In some applications, the receiver cannot utilise the RTP stream at all before it has received the signalling message describing the RTP stream and its usage. In other applications, there exists a default handling that is appropriate.

If all RTP streams in an RTP session are to be treated in the same way, identifying the session is enough. If SSRCs in a session are to be treated differently, signalling needs to identify both the session and the SSRC.

If this signalling affects how any RTP central node, like an RTP mixer or translator that selects, mixes or processes streams, treats the streams, the node will also need to receive the same signalling to know how to treat RTP streams with different usage in the right fashion.

Authors' Addresses

Magnus Westerlund
Ericsson
Torshamsgatan 23
SE-164 80 Kista
Sweden

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

Bo Burman
Ericsson
Gronlandsgatan 31
SE-164 60 Kista
Sweden

Phone: +46 10 714 13 11
Email: bo.burman@ericsson.com

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

Email: csp@csperkins.org

Harald Tveit Alvestrand
Google
Kungsbron 2
Stockholm 11122
Sweden

Email: harald@alvestrand.no

Roni Even
Huawei

Email: roni.even@huawei.com

AVTCore D. Singer
Internet-Draft Apple, Inc.
Obsoletes: 5285 (if approved) H. Desineni
Intended status: Standards Track Qualcomm
Expires: February 3, 2018 R. Even, Ed.
Huawei Technologies
August 2, 2017

A General Mechanism for RTP Header Extensions
draft-ietf-avtcore-rfc5285-bis-14.txt

Abstract

This document provides a general mechanism to use the header extension feature of RTP (the Real-Time Transport Protocol). It provides the option to use a small number of small extensions in each RTP packet, where the universe of possible extensions is large and registration is de-centralized. The actual extensions in use in a session are signaled in the setup information for that session. This document obsoletes RFC5285.

Status of This Memo

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

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

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

This Internet-Draft will expire on February 3, 2018.

Copyright Notice

Copyright (c) 2017 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust’s Legal Provisions Relating to IETF Documents (http://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction

The RTP specification [RFC3550] provides a capability to extend the RTP header. It defines the header extension format and rules for its use in Section 5.3.1. The existing header extension method permits at most one extension per RTP packet, identified by a 16-bit identifier and a 16-bit length field specifying the length of the header extension in 32-bit words.

This mechanism has two conspicuous drawbacks. First, it permits only one header extension in a single RTP packet. Second, the specification gives no guidance as to how the 16-bit header extension identifiers are allocated to avoid collisions.

This specification removes the first drawback by defining a backward-compatible and extensible means to carry multiple header extension elements in a single RTP packet. It removes the second drawback by defining that these extension elements are named by URIs, defining an IANA registry for extension elements defined in IETF specifications, and a Session Description Protocol (SDP) method for mapping between the naming URIs and the identifier values carried in the RTP packets.

This header extension applies to RTP/AVP (the Audio/Visual Profile) and its extensions.

This document obsoletes [RFC5285] and removes a limitation from RFC5285 that did not allow sending both one-byte and two-byte header extensions in the same RTP stream.

2. Requirements Notation

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

3. Design Goals

The goal of this design is to provide a simple mechanism whereby multiple identified extensions can be used in RTP packets, without the need for formal registration of those extensions but nonetheless avoiding collision.

This mechanism provides an alternative to the practice of burying associated metadata into the media format bit stream. This has often been done in media data sent over fixed-bandwidth channels. Once this is done, a decoder for the specific media format needs to extract the metadata. Also, depending on the media format, the metadata can be added at the time of encoding the media so that the bit-rate used for the metadata is taken into account. But the metadata can be unknown at that time. Inserting metadata at a later time can cause a decode and re-encode to meet bit-rate requirements.

In some cases, a more appropriate, higher-level mechanism may be available, and if so, it can be used. For cases where a higher-level mechanism is not available, it is better to provide a mechanism at the RTP level than have the metadata be tied to a specific form of media data.

4. Packet Design

4.1. General

The following design is fit into the "header extension" of the RTP extension, as described above.

The presence and format of this header extension and its contents are negotiated or defined out-of-band, such as through signaling (see below for SDP signaling). The 16-bit identifier for the two forms of RTP extension defined here is only an architectural constant (e.g., for use by network analyzers); it is the negotiation/definition (e.g., in SDP) that is the definitive indication that this header extension is present.

The RTP specification [RFC3550] states that RTP "is designed so that the header extension may be ignored by other interoperating implementations that have not been extended". The intent of this restriction is that RTP header extensions MUST NOT be used to extend RTP itself in a manner that is backwards incompatible with non-extended implementations. For example, a header extension is not allowed to change the meaning or interpretation of the standard RTP header fields, or of the RTCP Control Protocol (RTCP). Header extensions MAY carry metadata in addition to the usual RTP header information, provided the RTP layer can function if that metadata is missing. For example, RTP header extensions can be used to carry data that's also sent in RTCP, as an optimisation to lower latency, since they'll fall back to the original, non-optimised, behaviour if the header extension is not present. The use of header extensions to convey information that will, if missing, disrupt the behaviour of a higher layer application that builds on top of RTP is only acceptable if this doesn't affect interoperability at the RTP layer. For example, applications that use the SDP BUNDLE extension with the MID RTP header extension [I-D.ietf-mmusic-sdp-bundle-negotiation] to correlate RTP streams with SDP m= lines likely won't work with full functionality if the MID is missing, but the operation of the RTP layer of those applications will be unaffected. Support for RTP header extensions based on this memo is negotiated using, for example, SDP Offer/Answer [RFC3264]; intermediaries aware of the RTP header extensions are advised to be cautious when removing or generating RTP header extensions see section 4.7 of [RFC7667].

The RTP header extension is formed as a sequence of extension elements, with possible padding. Each extension element has a local identifier and a length. The local identifiers MAY be mapped to a larger namespace in the negotiation (e.g., session signaling).

4.1.1. Transmission Considerations

As is good network practice, data should only be transmitted when needed. The RTP header extension SHOULD only be present in a packet if that packet also contains one or more extension elements, as defined here. An extension element SHOULD only be present in a packet when needed; the signaling setup of extension elements indicates only that those elements can be present in some packets, not that they are in fact present in all (or indeed, any) packets.

Some general considerations for getting the header extensions delivered to the receiver:

1. The probability for packet loss and burst loss determine how many repetitions of the header extensions will be required to reach a targeted delivery probability, and if burst loss is likely, what distribution would be needed to avoid getting all repetitions of the header extensions lost in a single burst.

2. If a set of packets are all needed to enable decoding, there is commonly no reason for including the header extension in all of these packets, as they share fate. Instead, at most one instance of the header extension per independently decodable set of media data would be a more efficient use of the bandwidth.

3. How early the Header Extension item information is needed, from the first received RTP data or only after some set of packets are received, can guide if the header extension(s) should be in all of the first N packets or be included only once per set of packets, for example once per video frame.

4. The use of RTP level robustness mechanisms, such as RTP retransmission [RFC4588], or Forward Error Correction, e.g., [RFC5109] may treat packets differently from a robustness perspective, and header extensions should be added to packets that get a treatment corresponding to the relative importance of receiving the information.

As a summary, the number of header extension transmissions should be tailored to a desired probability of delivery taking the receiver population size into account. For the very basic case, N repetitions of the header extensions should be sufficient, but may not be optimal. N is selected so that the header extension target delivery probability reaches 1-P^N, where P is the probability of packet loss. For point to point or small receiver populations, it might also be possible to use feedback, such as RTCP, to determine when the information in the header extensions has reached all receivers and stop further repetitions. Feedback that can be used includes the

RTCP XR Loss RLE report block [RFC3611], which will indicate successful delivery of particular packets. If the RTP/AVPF Transport Layer Feedback Messages for generic NACK [RFC4585] is used, it can indicate the failure to deliver an RTP packet with the header extension, thus indicating the need for further repetitions. The normal RTCP report blocks can also provide an indicator of successful delivery, if no losses are indicated for a reporting interval covering the RTP packets with the header extension. Note that loss of an RTCP packet reporting on an interval where RTP header extension packets were sent, does not necessarily mean that the RTP header extension packets themselves were lost.

4.1.2. Header Extension Type Considerations

Each extension element in a packet has a local identifier (ID) and a length. The local identifiers present in the stream MUST have been negotiated or defined out-of-band. There are no static allocations of local identifiers. Each distinct extension MUST have a unique ID. The ID value 0 is reserved for padding and MUST NOT be used as a local identifier.

An extension element with an ID value equal 0 MUST NOT have len field greater than 0. If such an extension element is encountered, its length field MUST be ignored, processing of the entire extension MUST terminate at that point, and only the extension elements present prior to the element with ID 0 and len field greater than 0 SHOULD be considered.

There are two variants of the extension: one-byte and two-byte headers. Since it is expected that (a) the number of extensions in any given RTP session is small and (b) the extensions themselves are small, the one-byte header form is preferred and MUST be supported by all receivers. A stream MUST contain only one-byte or only two-byte headers unless it is known that all recipients support mixing, either by SDP Offer/Answer [RFC3264] negotiation (see section 6) or by out-of-band knowledge. Each RTP packet with an RTP header extension following this specification will indicate if it contains one or two byte header extensions through the use of the "defined by profile" field. Extension element types that do not match the header extension format, i.e. one- or two-byte, MUST NOT be used in that RTP packet. Transmitters SHOULD NOT use the two-byte form when all extensions are small enough for the one-byte header form. Transmitters that intend to send the two-byte form SHOULD negotiate the use of IDs above 14 if they want to let the Receivers know that they intend to use two-byte form, for example if the RTP header extension is longer than 16 bytes. A transmitter may be aware that an intermediary may add RTP header extensions; in this case the transmitter SHOULD use two-byte form.

A sequence of extension elements, possibly with padding, forms the header extension defined in the RTP specification. There are as many extension elements as fit into the length as indicated in the RTP header extension length. Since this length is signaled in full 32-bit words, padding bytes are used to pad to a 32-bit boundary. The entire extension is parsed byte-by-byte to find each extension element (no alignment is needed), and parsing stops at the earlier of the end of the entire header extension, or in "one-byte headers only" case, on encountering an identifier with the reserved value of 15.

In both forms, padding bytes have the value of 0 (zero). They MAY be placed between extension elements, if desired for alignment, or after the last extension element, if needed for padding. A padding byte does not supply the ID of an element, nor the length field. When a padding byte is found, it is ignored and the parser moves on to interpreting the next byte.

Note carefully that the one-byte header form allows for data lengths between 1 and 16 bytes, by adding 1 to the signaled length value (thus, 0 in the length field indicates 1 byte of data follows). This allows for the important case of 16-byte payloads. This addition is not performed for the two-byte headers, where the length field signals data lengths between 0 and 255 bytes.

Use of RTP header extensions will reduce the efficiency of RTP header compression, since the header extension will be sent uncompressed unless the RTP header compression module is updated to recognize the extension header. If header extensions are present in some packets, but not in others, this can also reduce compression efficiency by requiring an update to the fixed header to be conveyed when header extensions start or stop being sent. The interactions of the RTP header extension and header compression is explored further in [RFC2508] and [RFC3095].

4.2. One-Byte Header

In the one-byte header form of extensions, the 16-bit value required by the RTP specification for a header extension, labeled in the RTP specification as "defined by profile", MUST have the fixed bit pattern 0xBEDE (the pattern was picked for the trivial reason that the first version of this specification was written on May 25th the feast day of the Venerable Bede).

Each extension element MUST start with a byte containing an ID and a length:

```
    0
    0 1 2 3 4 5 6 7
   +-+-+-+-+-+-+-+-+
   |  ID   |  len  |
   +-+-+-+-+-+-+-+-+
```

The 4-bit ID is the local identifier of this element in the range 1-14 inclusive. In the signaling section, this is referred to as the valid range.

The local identifier value 15 is reserved for future extension and MUST NOT be used as an identifier. If the ID value 15 is encountered, its length field MUST be ignored, processing of the entire extension MUST terminate at that point, and only the extension elements present prior to the element with ID 15 SHOULD be considered.

The 4-bit length is the number minus one of data bytes of this header extension element following the one-byte header. Therefore, the value zero in this field indicates that one byte of data follows, and a value of 15 (the maximum) indicates element data of 16 bytes. (This permits carriage of 16-byte values, which is a common length of labels and identifiers, while losing the possibility of zero-length values -- which would often be padded anyway.)

An example header extension, with three extension elements, and some padding follows:

```
    0                   1                   2                   3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |       0xBE    |    0xDE       |           length=3            |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |  ID   | L=0   |     data      |  ID   |  L=1  |   data...
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
         ...data   |    0 (pad)    |    0 (pad)    |  ID   | L=3   |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                          data                                 |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

4.3. Two-Byte Header

In the two-byte header form, the 16-bit value defined by the RTP specification for a header extension, labeled in the RTP specification as "defined by profile", is defined as shown below.

```
 0                   1
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         0x100         |appbits|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The appbits field is 4 bits that are application-dependent and MAY be defined to be any value or meaning, and are outside the scope of this specification. For the purposes of signaling, this field is treated as a special extension value assigned to the local identifier 256. If no extension has been specified through configuration or signaling for this local identifier value 256, the appbits field SHOULD be set to all 0s by the sender and MUST be ignored by the receiver.

Each extension element starts with a byte containing an ID and a byte containing a length:

```
 0                   1
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|       ID      |     length    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The 8-bit ID is the local identifier of this element in the range 1-255 inclusive. In the signaling section, the range 1-256 is referred to as the valid range, with the values 1-255 referring to extension elements, and the value 256 referring to the 4-bit field 'appbits' (above). Note that there is one ID space for both one-byte and two-byte form. This means that the lower values (1-14) can be used in the 4-bit ID field in the one-byte header format with the same meanings.

The 8-bit length field is the length of extension data in bytes not including the ID and length fields. The value zero indicates there is no data following.

An example header extension, with three extension elements, and some padding follows:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|       0x10    |    0x00       |           length=3            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|      ID       |     L=0       |     ID        |     L=1       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|       data    |    0 (pad)    |       ID      |      L=4      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                          data                                 |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

5. SDP Signaling Design

The indication of the presence of this extension, and the mapping of local identifiers used in the header extension to a larger namespace, MUST be performed out-of-band, for example, as part of an SDP Offer/ Answer [RFC3264]. This section defines such signaling in SDP.

A usable mapping MUST use IDs in the valid range, and each ID in this range MUST be used only once for each media (or only once if the mappings are session level). Mappings that do not conform to these rules MAY be presented, for instance, during SDP Offer/Answer [RFC3264] negotiation as described in the next section, but remapping to conformant values is necessary before they can be applied.

Each extension is named by a URI. That URI MUST be absolute, and precisely identifies the format and meaning of the extension. URIs that contain a domain name SHOULD also contain a month-date in the form mmyyyy. The definition of the element and assignment of the URI MUST have been authorized by the owner of the domain name on or very close to that date. (This avoids problems when domain names change ownership.) If the resource or document defines several extensions, then the URI MUST identify the actual extension in use, e.g., using a fragment or query identifier (characters after a '#' or '?' in the URI).

Rationale: the use of URIs provides for a large, unallocated space, and gives documentation on the extension. The URIs do not have to be de-referencable, in order to permit confidential or experimental use, and to cover the case when extensions continue to be used after the organization that defined them ceases to exist.

An extension URI with the same attributes MUST NOT appear more than once applying to the same stream, i.e., at session level or in the declarations for a single stream at media level. (The same extension can, of course, be used for several streams, and can appear with different extensionattributes for the same stream.)

For extensions defined in RFCs, the URI used SHOULD be a URN starting "urn:ietf:params:rtp-hdrext:" and followed by a registered, descriptive name.

The registration requirements are detailed in the IANA Considerations section, below.

An example (this is only an example), where 'avt-example-metadata' is the hypothetical name of a header extension, might be:

    urn:ietf:params:rtp-hdrext:avt-example-metadata

An example name not from the IETF (this is only an example) might be:

    http://example.com/082005/ext.htm#example-metadata

The mapping MAY be provided per media stream (in the media-level section(s) of SDP, i.e., after an "m=" line) or globally for all streams (i.e., before the first "m=" line, at session level). The definitions MUST be either all session level or all media level; it is not permitted to mix the two styles. In addition, as noted above, the IDs used MUST be unique in each media section of the SDP, or unique in the session for session-level SDP declarations.

Each local identifier potentially used in the stream is mapped to an extension identified by a URI using an attribute of the form:

    a=extmap:<value>["/"<direction>] <URI> <extensionattributes>

where <URI> is a URI, as above, <value> is the local identifier (ID) of this extension and is an integer in the valid range (0 is reserved for padding in both forms, and 15 is reserved in the one-byte header form, as noted above), and <direction> is one of "sendonly", "recvonly", "sendrecv", or "inactive" (without the quotes) with relation to the device being configured.

The formal BNF syntax is presented in a later section of this specification.

Example:

    a=extmap:1 http://example.com/082005/ext.htm#ttime

    a=extmap:2/sendrecv http://example.com/082005/ext.htm#xmeta short

When SDP signaling is used for the RTP session, it is the presence of the ’extmap’ attribute(s) that is diagnostic that this style of header extensions is used, not the magic number ("BEDE" or "100") indicated above.

6. SDP Signaling for support of mixed one byte and two bytes header extensions.

In order to allow for backward interoperability with systems that do not support mixing of one byte and two bytes header extensions this document defines the "a=extmap-allow-mixed" Session Description Protocol (SDP) [RFC4566] attribute to indicate if the participant is capable of supporting this new mode. The attribute takes no value. This attribute can be used at the session or media levels. A participant that proposes the use of this mode SHALL itself support the reception of mixed one byte and two bytes header extensions.

If SDP Offer/Answer [RFC3264] is supported and used,the negotiation for mixed one byte and two bytes extension MUST be negotiated using SDP Offer/Answer [RFC3264]. In the absence of negotiations using SDP Offer/Answer, for example when declarative SDP is used, mixed headers MUST NOT occur unless the transmitter has some (out of band) knowledge that all potential recipients support this mode.

The formal definition of this attribute is:

Name: extmap-allow-mixed

Value: none

Usage Level: session, media

Charset Dependent: no

Example:

a=extmap-allow-mixed

When doing SDP Offer/Answer [RFC3264] an offering client that wishes to use both one and two bytes extensions MUST include the attribute "a= extmap-allow-mixed " in the SDP offer. If "a= extmap-allow-mixed " is present in the offer SDP, the answerer that supports this mode and wishes to use it SHALL include the "a=extmap-allow-mixed " attribute in the answer. In the cases where the attribute has been excluded, both clients SHALL NOT use mixed one bytes and two bytes extensions in the same RTP stream but MAY use one-byte or two-bytes form exclusively (see section 4.1.2).

When used in [I-D.ietf-mmusic-sdp-bundle-negotiation] this attribute is specified as identical category for the [I-D.ietf-mmusic-sdp-mux-attributes]. This allows for only a subset of the m-lines in the bundle group to offer extmap-allow-mixed. When an answerer supporting the extmap-allow-mix attribute receives an offer where only some of the m-lines in the bundle group include the extmap-allow-mixed attribute, the answerer MUST receive this offer and support mixed one-byte and two-bytes only for those m-lines. Transmitters MUST only send RTP header extensions using mixed on those RTP streams originating from those media sources (m=) blocks that includes extmap-allow-mixed, and are RECOMMENDED to support receiving mixed on all RTP streams being received in an RTP session where at least one bundled m= block is indicating extmap-allow-mixed.

7. SDP Offer/Answer

The simple signaling described above for the extmap attribute MAY be enhanced in an SDP Offer/Answer [RFC3264] context, to permit:

- o asymmetric behavior (extensions sent in only one direction),
- o the offer of mutually exclusive alternatives, or
- o the offer of more extensions than can be sent in a single session.

A direction attribute MAY be included in an extmap; without it, the direction implicitly inherits, of course, from the stream direction, or is "sendrecv" for session-level attributes or extensions of "inactive" streams. The direction MUST be one of "sendonly", "recvonly", "sendrecv", or "inactive" as specified in [RFC3264]

Extensions, with their directions, MAY be signaled for an "inactive" stream. It is an error to use an extension direction incompatible with the stream direction (e.g., a "sendonly" attribute for a "recvonly" stream).

If an offer or answer contains session-level mappings (and hence no media-level mappings), and different behavior is desired for each stream, then the entire set of extension map declarations MAY be moved into the media-level section(s) of the SDP. (Note that this specification does not permit mixing global and local declarations, to make identifier management easier.)

If an extension map is offered as "sendrecv", explicitly or implicitly, and asymmetric behavior is desired, the SDP answer MAY be changed to modify or add direction qualifiers for that extension.

If an extension is marked as "sendonly" and the answerer desires to receive it, the extension MUST be marked as "recvonly" in the SDP answer. An answerer that has no desire to receive the extension or does not understand the extension SHOULD remove it from the SDP answer. An answerer MAY want to respond that he supports the extension and does not want to receive it at the moment but may offer to receive it in a future offer, will mark the extension as "inactive"

If an extension is marked as "recvonly" and the answerer desires to send it, the extension MUST be marked as "sendonly" in the SDP answer. An answerer that has no desire to, or is unable to, send the extension SHOULD remove it from the SDP answer. An answerer MAY want to respond that he support this extension yet has no intention of sending it now but may offer to send it in a future offer by marking the extension as "inactive"

Local identifiers in the valid range inclusive in an offer or answer must not be used more than once per media section (including the session-level section). The local identifiers MUST be unique in an RTP session and the same identifier MUST be used for the same offered extension in the answer. A session update MAY change the direction qualifiers of extensions under use. A session update MAY add or remove extension(s). Identifiers values in the valid range MUST NOT be altered (remapped).

Note that, under this rule, the same local identifier cannot be used for two extensions for the same media, even when one is "sendonly" and the other "recvonly", as it would then be impossible to make either of them sendrecv (since re-numbering is not permitted either).

If a party wishes to offer mutually exclusive alternatives, then multiple extensions with the same identifier in the extended range 4096-4351 MAY be offered; the answerer SHOULD select at most one of the offered extensions with the same identifier, and remap it to a free identifier in the valid range, for that extension to be usable.

Similarly, if more extensions are offered than can be fit in the valid range, identifiers in the range 4096-4351 MAY be offered; the answerer SHOULD choose those that are desired, and remap them to a free identifier in the valid range.

An answerer may copy an extmap for an identifier in the extended range into the answer to indicate to the offerer that it supports that extension. Of course, such an extension cannot be used, since there is no way to specify them in an extension header. If needed, the offerer or answerer can update the session to assign a valid identifier to that extension URI.

Rationale: the range 4096-4351 for these negotiation identifiers is deliberately restricted to allow expansion of the range of valid identifiers in future.

Either party MAY include extensions in the stream other than those negotiated, or those negotiated as "inactive", for example, for the benefit of intermediate nodes. Only extensions that appeared with an identifier in the valid range in SDP originated by the sender can be sent.

Example (port numbers, RTP profiles, payload IDs and rtpmaps, etc. all omitted for brevity):

The offer:

