

AVTCORE Working Group
INTERNET-DRAFT
Updates: 7983, 5764
Category: Standards Track
Expires: May 19, 2021

B. Aboba
Microsoft Corporation
G. Salgueiro
Cisco Systems
C. Perkins
University of Glasgow
19 November 2020

Multiplexing Scheme Updates for QUIC
draft-aboba-avtcore-rfc7983bis-01.txt

Abstract

This document defines how QUIC, Datagram Transport Layer Security (DTLS), Real-time Transport Protocol (RTP), RTP Control Protocol (RTCP), Session Traversal Utilities for NAT (STUN), Traversal Using Relays around NAT (TURN), and ZRTP packets are multiplexed on a single receiving socket.

This document updates RFC 7983 and RFC 5764.

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 19, 2021.

Copyright Notice

Copyright (c) 2020 IETF Trust and the persons identified as the document authors. All rights reserved.

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

Table of Contents

1. Introduction	3
1.1. Terminology	3
2. Multiplexing of TURN Channels	3
3. Updates to RFC 7983	4
4. Security Considerations	5
5. IANA Considerations	6
6. References	6
6.1. Normative References	6
6.2. Informative References	7
Acknowledgements	7
Authors' Addresses	8

1. Introduction

"Multiplexing Scheme Updates for Secure Real-time Transport Protocol (SRTP) Extension for Datagram Transport Layer Security (DTLS)" [RFC7983] defines a scheme for a Real-time Transport Protocol (RTP) [RFC3550] receiver to demultiplex DTLS [RFC6347], Session Traversal Utilities for NAT (STUN) [RFC5389], Secure Real-time Transport Protocol (SRTP) / Secure Real-time Transport Control Protocol (SRTCP) [RFC3711], ZRTP [RFC6189] and TURN Channel packets arriving on a single port.

This document updates [RFC7983] and [RFC5764] to also allow QUIC [I-D.ietf-quic-transport] to be multiplexed on the same port. For peer-to-peer operation in WebRTC scenarios as described in [WEBRTC-QUIC][WEBRTC-QUIC-TRIAL], RTP is used to transport audio and video and QUIC is used for data exchange, SRTP [RFC3711] is keyed using DTLS-SRTP [RFC5764] and therefore SRTP/SRTCP [RFC3550], STUN, TURN, DTLS [RFC6347] and QUIC need to be multiplexed on the same port.

Since new versions of QUIC are allowed to change aspects of the wire image, there is no guarantee that future versions of QUIC beyond version 1 will adhere to the multiplexing scheme described in this document.

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. Multiplexing of TURN Channels

TURN channels are an optimization where data packets are exchanged with a 4-byte prefix instead of the standard 36-byte STUN overhead (see Section 2.5 of [RFC5766]). [RFC7983] allocated the values from 64 to 79 in order to allow TURN channels to be demultiplexed when the TURN Client does the channel binding request in combination with the demultiplexing scheme described in [RFC7983].

As noted in [I-D.aboba-avtcore-quic-multiplexing], the first octet of a QUIC short header packet falls in the range 64 to 127, thereby overlapping with the allocated range for TURN channels of 64 to 79.

The first octet of QUIC long header packets fall in the range 192 to 255. Since QUIC long header packets precede QUIC short header packets, if no packets with a first octet in the range of 192 to 255 have been received, a packet whose first octet is in the range of 64 to 79 can be demultiplexed unambiguously as TURN Channel traffic.

RTP (or RTCP, if both RTCP and RTP are being multiplexed over the same destination port). If the value is between 80 and 127 or between 192 and 255 (inclusive) then the packet is QUIC. If the value is between 64 and 79 inclusive, then if a packet has been previously forwarded that is in the range of 192 and 255, then the packet is QUIC, otherwise it is TURN Channel.

If the value does not match any known range, then the packet MUST be dropped and an alert MAY be logged. This process is summarized in Figure 3.

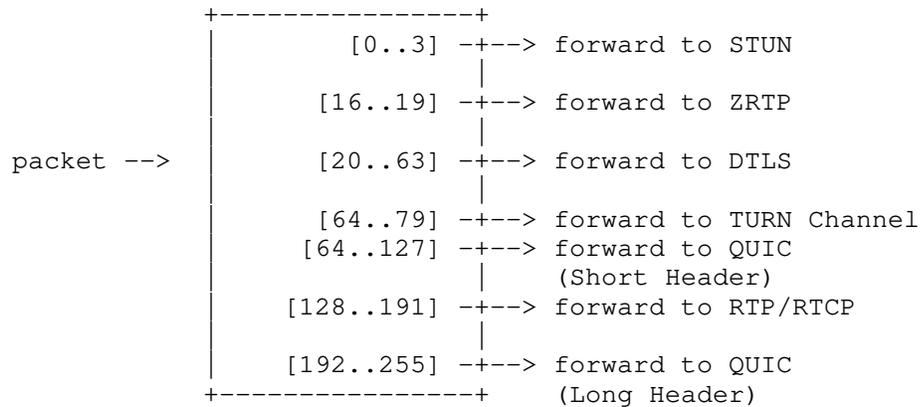


Figure 3: The receiver's packet demultiplexing algorithm.

END NEW TEXT

4. Security Considerations

The solution 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.

5. IANA Considerations

This document does not require actions by IANA.

6. References

6.1. Normative References

[I-D.ietf-quic-tls]

Thomson, M. and S. Turner, "Using Transport Layer Security (TLS) to Secure QUIC", draft-ietf-quic-tls-32 (work in progress), October 20, 2020.

[I-D.ietf-quic-transport]

Iyengar, J. and M. Thomson, "QUIC: A UDP-Based Multiplexed and Secure Transport", draft-ietf-quic-transport-32 (work in progress), October 20, 2020.

[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>>.

[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](http://www.rfc-editor.org/info/rfc5766)>.

- [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, <<http://www.rfc-editor.org/info/rfc7983>>.

6.2. Informative References

- [I-D.aboba-avtcore-quick-multiplexing] Aboba, B., Thatcher, P. and C. Perkins, "QUIC Multiplexing", draft-aboba-avtcore-quick-multiplexing-04 (work in progress), January 28, 2020.
- [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>>.
- [WEBRTC-QUIC] Thatcher, P. and B. Aboba, "QUIC API For Peer-to-Peer Connections", W3C Community Group Draft (work in progress), January 2020, <<https://w3c.github.io/webrtc-quick>>
- [WEBRTC-QUIC-TRIAL] Hampson, S., "RTCQuicTransport Coming to an Origin Trial Near You (Chrome 73)", January 2019, <<https://developers.google.com/web/updates/2019/01/rtcquictransport-api>>

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

Gonzalo Salgueiro
Cisco Systems
7200-12 Kit Creek Road
Research Triangle Park, NC 27709
United States of America

Email: gsalguei@cisco.com

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

Email: csp@csp Perkins.org

Internet Engineering Task Force
Internet-Draft
Intended status: Informational
Expires: 23 May 2021

G. Hellstrom
Gunnar Hellstrom Accessible Communication
19 November 2020

Real-time text solutions for multi-party sessions
draft-hellstrom-avtcore-multi-party-rtt-solutions-05

Abstract

This document specifies methods for Real-Time Text (RTT) media handling in multi-party calls. The main discussed transport is to carry Real-Time text by the RTP protocol in a time-sampled mode according to RFC 4103. The mechanisms enable the receiving application to present the received real-time text media, separated per source, in different ways according to user preferences. Some presentation related features are also described explaining suitable variations of transmission and presentation of text.

Call control features are described for the SIP environment. A number of alternative methods for providing the multi-party negotiation, transmission and presentation are discussed and a recommendation for the main ones is provided. The main solution for SIP based centralized multi-party handling of real-time text is achieved through a media control unit coordinating multiple RTP text streams into one RTP stream.

Alternative methods using a single RTP stream and source identification inline in the text stream are also described, one of them being provided as a lower functionality fallback method for endpoints with no multi-party awareness for RTT.

Bridging methods where the text stream is carried without the contents being dealt with in detail by the bridge are also discussed.

Brief information is also provided for multi-party RTT in the WebRTC environment.

The intention is to provide background for decisions, specification and implementation of selected methods.

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 23 May 2021.