```
a=extmap:1 URI-toffset
a=extmap:14 URI-obscure
a=extmap:4096 URI-gps-string
a=extmap:4096 URI-gps-binary
a=extmap:4097 URI-frametype
m=video
a=sendrecv
m=audio
a=sendrecv
```

The answerer is interested in receiving GPS in string format only on video, but cannot send GPS at all. It is not interested in transmission offsets on audio, and does not understand the URI-obscure extension. It therefore moves the extensions from session level to media level, and adjusts the declarations:

```
m=video
a=sendrecv
a=extmap:1 URI-toffset
a=extmap:2/recvonly URI-gps-string
a=extmap:3 URI-frametype
m=audio
a=sendrecv
a=extmap:1/sendonly URI-toffset
```

When using [I-D.ietf-mmusic-sdp-bundle-negotiation] to bundle multiple m-lines the extmap attribute falls under the special category of [I-D.ietf-mmusic-sdp-mux-attributes]. All the m-lines in a bundle group are considered to be part of the same local identifier (ID) space. If an RTP header extension, i.e. a particular extension URI and configuration using <extensionattributes>, is offered in multiple m-lines that are part of the same bundle group it MUST use the same ID in all of these m-lines. Each m-line in a bundle group can include different RTP header extensions allowing for example audio and video sources to use different sets of RTP header extensions. It SHALL be assumed that for any RTP header extension, difference in configuration using any of the <extensionattributes> is important and need to be preserved to any receiver, thus requiring assignment of different IDs. Any RTP header extension that does not match this assumption MUST explicitly provide rules for what are compatible configurations that can be sent with the same ID. The directionality of the RTP header extensions in each m-line of the bundle group are handled as the non-bundled case. This allows for specifying different directionality for each of the repeated extension URI in bundled group.

8. BNF Syntax

The syntax definition below uses ABNF according to [RFC5234]. The syntax element 'URI' is defined in [RFC3986] (only absolute URIs are permitted here). The syntax element 'extmap' is an attribute as defined in [RFC4566], i.e., "a=" precedes the extmap definition. Specific extensionattributes are defined by the specification that defines a specific extension name; there can be several.

```
    Name: extmap

    Value: extmap-value

    Syntax:

     extmap-value = mapentry SP extensionname
                    [SP extensionattributes]

     mapentry = "extmap:" 1*5DIGIT ["/" direction]

     extensionname = URI

     extensionattributes = byte-string

     direction = "sendonly" / "recvonly" / "sendrecv" / "inactive"

     URI = <Defined in RFC 3986>

     byte-string = <Defined in RFC 4566>

     SP = <Defined in RFC 5234>

     DIGIT = <Defined in RFC 5234>
```

9. Security Considerations

This document defines only a place to transmit information; the security implications of each of the extensions must be discussed with those extensions.

Extensions usage is negotiated using [RFC3264] so integrity protection and end-to-end authentication MUST be implemented. The security considerations of [RFC3264] MUST be followed, to prevent, for example, extension usage blocking.

Header extensions have the same security coverage as the RTP header itself. When Secure Real-time Transport Protocol (SRTP) [RFC3711] is used to protect RTP sessions, the RTP payload can be both encrypted and integrity protected, while the RTP header is either unprotected or integrity protected. In order to prevent DOS attacks, for example, by changing the header extension, integrity protection SHOULD be used. Lower layer security protection like DTLS[RFC6347] MAY be used. RTP header extensions can carry sensitive information for which participants in multimedia sessions want confidentiality. RFC6904 [RFC6904] provides a mechanism, extending the mechanisms of SRTP, to selectively encrypt RTP header extensions in SRTP.

The RTP application designer needs to consider their security needs, that includes cipher strength for SRTP packets in general and what that means for the integrity and confidentiality of the RTP header extensions. As defined by RFC6904 [RFC6904] the encryption stream cipher for the header extension is dependent on the chosen SRTP cipher.

Other security options for securing RTP are discussed in [RFC7201].

10. IANA Considerations

This document updates the IANA consideration to reference this document and adds a new SDP attribute in section 10.3

Note to IANA : change RFCxxxx to this RFC number and remove the note.

10.1. Identifier Space for IANA to Manage

The mapping from the naming URI form to a reference to a specification is managed by IANA. Insertion into this registry is under the requirements of "Expert Review" as defined in [RFC8126].

The IANA will also maintain a server that contains all of the registered elements in a publicly accessible space.

Here is the formal declaration to comply with the IETF URN Sub-namespace specification [RFC3553].

- o Registry name: RTP Compact Header Extensions

- o Specification: RFC 5285 and RFCs updating RFC 5285.

- o Information required:

  A. The desired extension naming URI

  B. A formal reference to the publicly available specification

  C. A short phrase describing the function of the extension

  D. Contact information for the organization or person making the registration

  For extensions defined in RFCs, the URI SHOULD be of the form urn:ietf:params:rtp-hdrext:, and the formal reference is the RFC number of the RFC documenting the extension.

- o Review process: Expert review is REQUIRED. The expert review SHOULD check the following requirements:

  1. that the specification is publicly available;

  2. that the extension complies with the requirements of RTP, and this specification, for header extensions (specifically, that the header extension can be ignored or discarded without breaking the RTP layer);

  3. that the extension specification is technically consistent (in itself and with RTP), complete, and comprehensible;

  4. that the extension does not duplicate functionality in existing IETF specifications (including RTP itself), or other extensions already registered;

  5. that the specification contains a security analysis regarding the content of the header extension;

  6. that the extension is generally applicable, for example point-to-multipoint safe, and the specification correctly describes limitations if they exist; and

  7. that the suggested naming URI form is appropriately chosen and unique.

8. That for [I-D.ietf-mmusic-sdp-bundle-negotiation] multiplexed m-lines, any RTP header extension with difference in configurations of <extensionattributes> that do not require assignment of different IDs, MUST explicitly indicate this and provide rules for what are compatible configurations that can be sent with the same ID.

- Size and format of entries: a mapping from a naming URI string to a formal reference to a publicly available specification, with a descriptive phrase and contact information.

- Initial assignments: none.

10.2. Registration of the SDP extmap Attribute

IANA is requested to update the registration of the extmap SDP [RFC4566] attribute.

- Contact Name and email address: IETF, contacted via mmusic@ietf.org, or a successor address designated by IESG Attribute Name: extmap

- Attribute Syntax: See section 8 of [RFCXXXX].

- Attribute Semantics: The details of appropriate values are given in [RFC XXXX].

- Usage Level: Media or session level.

- Charset Dependent: No.

- Purpose: defines the mapping from the extension numbers used in packet headers into extension names.

- O/A Procedures: See section 7 of [RFCXXXX].

- Mux Category: Special.

- Reference: [RFCXXXX]

10.3. Registration of the SDP extmap-allow-mixed Attribute

The IANA is requested to register one new SDP attribute:

- Contact Name and email address: IETF, contacted via mmusic@ietf.org, or a successor address designated by IESG.

- Attribute Name: extmap-allow-mixed.

- o Attribute Syntax: See section 6 of [RFCXXXX].
- o Attribute Semantics: See section 6 of [RFCXXXX].
- o Attribute Value: None.
- o Usage Level: Media or session level.
- o Charset Dependent: no.
- o Purpose: Negotiate the use of One and Two bytes in the same RTP stream.
- o O/A Procedures: See section 6 of [RFCXXXX].
- o Mux Category: Identical
- o Reference: [RFCXXXX]

11. Changes from RFC5285

The major motivation for updating [RFC5285] was to allow having one byte and two bytes RTP header extensions in the same RTP stream (but not in the same RTP packet). The support for this case is negotiated using a new SDP attribute "extmap-allow-mixed" specified in this document.

The other major change is to update the requirement from the RTP specification [RFC3550] and [RFC5285] that the header extension "is designed so that the header extension MAY be ignored". This is described in section 4.1.

The transmission consideration section (4.1.1) adds more text to clarify when and how many times to send the RTP header extension to provide higher probability of delivery

>The security section was expanded

The rest of the changes are editorial.

12. Acknowledgments

Both Brian Link and John Lazzaro provided helpful comments on an initial draft of this document. Colin Perkins was helpful in reviewing and dealing with the details. The use of URNs for IETF-defined extensions was suggested by Jonathan Lennox, and Pete Cordell was instrumental in improving the padding wording. Dave Oran provided feedback and text in the review. Mike Dolan contributed the two-byte header form. Magnus Westerlund and Tom Taylor were instrumental in managing the registration text.

13. References

13.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <http://www.rfc-editor.org/info/rfc2119>.

[RFC2508] Casner, S. and V. Jacobson, "Compressing IP/UDP/RTP Headers for Low-Speed Serial Links", RFC 2508, DOI 10.17487/RFC2508, February 1999, <http://www.rfc-editor.org/info/rfc2508>.

[RFC3095] Bormann, C., Burmeister, C., Degermark, M., Fukushima, H., Hannu, H., Jonsson, L-E., Hakenberg, R., Koren, T., Le, K., Liu, Z., Martensson, A., Miyazaki, A., Svanbro, K., Wiebke, T., Yoshimura, T., and H. Zheng, "RObust Header Compression (ROHC): Framework and four profiles: RTP, UDP, ESP, and uncompressed", RFC 3095, DOI 10.17487/RFC3095, July 2001, <http://www.rfc-editor.org/info/rfc3095>.

[RFC3264] Rosenberg, J. and H. Schulzrinne, "An Offer/Answer Model with Session Description Protocol (SDP)", RFC 3264, DOI 10.17487/RFC3264, June 2002, <http://www.rfc-editor.org/info/rfc3264>.

[RFC3711] Baugher, M., McGrew, D., Naslund, M., Carrara, E., and K. Norrman, "The Secure Real-time Transport Protocol (SRTP)", RFC 3711, DOI 10.17487/RFC3711, March 2004, <http://www.rfc-editor.org/info/rfc3711>.

[RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, DOI 10.17487/RFC3986, January 2005, <http://www.rfc-editor.org/info/rfc3986>.

[RFC4566] Handley, M., Jacobson, V., and C. Perkins, "SDP: Session Description Protocol", RFC 4566, DOI 10.17487/RFC4566, July 2006, <http://www.rfc-editor.org/info/rfc4566>.

[RFC5234] Crocker, D., Ed. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", STD 68, RFC 5234, DOI 10.17487/RFC5234, January 2008, <http://www.rfc-editor.org/info/rfc5234>.

[RFC6904] Lennox, J., "Encryption of Header Extensions in the Secure Real-time Transport Protocol (SRTP)", RFC 6904, DOI 10.17487/RFC6904, April 2013, <http://www.rfc-editor.org/info/rfc6904>.

13.2. Informative References

[I-D.ietf-mmusic-sdp-bundle-negotiation]
Holmberg, C., Alvestrand, H., and C. Jennings, "Negotiating Media Multiplexing Using the Session Description Protocol (SDP)", draft-ietf-mmusic-sdp-bundle-negotiation-38 (work in progress), April 2017.

[I-D.ietf-mmusic-sdp-mux-attributes]
Nandakumar, S., "A Framework for SDP Attributes when Multiplexing", draft-ietf-mmusic-sdp-mux-attributes-16 (work in progress), December 2016.

[RFC3550] Schulzrinne, H., Casner, S., Frederick, R., and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications", STD 64, RFC 3550, DOI 10.17487/RFC3550, July 2003, <http://www.rfc-editor.org/info/rfc3550>.

[RFC3553] Mealling, M., Masinter, L., Hardie, T., and G. Klyne, "An IETF URN Sub-namespace for Registered Protocol Parameters", BCP 73, RFC 3553, DOI 10.17487/RFC3553, June 2003, <http://www.rfc-editor.org/info/rfc3553>.

[RFC3611] Friedman, T., Ed., Caceres, R., Ed., and A. Clark, Ed., "RTP Control Protocol Extended Reports (RTCP XR)", RFC 3611, DOI 10.17487/RFC3611, November 2003, <http://www.rfc-editor.org/info/rfc3611>.

[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, DOI 10.17487/RFC4585, July 2006, <http://www.rfc-editor.org/info/rfc4585>.

[RFC4588] Rey, J., Leon, D., Miyazaki, A., Varsa, V., and R. Hakenberg, "RTP Retransmission Payload Format", RFC 4588, DOI 10.17487/RFC4588, July 2006, <http://www.rfc-editor.org/info/rfc4588>.

[RFC5109] Li, A., Ed., "RTP Payload Format for Generic Forward Error Correction", RFC 5109, DOI 10.17487/RFC5109, December 2007, <http://www.rfc-editor.org/info/rfc5109>.

[RFC5285] Singer, D. and H. Desineni, "A General Mechanism for RTP Header Extensions", RFC 5285, DOI 10.17487/RFC5285, July 2008, <http://www.rfc-editor.org/info/rfc5285>.

[RFC6347] Rescorla, E. and N. Modadugu, "Datagram Transport Layer Security Version 1.2", RFC 6347, DOI 10.17487/RFC6347, January 2012, <http://www.rfc-editor.org/info/rfc6347>.

[RFC7201] Westerlund, M. and C. Perkins, "Options for Securing RTP Sessions", RFC 7201, DOI 10.17487/RFC7201, April 2014, <http://www.rfc-editor.org/info/rfc7201>.

[RFC7667] Westerlund, M. and S. Wenger, "RTP Topologies", RFC 7667, DOI 10.17487/RFC7667, November 2015, <http://www.rfc-editor.org/info/rfc7667>.

[RFC8126] Cotton, M., Leiba, B., and T. Narten, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 8126, DOI 10.17487/RFC8126, June 2017, <http://www.rfc-editor.org/info/rfc8126>.

Authors' Addresses

David Singer
Apple, Inc.
1 Infinite Loop
Cupertino, CA 95014
USA

Phone: +1 408 996 1010
Email: singer@apple.com
URI: http://www.apple.com/quicktime

Harikishan Desineni
Qualcomm
10001 Pacific Heights Blvd
San Diego, CA 92121
USA

Phone: +1 858 845 8996
Email: hdesinen@quicinc.com

Roni Even (editor)
Huawei Technologies
Tel Aviv
Israel

Email: Roni.even@huawei.com

AVTCORE WG
Internet-Draft
Intended status: Standards Track
Expires: September 3, 2016

J. Lennox
Vidyo
M. Westerlund
Ericsson
Q. Wu
Huawei
C. Perkins
University of Glasgow
March 2, 2016

Sending Multiple RTP Streams in a Single RTP Session: Grouping RTCP Reception Statistics and Other Feedback

draft-ietf-avtcore-rtp-multi-stream-optimisation-12

Abstract

RTP allows multiple RTP streams to be sent in a single session, but requires each Synchronisation Source (SSRC) to send RTCP reception quality reports for every other SSRC visible in the session. This causes the number of RTCP reception reports to grow with the number of SSRCs, rather than the number of endpoints. In many cases most of these RTCP reception reports are unnecessary, since all SSRCs of an endpoint are normally co-located and see the same reception quality. This memo defines a Reporting Group extension to RTCP to reduce the reporting overhead in such scenarios.

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 3, 2016.

Copyright Notice

Copyright (c) 2016 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (http://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction

The Real-time Transport Protocol (RTP) [RFC3550] is a protocol for group communication, supporting multiparty multimedia sessions. A single RTP session can support multiple participants sending at once, and can also support participants sending multiple simultaneous RTP streams. Examples of the latter might include a participant with multiple cameras who chooses to send multiple views of a scene, or a participant that sends audio and video flows multiplexed in a single RTP session. Rules for handling RTP sessions containing multiple RTP streams are described in [RFC3550] with some clarifications in [I-D.ietf-avtcore-rtp-multi-stream].

An RTP endpoint will have one or more synchronisation sources (SSRCs). It will have at least one RTP Stream, and thus SSRC, for each media source it sends, and might use multiple SSRCs per media source when using media scalability features [RFC6190], forward error correction, RTP retransmission [RFC4588], or similar mechanisms. An endpoint that is not sending any RTP stream, will have at least one SSRC to use for reporting and any feedback messages. Each SSRC has to send RTCP sender reports corresponding to the RTP packets it sends, and receiver reports for traffic it receives. That is, every SSRC will send RTCP packets to report on every other SSRC. This rule is simple, but can be quite inefficient for endpoints that send large numbers of RTP streams in a single RTP session. Consider a session comprising ten participants, each sending three media sources, each with their own RTP stream. There will be 30 SSRCs in such an RTP session, and each of those 30 SSRCs will send an RTCP Sender Report/ Receiver Report packet (containing several report blocks) per reporting interval as each SSRC reports on all the others. However, the three SSRCs comprising each participant are commonly co-located such that they see identical reception quality. If there was a way to indicate that several SSRCs are co-located, and see the same reception quality, then two-thirds of those RTCP reports could be suppressed. This would allow the remaining RTCP reports to be sent more often, while keeping within the same RTCP bandwidth fraction.

This memo defines such an RTCP extension, RTCP Reporting Groups. This extension is used to indicate the SSRCs that originate from the same endpoint, and therefore have identical reception quality, hence allowing the endpoints to suppress unnecessary RTCP reception quality reports.

2. Terminology

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

3. RTCP Reporting Groups

An RTCP Reporting Group is a set of synchronization sources (SSRCs) that are co-located at a single endpoint (which could be an end host or a middlebox) in an RTP session. Since they are co-located, every SSRC in the RTCP reporting group will have an identical view of the network conditions, and see the same lost packets, jitter, etc. This allows a single representative to send RTCP reception quality reports on behalf of the rest of the reporting group, reducing the number of RTCP packets that need to be sent without loss of information.

3.1. Semantics and Behaviour of RTCP Reporting Groups

A group of co-located SSRCs that see identical network conditions can form an RTCP reporting group. If reporting groups are in use, an RTP endpoint with multiple SSRCs MAY put those SSRCs into a reporting group if their view of the network is identical; i.e., if they report on traffic received at the same interface of an RTP endpoint. SSRCs with different views of the network MUST NOT be put into the same reporting group.

An endpoint that has combined its SSRCs into an RTCP reporting group will choose one (or a subset) of those SSRCs to act as "reporting source(s)" for that RTCP reporting group. A reporting source will send RTCP SR/RR reception quality reports on behalf of the other members of the RTCP reporting group. A reporting source MUST suppress the RTCP SR/RR reports that relate to other members of the reporting group, and only report on remote SSRCs. The other members (non reporting sources) of the RTCP reporting group will suppress their RTCP reception quality reports, and instead send an RTCP RGRS packet (see Section 3.2.2) to indicate that they are part of an RTCP reporting group and give the SSRCs of the reporting sources.

If there are large numbers of remote SSRCs in the RTP session, then the reception quality reports generated by the reporting source might grow too large to fit into a single compound RTCP packet, forcing the reporting source to use a round-robin policy to determine what remote SSRCs it includes in each compound RTCP packet, and so reducing the frequency of reports on each SSRC. To avoid this, in sessions with large numbers of remote SSRCs, an RTCP reporting group MAY use more than one reporting source. If several SSRCs are acting as reporting sources for an RTCP reporting group, then each reporting source MUST have non-overlapping sets of remote SSRCs it reports on.

An endpoint MUST NOT create an RTCP reporting group that comprises only a single local SSRC (i.e., an RTCP reporting group where the reporting source is the only member of the group), unless it is anticipated that the group might have additional SSRCs added to it in the future.

If a reporting source leaves the RTP session (i.e., if it sends a RTCP BYE packet, or leaves the session without sending BYE under the rules of [RFC3550] section 6.3.7), the remaining members of the RTCP reporting group MUST either (a) have another reporting source, if one exists, report on the remote SSRCs the leaving SSRC reported on, (b) choose a new reporting source, or (c) disband the RTCP reporting group and begin sending reception quality reports following [RFC3550] and [I-D.ietf-avtcore-rtp-multi-stream].

The RTCP timing rules assign different bandwidth fractions to senders and receivers. This lets senders transmit RTCP reception quality reports more often than receivers. If a reporting source in an RTCP reporting group is a receiver, but one or more non-reporting SSRCs in the RTCP reporting group are senders, then the endpoint MAY treat the reporting source as a sender for the purpose of RTCP bandwidth allocation, increasing its RTCP bandwidth allocation, provided it also treats one of the senders as if it were a receiver and makes the corresponding reduction in RTCP bandwidth for that SSRC. However, the application needs to consider the impact on the frequency of transmitting of the synchronization information included in RTCP Sender Reports.

3.2. Identifying Members of an RTCP Reporting Group

When RTCP Reporting Groups are in use, the other SSRCs in the RTP session need to be able to identify which SSRCs are members of an RTCP reporting group. Two RTCP extensions are defined to support this: the RTCP RGRP SDES item is used by the reporting source(s) to identify an RTCP reporting group, and the RTCP RGRS packet is used by other members of an RTCP reporting group to identify the reporting source(s).

3.2.1. Definition and Use of the RTCP RGRP SDES Item

This document defines a new RTCP SDES item to identify an RTCP reporting group. The motivation for giving a reporting group an identify is to ensure that the RTCP reporting group and its member SSRCs can be correctly associated when there are multiple reporting sources, and to ensure that a reporting SSRC can be associated with the correct reporting group if an SSRC collision occurs.

This document defines the RTCP Source Description (SDES) RGRP item. The RTCP SDES RGRP item MUST be sent by the reporting sources in a reporting group, and MUST NOT be sent by other members of the reporting group or by SSRCs that are not members of any RTCP reporting group. Specifically, every reporting source in an RTCP reporting group MUST include an RTCP SDES packet containing an RGRP item in every compound RTCP packet in which it sends an RR or SR packet (i.e., in every RTCP packet it sends, unless Reduced-Size RTCP [RFC5506] is in use).

Syntactically, the format of the RTCP SDES RGRP item is identical to that of the RTCP SDES CNAME item [RFC7022], except that the SDES item type field MUST have value RGRP=(TBA) instead of CNAME=1. The value of the RTCP SDES RGRP item MUST be chosen with the same concerns about global uniqueness and the same privacy considerations as the RTCP SDES CNAME. The value of the RTCP SDES RGRP item MUST be stable throughout the lifetime of the reporting group, even if some or all of the reporting sources change their SSRC due to collisions, or if the set of reporting sources changes.

Note to RFC Editor: please replace (TBA) in the above paragraph with the RTCP SDES item type number assigned to the RGRP item, then delete this note.

An RTP mixer or translator that forwards RTCP SR or RR packets from members of a reporting group MUST forward the corresponding RTCP SDES RGRP items as well, even if it otherwise strips SDES items other than the CNAME item.

3.2.2. Definition and Use of the RTCP RGRS Packet

A new RTCP packet type is defined to allow the members of an RTCP reporting group to identify the reporting sources for that group. This allows participants in an RTP session to distinguish an SSRC that is sending empty RTCP reception reports because it is a member of an RTCP reporting group, from an SSRC that is sending empty RTCP reception reports because it is not receiving any traffic. It also explicitly identifies the reporting sources, allowing other members of the RTP session to know which SSRCs are acting as the reporting sources for an RTCP reporting group, and allowing them to detect if RTCP packets from any of the reporting sources are being lost.

The format of the RTCP RGRS packet is defined below. It comprises the fixed RTCP header that indicates the packet type and length, the SSRC of the packet sender, and a list of reporting sources for the RTCP reporting group of which the packet sender is a member.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|V=2|P|    SC   | PT=RGRS(TBA)  |             length            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                     SSRC of packet sender                     |
+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
:          List of SSRC(s) for the Reporting Source(s)          :
:                              ...                              :
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The fields in the RTCP RGRS packet have the following definition:

version (V): 2 bits unsigned integer. This field identifies the RTP version. The current RTP version is 2.

padding (P): 1 bit. If set, the padding bit indicates that the RTCP packet contains additional padding octets at the end that are not part of the control information but are included in the length field. See [RFC3550].

Source Count (SC): 5 bits unsigned integer. Indicates the number of reporting source SSRCs that are included in this RTCP packet. As the RTCP RGRS packet MUST NOT be not sent by reporting sources, all the SSRCs in the list of reporting sources will be different from the SSRC of the packet sender. Every RTCP RGRS packet MUST contain at least one reporting source SSRC.

Payload type (PT): 8 bits unsigned integer. The RTCP packet type number that identifies the packet as being an RTCP RGRS packet. The RGRS RTCP packet has the value [TBA].

Note to RFC Editor: please replace [TBA] here, and in the packet format diagram above, with the RTCP packet type that IANA assigns to the RTCP RGRS packet.

Length: 16 bits unsigned integer. The length of this packet in 32-bit words minus one, including the header and any padding. This is in line with the definition of the length field used in RTCP sender and receiver reports [RFC3550]. Since all RTCP RGRS packets include at least one reporting source SSRC, the length will always be 2 or greater.

SSRC of packet sender: 32 bits. The SSRC of the sender of this packet.

List of SSRCs for the Reporting Source(s): A variable length size (as indicated by SC header field) of the 32 bit SSRC values of the reporting sources for the RTCP Reporting Group of which the packet sender is a member.

Every source that belongs to an RTCP reporting group but is not a reporting source MUST include an RTCP RGRS packet in every compound RTCP packet in which it sends an RR or SR packet (i.e., in every RTCP packet it sends, unless Reduced-Size RTCP [RFC5506] is in use). Each RTCP RGRS packet MUST contain the SSRC identifier of at least one reporting source. If there are more reporting sources in an RTCP reporting group than can fit into an RTCP RGRS packet, the members of that reporting group MUST send the SSRCs of the reporting sources in a round-robin fashion in consecutive RTCP RGRS packets, such that all the SSRCs of the reporting sources are included over the course of several RTCP reporting intervals.

An RTP mixer or translator that forwards RTCP SR or RR packets from members of a reporting group MUST also forward the corresponding RGRS RTCP packets. If the RTP mixer or translator rewrites SSRC values of the packets it forwards, it MUST make the corresponding changes to the RTCP RGRS packets.

3.3. Interactions with the RTP/AVPF Feedback Profile

Use of the RTP/AVPF Feedback Profile [RFC4585] allows SSRCs to send rapid RTCP feedback requests and codec control messages. If use of the RTP/AVPF profile has been negotiated in an RTP session, members of an RTCP reporting group can send rapid RTCP feedback and codec control messages following [RFC4585] and [RFC5104], as updated by Section 5.4 of [I-D.ietf-avtcore-rtp-multi-stream], and by the following considerations.

The members of an RTCP reporting group will all see identical network conditions. Accordingly, one might therefore think that it doesn’t matter which SSRC in the reporting group sends the RTP/AVPF feedback or codec control messages. There might be, however, cases where the sender of the feedback/codec control message has semantic importance, or when only a subset of the members of an RTCP reporting group might want to send RTP/AVPF feedback or a codec control message in response to a particular event. For example, an RTP video sender might choose to treat packet loss feedback received from SSRCs known to be audio receivers with less urgency than feedback that it receives from video receivers when deciding what packets to retransmit, and a multimedia receiver using reporting groups might want to choose the outgoing SSRC for feedback packets to reflect this.

Each member of an RTCP reporting group SHOULD therefore send RTP/AVPF feedback/codec control messages independently of the other members of the reporting group, to respect the semantic meaning of the message sender. The suppression rules of [RFC4585] will ensure that only a single copy of each feedback packet is (typically) generated, even if several members of a reporting group send the same feedback. When an endpoint knows that several members of its RTCP reporting group will be sending identical feedback, and that the sender of the feedback is not semantically important, then that endpoint MAY choose to send all its feedback from the reporting source and deterministically suppress feedback packets generated by the other sources in the reporting group.

It is important to note that the RTP/AVPF timing rules operate on a per-SSRC basis. Using a single reporting source to send all feedback for a reporting group will hence limit the amount of feedback that can be sent to that which can be sent by one SSRC. If this limit is a problem, then the reporting group can allow each of its members to send its own feedback, using its own SSRC.

If the RTP/AVPF feedback messages or codec control requests are sent as compound RTCP packets, then those compound RTCP packets MUST include either an RTCP RGRS packet or an RTCP SDES RGRP item, depending on whether they are sent by the reporting source or a non-reporting source in the RTCP reporting group respectively. The contents of non-compound RTCP feedback or codec control messages are not affected by the use of RTCP reporting groups.

3.4. Interactions with RTCP Extended Report (XR) Packets

When using RTCP Extended Reports (XR) [RFC3611] with RTCP reporting groups, it is RECOMMENDED that the reporting source is used to send the RTCP XR packets. If multiple reporting sources are in use, the reporting source that sends the SR/RR packets that relate to a particular remote SSRC SHOULD send the RTCP XR reports about that SSRC. This is motivated as one commonly combine the RTCP XR metrics with the regular report block to more fully understand the situation. Receiving these blocks in different compound packets reduces their value as the measuring intervals are not synchronized in those cases.

Some RTCP XR report blocks are specific to particular types of media, and might be relevant to only some members of a reporting group. For example, it would make no sense for an SSRC that is receiving video to send a VoIP metric RTCP XR report block. Such media specific RTCP XR report blocks MUST be sent by the SSRC to which they are relevant, and MUST NOT be included in the common report sent by the reporting source. This might mean that some SSRCs send RTCP XR packets in compound RTCP packets that contain an empty RTCP SR/RR packet, and that the time period covered by the RTCP XR packet is different to that covered by the RTCP SR/RR packet. If it is important that the RTCP XR packet and RTCP SR/RR packet cover the same time period, then that source SHOULD be removed from the RTCP reporting group, and send standard RTCP packets instead.

3.5. Middlebox Considerations

Many different types of middlebox are used with RTP. RTCP reporting groups are potentially relevant to those types of RTP middlebox that have their own SSRCs and generate RTCP reports for the traffic they receive. RTP middleboxes that do not have their own SSRC, and that don’t send RTCP reports on the traffic they receive, cannot use the RTCP reporting groups extension, since they generate no RTCP reports to group.

An RTP middlebox that has several SSRCs of its own can use the RTCP reporting groups extension to group the RTCP reports it generates. This can occur, for example, if a middlebox is acting as an RTP mixer for both audio and video flows that are multiplexed onto a single RTP session, where the middlebox has one SSRC for the audio mixer and one for the video mixer part, and when the middlebox wants to avoid cross reporting between audio and video.

A middlebox cannot use the RTCP reporting groups extension to group RTCP packets from the SSRCs that it is forwarding. It can, however, group the RTCP packets from the SSRCs it is forwarding into compound RTCP packets following the rules in Section 6.1 of [RFC3550] and Section 5.3 of [I-D.ietf-avtcore-rtp-multi-stream]. If the middlebox is using RTCP reporting groups for its own SSRCs, it MAY include RTCP packets from the SSRCs that it is forwarding as part of the compound RTCP packets its reporting source generates.

A middlebox that forwards RTCP SR or RR packets sent by members of a reporting group MUST forward the corresponding RTCP SDES RGRP items, as described in Section 3.2.1. A middlebox that forwards RTCP SR or RR packets sent by member of a reporting group MUST also forward the corresponding RTCP RGRS packets, as described in Section 3.2.2. Failure to forward these packets can cause compatibility problems, as described in Section 4.2.

If a middlebox rewrites SSRC values in the RTP and RTCP packets that it is forwarding, then it MUST make the corresponding changes in RTCP SDES packets containing RGRP items and in RTCP RGRS packets, to allow them to be associated with the rewritten SSRCs.

3.6. SDP Signalling for Reporting Groups

This document defines the "a=rtcp-rgrp" Session Description Protocol (SDP) [RFC4566] attribute to indicate if the session participant is capable of supporting RTCP Reporting Groups for applications that use SDP for configuration of RTP sessions. It is a property attribute, and hence takes no value. The multiplexing category [I-D.ietf-mmusic-sdp-mux-attributes] is IDENTICAL, as the functionality applies on RTP session level. A participant that proposes the use of RTCP Reporting Groups SHALL itself support the reception of RTCP Reporting Groups. The formal definition of this attribute is:

Name: rtcp-rgrp
Value:
Usage Level: session, media
Charset Dependent: no
Example:
   a=rtcp-rgrp

When using SDP Offer/Answer [RFC3264], the following procedures are to be used:

- Generating the initial SDP offer: If the offerer supports the RTCP reporting group extensions, and is willing to accept RTCP packets containing those extensions, then it MUST include an "a=rtcp-rgrp" attribute in the initial offer. If the offerer does not support RTCP reporting groups extensions, or is not willing to accept RTCP packets containing those extensions, then it MUST NOT include the "a=rtcp-rgrp" attribute in the offer.

- Generating the SDP answer: If the SDP offer contains an "a=rtcp-rgrp" attribute, and if the answerer supports RTCP reporting groups and is willing to receive RTCP packets using the RTCP reporting groups extensions, then the answerer MAY include an "a=rtcp-rgrp" attribute in the answer and MAY send RTCP packets containing the RTCP reporting groups extensions. If the offer does not contain an "a=rtcp-rgrp" attribute, or if the offer does contain such an attribute but the answerer does not wish to accept RTCP packets using the RTCP reporting groups extensions, then the answer MUST NOT include an "a=rtcp-rgrp" attribute.

- Offerer Processing of the SDP Answer: If the SDP answer contains an "a=rtcp-rgrp" attribute, and the corresponding offer also contained an "a=rtcp-rgrp" attribute, then the offerer MUST be prepared to accept and process RTCP packets that contain the reporting groups extension, and MAY send RTCP packets that contain the reporting groups extension. If the SDP answer contains an "a=rtcp-rgrp" attribute, but the corresponding offer did not contain the "a=rtcp-rgrp" attribute, then the offerer MUST reject the call. If the SDP answer does not contain an "a=rtcp-rgrp" attribute, then the offerer MUST NOT send packets containing the RTCP reporting groups extensions, and does not need to process packet containing the RTCP reporting groups extensions.

In declarative usage of SDP, such as the Real Time Streaming Protocol (RTSP) [RFC2326] and the Session Announcement Protocol (SAP) [RFC2974], the presence of the attribute indicates that the session participant MAY use RTCP Reporting Groups in its RTCP transmissions. An implementation that doesn’t explicitly support RTCP Reporting Groups MAY join a RTP session as long as it has been verified that the implementation doesn’t suffer from the problems discussed in Section 4.2.

4. Properties of RTCP Reporting Groups

This section provides additional information on what the resulting properties are with the design specified in Section 3. The content of this section is non-normative.

4.1. Bandwidth Benefits of RTCP Reporting Groups

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

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

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

Using the original [RFC3550] everyone-reports-on-every-sender feedback rules, each of the 184 receivers will send 16 report blocks, and each of the 16 senders will send 15. This amounts to approximately 76 kB of report block traffic per interval; 92% of RTCP traffic consists of report blocks.

If reporting groups are used, however, there is only 0.4 kB of reports per interval, with no loss of useful information. Additionally, there will be (assuming 16-byte RGRPs, and a single reporting source per reporting group) an additional 2.4 kB per cycle of RGRP SDES items and RGRS packets. Put another way, the unmodified [RFC3550] reporting interval is approximately 9 times longer than if reporting groups are in use.

4.2. Compatibility of RTCP Reporting Groups

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

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

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

Thus, it is unclear to what extent such backward compatibility issues would actually cause trouble in practice.

5. Security Considerations

The security considerations of [RFC3550] and [I-D.ietf-avtcore-rtp-multi-stream] apply. If the RTP/AVPF profile is in use, then the security considerations of [RFC4585] (and [RFC5104], if used) also apply. If RTCP XR is used, the security consideration of [RFC3611] and any XR report blocks used also apply.

The RTCP SDES RGRP item is vulnerable to malicious modifications unless integrity protected is used. A modification of this item’s length field cause the parsing of the RTCP packet in which it is contained to fail. Depending on the implementation, parsing of the full compound RTCP packet can also fail causing the whole packet to be discarded. A modification to the value of this SDES item would make the receiver of the report think that the sender of the report was a member of a different RTCP reporting group. This will potentially create an inconsistency, when the RGRS reports the source as being in the same reporting group as another source with another reporting group identifier. What impact on a receiver implementation such inconsistencies would have are difficult to fully predict. One case is when congestion control or other adaptation mechanisms are used, an inconsistent report can result in a media sender to reduce its bit-rate. However, a direct modification of the receiver report or a feedback message itself would be a more efficient attack, and equally costly to perform.

The new RGRS RTCP Packet type is very simple. The common RTCP packet type header shares the security risks with previous RTCP packet types. Errors or modification of the length field can cause the full compound packet to fail header validation (see Appendix A.2 in [RFC3550]) resulting in the whole compound RTCP packet being discarded. Modification of the SC or P fields would cause inconsistency when processing the RTCP packet, likely resulting it being classified as invalid. A modification of the PT field would cause the packet being interpreted under some other packet type’s rules. In such case the result might be more or less predictable but packet type specific. Modification of the SSRC of packet sender would attribute this packet to another sender. Resulting in a receiver believing the reporting group applies also for this SSRC, if it exists. If it doesn’t exist, unless also corresponding modifications are done on a SR/RR packet and a SDES packet the RTCP packet SHOULD be discarded. If consistent changes are done, that could be part of a resource exhaustion attack on a receiver implementation. Modification of the "List of SSRCs for the Reporting Source(s)" would change the SSRC the receiver expect to report on behalf of this SSRC. If that SSRC exist, that could potentially change the report group used for this SSRC. A change to another reporting group belonging to another endpoint is likely detectable as there would be a mismatch between the SSRC of the packet sender’s endpoint information, transport addresses, SDES CNAME etc and the corresponding information from the reporting group indicated.

In general the reporting group is providing limited impacts attacks. The most significant result from an deliberate attack would be to cause the information to be discarded or be inconsistent, including discard of all RTCP packets that are modified. This causes a lack of information at any receiver entity, possibly disregarding the endpoints participation in the session.

To protect against this type of attacks from external non trusted entities, integrity and source authentication SHOULD be applied. This can be done, for example, by using SRTP [RFC3711] with appropriate key-management, other options exist as discussed in RTP Security Options [RFC7201].

The Report Group Identifier has a potential privacy impacting properties. If this would be generated by an implementation in such a way that is long term stable or predictable, it could be used for tracking a particular end-point. Therefore it is RECOMMENDED that it be generated as a short-term persistent RGRP, following the rules for short-term persistent CNAMEs in [RFC7022]. The rest of the information revealed, i.e. the SSRCs, the size of reporting group and the number of reporting sources in a reporting group is of less sensitive nature, considering that the SSRCs and the communication would anyway be revealed without this extension. By encrypting the report group extensions the SSRC values would preserved confidential, but can still be revealed if SRTP [RFC3711] is used. The size of the reporting groups and number of reporting sources are likely determinable from analysis of the packet pattern and sizes. However, this information appears to have limited value.

6. IANA Considerations

(Note to the RFC-Editor: in the following, please replace "TBA" with the IANA-assigned value, and "XXXX" with the number of this document, then delete this note)

The IANA is requested to register one new RTCP SDES items in the "RTCP SDES Item Types" registry, as follows:

Value	Abbrev	Name	Reference
TBA	RGRP	Reporting Group Identifier	[RFCXXXX]

The definition of the RTCP SDES RGRP item is given in Section 3.2.1 of this memo.

The IANA is also requested to register one new RTCP packet type in the "RTCP Control Packet Types (PT)" Registry as follows:

Value	Abbrev	Name	Reference
TBA	RGRS	Reporting Group Reporting Sources	[RFCXXXX]

The definition of the RTCP RGRS packet type is given in Section 3.2.2 of this memo.

The IANA is also requested to register one new SDP attribute:

```
SDP Attribute ("att-field"):
   Attribute name:     rtcp-rgrp
   Long form:          RTCP Reporting Groups
   Type of name:       att-field
   Type of attribute:  Media or session level
   Subject to charset: No
   Purpose:            Negotiate or configure the use of the RTCP
                       Reporting Group Extension.
   Reference:          [RFCXXXX]
   Values:             None
```

The definition of the "a=rtcp-rgrp" SDES attribute is given in Section 3.6 of this memo.

7. References

7.1. Normative References

[I-D.ietf-avtcore-rtp-multi-stream]
Lennox, J., Westerlund, M., Wu, Q., and C. Perkins, "Sending Multiple RTP Streams in a Single RTP Session", draft-ietf-avtcore-rtp-multi-stream-11 (work in progress), December 2015.

[I-D.ietf-mmusic-sdp-mux-attributes]
Nandakumar, S., "A Framework for SDP Attributes when Multiplexing", draft-ietf-mmusic-sdp-mux-attributes-12 (work in progress), January 2016.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <http://www.rfc-editor.org/info/rfc2119>.

[RFC3264] Rosenberg, J. and H. Schulzrinne, "An Offer/Answer Model with Session Description Protocol (SDP)", RFC 3264, DOI 10.17487/RFC3264, June 2002, <http://www.rfc-editor.org/info/rfc3264>.

[RFC3550] Schulzrinne, H., Casner, S., Frederick, R., and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications", STD 64, RFC 3550, DOI 10.17487/RFC3550, July 2003, <http://www.rfc-editor.org/info/rfc3550>.

[RFC4566] Handley, M., Jacobson, V., and C. Perkins, "SDP: Session Description Protocol", RFC 4566, DOI 10.17487/RFC4566, July 2006, <http://www.rfc-editor.org/info/rfc4566>.

[RFC7022] Begen, A., Perkins, C., Wing, D., and E. Rescorla, "Guidelines for Choosing RTP Control Protocol (RTCP) Canonical Names (CNAMEs)", RFC 7022, DOI 10.17487/RFC7022, September 2013, <http://www.rfc-editor.org/info/rfc7022>.

7.2. Informative References

[RFC2326] Schulzrinne, H., Rao, A., and R. Lanphier, "Real Time Streaming Protocol (RTSP)", RFC 2326, DOI 10.17487/RFC2326, April 1998, <http://www.rfc-editor.org/info/rfc2326>.

[RFC2974] Handley, M., Perkins, C., and E. Whelan, "Session Announcement Protocol", RFC 2974, DOI 10.17487/RFC2974, October 2000, <http://www.rfc-editor.org/info/rfc2974>.

[RFC3611] Friedman, T., Ed., Caceres, R., Ed., and A. Clark, Ed., "RTP Control Protocol Extended Reports (RTCP XR)", RFC 3611, DOI 10.17487/RFC3611, November 2003, <http://www.rfc-editor.org/info/rfc3611>.

[RFC3711] Baugher, M., McGrew, D., Naslund, M., Carrara, E., and K. Norrman, "The Secure Real-time Transport Protocol (SRTP)", RFC 3711, DOI 10.17487/RFC3711, March 2004, <http://www.rfc-editor.org/info/rfc3711>.

[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, DOI 10.17487/RFC4585, July 2006, <http://www.rfc-editor.org/info/rfc4585>.

[RFC4588] Rey, J., Leon, D., Miyazaki, A., Varsa, V., and R. Hakenberg, "RTP Retransmission Payload Format", RFC 4588, DOI 10.17487/RFC4588, July 2006, <http://www.rfc-editor.org/info/rfc4588>.

[RFC5104] Wenger, S., Chandra, U., Westerlund, M., and B. Burman, "Codec Control Messages in the RTP Audio-Visual Profile with Feedback (AVPF)", RFC 5104, DOI 10.17487/RFC5104, February 2008, <http://www.rfc-editor.org/info/rfc5104>.

[RFC5506] Johansson, I. and M. Westerlund, "Support for Reduced-Size Real-Time Transport Control Protocol (RTCP): Opportunities and Consequences", RFC 5506, DOI 10.17487/RFC5506, April 2009, <http://www.rfc-editor.org/info/rfc5506>.

[RFC6190] Wenger, S., Wang, Y., Schierl, T., and A. Eleftheriadis, "RTP Payload Format for Scalable Video Coding", RFC 6190, DOI 10.17487/RFC6190, May 2011, <http://www.rfc-editor.org/info/rfc6190>.

[RFC7201] Westerlund, M. and C. Perkins, "Options for Securing RTP Sessions", RFC 7201, DOI 10.17487/RFC7201, April 2014, <http://www.rfc-editor.org/info/rfc7201>.

Authors' Addresses

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

Email: jonathan@vidyo.com

Magnus Westerlund
Ericsson
Farogatan 2
SE-164 80 Kista
Sweden

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

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

Email: bill.wu@huawei.com

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

Email: csp@csperkins.org

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: May 3, 2018

M. Zanaty
E. Berger
S. Nandakumar
Cisco Systems

October 30, 2017

Frame Marking RTP Header Extension
draft-ietf-avtext-framemarking-06

Abstract

This document describes a Frame Marking RTP header extension used to convey information about video frames that is critical for error recovery and packet forwarding in RTP middleboxes or network nodes. It is most useful when media is encrypted, and essential when the middlebox or node has no access to the media decryption keys. It is also useful for codec-agnostic processing of encrypted or unencrypted media, while it also supports extensions for codec-specific information.

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 https://datatracker.ietf.org/drafts/current/.

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

This Internet-Draft will expire on May 3, 2018.

Copyright Notice

Copyright (c) 2017 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust’s Legal Provisions Relating to IETF Documents (https://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction

Many widely deployed RTP [RFC3550] topologies [RFC7667] used in modern voice and video conferencing systems include a centralized component that acts as an RTP switch. It receives voice and video streams from each participant, which may be encrypted using SRTP [RFC3711], or extensions that provide participants with private media via end-to-end encryption where the switch has no access to media decryption keys. The goal is to provide a set of streams back to the participants which enable them to render the right media content. In a simple video configuration, for example, the goal will be that each participant sees and hears just the active speaker. In that case, the goal of the switch is to receive the voice and video streams from each participant, determine the active speaker based on energy in the voice packets, possibly using the client-to-mixer audio level RTP header extension [RFC6464], and select the corresponding video stream for transmission to participants; see Figure 1.

In this document, an "RTP switch" is used as a common short term for the terms "switching RTP mixer", "source projecting middlebox", "source forwarding unit/middlebox" and "video switching MCU" as discussed in [RFC7667].

```
+---+      +------------+      +---+
| A |<---->|            |<---->| B |
+---+      |            |      +---+
           |   RTP      |
+---+      |  Switch    |      +---+
| C |<---->|            |<---->| D |
+---+      +------------+      +---+
```

Figure 1: RTP switch

In order to properly support switching of video streams, the RTP switch typically needs some critical information about video frames in order to start and stop forwarding streams.

- o Because of inter-frame dependencies, it should ideally switch video streams at a point where the first frame from the new speaker can be decoded by recipients without prior frames, e.g switch on an intra-frame.
- o In many cases, the switch may need to drop frames in order to realize congestion control techniques, and needs to know which frames can be dropped with minimal impact to video quality.
- o Furthermore, it is highly desirable to do this in a payload format-agnostic way which is not specific to each different video codec. Most modern video codecs share common concepts around frame types and other critical information to make this codec-agnostic handling possible.
- o It is also desirable to be able to do this for SRTP without requiring the video switch to decrypt the packets. SRTP will encrypt the RTP payload format contents and consequently this data is not usable for the switching function without decryption, which may not even be possible in the case of end-to-end encryption of private media.

By providing meta-information about the RTP streams outside the encrypted media payload, an RTP switch can do codec-agnostic selective forwarding without decrypting the payload. This document specifies the necessary meta-information in an RTP header extension.

2. Key Words for Normative Requirements

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

3. Frame Marking RTP Header Extension

This specification uses RTP header extensions as defined in [RFC5285]. A subset of meta-information from the video stream is provided as an RTP header extension to allow an RTP switch to do generic selective forwarding of video streams encoded with potentially different video codecs.

The Frame Marking RTP header extension is encoded using the one-byte header or two-byte header as described in [RFC5285]. The one-byte header format is used for examples in this memo. The two-byte header format is used when other two-byte header extensions are present in the same RTP packet, since mixing one-byte and two-byte extensions is not possible in the same RTP packet.

This extension is only specified for Source (not Redunadancy) RTP Streams [RFC7656] that carry video payloads. It is not specified for audio payloads, nor is it specified for Redundancy RTP Streams. The (separate) specifications for Redudancy RTP Streams often include provisions for recovering any header extensions that were part of the original source packet. Such provisions SHALL be followed to recover the Frame Marking RTP header extension of the original source packet.

3.1. Extension for Non-Scalable Streams

The following RTP header extension is RECOMMENDED for non-scalable streams. It MAY also be used for scalable streams if the sender has limited or no information about stream scalability. The ID is assigned per [RFC5285], and the length is encoded as L=0 which indicates 1 octet of data.

```
 0                   1
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  ID=? |  L=0  |S|E|I|D|0 0 0 0|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The following information are extracted from the media payload and sent in the Frame Marking RTP header extension.

- o S: Start of Frame (1 bit) - MUST be 1 in the first packet in a frame; otherwise MUST be 0.
- o E: End of Frame (1 bit) - MUST be 1 in the last packet in a frame; otherwise MUST be 0.
- o I: Independent Frame (1 bit) - MUST be 1 for frames that can be decoded independent of temporally prior frames, e.g. intra-frame, VPX keyframe, H.264 IDR [RFC6184], H.265 IDR/CRA/BLA/RAP [RFC7798]; otherwise MUST be 0.
- o D: Discardable Frame (1 bit) - MUST be 1 for frames that can be discarded, and still provide a decodable media stream; otherwise MUST be 0.
- o The remaining (4 bits) - MUST be 0 for non-scalable streams.

3.2. Extension for Scalable Streams

The following RTP header extension is RECOMMENDED for scalable streams. It MAY also be used for non-scalable streams, in which case TID, LID and TL0PICIDX MUST be 0. The ID is assigned per [RFC5285], and the length is encoded as L=2 which indicates 3 octets of data.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  ID=? |  L=2  |S|E|I|D|B| TID |   LID         |    TL0PICIDX  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The following information are extracted from the media payload and sent in the Frame Marking RTP header extension.

- o S: Start of Frame (1 bit) - MUST be 1 in the first packet in a frame within a layer; otherwise MUST be 0.
- o E: End of Frame (1 bit) - MUST be 1 in the last packet in a frame within a layer; otherwise MUST be 0.
- o I: Independent Frame (1 bit) - MUST be 1 for frames that can be decoded independent of temporally prior frames, e.g. intra-frame, VPX keyframe, H.264 IDR [RFC6184], H.265 IDR/CRA/BLA/RAP [RFC7798]; otherwise MUST be 0. Note that this bit only signals temporal independence, so it can be 1 in spatial or quality enhancement layers that depend on temporally co-located layers but not temporally prior frames.
- o D: Discardable Frame (1 bit) - MUST be 1 for frames that can be discarded, and still provide a decodable media stream; otherwise MUST be 0.
- o B: Base Layer Sync (1 bit) - MUST be 1 if this frame only depends on the base layer; otherwise MUST be 0. If no scalability is used, this MUST be 0.

- o TID: Temporal ID (3 bits) - The base temporal layer starts with 0, and increases with 1 for each higher temporal layer/sub-layer. If no scalability is used, this MUST be 0.
- o LID: Layer ID (8 bits) - Identifies the spatial and quality layer encoded. If no scalability is used, this MUST be 0 or omitted. When omitted, TL0PICIDX MUST also be omitted.
- o TL0PICIDX: Temporal Layer 0 Picture Index (8 bits) - Running index of base temporal layer 0 frames when TID is 0. When TID is not 0, this indicates a dependency on the given index. If no scalability is used, this MUST be 0 or omitted. When omitted, LID MUST also be omitted.

The layer information contained in TID and LID convey useful aspects of the layer structure that can be utilized in selective forwarding. Without further information about the layer structure, these identifiers can only be used for relative priority of layers. They convey a layer hierarchy with TID=0 and LID=0 identifying the base layer. Higher values of TID identify higher temporal layers with higher frame rates. Higher values of LID identify higher spatial and/or quality layers with higher resolutions and/or bitrates.

With further information, for example, possible future RTCP SDES items that convey full layer structure information, it may be possible to map these TIDs and LIDs to specific frame rates, resolutions and bitrates. Such additional layer information may be useful for forwarding decisions in the RTP switch, but is beyond the scope of this memo. The relative layer information is still useful for many selective forwarding decisions even without such additional layer information.

3.2.1. Layer ID Mappings for Scalable Streams

3.2.1.1. H265 LID Mapping

The following shows the H265 [RFC7798] LayerID (6 bits) and TID (3 bits) from the NAL unit header mapped to the generic LID and TID fields.

The I bit MUST be 1 when the NAL unit type is 16-23 (inclusive), otherwise it MUST be 0.

The S and E bits MUST match the corresponding bits in PACI:PHES:TSCI payload structures.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  ID=2 |  L=2  |S|E|I|D|B| TID |0|0|  LayerID  |    TL0PICIDX  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

3.2.1.2. H264-SVC LID Mapping

The following shows H264-SVC [RFC6190] Layer encoding information (3 bits for spatial/dependency layer, 4 bits for quality layer and 3 bits for temporal layer) mapped to the generic LID and TID fields.

The S, E, I and D bits MUST match the corresponding bits in PACSI payload structures.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  ID=2 |  L=2  |S|E|I|D|B| TID |0| DID |  QID  |    TL0PICIDX  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

3.2.1.3. H264 (AVC) LID Mapping

The following shows the header extension for H264 (AVC) [RFC6184] that contains only temporal layer information.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  ID=2 |  L=2  |S|E|I|D|B| TID |0|0|0|0|0|0|0|0|    TL0PICIDX  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

3.2.1.4. VP8 LID Mapping

The following shows the header extension for VP8 [RFC7741] that contains only temporal layer information.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  ID=2 |  L=2  |S|E|I|D|B| TID |0|0|0|0|0|0|0|0|    TL0PICIDX  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

3.2.1.5. Future Codec LID Mapping

The RTP payload format specification for future video codecs SHOULD include a section describing the LID mapping and TID mapping for the codec. For example, the LID/TID mapping for the VP9 codec is described in the VP9 RTP Payload Format [I-D.ietf-payload-vp9].

3.3. Signaling Information

The URI for declaring this header extension in an extmap attribute is "urn:ietf:params:rtp-hdrext:framemarking". It does not contain any extension attributes.

An example attribute line in SDP:

```
a=extmap:3 urn:ietf:params:rtp-hdrext:framemarking
```

3.4. Usage Considerations

The header extension values MUST represent what is already in the RTP payload.

When an RTP switch needs to discard a received video frame due to congestion control considerations, it is RECOMMENDED that it preferably drop frames marked with the D (Discardable) bit set, or the highest values of TID and LID, which indicate the highest temporal and spatial/quality enhancement layers, since those typically have fewer dependenices on them than lower layers.

When an RTP switch wants to forward a new video stream to a receiver, it is RECOMMENDED to select the new video stream from the first switching point with the I (Independent) bit set and forward the same. An RTP switch can request a media source to generate a switching point by sending Full Intra Request (RTCP FIR) as defined in [RFC5104], for example.

3.4.1. Relation to Layer Refresh Request (LRR)

Receivers can use the Layer Refresh Request (LRR) [I-D.ietf-avtext-lrr] RTCP feedback message to upgrade to a higher layer in scalable encodings. The TID/LID values and formats used in LRR messages MUST correspond to the same values and formats specified in Section 3.2.

3.4.2. Scalability Structures

The LID and TID information is most useful for fixed scalability structures, such as nested hierarchical temporal layering structures, where each temporal layer only references lower temporal layers or the base temporal layer. The LID and TID information is less useful, or even not useful at all, for complex, irregular scalability structures that do not conform to common, fixed patterns of interlayer dependencies and referencing structures. Therefore it is RECOMMENDED to use LID and TID information for RTP switch forwarding decisions only in the case of temporally nested scalability structures, and it is NOT RECOMMENDED for other (more complex or irregular) scalability structures.

4. Security Considerations

In the Secure Real-Time Transport Protocol (SRTP) [RFC3711], RTP header extensions are authenticated but usually not encrypted. When header extensions are used some of the payload type information are exposed and visible to middle boxes. The encrypted media data is not exposed, so this is not seen as a high risk exposure.

5. Acknowledgements

Many thanks to Bernard Aboba, Jonathan Lennox, and Stephan Wenger for their inputs.

6. IANA Considerations

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

Extension URI: urn:ietf:params:rtp-hdrext:framemarkinginfo
Description: Frame marking information for video streams
Contact: mzanaty@cisco.com
Reference: RFC XXXX

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

7. References

7.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

7.2. Informative References

[I-D.ietf-avtext-lrr]
Lennox, J., Hong, D., Uberti, J., Holmer, S., and M. Flodman, "The Layer Refresh Request (LRR) RTCP Feedback Message", draft-ietf-avtext-lrr-07 (work in progress), July 2017.

[I-D.ietf-payload-vp9]
Uberti, J., Holmer, S., Flodman, M., Lennox, J., and D. Hong, "RTP Payload Format for VP9 Video", draft-ietf-payload-vp9-04 (work in progress), July 2017.

[RFC3550] Schulzrinne, H., Casner, S., Frederick, R., and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications", STD 64, RFC 3550, DOI 10.17487/RFC3550, July 2003, <https://www.rfc-editor.org/info/rfc3550>.

[RFC3711] Baugher, M., McGrew, D., Naslund, M., Carrara, E., and K. Norrman, "The Secure Real-time Transport Protocol (SRTP)", RFC 3711, DOI 10.17487/RFC3711, March 2004, <https://www.rfc-editor.org/info/rfc3711>.