Copyright Notice

Copyright (c) 2020 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	4
1.1. Requirements Language	5
2. Centralized conference model	5
3. Requirements on multi-party RTT	6
3.1. General requirements	6
3.2. Performance requirements	7
4. RTP based solutions	8
4.1. Coordination of text RTP streams	8
4.1.1. RTP-based solutions with a central mixer	8
4.1.1.1. RTP Mixer using default RFC 4103 methods	8
4.1.1.2. RTP Mixer using the default method but decreased transmission interval	9
4.1.1.3. RTP Mixer with frequent transmission and indicating sources in CSRC-list	10
4.1.1.4. RTP Mixer interleaving packets, receiver using timestamp to recover from loss	11
4.1.1.5. RTP Mixer with multiple primary data in each packet and individual sequence numbers	12
4.1.1.6. RTP Mixer with multiple primary data in each packet	14

- 4.1.1.7. RTP Mixer with RFC 5109 FEC and RFC 2198 redundancy in the packets 15
- 4.1.1.8. RTP Mixer with RFC 5109 FEC and RFC 2198 redundancy and separate sequence number in the packets 17
- 4.1.1.9. RTP Mixer indicating participants by a control code in the stream 19
- 4.1.1.10. Mixing for multi-party unaware user agents 20
- 4.1.2. RTP-based bridging with minor RTT media contents reformatting by the bridge 21
 - 4.1.2.1. RTP Translator sending one RTT stream per participant 21
 - 4.1.2.2. Distributing packets in an end-to-end encryption structure 24
 - 4.1.2.3. Mesh of RTP endpoints 25
 - 4.1.2.4. Multiple RTP sessions, one for each participant 26
- 5. Preferred RTP-based multi-party RTT transport method 26
- 6. Session control of RTP-based multi-party RTT sessions 27
 - 6.1. Implicit RTT multi-party capability indication 27
 - 6.2. RTT multi-party capability declared by SIP media-tags 28
 - 6.3. SDP media attribute for RTT multi-party capability indication 30
 - 6.4. Simplified SDP media attribute for RTT multi-party capability indication 31
 - 6.5. SDP format parameter for RTT multi-party capability indication 32
 - 6.6. A text media subtype for support of multi-party rtt 33
 - 6.7. Preferred capability declaration method for RTP-based transport. 33
 - 6.8. Identification of the source of text for RTP-based solutions 33
- 7. RTT bridging in WebRTC 34
 - 7.1. RTT bridging in WebRTC with one data channel per source 34
 - 7.2. RTT bridging in WebRTC with one common data channel 35
 - 7.3. Preferred rtt multi-party method for WebRTC 36
- 8. Presentation of multi-party text 36
 - 8.1. Associating identities with text streams 36
 - 8.2. Presentation details for multi-party aware endpoints. 37
 - 8.2.1. Bubble style presentation 37
 - 8.2.2. Other presentation styles 39
- 9. Presentation details for multi-party unaware endpoints. 39
- 10. Security Considerations 39
- 11. IANA Considerations 40
- 12. Congestion considerations 40
- 13. Acknowledgements 40
- 14. Change history 40

- 14.1. Changes to draft-hellstrom-avtcore-multi-party-rtt-solutions-05 . . . 40
- 14.2. Changes to draft-hellstrom-avtcore-multi-party-rtt-solutions-04 . . . 40
- 14.3. Changes to draft-hellstrom-avtcore-multi-party-rtt-solutions-03 . . . 40
- 14.4. Changes to draft-hellstrom-avtcore-multi-party-rtt-solutions-02 . . . 41
- 14.5. Changes to draft-hellstrom-avtcore-multi-party-rtt-solutions-01 . . . 41
- 14.6. Changes from draft-hellstrom-mmusic-multi-party-rtt-02 to draft-hellstrom-avtcore-multi-party-rtt-solutions-00 . . . 41
- 14.7. Changes from version draft-hellstrom-mmusic-multi-party-rtt-01 to -02 41
- 15. References 42
 - 15.1. Normative References 42
 - 15.2. Informative References 42
- Author's Address 45

1. Introduction

Real-time text (RTT) is a medium in real-time conversational sessions. Text entered by participants in a session is transmitted in a time-sampled fashion, so that no specific user action is needed to cause transmission. This gives a direct flow of text in the rate it is created, that is suitable in a real-time conversational setting. The real-time text medium can be combined with other media in multimedia sessions.

Media from a number of multimedia session participants can be combined in a multi-party session. The present document specifies how the real-time text streams can be handled in multi-party sessions. Recommendations are provided for preferred methods.

The description is mainly focused on the transport level, but also describes a few session and presentation level aspects.

Transport of real-time text is specified in RFC 4103 [RFC4103] RTP Payload for text conversation. It makes use of RFC 3550 [RFC3550] Real Time Protocol, for transport. Robustness against network transmission problems is normally achieved through redundant transmission based on the principle from RFC 2198 [RFC2198], with one primary and two redundant transmission of each text element. Primary and redundant transmissions are combined in packets and described by a redundancy header. This transport is usually used in the SIP Session Initiation Protocol RFC 3261 [RFC3261] environment.

A very brief overview of functions for real-time text handling in multi-party sessions is described in RFC 4597 [RFC4597] Conferencing Scenarios, sections 4.8 and 4.10. The present specification builds on that description and indicates which protocol mechanisms should be used to implement multi-party handling of real-time text.

Real-time text can also be transported in the WebRTC environment, by using WebRTC data channels according to RFC-to-be 8865 [I-D.ietf-mmusic-t140-usage-data-channel]. Multi-party aspects for WebRTC solutions are briefly covered.

1.1. Requirements Language

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

2. Centralized conference model

In the centralized conference model for SIP, introduced in RFC 4353 [RFC4353] "A Framework for Conferencing with the Session Initiation Protocol (SIP)", one function co-ordinates the communication with participants in the multi-party session. This function also controls media mixer functions for the media appearing in the session. The central function is common for control of all media, while the media mixers may work differently for each media.

The central function is called the Focus UA. Many variants exist for setting up sessions including the multipoint control centre. It is not within scope of this description to describe these, but rather the media specific handling in the mixer required to handle multi-party calls with RTT.

The main principle for handling real-time text media in a centralized conference is that one RTP session for real-time text is established including the multipoint media control centre and the participating endpoints which are going to have real-time text exchange with the others.

The different possible mechanisms for mixing and transporting RTT differs in the way they multiplex the text streams and how they identify the sources of the streams. RFC 7667 [RFC7667] describes a number of possible use cases for RTP. This specification refers to different sections of RFC 7667 for further reading of the situations caused by the different possible design choices.

The recommended method for using RTP based RTT in a centralized conference model is specified in [I-D.ietf-avtcore-multi-party-rtt-mix] based on the recommendations in this document.

Real-time text can also be transported in the WebRTC environment, by using WebRTC data channels according to [I-D.ietf-mmusic-t140-usage-data-channel]. Ways to handle multi-party calls in that environment are also specified.

3. Requirements on multi-party RTT

3.1. General requirements

The following general requirements are placed on multi-party RTT:

A solution shall be applicable to IMS (3GPP TS 22.173) [TS22173], SIP based VoIP and Next Generation Emergency Services (NENA i3 [NENAI3], ETSI TS 103 479 [TS103479], RFC 6443[RFC6443]).

The transmission interval for text should not be longer than 500 milliseconds when there is anything available to send. Ref ITU-T T.140 [T140].

If text loss is detected or suspected, a missing text marker should be inserted in the text stream. Ref ITU-T T.140 Amendment 1 [T140ad1]. ETSI EN 301 549 [EN301549]

The display of text from the members of the conversation shall be arranged so that the text from each participant is clearly readable, and its source and the relative timing of entered text is visualized in the display. Mechanisms for looking back in the contents from the current session should be provided. The text should be displayed as soon as it is received. Ref ITU-T T.140 [T140]

Bridges must be multimedia capable (voice, video, text). Ref NENA i3 STA-010.2. [NENAI3]

It MUST be possible to use real-time text in conferences both as a medium of discussion between individual participants (for example, for sidebar discussions in real-time text while listening to the main conference audio) and for central support of the conference with real-time text interpretation of speech. Ref (R7) in RFC 5194. [RFC5194]

It should be possible to protect RTT contents with usual means for privacy and integrity. Ref RFC 6881 section 16. [RFC6881]

Conferencing procedures are documented in RFC 4579 [RFC4579]. Ref NENA i3 STA-010.2.[NENAI3]

Conferencing applies to any kind of media stream by which users may want to communicate. Ref 3GPP TS 24.147 [TS24147]

The framework for SIP conferences is specified in RFC 4353 [RFC4353]. Ref 3GPP TS 24.147 [TS24147]

3.2. Performance requirements

The mixer performance requirements can be expressed in one number, extracted from the user requirements on real-time text expressed in ITU-T F.700, where it is stated that for "good" usability, text characters should not be delayed more than 1 second from creation to presentation. For "usable" usability the figure is 2 seconds. The main factor behind these limits is from when taking turns in a conversation gets disturbed by a delay of when a response gets visible to the receiving part. If that times get too long, the receiving part gets unsure if the previous utterance was well perceived and the receiving part maybe prepares for repetition. This is similar to the same effect in voice communication, where the usability limit is 400 ms delay.