[RFC5104] Wenger, S., Chandra, U., Westerlund, M., and B. Burman, "Codec Control Messages in the RTP Audio-Visual Profile with Feedback (AVPF)", RFC 5104, DOI 10.17487/RFC5104, February 2008, <https://www.rfc-editor.org/info/rfc5104>.

[RFC5285] Singer, D. and H. Desineni, "A General Mechanism for RTP Header Extensions", RFC 5285, DOI 10.17487/RFC5285, July 2008, <https://www.rfc-editor.org/info/rfc5285>.

[RFC6184] Wang, Y., Even, R., Kristensen, T., and R. Jesup, "RTP Payload Format for H.264 Video", RFC 6184, DOI 10.17487/RFC6184, May 2011, <https://www.rfc-editor.org/info/rfc6184>.

[RFC6190] Wenger, S., Wang, Y., Schierl, T., and A. Eleftheriadis, "RTP Payload Format for Scalable Video Coding", RFC 6190, DOI 10.17487/RFC6190, May 2011, <https://www.rfc-editor.org/info/rfc6190>.

[RFC6464] Lennox, J., Ed., Ivov, E., and E. Marocco, "A Real-time Transport Protocol (RTP) Header Extension for Client-to-Mixer Audio Level Indication", RFC 6464, DOI 10.17487/RFC6464, December 2011, <https://www.rfc-editor.org/info/rfc6464>.

[RFC7656] Lennox, J., Gross, K., Nandakumar, S., Salgueiro, G., and B. Burman, Ed., "A Taxonomy of Semantics and Mechanisms for Real-Time Transport Protocol (RTP) Sources", RFC 7656, DOI 10.17487/RFC7656, November 2015, <https://www.rfc-editor.org/info/rfc7656>.

[RFC7667] Westerlund, M. and S. Wenger, "RTP Topologies", RFC 7667, DOI 10.17487/RFC7667, November 2015, <https://www.rfc-editor.org/info/rfc7667>.

[RFC7741] Westin, P., Lundin, H., Glover, M., Uberti, J., and F. Galligan, "RTP Payload Format for VP8 Video", RFC 7741, DOI 10.17487/RFC7741, March 2016, <https://www.rfc-editor.org/info/rfc7741>.

[RFC7798] Wang, Y., Sanchez, Y., Schierl, T., Wenger, S., and M. Hannuksela, "RTP Payload Format for High Efficiency Video Coding (HEVC)", RFC 7798, DOI 10.17487/RFC7798, March 2016, <https://www.rfc-editor.org/info/rfc7798>.

Authors' Addresses

Mo Zanaty
Cisco Systems
170 West Tasman Drive
San Jose, CA 95134
US

Email: mzanaty@cisco.com

Espen Berger
Cisco Systems

Phone: +47 98228179
Email: espeberg@cisco.com

Suhas Nandakumar
Cisco Systems
170 West Tasman Drive
San Jose, CA 95134
US

Email: snandaku@cisco.com

Payload Working Group
Internet-Draft
Intended status: Standards Track
Expires: December 31, 2017

J. Lennox
D. Hong
Vidyo
J. Uberti
S. Holmer
M. Flodman
Google

June 29, 2017

The Layer Refresh Request (LRR) RTCP Feedback Message
draft-ietf-avtext-lrr-07

Abstract

This memo describes the RTCP Payload-Specific Feedback Message "Layer Refresh Request" (LRR), which can be used to request a state refresh of one or more substreams of a layered media stream. It also defines its use with several RTP payloads for scalable media formats.

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 31, 2017.

Copyright Notice

Copyright (c) 2017 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust’s Legal Provisions Relating to IETF Documents (http://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction

This memo describes an RTCP [RFC3550] Payload-Specific Feedback Message [RFC4585] "Layer Refresh Request" (LRR). It is designed to allow a receiver of a layered media stream to request that one or more of its substreams be refreshed, such that it can then be decoded by an endpoint which previously was not receiving those layers, without requiring that the entire stream be refreshed (as it would be if the receiver sent a Full Intra Request (FIR); [RFC5104] see also [RFC8082]).

The feedback message is applicable both to temporally and spatially scaled streams, and to both single-stream and multi-stream scalability modes.

2. Conventions, Definitions and Acronyms

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

2.1. Terminology

A "Layer Refresh Point" is a point in a scalable stream after which a decoder, which previously had been able to decode only some (possibly none) of the available layers of stream, is able to decode a greater number of the layers.

For spatial (or quality) layers, in normal encoding, a subpicture can depend both on earlier pictures of that spatial layer and also on lower-layer pictures of the current picture. A layer refresh, however, typically requires that a spatial layer picture be encoded in a way that references only the lower-layer subpictures of the current picture, not any earlier pictures of that spatial layer. Additionally, the encoder must promise that no earlier pictures of that spatial layer will be used as reference in the future.

However, even in a layer refresh, layers other than the ones being refreshed may still maintain dependency on earlier content of the stream. This is the difference between a layer refresh and a Full Intra Request [RFC5104]. This minimizes the coding overhead of refresh to only those parts of the stream that actually need to be refreshed at any given time.

An illustration of spatial layer refresh of an enhancement layer is shown below. <-- indicates a coding dependency.

```
... <--  S1  <--  S1       S1  <--  S1  <-- ...
          |        |        |        |
         \/       \/       \/       \/
... <--  S0  <--  S0  <--  S0  <--  S0  <-- ...

          1        2        3        4
```

Figure 1

In Figure 1, frame 3 is a layer refresh point for spatial layer S1; a decoder which had previously only been decoding spatial layer S0 would be able to decode layer S1 starting at frame 3.

An illustration of spatial layer refresh of a base layer is shown below. <-- indicates a coding dependency.

```
     ... <--  S1  <--  S1  <--  S1  <--  S1  <-- ...
               |        |        |        |
              \/       \/       \/       \/
     ... <--  S0  <--  S0       S0  <--  S0  <-- ...

               1        2        3        4
```

Figure 2

In Figure 2, frame 3 is a layer refresh point for spatial layer S0; a decoder which had previously not been decoding the stream at all could decode layer S0 starting at frame 3.

For temporal layers, while normal encoding allows frames to depend on earlier frames of the same temporal layer, layer refresh requires that the layer be "temporally nested", i.e. use as reference only earlier frames of a lower temporal layer, not any earlier frames of this temporal layer, and also promise that no future frames of this temporal layer will reference frames of this temporal layer before the refresh point. In many cases, the temporal structure of the stream will mean that all frames are temporally nested, in which case decoders will have no need to send LRR messages for the stream.

An illustration of temporal layer refresh is shown below. <-- indicates a coding dependency.

```
        ...  <----- T1  <------ T1          T1  <------ ...
                    /           /           /
                  |_          |_          |_
     ... <--  T0  <------ T0  <------ T0  <------ T0  <--- ...

               1     2     3     4     5     6     7
```

Figure 3

In Figure 3, frame 6 is a layer refresh point for temporal layer T1; a decoder which had previously only been decoding temporal layer T0 would be able to decode layer T1 starting at frame 6.

An illustration of an inherently temporally nested stream is shown below. <-- indicates a coding dependency.

```
                   T1          T1          T1
                  /           /           /
                 |_          |_          |_
    ... <--  T0  <------ T0  <------ T0  <------ T0  <--- ...

              1     2     3     4     5     6     7
```

Figure 4

In Figure 4, the stream is temporally nested in its ordinary structure; a decoder receiving layer T0 can begin decoding layer T1 at any point.

A "Layer Index" is a numeric label for a specific spatial and temporal layer of a scalable stream. It consists of the pair of a "temporal ID" identifying the temporal layer, and a "layer ID" identifying the spatial or quality layer. The details of how layers of a scalable stream are labeled are codec-specific. Details for several codecs are defined in Section 4.

3. Layer Refresh Request

A layer refresh frame can be requested by sending a Layer Refresh Request (LRR), which is an RTP Control Protocol (RTCP) [RFC3550] payload-specific feedback message [RFC4585] asking the encoder to encode a frame which makes it possible to upgrade to a higher layer. The LRR contains one or two tuples, indicating the temporal and spatial layer the decoder wants to upgrade to, and (optionally) the currently highest temporal and spatial layer the decoder can decode.

The specific format of the tuples, and the mechanism by which a receiver recognizes a refresh frame, is codec-dependent. Usage for several codecs is discussed in Section 4.

LRR follows the model of the Full Intra Request (FIR) [RFC5104] (Section 3.5.1) for its retransmission, reliability, and use in multipoint conferences.

The LRR message is identified by RTCP packet type value PT=PSFB and FMT=TBD. The FCI field MUST contain one or more LRR entries. Each entry applies to a different media sender, identified by its SSRC.

[NOTE TO RFC Editor: Please replace "TBD" with the IANA-assigned payload-specific feedback number.]

3.1. Message Format

The Feedback Control Information (FCI) for the Layer Refresh Request consists of one or more FCI entries, the content of which is depicted in Figure 5. The length of the LRR feedback message MUST be set to 2+3*N 32-bit words, where N is the number of FCI entries.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                              SSRC                             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Seq nr.       |C| Payload Type| Reserved                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| RES     | TTID| TLID          | RES     | CTID| CLID          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 5

SSRC (32 bits) The SSRC value of the media sender that is requested to send a layer refresh point.

Seq nr. (8 bits) Command sequence number. The sequence number space is unique for each pairing of the SSRC of command source and the SSRC of the command target. The sequence number SHALL be increased by 1 for each new command (modulo 256, so the value after 255 is 0). A repetition SHALL NOT increase the sequence number. The initial value is arbitrary.

C (1 bit) A flag bit indicating whether the "Current Temporal Layer ID (CTID)" and "Current Layer ID (CLID)" fields are present in the FCI. If this bit is 0, the sender of the LRR message is requesting refresh of all layers up to and including the target layer.

Payload Type (7 bits) The RTP payload type for which the LRR is being requested. This gives the context in which the target layer index is to be interpreted.

Reserved (RES) (three separate fields, 16 bits / 5 bits / 5 bits) All bits SHALL be set to 0 by the sender and SHALL be ignored on reception.

Target Temporal Layer ID (TTID) (3 bits) The temporal ID of the target layer for which the receiver wishes a refresh point.

Target Layer ID (TLID) (8 bits) The layer ID of the target spatial or quality layer for which the receiver wishes a refresh point. Its format is dependent on the payload type field.

Current Temporal Layer ID (CTID) (3 bits) If C is 1, the ID of the current temporal layer being decoded by the receiver. This message is not requesting refresh of layers at or below this layer. If C is 0, this field SHALL be set to 0 by the sender and SHALL be ignored on reception.

Current Layer ID (CLID) (8 bits) If C is 1, the layer ID of the current spatial or quality layer being decoded by the receiver. This message is not requesting refresh of layers at or below this layer. If C is 0, this field SHALL be set to 0 by the sender and SHALL be ignored on reception.

When C is 1, TTID MUST NOT be less than CTID, and TLID MUST NOT be less than CLID; at least one of TTID or TLID MUST be greater than CTID or CLID respectively. That is to say, the target layer index <TTID, TLID> MUST be a layer upgrade from the current layer index <CTID, CLID>. A sender MAY request an upgrade in both temporal and spatial/quality layers simultaneously.

A receiver receiving an LRR feedback packet which does not satisfy the requirements of the previous paragraph, i.e. one where the C bit is present but TTID is less than CTID or TLID is less than CLID, MUST discard the request.

Note: the syntax of the TTID, TLID, CTID, and CLID fields match, by design, the TID and LID fields in [I-D.ietf-avtext-framemarking].

3.2. Semantics

Within the common packet header for feedback messages (as defined in section 6.1 of [RFC4585]), the "SSRC of packet sender" field indicates the source of the request, and the "SSRC of media source" is not used and SHALL be set to 0. The SSRCs of the media senders to which the LRR command applies are in the corresponding FCI entries. A LRR message MAY contain requests to multiple media senders, using one FCI entry per target media sender.

Upon reception of LRR, the encoder MUST send a decoder refresh point (see Section 2.1) as soon as possible.

The sender MUST respect bandwidth limits provided by the application of congestion control, as described in Section 5 of [RFC5104]. As layer refresh points will often be larger than non-refreshing frames, this may restrict a sender's ability to send a layer refresh point quickly.

LRR MUST NOT be sent as a reaction to picture losses due to packet loss or corruption -- it is RECOMMENDED to use PLI [RFC4585] instead. LRR SHOULD be used only in situations where there is an explicit change in decoders' behavior, for example when a receiver will start decoding a layer which it previously had been discarding.

4. Usage with specific codecs

In order for LRR to be used with a scalable codec, the format of the temporal and layer ID fields (for both the target and current layer indices) needs to be specified for that codec's RTP packetization. New RTP packetization specifications for scalable codecs SHOULD define how this is done. (The VP9 payload [I-D.ietf-payload-vp9], for instance, has done so.) If the payload also specifies how it is used with the Frame Marking RTP Header Extension [I-D.ietf-avtext-framemarking], the syntax MUST be defined in the same manner as the TID and LID fields in that header.

4.1. H264 SVC

H.264 SVC [RFC6190] defines temporal, dependency (spatial), and quality scalability modes.

```
+---------------+---------------+
|0|1|2|3|4|5|6|7|0|1|2|3|4|5|6|7|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| RES     | TID |R|  DID  | QID |
+---------------+---------------+
```

Figure 6

Figure 6 shows the format of the layer index fields for H.264 SVC streams. The "R" and "RES" fields MUST be set to 0 on transmission and ignored on reception. See [RFC6190] Section 1.1.3 for details on the DID, QID, and TID fields.

A dependency or quality layer refresh of a given layer in H.264 SVC can be identified by the "I" bit (idr_flag) in the extended NAL unit header, present in NAL unit types 14 (prefix NAL unit) and 20 (coded scalable slice). Layer refresh of the base layer can also be identified by its NAL unit type of its coded slices, which is "5" rather than "1". A dependency or quality layer refresh is complete once this bit has been seen on all the appropriate layers (in decoding order) above the current layer index (if any, or beginning from the base layer if not) through the target layer index.

Note that as the "I" bit in a PACSI header is set if the corresponding bit is set in any of the aggregated NAL units it describes; thus, it is not sufficient to identify layer refresh when NAL units of multiple dependency or quality layers are aggregated.

In H.264 SVC, temporal layer refresh information can be determined from various Supplemental Encoding Information (SEI) messages in the bitstream.

Whether an H.264 SVC stream is scalably nested can be determined from the Scalability Information SEI message's temporal_id_nesting flag. If this flag is set in a stream's currently applicable Scalability Information SEI, receivers SHOULD NOT send temporal LRR messages for that stream, as every frame is implicitly a temporal layer refresh point. (The Scalability Information SEI message may also be available in the signaling negotiation of H.264 SVC, as the sprop-scalability-info parameter.)

If a stream's temporal_id_nesting flag is not set, the Temporal Level Switching Point SEI message identifies temporal layer switching points. A temporal layer refresh is satisfied when this SEI message is present in a frame with the target layer index, if the message's delta_frame_num refers to a frame with the requested current layer index. (Alternately, temporal layer refresh can also be satisfied by a complete state refresh, such as an IDR.) Senders which support receiving LRR for non-temporally-nested streams MUST insert Temporal Level Switching Point SEI messages as appropriate.

4.2. VP8

The VP8 RTP payload format [RFC7741] defines temporal scalability modes. It does not support spatial scalability.

```
            +---------------+---------------+
            |0|1|2|3|4|5|6|7|0|1|2|3|4|5|6|7|
            +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
            | RES     | TID | RES           |
            +---------------+---------------+
```

Figure 7

Figure 7 shows the format of the layer index field for VP8 streams. The "RES" fields MUST be set to 0 on transmission and be ignored on reception. See [RFC7741] Section 4.2 for details on the TID field.

A VP8 layer refresh point can be identified by the presence of the "Y" bit in the VP8 payload header. When this bit is set, this and all subsequent frames depend only on the current base temporal layer.

On receipt of an LRR for a VP8 stream, A sender which supports LRR MUST encode the stream so it can set the Y bit in a packet whose temporal layer is at or below the target layer index.

Note that in VP8, not every layer switch point can be identified by the Y bit, since the Y bit implies layer switch of all layers, not just the layer in which it is sent. Thus the use of LRR with VP8 can result in some inefficiency in transmision. However, this is not expected to be a major issue for temporal structures in normal use.

4.3. H265

The initial version of the H.265 payload format [RFC7798] defines temporal scalability, with protocol elements reserved for spatial or other scalability modes (which are expected to be defined in a future version of the specification).

```
+---------------+---------------+
|0|1|2|3|4|5|6|7|0|1|2|3|4|5|6|7|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| RES     | TID |RES|  LayerId  |
+---------------+---------------+
```

Figure 8

Figure 8 shows the format of the layer index field for H.265 streams. The "RES" fields MUST be set to 0 on transmission and ignored on reception. See [RFC7798] Section 1.1.4 for details on the LayerId and TID fields.

H.265 streams signal whether they are temporally nested, using the vps_temporal_id_nesting_flag in the Video Parameter Set (VPS), and the sps_temporal_id_nesting_flag in the Sequence Parameter Set (SPS). If this flag is set in a stream's currently applicable VPS or SPS, receivers SHOULD NOT send temporal LRR messages for that stream, as every frame is implicitly a temporal layer refresh point.

If a stream's sps_temporal_id_nesting_flag is not set, the NAL unit types 2 to 5 inclusively identify temporal layer switching points. A layer refresh to any higher target temporal layer is satisfied when a NAL unit type of 4 or 5 with TID equal to 1 more than current TID is seen. Alternatively, layer refresh to a target temporal layer can be incrementally satisfied with NAL unit type of 2 or 3. In this case, given current TID = TO and target TID = TN, layer refresh to TN is satisfied when NAL unit type of 2 or 3 is seen for TID = T1, then TID = T2, all the way up to TID = TN. During this incremental process, layer refresh to TN can be completely satisfied as soon as a NAL unit type of 2 or 3 is seen.

Of course, temporal layer refresh can also be satisfied whenever any Intra Random Access Point (IRAP) NAL unit type (with values 16-23, inclusively) is seen. An IRAP picture is similar to an IDR picture in H.264 (NAL unit type of 5 in H.264) where decoding of the picture can start without any older pictures.

In the (future) H.265 payloads that support spatial scalability, a spatial layer refresh of a specific layer can be identified by NAL units with the requested layer ID and NAL unit types between 16 and 21 inclusive. A dependency or quality layer refresh is complete once NAL units of this type have been seen on all the appropriate layers (in decoding order) above the current layer index (if any, or beginning from the base layer if not) through the target layer index.

5. Usage with different scalability transmission mechanisms

Several different mechanisms are defined for how scalable streams can be transmitted in RTP. The RTP Taxonomy [RFC7656] Section 3.7 defines three mechanisms: Single RTP Stream on a Single Media Transport (SRST), Multiple RTP Streams on a Single Media Transport (MRST), and Multiple RTP Streams on Multiple Media Transports (MRMT).

The LRR message is applicable to all these mechanisms. For MRST and MRMT mechanisms, the "media source" field of the LRR FCI is set to the SSRC of the RTP stream containing the layer indicated by the Current Layer Index (if "C" is 1), or the stream containing the base encoded stream (if "C" is 0). For MRMT, it is sent on the RTP session on which this stream is sent. On receipt, the sender MUST refresh all the layers requested in the stream, simultaneously in decode order.

6. SDP Definitions

Section 7 of [RFC5104] defines SDP procedures for indicating and negotiating support for codec control messages (CCM) in SDP. This document extends this with a new codec control command, "lrr", which indicates support of the Layer Refresh Request (LRR).

Figure 9 gives a formal Augmented Backus-Naur Form (ABNF) [RFC5234] showing this grammar extension, extending the grammar defined in [RFC5104].

```
rtcp-fb-ccm-param =/ SP "lrr"    ; Layer Refresh Request
```

Figure 9: Syntax of the "lrr" ccm

The Offer-Answer considerations defined in [RFC5104] Section 7.2 apply.