Another important factor in a multi-party conference is the opportunity for a participant using real-time text to provide timely comments and get a chance to enter the discussion if the majority of participants use voice in the conference. A complicating factor when stating the requirements is that some transport methods do not cause a total delay, but instead an increasing jerkiness when the number of simultaneously sending participants is increased.

It should however be remembered that the expected number of participants sending real-time text simultaneously is low. Just as with voice or sign language, the capability of the participants to perceive utterances from more than one participant at a time is very limited. Therefore the normal case in multi-party situations is that one participant at a time is the main provider of text. Others might usually just provide very brief comments such as "yes" or "no" or "may I comment?". Only at very rare situations two participants provide more information simultaneously.

* The number of expected simultaneously transmitting users is different for different applications. In all cases, just one transmitting user is the normal case. Two simultaneously transmitting participants can occasionally be expected in emergency services, relay services, small unmanaged conferences and group calls and large managed conferences. Three

simultaneously transmitting participants may appear occasionally in large unmanaged conferences. The following can therefore express the performance requirement.

- * The mean delay of text passing the mixer introduced when only one participant is sending text should be kept to a minimum and should not be more than 400 ms.
- * The mean delay of text passing the mixer should not be more than 1 second during moments when up to three users are sending text simultaneously.
- * For the very rare case that more than three participants send text simultaneously, the mixer may take action to limit the introduced delay of the text passing the mixer to 7 seconds e.g. by discarding text from some participants and instead inserting a general warning about possible text loss in the stream.

4. RTP based solutions

4.1. Coordination of text RTP streams

Coordinating and sending text RTP streams in the multi-party session can be done in a number of ways. The most suitable methods are specified here with pros and cons.

A receiving and presenting endpoint **MUST** separate text from the different sources and identify and display them accordingly.

4.1.1. RTP-based solutions with a central mixer

A set of solutions can be based on the central RTP mixer. They are described here and a preferred method selected.

4.1.1.1. RTP Mixer using default RFC 4103 methods

Without any extra specifications, a mixer would transmit with 300 milliseconds intervals, and use RFC 4103 [RFC4103] with the default redundancy of one original and two redundant transmissions. The source of the text would be indicated by a single member in the CSRC list. Text from different sources cannot be transmitted in the same packet. Therefore, from the time when the mixer sent one piece of new text from one source, it will need to transmit that text again twice as redundant data, before it can send text from another source. The jerkiness = time between transmission of new text is 900 ms. This is clearly insufficient.

Pros:

Only a capability negotiation method is needed. No other update of standards are needed, just a general remark that traditional RTP-mixing is used.

Cons:

Clearly insufficient mixer switching performance.

A bit complex handling of transmission when there is new text available from more than one source. The mixer needs to send two packets more with redundant text from the current source before starting to send anything from the other source.

4.1.1.2. RTP Mixer using the default method but decreased transmission interval

This method makes use of the default RTP-mixing method briefly described in Section 4.1.1.1. The only difference is that the transmission interval is decreased to 100 milliseconds when there is text from more than one source available for transmission. The jerkiness is 300 ms. The mean delay with two simultaneously sending participants is 250 ms, and with three simultaneously sending participants 500 ms. This is acceptable performance.

Pros:

Minor influence on standards

Can be relatively rapidly be introduced in the intended technical environments.

Can be declared in sdp as the already existing "text/red" format with a multi-party attribute for capability negotiation.

Cons:

The introduced jerkiness of new text from more than the required three simultaneously sending sources is high.

Slightly higher risk for loss of text at bursty packet loss than for the recommended transmission interval (300 ms) for RFC 4103.

When complete loss of packets occur (beyond recovery), it is not possible to deduce from which source text was lost.

A bit complex handling of transmission when there is new text available from more than one source. The mixer needs to send two packets more with redundant text from the current source before starting to send anything from the other source.

4.1.1.3. RTP Mixer with frequent transmission and indicating sources in CSRC-list

An RTP media mixer combines text from participants into one RTP stream, thus all using the same destination address/port combination, the same RTP SSRC, and one sequence number series as described in Section 7.1 and 7.3 of RTP RFC 3550 [RFC3550] about the Mixer function. This method is also briefly described in RFC 7667, section 3.6.1 Media mixing mixer [RFC7667].