7. Security Considerations

All the security considerations of FIR feedback packets [RFC5104] apply to LRR feedback packets as well. Additionally, media senders receiving LRR feedback packets MUST validate that the payload types and layer indices they are receiving are valid for the stream they are currently sending, and discard the requests if not.

8. IANA Considerations

This document defines a new entry to the "Codec Control Messages" subregistry of the "Session Description Protocol (SDP) Parameters" registry, according to the following data:

Value name: lrr

Long name: Layer Refresh Request Command

Usable with: ccm

Mux: IDENTICAL-PER-PT

Reference: RFC XXXX

This document also defines a new entry to the "FMT Values for PSFB Payload Types" subregistry of the "Real-Time Transport Protocol (RTP) Parameters" registry, according to the following data:

Name: LRR

Long Name: Layer Refresh Request Command

Value: TBD

Reference: RFC XXXX

9. References

9.1. Normative References

[I-D.ietf-avtext-framemarking]
Berger, E., Nandakumar, S., and M. Zanaty, "Frame Marking RTP Header Extension", draft-ietf-avtext-framemarking-04 (work in progress), March 2017.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <http://www.rfc-editor.org/info/rfc2119>.

[RFC3550] Schulzrinne, H., Casner, S., Frederick, R., and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications", STD 64, RFC 3550, DOI 10.17487/RFC3550, July 2003, <http://www.rfc-editor.org/info/rfc3550>.

[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, DOI 10.17487/RFC4585, July 2006, <http://www.rfc-editor.org/info/rfc4585>.

[RFC5104] Wenger, S., Chandra, U., Westerlund, M., and B. Burman, "Codec Control Messages in the RTP Audio-Visual Profile with Feedback (AVPF)", RFC 5104, DOI 10.17487/RFC5104, February 2008, <http://www.rfc-editor.org/info/rfc5104>.

[RFC5234] Crocker, D., Ed. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", STD 68, RFC 5234, DOI 10.17487/RFC5234, January 2008, <http://www.rfc-editor.org/info/rfc5234>.

[RFC6190] Wenger, S., Wang, Y., Schierl, T., and A. Eleftheriadis, "RTP Payload Format for Scalable Video Coding", RFC 6190, DOI 10.17487/RFC6190, May 2011, <http://www.rfc-editor.org/info/rfc6190>.

[RFC7741] Westin, P., Lundin, H., Glover, M., Uberti, J., and F. Galligan, "RTP Payload Format for VP8 Video", RFC 7741, DOI 10.17487/RFC7741, March 2016, <http://www.rfc-editor.org/info/rfc7741>.

[RFC7798] Wang, Y., Sanchez, Y., Schierl, T., Wenger, S., and M. Hannuksela, "RTP Payload Format for High Efficiency Video Coding (HEVC)", RFC 7798, DOI 10.17487/RFC7798, March 2016, <http://www.rfc-editor.org/info/rfc7798>.

9.2. Informative References

[I-D.ietf-payload-vp9] Uberti, J., Holmer, S., Flodman, M., Lennox, J., and D. Hong, "RTP Payload Format for VP9 Video", draft-ietf-payload-vp9-03 (work in progress), March 2017.

[RFC7656] Lennox, J., Gross, K., Nandakumar, S., Salgueiro, G., and B. Burman, Ed., "A Taxonomy of Semantics and Mechanisms for Real-Time Transport Protocol (RTP) Sources", RFC 7656, DOI 10.17487/RFC7656, November 2015, <http://www.rfc-editor.org/info/rfc7656>.

[RFC8082] Wenger, S., Lennox, J., Burman, B., and M. Westerlund, "Using Codec Control Messages in the RTP Audio-Visual Profile with Feedback with Layered Codecs", RFC 8082, DOI 10.17487/RFC8082, March 2017, <http://www.rfc-editor.org/info/rfc8082>.

Authors' Addresses

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

Email: jonathan@vidyo.com

Danny Hong
Vidyo, Inc.
433 Hackensack Avenue
Seventh Floor
Hackensack, NJ 07601
US

Email: danny@vidyo.com

Justin Uberti
Google, Inc.
747 6th Street South
Kirkland, WA 98033
USA

Email: justin@uberti.name

Stefan Holmer
Google, Inc.
Kungsbron 2
Stockholm 111 22
Sweden

Email: holmer@google.com

Magnus Flodman
Google, Inc.
Kungsbron 2
Stockholm 111 22
Sweden

Email: mflodman@google.com

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: April 9, 2017

A. Roach
Mozilla
S. Nandakumar
Cisco Systems
P. Thatcher
Google

October 06, 2016

RTP Stream Identifier Source Description (SDES)
draft-ietf-avtext-rid-09

Abstract

This document defines and registers two new RTCP Stream Identifier Source Description (SDES) items. One, named RtpStreamId, is used for unique identification of RTP streams. The other, RepairedRtpStreamId, can be used to identify which stream a redundancy RTP stream is to be used to repair.

Status of This Memo

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

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

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

This Internet-Draft will expire on April 9, 2017.

Copyright Notice

Copyright (c) 2016 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust’s Legal Provisions Relating to IETF Documents (http://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction

RTP sessions frequently consist of multiple streams, each of which is identified at any given time by its SSRC; however, the SSRC associated with a stream is not guaranteed to be stable over its lifetime. Within a session, these streams can be tagged with a number of identifiers, including CNAMEs and MSIDs [I-D.ietf-mmusic-msid]. Unfortunately, none of these have the proper ordinality to refer to an individual stream; all such identifiers can appear in more than one stream at a time. While approaches that use unique Payload Types (PTs) per stream have been used in some applications, this is a semantic overloading of that field, and one for which its size is inadequate: in moderately complex systems that use PT to uniquely identify every potential combination of codec configuration and unique stream, it is possible to simply run out of values.

To address this situation, we define a new RTCP Stream Identifier Source Description (SDES) identifier, RtpStreamId, that uniquely identifies a single RTP stream. A key motivator for defining this identifier is the ability to differentiate among different encodings of a single Source Stream that are sent simultaneously (i.e., simulcast). This need for unique identification extends to dependent streams (e.g., where layers used by a layered codec are transmitted on separate streams).

At the same time, when redundancy RTP streams are in use, we also need an identifier that connects such streams to the RTP stream for which they are providing redundancy. For this purpose, we define an additional SDES identifier, RepairedRtpStreamId. This identifier can appear only in packets associated with a redundancy RTP stream. They carry the same value as the RtpStreamId of the RTP stream that the redundant RTP stream is correcting.

2. Terminology

In this document, the terms "source stream", "RTP stream", "source RTP stream", "dependent stream", "received RTP stream", and "redundancy RTP stream" are used as defined in [RFC7656].

The following acronyms are also used:

- CNAME: Canonical End-Point Identifier, defined in [RFC3550]
- MID: Media Identification, defined in [I-D.ietf-mmusic-sdp-bundle-negotiation]
- MSID: Media Stream Identifier, defined in [I-D.ietf-mmusic-msid]
- RTCP: Real-time Transport Control Protocol, defined in [RFC3550]
- RTP: Real-time Transport Protocol, defined in [RFC3550]
- SDES: Source Description, defined in [RFC3550]
- SSRC: Synchronization Source, defined in [RFC3550]

3. Usage of RtpStreamId and RepairedRtpStreamId in RTP and RTCP

The RTP fixed header includes the payload type number and the SSRC values of the RTP stream. RTP defines how you de-multiplex streams within an RTP session; however, in some use cases, applications need further identifiers in order to effectively map the individual RTP Streams to their equivalent payload configurations in the SDP.

This specification defines two new RTCP SDES items [RFC3550]. The first item is 'RtpStreamId', which is used to carry RTP stream identifiers within RTCP SDES packets. This makes it possible for a receiver to associate received RTP packets (identifying the RTP stream) with a media description having the format constraint specified. The second is 'RepairedRtpStreamId', which can be used in redundancy RTP streams to indicate the RTP stream repaired by a redundancy RTP stream.

To be clear: the value carried in a RepairedRtpStreamId will always match the RtpStreamId value from another RTP stream in the same session. For example, if a source RTP stream is identified by RtpStreamId "A", then any redundancy RTP stream that repairs that source RTP stream will contain a RepairedRtpStreamId of "A" (if this mechanism is being used to perform such correlation). These redundant RTP streams may also contain their own unique RtpStreamId.

This specification also uses the RTP header extension for RTCP SDES items [I-D.ietf-avtext-sdes-hdr-ext] to allow carrying RtpStreamId and RepairedRtpStreamId values in RTP packets. This allows correlation at stream startup, or after stream changes where the use of RTCP may not be sufficiently responsive. This speed of response is necessary since, in many cases, the stream cannot be properly processed until it can be identified.

RtpStreamId and RepairedRtpStreamId values are scoped by source identifier (e.g., CNAME) and by media session. When the media is multiplexed using the BUNDLE extension [I-D.ietf-mmusic-sdp-bundle-negotiation], these values are further scoped by their associated MID values. For example: an RtpStreamId of "1" may be present in the stream identified with a CNAME of "1234@example.com", and may also be present in a stream with a CNAME of "5678@example.org", and these would refer to different streams. Similarly, an RtpStreamId of "1" may be present with an MID of "A", and again with a MID of "B", and also refer to two different streams.

Note that the RepairedRtpStreamId mechanism is limited to indicating one repaired stream per redundancy stream. If systems require correlation for schemes in which a redundancy stream contains information used to repair more than one stream, they will have to use a more complex mechanism than the one defined in this specification.

As with all SDES items, RtpStreamId and RepairedRtpStreamId are limited to a total of 255 octets in length. RtpStreamId and RepairedStreamId are constrained to contain only alphanumeric characters. For avoidance of doubt, the only allowed byte values for these IDs are decimal 48 through 57, 65 through 90, and 97 through 122.

3.1. RTCP 'RtpStreamId' SDES Extension

```
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|RtpStreamId=TBD|     length    | RtpStreamId                 ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The RtpStreamId payload is ASCII encoded and is not null-terminated.

RFC EDITOR NOTE: Please replace TBD with the assigned SDES identifier value.

3.2. RTCP 'RepairedRtpStreamId' SDES Extension

```
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|Repaired...=TBD|     length    | RepairRtpStreamId           ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The RepairedRtpStreamId payload is ASCII encoded and is not null-terminated.

RFC EDITOR NOTE: Please replace TBD with the assigned SDES identifier value.

3.3. RTP 'RtpStreamId' and 'RepairedRtpStreamId' Header Extensions

Because recipients of RTP packets will typically need to know which streams they correspond to immediately upon receipt, this specification also defines a means of carrying RtpStreamId and RepairedRtpStreamId identifiers in RTP extension headers, using the technique described in [I-D.ietf-avtext-sdes-hdr-ext].

As described in that document, the header extension element can be encoded using either the one-byte or two-byte header, and the identification-tag payload is ASCII-encoded.

As the identifier is included in an RTP header extension, there should be some consideration given to the packet expansion caused by the identifier. To avoid Maximum Transmission Unit (MTU) issues for the RTP packets, the header extension's size needs to be taken into account when encoding media. Note that the set of header extensions included in the packet needs to be padded to the next 32-bit boundary [RFC5285].

In many cases, a one-byte identifier will be sufficient to distinguish streams in a session; implementations are strongly encouraged to use the shortest identifier that fits their purposes. Implementors are warned, in particular, not to include any information in the identifier that is derived from potentially user-identifying information, such as user ID or IP address. To avoid identification of specific implementations based on their pattern of tag generation, implementations are encouraged to use a simple scheme that starts with the ASCII digit "1", and increments by one for each subsequent identifier.

4. IANA Considerations

4.1. New RtpStreamId SDES item

RFC EDITOR NOTE: Please replace RFCXXXX with the RFC number of this document.

RFC EDITOR NOTE: Please replace TBD with the assigned SDES identifier value.

This document adds the RtpStreamId SDES item to the IANA "RTP SDES item types" registry as follows:

```
Value:          TBD
Abbrev.:        RtpStreamId
Name:           RTP Stream Identifier
Reference:      RFCXXXX
```

4.2. New RepairRtpStreamId SDES item

RFC EDITOR NOTE: Please replace RFCXXXX with the RFC number of this document.

RFC EDITOR NOTE: Please replace TBD with the assigned SDES identifier value.

This document adds the RepairedRtpStreamId SDES item to the IANA "RTP SDES item types" registry as follows:

```
Value:          TBD
Abbrev.:        RepairedRtpStreamId
Name:           Repaired RTP Stream Identifier
Reference:      RFCXXXX
```

4.3. New RtpStreamId Header Extension URI

RFC EDITOR NOTE: Please replace RFCXXXX with the RFC number of this document.

This document defines a new extension URI in the RTP SDES Compact Header Extensions sub-registry of the RTP Compact Header Extensions registry sub-registry, as follows

Extension URI: urn:ietf:params:rtp-hdrext:sdes:rtp-stream-id
Description: RTP Stream Identifier Contact: adam@nostrum.com
Reference: RFCXXXX

4.4. New RepairRtpStreamId Header Extension URI

RFC EDITOR NOTE: Please replace RFCXXXX with the RFC number of this document.

This document defines a new extension URI in the RTP SDES Compact Header Extensions sub-registry of the RTP Compact Header Extensions registry sub-registry, as follows

Extension URI: urn:ietf:params:rtp-hdrext:sdes:repaired-rtp-sream-id
Description: RTP Repaired Stream Identifier Contact: adam@nostrum.com
Reference: RFCXXXX

5. Security Considerations

Although the identifiers defined in this document are limited to be strictly alphanumeric, SDES items have the potential to carry any string. As a consequence, there exists a risk that it might carry privacy-sensitive information. Implementations need to take care when generating identifiers so that they do not contain information that can identify the user or allow for long term tracking of the device. Following the generation recommendations in Section 3.3 will result in non-instance-specific labels, with only minor fingerprinting possibilities in the total number of used RtpStreamIds and RepairedRtpStreamIds.

Even if the SDES items are generated to convey as little information as possible, implementors are strongly encouraged to encrypt SDES items - both in RTCP and RTP header extensions - so as to preserve privacy against third parties.

As the SDES items are used for identification of the RTP streams for different application purposes, it is important that the intended values are received. An attacker, either a third party or malicious RTP middlebox, that removes, or changes the values for these SDES items, can severely impact the application. The impact can include failure to decode or display the media content of the RTP stream. It can also result in incorrectly attributing media content to identifiers of the media source, such as incorrectly identifying the speaker. To prevent this from occurring due to third party attacks, integrity and source authentication is needed.

Options for Securing RTP Sessions [RFC7201] discusses options for how encryption, integrity and source authentication can be accomplished.

6. Acknowledgements

Many thanks for review and input from Cullen Jennings, Magnus Westerlund, Colin Perkins, Jonathan Lennox, and Paul Kyzivat. Magnus Westerlund provided substantially all of the Security Considerations section.

7. References

7.1. Normative References

[I-D.ietf-avtext-sdes-hdr-ext]
Westerlund, M., Burman, B., Even, R., and M. Zanaty, "RTP Header Extension for RTCP Source Description Items", draft-ietf-avtext-sdes-hdr-ext-07 (work in progress), June 2016.

[I-D.ietf-mmusic-sdp-bundle-negotiation]
Holmberg, C., Alvestrand, H., and C. Jennings, "Negotiating Media Multiplexing Using the Session Description Protocol (SDP)", draft-ietf-mmusic-sdp-bundle-negotiation-32 (work in progress), August 2016.

[RFC3550] Schulzrinne, H., Casner, S., Frederick, R., and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications", STD 64, RFC 3550, DOI 10.17487/RFC3550, July 2003, <http://www.rfc-editor.org/info/rfc3550>.

[RFC5285] Singer, D. and H. Desineni, "A General Mechanism for RTP Header Extensions", RFC 5285, DOI 10.17487/RFC5285, July 2008, <http://www.rfc-editor.org/info/rfc5285>.

[RFC7656] Lennox, J., Gross, K., Nandakumar, S., Salgueiro, G., and B. Burman, Ed., "A Taxonomy of Semantics and Mechanisms for Real-Time Transport Protocol (RTP) Sources", RFC 7656, DOI 10.17487/RFC7656, November 2015, <http://www.rfc-editor.org/info/rfc7656>.

7.2. Informative References

[I-D.ietf-mmusic-msid]
Alvestrand, H., "WebRTC MediaStream Identification in the Session Description Protocol", draft-ietf-mmusic-msid-15 (work in progress), July 2016.

[RFC7201] Westerlund, M. and C. Perkins, "Options for Securing RTP Sessions", RFC 7201, DOI 10.17487/RFC7201, April 2014, <http://www.rfc-editor.org/info/rfc7201>.

Authors' Addresses

Adam Roach
Mozilla

Email: adam@nostrum.com

Suhas Nandakumar
Cisco Systems

Email: snandaku@cisco.com

Peter Thatcher
Google

Email: pthatcher@google.com

AVTEXT Working Group
INTERNET-DRAFT
Intended Status: Standards Track
Expires: February 4, 2017
J. Xia
R. Even
R. Huang
Huawei
L. Deng
China Mobile
August 3, 2016

RTP/RTCP extension for RTP Splicing Notification
draft-ietf-avtext-splicing-notification-09

Abstract

Content splicing is a process that replaces the content of a main multimedia stream with other multimedia content, and delivers the substitutive multimedia content to the receivers for a period of time. The splicer is designed to handle RTP splicing and needs to know when to start and end the splicing.

This memo defines two RTP/RTCP extensions to indicate the splicing related information to the splicer: an RTP header extension that conveys the information in-band and an RTCP packet that conveys the information out-of-band.

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), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

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."

The list of current Internet-Drafts can be accessed at http://www.ietf.org/1id-abstracts.html

The list of Internet-Draft Shadow Directories can be accessed at http://www.ietf.org/shadow.html

Copyright and License Notice

Copyright (c) 2016 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (http://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1 Introduction

Splicing is a process that replaces some multimedia content with other multimedia content and delivers the substitutive multimedia content to the receivers for a period of time. In some predictable splicing cases, e.g., advertisement insertion, the splicing duration needs to be inside of the specific, pre-designated time slot. Certain timing information about when to start and end the splicing must be first acquired by the splicer in order to start the splicing. This document refers to this information as the Splicing Interval.

[SCTE35] provides a method that encapsulates the Splicing Interval inside the MPEG2-TS layer in cable TV systems. When transported in RTP, an middle box designed as the splicer to decode the RTP packets and search for the Splicing Interval inside the payloads is required. The need for such processing increases the workload of the middle box and limits the number of RTP sessions the middle box can support.

The document defines an RTP header extension [RFC5285bis] used by the main RTP sender to provide the Splicing Interval by including it in the RTP packets.

However, the Splicing Interval conveyed in the RTP header extension might not reach the splicer successfully. Any splicing un-aware middlebox on the path between the RTP sender might strip this RTP header extension.

To increase robustness against such case, the document also defines a new RTCP packet type to carry the same Splicing Interval to the splicer. Since RTCP is also unreliable and may not be so immediate as the in-band way, it’s only considered as a complement to the RTP header extension.

1.1 Terminology

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

In addition, we define following terminologies:

Main RTP sender:

The sender of RTP packets carrying the main RTP stream.

Splicer:

An intermediary node that inserts substitutive content into a main

RTP stream. The splicer sends substitutive content to the RTP receiver instead of the main content during splicing. It is also responsible for processing RTCP traffic between the RTP sender and the RTP receiver.

Splicing-In Point

A virtual point in the RTP stream, suitable for substitutive content entry, typically in the boundary between two independently decodable frames.

Splicing-Out Point

A virtual point in the RTP stream, suitable for substitutive content exit, typically in the boundary between two independently decodable frames.

Splicing Interval:

The NTP-format timestamps, representing the main RTP sender wallclock time, for the Splicing-In point and Splicing-Out point per [RFC6828] allowing the splicer to know when to start and end the RTP splicing.

Substitutive RTP Sender:

The sender of RTP packets carrying the RTP stream that will replace the content in the main RTP stream.

2 Overview

2.1 Overview of RTP Splicing

RTP Splicing is intended to replace some multimedia content with certain substitutive multimedia content, and then forward it to the receivers for a period of time. This process is authorized by the main RTP sender that offers a specific time window for inserting the substitutive multimedia content in the main content. A typical usage is that IPTV service provider uses its own regional advertising content to replace national advertising content, the time window of which is explicitly indicated by the IPTV service provider.