The sources of the text in each RTP packet are identified by the CSRC list in the RTP packets, containing the SSRC of the initial sources of text. The order of the CSRC parameters is with the SSRC of the source of the primary text first, followed by the SSRC of the first level redundancy, and then the second level redundancy.

The transmission interval should be 100 milliseconds when there is text to transmit from more than one source, and otherwise 300 ms.

The identification of the sources is made through the CSRC fields and can be made more readable at the receiver through the RTCP SDES CNAME and NAME packets as described in RTP[RFC3550].

Information provided through the notification according to RFC 4575 [RFC4575] when the participant joined the conference provides also suitable information and a reference to the SSRC.

A receiving endpoint is supposed to separate text items from the different sources and identify and display them accordingly.

The ordered CSRC lists in the RFC 4103 [RFC4103] packets make it possible to recover from loss of one and two packets in sequence and assign the recovered text to the right source. For more loss, a marker for possible loss should be inserted or presented.

The conference server needs to have authority to decrypt the payload in the received RTP packets in order to be able to recover text from redundant data or insert the missing text marker in the stream, and repack the text in new packets.

Even if the format is very similar to "text/red" of RFC 4103, it needs to be declared as a new media subtype, e.g. "text/rex".

Pros:

This method has low overhead and less complexity than the methods in Section 4.1.1.1, Section 4.1.1.2, Section 4.1.1.4 and Section 4.1.1.6.

When loss of packets occur, it is possible to recover text from redundancy at loss of up to the number of redundancy levels carried in the RFC 4103 [RFC4103] stream (normally primary and two redundant levels).

This method can be implemented with most RTP implementations.

The source switching performance is sufficient for well-behaving conference participants. The jerkiness is 100 ms.

Cons:

When more consecutive packet loss than the number of generations of redundant data appears, it is not possible to deduce the sources of the totally lost data.

Slightly higher risk for loss of text at bursty packet loss than for the recommended transmission interval for RFC 4103.

Requires a different sub media format, e.g. "text/rex". This takes a long time in standardisation and releases of target technical environments.

The conference server needs to be allowed to decrypt/encrypt the packet payload. This is however normal for media mixers for other media.

4.1.1.4. RTP Mixer interleaving packets, receiver using timestamp to recover from loss

This method has text only from one source per packet, as the original RFC 4103 [RFC4103] specifies. Packets with text from different sources are instead allowed to be interleaved. The recovery procedure in the receiver makes use of the RTP timestamp and timestamp offsets in the redundancy headers to evaluate if a piece of redundant data was received earlier or not as a base for decision if the redundant data should be recovered or not in case of packet loss.

In this method, the transmission interval is 100 milliseconds when text (new or redundant) from more than one source is available for transmission. Otherwise it is 320 ms or following the timing of received packets.

Pros:

The format of each packet is equal to what is specified in RFC 4103 [RFC4103].

The source switching performance is sufficient and good. Text from five participants can be transmitted simultaneously with 500 milliseconds interval per source.

New text from five simultaneous sources can be transmitted within 500 milliseconds. This is sufficient.

Recovery from packet loss with five simultaneous sources takes 1 second. This is good and implies good protection against bursty packet loss causing resulting text loss.

Cons:

The recovery time in case of packet loss is long with more than ten simultaneously sending participants. Then it will be more than 2 seconds.

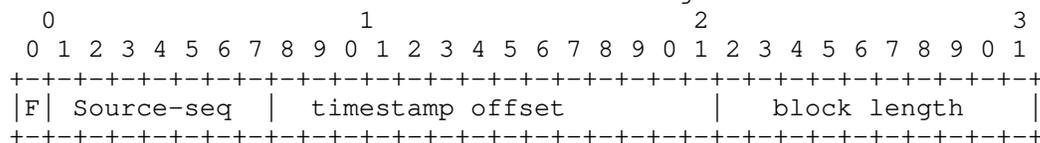
The recovery procedure is different from what is described in RFC 4103 [RFC4103].

It will in many cases of loss of multiple packets not be possible to deduce if there was any resulting loss of text. A mark for possible loss should be inserted in cases when there might have been resulting loss.

4.1.1.5. RTP Mixer with multiple primary data in each packet and individual sequence numbers

This method allows primary as well as redundant text from more than one source per packet. The packet payload contains an ordered set of redundant and primary data with the same number of generations of redundancy as once agreed in the SDP negotiation. The data header reflects these parts of the payload. The CSRC list contains one CSRC member per source in the payload and in the same order. An individual sequence number per source is included in the data header replacing the t140 payload type number that is instead assumed to be constant in this format. This allows an individual extra sequence number per source with maximum value 127, suitable for checking for which source loss of text appeared when recovery was not possible.

The data header would contain the following fields:



Where "Source-seq" is the sequence number per source.

The maximum number of members in the CSRC-list is 15, and that is therefore the maximum number of sources that can be represented in each packet provided that all data can be fitted into the size allowable in one packet.

Transmission is done as soon as there is new text available, but not with shorter interval than 150 ms and not longer than 300 ms while there is anything to send.

A new media subtype is needed, e.g. "text/rex".

This is an SDP offer example for both traditional "text/red" and multi-party "text/rex" format:

```
m=text 11000 RTP/AVP 101 100 98
a=rtpmap:98 t140/1000
a=rtpmap:100 red/1000
a=rtpmap:101 rex/1000
a=fmtp:100 98/98/98
a=fmtp:101 98/98/98
```

Pros:

The source switching performance is good. Text from 15 participants can be transmitted simultaneously.

New text from 15 simultaneous sources can be transmitted within 300 milliseconds. This is good performance.

When more consecutive packet loss than the number of generations of redundant data appears, it is still possible to deduce the sources of the totally lost data, when next text from these sources arrive.

Cons:

The format of each packet is different from what is specified in RFC 4103 [RFC4103].

The processing time in standard organisation will be long.

A new media subtype is needed, causing a bit complex negotiation.

The recovery procedure is a bit complex.

4.1.1.6. RTP Mixer with multiple primary data in each packet

This method allows primary as well as redundant text from more than one source per packet. The packet payload contains an ordered set of redundant and primary data with the same number of generations of redundancy as once agreed in the SDP negotiation. The data header reflects these parts of the payload. The CSRC list contains one CSRC member per source in the payload and in the same order.

The maximum number of members in the CSRC-list is 15, and that is therefore the maximum number of sources that can be represented in each packet provided that all data can be fitted into the size allowable in one packet.

Transmission is done as soon as there is new text available, but not with shorter interval than 150 ms and not longer than 300 ms while there is anything to send.

A new media subtype is needed, e.g. "text/rex".

SDP would be the same as in Section 4.1.1.6.

Pros:

The source switching performance is good. Text from 15 participants can be transmitted simultaneously.

New text from 15 simultaneous sources can be transmitted within 150 milliseconds. This is good performance.

Cons:

The format of each packet is different from what is specified in RFC 4103 [RFC4103].

A new media subtype is needed.

A new media subtype is needed, causing a bit complex negotiation.

The processing time in standard organisation will be long.

The recovery procedure is a bit complex [RFC4103].

When more consecutive packet loss than the number of generations of redundant data appears, it is not possible to deduce the sources of the totally lost data.

4.1.1.7. RTP Mixer with RFC 5109 FEC and RFC 2198 redundancy in the packets

This method allows primary data from one source and redundant text from other sources in each packet. The packet payload contains primary data in "text/t140" format, and redundant data in RFC 5109 FEC [RFC5109] format called "text/ulpfec". That means that the redundant data contains the sequence number and the CSRC and other characteristics from the RTP header when the data was sent as primary. The redundancy can be sent at a selected number of packets after when it was sent as primary, in order to improve the protection against bursty packet loss. The redundancy level is recommended to be the same as in original RFC 4103.

RFC 4103 says that the protection against loss can be made by other methods than plain redundancy, so this method is in line with that statement.

Transmission is done as soon as there is new text available, but not with shorter interval than 100 ms and not longer than 300 ms while there is anything to send (new or redundant text).

When more consecutive packet loss than the number of generations of redundant data appears, it is not possible to deduce the sources of the totally lost data.

The sdp can indicate the format as "text/red" with "text/ulpfec" redundant data in this way. with traditional RFC 4103 with "text/red" with "text/t140" as redundant data as a fallback.

```
m=text 49170 RTP/AVP 98 101 100 102
a=rtpmap:98 red/1000
a=fmtp:98 100/102/102
a=rtpmap:102 ulpfec/1000
a=rtpmap:100 t140/1000
a=rtpmap:101 red/1000
a=fmtp:101 100/100/100
a=fmtp:100 cps=200
```

The "text/ulpfec" format includes an indication of how far back the redundancy belongs, making it possible to cover bursty packet loss better than the other formats with short transmission intervals. For real-time text, it is recommended to send three packets between the primary and the redundant transmissions of text. That makes the

transmission cover between 500 and 1500 ms of bursty packet loss. The variation is because of the varying packet interval between many and one simultaneously transmitting source.