The splicer is a middlebox handling RTP splicing. It receives main content and substitutive content simultaneously but only chooses to send one of them to the receiver at any point of time. When RTP splicing begins, the splicer sends the substitutive content to the receivers instead of the main content. When RTP splicing ends, the splicer switches back to sending the main content to the receivers.

This implies that the receiver is explicitly configured to receive the traffic via the splicer, and will return any RTCP feedback to it in the presence of the splicer.

The middlebox working as the splicer can be implemented as either an RTP mixer or as an RTP translator. If implemented as an RTP mixer, the splicer will use its own SSRC, sequence number space, and timing model when generating the output stream to receivers, using the CSRC list to indicate whether the original or substitutive content is being delivered. The splicer, on behalf of the content provider, can omit the CSRC list from the RTP packets it generates. This simplifies the design of the receivers, since they don’t need to parse the CSRC list, but makes it harder to determine when the splicing is taking place (it requires inspection of the RTP payload data, rather than just the RTP headers). A splicer working as an RTP mixer splits the flow between the sender and receiver into two, and requires separate control loops, for RTCP and congestion control. [RFC6828] offers an example of an RTP mixer approach.

A splicer implemented as an RTP translator [RFC3550] will forward the RTP packets from the original and substitutive senders with their SSRCs intact, but will need to rewrite RTCP sender report packets to account for the splicing. In this case, the congestion control loops run between original sender and receiver, and between the substitutive sender and receiver. The splicer needs to ensure that the RTCP feedback message from the receiver are passed to the right sender to let the congestion control work.

2.2 Overview of Splicing Interval

To handle splicing on the RTP layer at the reserved time slots set by the main RTP sender, the splicer must first know the Splicing Interval from the main RTP sender before it can start splicing.

When a new splicing is forthcoming, the main RTP sender needs to send the Splicing Interval to the splicer. The Splicing Interval SHOULD be sent by RTP header extension or RTCP extension message more than once to mitigate the possible packet loss. To enable the splicer to get the substitutive content before the splicing starts, the main RTP sender MUST send the Splicing Interval far ahead. For example, the main RTP sender can estimate when to send the Splicing Interval based on the round-trip time (RTT) following the mechanisms in section 6.4.1 of [RFC3550] when the splicer sends RTCP RR to the main sender.

The substitutive sender also needs to learn the Splicing Interval from the main RTP sender in advance, and thus estimates when to transfer the substitutive content to the splicer. The Splicing Interval could be transmitted from the main RTP sender to the substitutive content using some out-of-band mechanisms, for example, a proprietary mechanism to exchange the Splicing Interval, or the substitutive sender is implemented together with the main RTP sender inside a single device. To ensure the Splicing Interval is valid for both the main RTP sender and the substitutive RTP sender, the two senders MUST share a common reference clock so that the splicer can achieve accurate splicing. The requirements for the common reference clock (e.g. resolution, skew) depend on the codec used by the media content.

In this document, the main RTP sender uses a pair of NTP-format timestamps, to indicate when to start and end the splicing to the splicer: the timestamp of the first substitutive RTP packet at the splicing in point, and the timestamp of the first main RTP packet at the splicing out point.

When the substitutive RTP sender gets the Splicing Interval, it must prepare the substitutive stream. The main and the substitutive content providers MUST ensure that the RTP timestamp of the first substitutive RTP packet that would be presented to the receivers corresponds to the same time instant as the former NTP-format timestamp in the Splicing Interval. To enable the splicer to know the first substitutive RTP packet it needs to send, the substitutive RTP sender MUST send the substitutive RTP packet ahead of the Splicing In point, allowing the splicer to find out the timestamp of this first RTP packet in the substitutive RTP stream, e.g., using a prior RTCP SR (Sender Report) message.

When the splicing will end, the main content provider and the substitutive content provider MUST ensure the RTP timestamp of the first main RTP packet that would be presented on the receivers corresponds to the same time instant as the latter NTP-format timestamp in the Splicing Interval.

3 Conveying Splicing Interval in RTP/RTCP extensions

This memo defines two backwards compatible RTP extensions to convey the Splicing Interval to the splicer: an RTP header extension and an RTCP splicing notification message.

3.1 RTP Header Extension

The RTP header extension mechanism defined in [RFC5285bis] can be adapted to carry the Splicing Interval consisting of a pair of NTP-format timestamps.

This RTP header extension carries the 7 octets splicing-out NTP-format timestamp (lower 24-bit part of the Seconds of a NTP-format timestamp and the 32 bits of the Fraction of a NTP-format timestamp as defined in [RFC5905]), followed by the 8 octets splicing-in NTP-format timestamp (64-bit NTP-format timestamp as defined in [RFC5905]). The top 8 bits of the splicing-out NTP timestamp are inferred from the top 8 bits of the splicing-in NTP timestamp, under the assumption that the splicing-out time is after the splicing-in time, and the splicing interval is less than 2^25 seconds. Therefore, if the value of 7 octets splicing-out NTP-format timestamp is smaller than the value of 7 lower octets splicing-in NTP-format, it implies a wrap of the 56-bit splicing-out NTP-format timestamp which means the top 8-bit value of the 64-bit splicing-out is equal to the top 8-bit value of splicing-in NTP Timestamp plus 0x01. Otherwise, the top 8 bits of splicing-out NTP timestamp is equal to the top 8 bits of splicing-in NTP Timestamp.

This RTP header extension can be encoded using either the one-byte or two-byte header defined in [RFC5285bis]. Figure 1 and 2 show the splicing interval header extension with each of the two header formats.

```
    0                   1                   2                   3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
  +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+E
  |   ID  | L=14  | OUT NTP timestamp format - Seconds (bit 8-31) |x
  +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+t
  |       OUT NTP timestamp format - Fraction (bit 0-31)          |e
  +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+n
  |        IN  NTP timestamp format - Seconds (bit 0-31)          |s
  +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+i
  |        IN  NTP timestamp format - Fraction (bit 0-31)         |o
  +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+n
```

Figure 1: Splicing Interval
Using the One-Byte Header Format

```
        0                   1                   2                   3
        0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+E
      |   ID          |    L=15       | OUT NTP timestamp  - Seconds  |x
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+t
      |Out Secds(cont)|      OUT NTP timestamp format - Fraction      |e
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+n
      |Out Fract(cont)|      IN  NTP timestamp format - Seconds       |s
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+i
      | In Secds(cont)|     IN  NTP timestamp format - Fraction       |o
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+n
      | In Fract(cont)| 0 (pad)       |              ...
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 2: Splicing Interval
Using the Two-Byte Header Format

Since the inclusion of an RTP header extension will reduce the efficiency of RTP header compression, it is RECOMMENDED that the main sender inserts the RTP header extensions into only a number of RTP packets, instead of all the RTP packets, prior to the splicing in.

After the splicer obtains the RTP header extension and derives the Splicing Interval, it generates its own stream and is not allowed to include the RTP header extension in outgoing packets to reduce header overhead.

3.2 RTCP Splicing Notification Message

In addition to the RTP header extension, the main RTP sender includes the Splicing Interval in an RTCP splicing notification message. Whether or not the timestamps are included in the RTP header extension, the main RTP sender MUST send the RTCP splicing notification message. This provide robustness in the case where a middlebox strips RTP header extensions. The main RTP sender MUST make sure the splicing information contained in the RTCP splicing notification message consistent with the information included in the RTP header extensions.

The RTCP splicing notification message is a new RTCP packet type. It has a fixed header followed by a pair of NTP-format timestamps:

```
         0                   1                   2                   3
        0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
       +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
       |V=2|P|reserved |    PT=TBA     |             length            |
```

```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                             SSRC                              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|             IN NTP Timestamp (most significant word)          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|             IN NTP Timestamp (least significant word)         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|             OUT NTP Timestamp (most significant word)         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|             OUT NTP Timestamp (least significant word)        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 2: RTCP Splicing Notification Message

The RSI packet includes the following fields:

Length: 16 bits

As defined in [RFC3550], the length of the RTCP packet in 32-bit words minus one, including the header and any padding.

SSRC: 32 bits

The SSRC of the Main RTP Sender.

Timestamp: 64 bits

Indicates the wallclock time when this splicing starts and ends. The full-resolution NTP-format timestamp is used, which is a 64-bit, unsigned, fixed-point number with the integer part in the first 32 bits and the fractional part in the last 32 bits. This format is same as the NTP timestamp field in the RTCP Sender Report (Section 6.4.1 of [RFC3550]).

The RTCP splicing notification message can be included in the RTCP compound packet together with RTCP SR generated at the main RTP sender, and hence follows the compound RTCP rules defined in Section 6.1 in [RFC3550].

If the use of non-compound RTCP [RFC5506] was previously negotiated between the sender and the splicer, the RTCP splicing notification message may be sent as non-compound RTCP packets. In some cases that the mapping from RTP timestamp to NTP timestamp changes, e.g., clock drift happening before the splicing event, it may be required to send RTCP SR or even updated Splicing Interval information timely to update the timestamp mapping for accurate splicing.

Since the RTCP splicing notification message is intentionally sent by the main RTP sender to the splicer, the splicer is not allowed to forward this message to the receivers so as to avoid their useless processing and additional RTCP bandwidth consumption in the downstream.

4 Reducing Splicing Latency

When splicing starts or ends, the splicer outputs the multimedia content from another sender to the receivers. Given that the receivers must first acquire certain information ([RFC6285] refers to this information as Reference Information) to start processing the multimedia data, either the main RTP sender or the substitutive sender SHOULD provide the Reference Information together with its multimedia content to reduce the delay caused by acquiring the Reference Information. The methods by which the Reference Information is distributed to the receivers is out of scope of this memo.

Another latency element is synchronization caused delay. The receivers must receive enough synchronization metadata prior to synchronizing the separate components of the multimedia streams when splicing starts or ends. Either the main RTP sender or the substitutive sender SHOULD send the synchronization metadata early enough so that the receivers can play out the multimedia in a synchronized fashion. The main RTP sender or the substitutive sender can estimate when to send the synchronization metadata based on, for example, the round-trip time (RTT) following the mechanisms in section 6.4.1 of [RFC3550] when the splicer sends RTCP RR to the main sender or the substitutive sender. The main RTP sender and the substitutive sender can also be coordinated by some proprietary out-of-band mechanisms to decide when and whom to send the metadata. If both send the information, the splicer SHOULD pick one based on the current situation, e.g., choosing main RTP sender when synchronizing the main media content while choosing the information from the substitutive sender when synchronizing the spliced content. The mechanisms defined in [RFC6051] are RECOMMENDED to be adopted to reduce the possible synchronization delay.

5 Failure Cases

This section examines the implications of losing RTCP splicing notification message and the other failure case, e.g., the RTP header extension is stripped on the path.

Given that there may be a splicing un-aware middlebox on the path between the main RTP sender and the splicer, the main and the substitutive RTP senders can use one heuristic to verify whether or not the Splicing Interval reaches the splicer.

The splicer can be implemented to have its own SSRC, and send RTCP reception reports to the senders of the main and substitutive RTP streams. This allows the senders to detect problems on the path to the splicer. Alternatively, it is possible to implement the splicer such that it has no SSRC, and does not send RTCP reports; this prevents the senders from being able to monitor the quality to the path to the splicer.

If the splicer has an SSRC and sends its own RTCP reports, it can choose not to pass RTCP reports it receives from the receivers to the senders. This will stop the senders from being able to monitor the quality of the paths from the splicer to the receivers.

A splicer that has an SSRC can choose to pass RTCP reception reports from the receivers back to the senders, after modifications to account for the splicing. This will allow the senders the monitor the quality of the paths from the splicer to the receivers. A splicer that does not have its own SSRC has to forward and translation RTCP reports from the receiver, otherwise the senders will not see any receivers in the RTP session.

If the splicer is implemented as a mixer, it will have its own SSRC and will send its own RTCP reports, and will forward translated RTCP reports from the receivers.

Upon the detection of a failure, the splicer can communicate with the main sender and the substitutive sender in some out of band signaling ways to fall back to the payload specific mechanisms it supports, e.g., MPEG-TS splicing solution defined in [SCTE35], or just abandon the splicing.

6 Session Description Protocol (SDP) Signaling

This document defines the URI for declaring this header extension in an extmap attribute to be "urn:ietf:params:rtp-hdrext:splicing-interval".

This document extends the standard semantics defined in SDP Grouping Framework [RFC5888] with a new semantic: SPLICE to represent the relationship between the main RTP stream and the substitutive RTP stream. Only 2 m-lines are allowed in the SPLICE group. The main RTP stream is the one with the extended extmap attribute, and the other one is substitutive stream. A single m-line MUST NOT be included in different SPLICE groups at the same time. The main RTP sender provides the information about both main and substitutive sources.

The extended SDP attribute specified in this document is applicable for offer/answer content [RFC3264] and do not affect any rules when negotiating offer and answer. When used with multiple m-lines, substitutive RTP MUST be applied only to the RTP packets whose SDP m-line is in the same group with the substitutive stream using SPLICE and has the extended splicing extmap attribute. This semantic is also applicable for BUNDLE cases.

The following examples show how SDP signaling could be used for splicing in different cases.

6.1 Declarative SDP

```
v=0
o=xia 1122334455 1122334466 IN IP4 splicing.example.com
s=RTP Splicing Example
t=0 0
a=group:SPLICE 1 2
m=video 30000 RTP/AVP 100
i=Main RTP Stream
c=IN IP4 233.252.0.1/127
a=rtpmap:100 MP2T/90000
a=extmap:1 urn:ietf:params:rtp-hdrext:splicing-interval
a=mid:1
m=video 30002 RTP/AVP 100
i=Substitutive RTP Stream
c=IN IP4 233.252.0.2/127
a=sendonly
a=rtpmap:100 MP2T/90000
a=mid:2
```

Figure 3: Example SDP for a single-channel splicing scenario

The splicer receiving the SDP message above receives one MPEG2-TS stream (payload 100) from the main RTP sender (with multicast destination address of 233.252.0.1) on port 30000, and/or receives another MPEG2-TS stream from the substitutive RTP sender (with multicast destination address of 233.252.0.2) on port 30002. But at a particular point in time, the splicer only selects one stream and outputs the content from the chosen stream to the downstream receivers.

6.2 Offer/Answer without BUNDLE

SDP Offer - from main RTP sender

```
v=0
o=xia 1122334455 1122334466 IN IP4 splicing.example.com
s=RTP Splicing Example
t=0 0
a=group:SPLICE 1 2
m=video 30000 RTP/AVP 31 100
i=Main RTP Stream
c=IN IP4 splicing.example.com
a=rtpmap:31 H261/90000
a=rtpmap:100 MP2T/90000
a=extmap:1 urn:ietf:params:rtp-hdrext:splicing-interval
a=sendonly
a=mid:1
m=video 40000 RTP/AVP 31 100
i=Substitutive RTP Stream
c=IN IP4 substitutive.example.com
a=rtpmap:31 H261/90000
a=rtpmap:100 MP2T/90000
a=sendonly
a=mid:2
```

SDP Answer - from splicer

```
v=0
o=xia 1122334455 1122334466 IN IP4 splicer.example.com
s=RTP Splicing Example
t=0 0
a=group:SPLICE 1 2
m=video 30000 RTP/AVP 100
i=Main RTP Stream
c=IN IP4 splicer.example.com
a=rtpmap:100 MP2T/90000
a=extmap:1 urn:ietf:params:rtp-hdrext:splicing-interval
a=recvonly
a=mid:1
m=video 40000 RTP/AVP 100
i=Substitutive RTP Stream
c=IN IP4 splicer.example.com
a=rtpmap:100 MP2T/90000
a=recvonly
a=mid:2
```

6.3 Offer/Answer with BUNDLE: All Media are spliced

In this example, the bundled audio and video media have their own substitutive media for splicing:

1. An Offer, in which the offerer assigns a unique address and a substitutive media to each bundled "m="line for splicing within the BUNDLE group.

2. An answer, in which the answerer selects its own BUNDLE address, and leave the substitutive media untouched.

SDP Offer - from main RTP sender

```
v=0
o=alice 1122334455 1122334466 IN IP4 splicing.example.com
s=RTP Splicing Example
c=IN IP4 splicing.example.com
t=0 0
a=group:SPLICE foo 1
a=group:SPLICE bar 2
a=group:BUNDLE foo bar
m=audio 10000 RTP/AVP 0 8 97
a=mid:foo
b=AS:200
a=rtpmap:0 PCMU/8000
a=rtpmap:8 PCMA/8000
a=rtpmap:97 iLBC/8000
a=extmap:1 urn:ietf:params:rtp-hdrext:splicing-interval
a=sendonly
m=video 10002 RTP/AVP 31 32
a=mid:bar
b=AS:1000
a=rtpmap:31 H261/90000
a=rtpmap:32 MPV/90000
a=extmap:2 urn:ietf:params:rtp-hdrext:splicing-interval
a=sendonly
m=audio 20000 RTP/AVP 0 8 97
i=Substitutive audio RTP Stream
c=IN IP4 substitive.example.com
a=rtpmap:0 PCMU/8000
a=rtpmap:8 PCMA/8000
a=rtpmap:97 iLBC/8000
a=sendonly
a=mid:1
m=video 20002 RTP/AVP 31 32
i=Substitutive video RTP Stream
```

```
          c=IN IP4 substitive.example.com
          a=rtpmap:31 H261/90000
          a=rtpmap:32 MPV/90000
          a=mid:2
          a=sendonly
```

SDP Answer - from the splicer

```
          v=0
          o=bob 2808844564 2808844564 IN IP4 splicer.example.com
          s=RTP Splicing Example
          c=IN IP4 splicer.example.com
          t=0 0
          a=group:SPLICE foo 1
          a=group:SPLICE bar 2
          a=group:BUNDLE foo bar
          m=audio 30000 RTP/AVP 0
          a=mid:foo
          b=AS:200
          a=rtpmap:0 PCMU/8000
          a=extmap:1 urn:ietf:params:rtp-hdrext:splicing-interval
          a=recvonly
          m=video 30000 RTP/AVP 32
          a=mid:bar
          b=AS:1000
          a=rtpmap:32 MPV/90000
          a=extmap:2 urn:ietf:params:rtp-hdrext:splicing-interval
          a=recvonly
          m=audio 30002 RTP/AVP 0
          i=Substitutive audio RTP Stream
          c=IN IP4 splicer.example.com
          a=rtpmap:0 PCMU/8000
          a=recvonly
          a=mid:1
          m=video 30004 RTP/AVP 32
          i=Substitutive video RTP Stream
          c=IN IP4 splicer.example.com
          a=rtpmap:32 MPV/90000
          a=mid:2
          a=recvonly
```

6.4 Offer/Answer with BUNDLE: a Subset of Media are Spliced

In this example, the substitutive media only applies for video when splicing:

1. An Offer, in which the offerer assigns a unique address to each bundled "m="line within the BUNDLE group, and assigns a substitutive

media to the bundled video "m=" line for splicing.

2. An answer, in which the answerer selects its own BUNDLE address, and leave the substitutive media untouched.

SDP Offer - from the main RTP sender:

```
v=0
o=alice 1122334455 1122334466 IN IP4 splicing.example.com
s=RTP Splicing Example
c=IN IP4 splicing.example.com
t=0 0
a=group:SPLICE bar 2
a=group:BUNDLE foo bar
m=audio 10000 RTP/AVP 0 8 97
a=mid:foo
b=AS:200
a=rtpmap:0 PCMU/8000
a=rtpmap:8 PCMA/8000
a=rtpmap:97 iLBC/8000
a=sendonly
m=video 10002 RTP/AVP 31 32
a=mid:bar
b=AS:1000
a=rtpmap:31 H261/90000
a=rtpmap:32 MPV/90000
a=extmap:2 urn:ietf:params:rtp-hdrext:splicing-interval
a=sendonly
m=video 20000 RTP/AVP 31 32
i=Substitutive video RTP Stream
c=IN IP4 substitutive.example.com
a=rtpmap:31 H261/90000
a=rtpmap:32 MPV/90000
a=mid:2
a=sendonly
```

SDP Answer - from the splicer:

```
v=0
o=bob 2808844564 2808844564 IN IP4 splicer.example.com
s=RTP Splicing Example
c=IN IP4 splicer.example.com
t=0 0
a=group:SPLICE bar 2
a=group:BUNDLE foo bar
m=audio 30000 RTP/AVP 0
a=mid:foo
b=AS:200
```

```
a=rtpmap:0 PCMU/8000
a=recvonly
m=video 30000 RTP/AVP 32
a=mid:bar
b=AS:1000
a=rtpmap:32 MPV/90000
a=extmap:2 urn:ietf:params:rtp-hdrext:splicing-interval
a=recvonly
m=video 30004 RTP/AVP 32
i=Substitutive video RTP Stream
c=IN IP4 splicer.example.com
a=rtpmap:32 MPV/90000
a=mid:2
a=recvonly
```

7 Security Considerations

The security considerations of the RTP specification [RFC3550] and the general mechanism for RTP header extensions [RFC5285bis] apply. The splicer can either be a mixer or a translator, and all the security considerations of these two RTP intermediaries topologies described in [RFC7667] and [RFC7201] are applicable for the splicer.

The splicer replaces some content with other content in RTP packet, thus breaking any RTP-level end-to-end security, such as source authentication and integrity protection. End to end source authentication is not possible with any known existing splicing solution. A new solution can theoretically be developed that enables identification of the participating entities and what each provides, i.e., the different media sources, main and substituting, and the splicer which provides the RTP-level integration of the media payloads in a common timeline and synchronization context.

Since the splicer breaks RTP-level end-to-end security, it needs to be part of the signaling context and the necessary security associations (e.g., SRTP crypto contexts) established for the RTP session participants. When using the Secure Real-Time Transport Protocol (SRTP) [RFC3711], the splicer would have to be provisioned with the same security association as the main RTP sender.

If there is a concern about the confidentiality of the splicing time information, the header extension defined in this document MUST be also protected, for example, header extension encryption [RFC6904] can be used in this case. However, the malicious endpoint may get the splicing time information by other means, e.g., inferring from the communication between the main and substitutive content sources. To avoid the insertion of invalid substitutive content, the splicer MUST have some mechanisms to authenticate the substitutive stream source.

For cases that the splicing time information is changed by a malicious endpoint, the splicing, for example, may fail since it will not be available at the right time for the substitutive media to arrive. Another case is that an attacker may prevent the receivers receiving the content from the main sender by inserting extra splicing time information. To avoid the above cases happening, the authentication of the RTP header extension for splicing time information SHOULD be considered.

When a splicer implemented as a mixer sends the stream to the receivers, CSRC list, which can be used to detect RTP-level forwarding loops as defined in Section 8.2 of [RFC3550], may be removed for simplifying the receivers that can not handle multiple sources in the RTP stream. Hence, loops may occur to cause packets to loop back to upstream of the splicer and may form a serious denial-of-service threat. In such a case, non-RTP means, e.g., signaling among all the participants, MUST be used to detect and resolve loops.

8 IANA Considerations

8.1 RTCP Control Packet Types

Based on the guidelines suggested in [RFC5226], a new RTCP packet format has been registered with the RTCP Control Packet Type (PT) Registry:

Name: SNM

Long name: Splicing Notification Message

Value: TBA

Reference: This document

8.2 RTP Compact Header Extensions

The IANA has also registered a new RTP Compact Header Extension [RFC5285bis], according to the following:

Extension URI: urn:ietf:params:rtp-hdrext:splicing-interval

Description: Splicing Interval

Contact: Jinwei Xia <xiajinwei@huawei.com>

Reference: This document

8.3 SDP Grouping Semantic Extension

This document request IANA to register the new SDP grouping semantic extension called "SPLICE".

Semantics: Splice

Token:SPLICE

Reference: This document

9 Acknowledgement

The authors would like to thank the following individuals who help to review this document and provide very valuable comments: Colin Perkins, Bo Burman, Stephen Botzko, Ben Campbell.

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.

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

[RFC5285bis] Even, R., Singer, D. and H. Desineni, "A General Mechanism for RTP Header Extensions", draft-ietf-avtcore-rfc5285-bis-02, May 2016.

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

[RFC5905] Mills, D., Martin, J., Ed., Burbank, J., and W. Kasch, "Network Time Protocol Version 4: Protocol and Algorithms Specification", RFC 5905, June 2010.

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

[RFC7201] Westerlund, M. and C. Perkins, "Options for Securing RTP Sessions", RFC 7201, April 2014.

[RFC7667] Westerlund, M. and S. Wenger, "RTP Topologies", RFC 7667, November 2015.

10.2 Informative References

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

[RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 5226, May 2008.

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

[RFC6285] Ver Steeg, B., Begen, A., Van Caenegem, T., and Z. Vax, "Unicast-Based Rapid Acquisition of Multicast RTP Sessions", RFC 6285, June 2011.

[RFC6904] Lennox, J.,"Encryption of Header Extensions in the Secure Real-Time Transport Protocol (SRTP)", April 2013.

[SCTE35] Society of Cable Telecommunications Engineers (SCTE), "Digital Program Insertion Cueing Message for Cable", 2011.

[RFC6828] Xia, J., "Content Splicing for RTP Sessions", RFC 6828, January 2013.

Authors' Addresses

Jinwei Xia
Huawei

Email: xiajinwei@huawei.com

Roni Even
Huawei

Email: ron.even.tlv@gmail.com

Rachel Huang

Huawei

Email: rachel.huang@huawei.com

Lingli Deng
China Mobile

Email: denglingli@chinamobile.com

Network Working Group
Internet-Draft
Intended status: Informational
Expires: May 3, 2018

H. Zheng
R. Even
Q. Wu
Huawei
R. Gu
China Mobile

October 30, 2017

RTP Control Protocol (RTCP) Extended Report (XR) Block for Effective Loss Index Reporting

draft-zheng-xrblock-effective-loss-index-00

Abstract

This document defines a new metric for RTP applications to measure the effectiveness of stream repair means, and an RTP Control Protocol (RTCP) Extended Report (XR) Block to report the metric.

Status of This Memo

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

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

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

This Internet-Draft will expire on May 3, 2018.

Copyright Notice

Copyright (c) 2017 IETF Trust and the persons identified as the document authors. All rights reserved.

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

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

Table of Contents

1. Introduction

RTP applications often use stream repair means, e.g. FEC (Forward Error Correction) [RFC5109] and/or retransmission [RFC4588] to improve the robustness of media streams. With the presence of those stream repair means, a degree of packet loss can be recovered for a media stream. In the past, some RTCP Extend Reports (XRs) were defined to reflect the situation of post-repair loss. For example, [RFC5725] defines an XR block using Run Length Encoding (RLE) to report post-repair loss; [RFC7509] defines count metrics for post-repair loss.

This document proposes a new metric Effective Loss Index (ELI) to measure the effectiveness of stream repair means. The new metric provides a simpler view on the post-repair loss than the mechanisms documented in [RFC5725] and [RFC7509]. EFI is an index, so the values reported from different RTP sources can be compared directly, which makes it easier to rank the effectiveness of loss repair means. An example use case is to find endpoints whose ELI values are at bottom 10%. For those endpoints, more informative XR reports such as those in [RFC5725] and [RFC7509] can then be used to discover more details about the loss situations.

This document also defines an XR block to report the metric, which can be found out in Section 3.

1.1. Effective Loss Index

Effective Loss Index (ELI) uses a simple model to measure the effectiveness of loss repair. The model assumes that repair means are applied onto packets by batches of equal size. Lower ELI means that the repair was more successful. Specifically, a batch is identified by a range of RTP sequence numbers. The size of a batch is number of packets. An application can agree upon a default batch size, or use the SDP signaling defined in Section 4.1 to communicate one.

An RTP endpoint is thought to process received packets and apply repair means batch by batch. For each batch, if there is still some unrecoverable loss after having applied the repair means, then the repair means are deemed as ineffective. The ineffectiveness is denoted by Effective Loss Factor (ELF), along with a parameter Effective Loss Threshold, showing below:

```
if Post-Repair Loss > Effective Loss Threshold
   Effective Loss Factor = 1
else
   Effective Loss Factor = 0
endif
```

Figure 1: Calculation of Effective Loss Factor

The parameters in Figure 1 are explained below:

- o Post-Repair Loss is the number of packet lost after repair in the batch.

- o Effective Loss Threshold is in number of packets.

The minimum value of Effective Loss Threshold is zero. This document does not mandate any value for Effective Loss Threshold. Applications can prescribe a value for themselves without signaling. On the other hand, SDP signaling defined in Section 4.1 can be used to communicate the value. Determining an Effective Loss Threshold value for use can be empirical, applications may have to try out and change the value from time to time, depending on their needs.

Effective Loss Index is an integer derived by calculating the average Effective Loss Factor across a sequence of consecutive batches of RTP packets. Let ELF(i) be the Effective Loss Factor calculated for i-th batch, and N as number of batches in the sequence, then Effective Loss Index is calculated as:

```
                        ELF(1)+ELF(2)+ ...+ELF(N)
Effective Loss Index =  ------------------------- x 10000
                                    N
```

Figure 2: Calculation of Effective Loss Index

The following is an example of how to calculate Effective Loss Index. For simplicity and demonstration purpose, the size of batches is assumed to be 3, and the Effective Loss Threshold is assumed to be 1. The example processes a sequence of 9 RTP packets in 3 batches.

```
              Batch     Post-Repair    Effective
                        Loss           Loss Factor
            | 1 2 3 |   2, 3           1
            | 4 5 6 |   5              0
            | 7 8 9 |   7              0

                                1 + 0 + 0
            Effective Loss Index = ----------- x 10000 = 3333
                                    3
```

1.2. Applicability

The metric defined by this document is applicable to a range of RTP applications that send packets in batches of equal length, probably with stream repair means (e.g., Forward Error Correction (FEC) [RFC5109] and/or retransmission [RFC4588]) applied on the batches. Note that in order to not interfere with the batches being protected, any additional packets generated by the stream repair means SHOULD be in a different RTP stream.

The number of batches among which ELI is calculated should not be too few, otherwise the result may be too biased. However, specifying a minimal number of batches seems unrealistic, due to the stream repair means used by applications can be quite different. This document leaves it to applications to choose a suitable minimal value for the number of batches.

1.3. RTCP and RTCP XR Reports

The use of RTCP for reporting is defined in [RFC3550]. [RFC3611] defines an extensible structure for reporting by using an RTCP Extended Report (XR). This document defines a new Extended Report block for use with [RFC3550] and [RFC3611].

1.4. Performance Metrics Framework

The Performance Metrics Framework [RFC6390] provides guidance on the definition and specification of performance metrics. The "Guidelines for Use of the RTP Monitoring Framework" [RFC6792] provides guidelines for reporting block format using RTCP XR. The Metrics Block described in this document is in accordance with the guidelines in [RFC6390] and [RFC6792].

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. Effective Loss Index Report Block

The Effective Loss Index Report Block has the following format:

```
 0               1               2               3               4
 0 1 2 3 4 5 6 7 0 1 2 3 4 5 6 7 0 1 2 3 4 5 6 7 0 1 2 3 4 5 6 7 0
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |     BT=TBD    |   Reserved    |      Block length = 3         |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |                        SSRC of Source                         |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |    Effective Loss Index       |          Padding              |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Block Type (BT): 8 bits: An Effect Loss Index Report Block is identified by the constant 'TBD'.

[[Editor Note: should replace 'TBD' with assigned value]]

Reserved: 8 bits: These bits are reserved for future use. They MUST be set to zero by senders and ignored by receivers (see Section 4.2 of [RFC6709]).

Block length: 16 bits: This field is in accordance with the definition in [RFC3611]. In this report block, it MUST be set to

3. The block MUST be discarded if the block length is set to a different value.

SSRC of source: 32 bits: As defined in Section 4.1 of [RFC3611].

Effective Loss Index: 16 bits: The value of this field SHOULD be set to the calculated result of Effective Loss Index (as in Figure 2).

Padding: 16 bits: These bits MUST be set to zero by senders and ignored by receivers.

4. SDP Signaling

[RFC3611] defines the use of SDP (Session Description Protocol) for signaling the use of RTCP XR blocks. However, XR blocks MAY be used without prior signaling (see Section 5 of [RFC3611]).

4.1. SDP rtcp-xr-attrib Attribute Extension

This session augments the SDP attribute "rtcp-xr" defined in Section 5.1 of [RFC3611] by providing an additional value of "xr-format" to signal the use of the report block defined in this document. The ABNF [RFC5234] syntax is as follows.

```
xr-format =/ xr-eli-block

xr-eli-block = "effective-loss-index"
               [ ":" effective-loss-batch-size]
               [ ">" effective-loss-threshold]

effective-loss-batch-size  = 1*DIGIT
                             ; the batch size is in number of packets

effective-loss-threshold   = 1*DIGIT
                             ; the threshold is in number of packets

DIGIT = %x30-39
```

The SDP attribute "xr-eli-block" is designed to contain two optional values, one for signaling the batch size, another for the Effective Loss Threshold. Here are some examples:

1. signaling both batch size (100) and Effective Loss Threshold (2)

xr-eli-block = "effective-loss-index" : "100" > "2"

2. signaling only batch size (100)

xr-eli-block = "effective-loss-index" : "100"

3. signaling only Effective Loss Threshold (2)

xr-eli-block = "effective-loss-index" > "2"

4.2. Offer/Answer Usage

When SDP is used in offer/answer context, the SDP Offer/Answer usage defined in [RFC3611] for the unilateral "rtcp-xr" attribute parameters applies. For detailed usage of Offer/Answer for unilateral parameters, refer to Section 5.2 of [RFC3611].

5. Security Considerations

This proposed RTCP XR block introduces no new security considerations beyond those described in [RFC3611] This block does not provide per-packet statistics, so the risk to confidentiality documented in Section 7, paragraph 3 of [RFC3611] does not apply.

An attacker may put incorrect information in the Effective Loss Index reports. Implementers should consider the guidance in [RFC7202] for using appropriate security mechanisms, i.e., where security is a concern, the implementation should apply encryption and authentication to the report block. For example, this can be achieved by using the AVPF profile together with the Secure RTP profile as defined in [RFC3711] an appropriate combination of the two profiles (an "SAVPF") is specified in [RFC5124] However, other mechanisms also exist (documented in [RFC7201] and might be more suitable.

6. IANA Considerations

New block types for RTCP XR are subject to IANA registration. For general guidelines on IANA considerations for RTCP XR, refer to [RFC3611].

6.1. New RTCP XR Block Type Value

This document assigns the block type value ’TBD’ in the IANA "RTP Control Protocol Extended Reports (RTCP XR) Block Type Registry" to the "Post-Repair Loss Count Metrics Report Block".

[[Editor Note: should replace ’TBD’ with assigned value]]

6.2. New RTCP XR SDP Parameter

This document also registers a new parameter "effective-loss-index" in the "RTP Control Protocol Extended Reports (RTCP XR) Session Description Protocol (SDP) Parameters Registry".

6.3. Contact Information for Registrations

The contact information for the registrations is:

RAI Area Directors <rai-ads@ietf.org>

7. Acknowledgements

This document has benefited greatly from the comments of various people. The following individuals have contributed to this document: Rachel Huang, Colin Perkins, Yanfang Zhang, Lingyan Wu.

8. References

8.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC3550] Schulzrinne, H., Casner, S., Frederick, R., and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications", STD 64, RFC 3550, DOI 10.17487/RFC3550, July 2003, <https://www.rfc-editor.org/info/rfc3550>.

8.2. Informative References

[RFC3611] Friedman, T., Ed., Caceres, R., Ed., and A. Clark, Ed., "RTP Control Protocol Extended Reports (RTCP XR)", RFC 3611, DOI 10.17487/RFC3611, November 2003, <https://www.rfc-editor.org/info/rfc3611>.

[RFC3711] Baugher, M., McGrew, D., Naslund, M., Carrara, E., and K. Norrman, "The Secure Real-time Transport Protocol (SRTP)", RFC 3711, DOI 10.17487/RFC3711, March 2004, <https://www.rfc-editor.org/info/rfc3711>.

[RFC4588] Rey, J., Leon, D., Miyazaki, A., Varsa, V., and R. Hakenberg, "RTP Retransmission Payload Format", RFC 4588, DOI 10.17487/RFC4588, July 2006, <https://www.rfc-editor.org/info/rfc4588>.

[RFC5109] Li, A., Ed., "RTP Payload Format for Generic Forward Error Correction", RFC 5109, DOI 10.17487/RFC5109, December 2007, <https://www.rfc-editor.org/info/rfc5109>.

[RFC5124] Ott, J. and E. Carrara, "Extended Secure RTP Profile for Real-time Transport Control Protocol (RTCP)-Based Feedback (RTP/SAVPF)", RFC 5124, DOI 10.17487/RFC5124, February 2008, <https://www.rfc-editor.org/info/rfc5124>.

[RFC5234] Crocker, D., Ed. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", STD 68, RFC 5234, DOI 10.17487/RFC5234, January 2008, <https://www.rfc-editor.org/info/rfc5234>.

[RFC5725] Begen, A., Hsu, D., and M. Lague, "Post-Repair Loss RLE Report Block Type for RTP Control Protocol (RTCP) Extended Reports (XRs)", RFC 5725, DOI 10.17487/RFC5725, February 2010, <https://www.rfc-editor.org/info/rfc5725>.

[RFC6390] Clark, A. and B. Claise, "Guidelines for Considering New Performance Metric Development", BCP 170, RFC 6390, DOI 10.17487/RFC6390, October 2011, <https://www.rfc-editor.org/info/rfc6390>.

[RFC6709] Carpenter, B., Aboba, B., Ed., and S. Cheshire, "Design Considerations for Protocol Extensions", RFC 6709, DOI 10.17487/RFC6709, September 2012, <https://www.rfc-editor.org/info/rfc6709>.

[RFC6792] Wu, Q., Ed., Hunt, G., and P. Arden, "Guidelines for Use of the RTP Monitoring Framework", RFC 6792, DOI 10.17487/RFC6792, November 2012, <https://www.rfc-editor.org/info/rfc6792>.

[RFC7201] Westerlund, M. and C. Perkins, "Options for Securing RTP Sessions", RFC 7201, DOI 10.17487/RFC7201, April 2014, <https://www.rfc-editor.org/info/rfc7201>.

[RFC7202] Perkins, C. and M. Westerlund, "Securing the RTP Framework: Why RTP Does Not Mandate a Single Media Security Solution", RFC 7202, DOI 10.17487/RFC7202, April 2014, <https://www.rfc-editor.org/info/rfc7202>.

[RFC7509] Huang, R. and V. Singh, "RTP Control Protocol (RTCP) Extended Report (XR) for Post-Repair Loss Count Metrics", RFC 7509, DOI 10.17487/RFC7509, May 2015, <https://www.rfc-editor.org/info/rfc7509>.

Appendix A. Metric Represented Using the Template from RFC 6390

A.1. Effective Loss Index

- Metric Name: RTP Effective Loss Index.

- Metric Description: The effectiveness of stream repair means applied on a sequence of RTP packets.

- Method of Measurement or Calculation: See the "Effective Loss Index" definition in Section 1.1. It is directly measured and must be measured for the primary source RTP packets with no further chance of repair.

- Units of Measurement: This metric is expressed as a 16-bit unsigned integer value representing the effectiveness of stream repair means.

- Measurement Point(s) with Potential Measurement Domain: It is measured at the receiving end of the RTP stream.

- Measurement Timing: This metric relies on the sequence number interval to determine measurement timing.

- Use and Applications: These metrics are applicable to any RTP application, especially those that use loss-repair mechanisms. See Section 1 for details.

- Reporting Model: See RFC 3611.

Authors' Addresses

Hui Zheng (Marvin)
Huawei

Email: marvin.zhenghui@huawei.com

Roni Even
Huawei

Email: roni.even@huawei.com

Qin Wu
Huawei

Email: bill.wu@huawei.com

Rong Gu
China Mobile

Email: gurong_cmcc@outlook.com