The "text/ulpfec" format has a number of parameters. One is the length of the data to be protected which in this case must be the whole t140block.

Pros:

The source switching performance is good. Text from 5 participants can be transmitted within 500 ms.

Good recovery from bursty packet loss.

The method is based on existing standards. No new registrations are needed.

Cons:

When more consecutive packet loss than the number of generations of redundant data appears, it is not possible to deduce the sources of the totally lost data.

Even if the switching performance is good, it is not as good as for the method called "RTP Mixer with multiple primary data in each packet" Section 4.1.1.6. With more than 5 simultaneously sending sources, there will be a noticeable delay of text of over 500 ms, with 100 ms added per simultaneous source. This is however beyond the requirements and would be a concern only in congestion situations.

The recovery procedure is a bit complex [RFC5109].

There is more overhead in terms of extra data and extra packets sent than in the other methods. With the recommended two redundant generations of data, each packet will be 36 bytes longer than with traditional RFC 4103, and at each pause in transmission five extra packets with only redundant data will be sent compared to two extra packets for the traditional RFC 4103 case.

4.1.1.8. RTP Mixer with RFC 5109 FEC and RFC 2198 redundancy and separate sequence number in the packets

This method allows primary data from one source and redundant text from other sources in each packet. The packet payload contains primary data in a new "text/t140e" format, and redundant data in RFC 5109 FEC [RFC5109] format called "text/ulpfec". That means that the redundant data contains the sequence number and the CSRC and other characteristics from the RTP header when the data was sent as primary. The redundancy can be sent at a selected number of packets after when it was sent as primary, in order to improve the protection against bursty packet loss. The redundancy level is recommended to be the same as in original RFC 4103. The "text/t140e" format contains a source-specific sequence number and the t140block.

RFC 4103 says that the protection against loss can be made by other methods than plain redundancy, so this method is in line with that statement.

Transmission is done as soon as there is new text available, but not with shorter interval than 100 ms and not longer than 300 ms while there is anything to send (new or redundant text).

When more consecutive packet loss than the number of generations of redundant data appears, it is possible to deduce which sources lost data when new data arrives from the sources. This is done by monitoring the received source specific sequence numbers preceding the text.

This is an example of how can indicate the format as "text/red" with "text/t140e" as primary and "text/ulpfec" redundant data, with traditional RFC 4103 with "text/red" with "text/t140" as redundant data as a fallback.

```
m=text 49170 RTP/AVP 98 101 100 102 103
a=rtpmap:98 red/1000
a=fmtp:98 100/102/102
a=rtpmap:102 ulpfec/1000
a=rtpmap:103 t140/1000
a=rtpmap:100 t140e/1000
a=rtpmap:101 red/1000
a=fmtp:101 103/103/103
a=fmtp:100 cps=200
```

The "text/ulpfec" format includes an indication of how far back the redundancy belongs, making it possible to cover bursty packet loss better than the other formats with short transmission intervals. For real-time text, it is recommended to send three packets between the

primary and the redundant transmissions of text. That makes the transmission cover between 500 and 1500 ms of bursty packet loss. The variation is because of the varying packet interval between many and one simultaneously transmitting source.

The "text/ulpfec" format has a number of parameters. One is the length of the data to be protected which in this case must be the whole t140block.

Pros:

The source switching performance is good. Text from 5 participants can be transmitted within 500 ms.

Good recovery from bursty packet loss.

The method is based on an existing standard for FEC.

When more consecutive packet loss than the number of generations of redundant data appears, it is possible to deduce the source of the lost data when new text arrives from the source.

Cons:

Even if the switching performance is good, it is not as good as for the method called "RTP Mixer with multiple primary data in each packet" Section 4.1.1.6. With more than 5 simultaneously sending sources, there will be a noticeable delay of text of over 500 ms, with 100 ms added per simultaneous source. This is however beyond the requirements and would be a concern only in congestion situations.

The recovery procedure is a bit complex [RFC5109].

There is more overhead in terms of extra data and extra packets sent than in the other methods. With the recommended two redundant generations of data, each packet will be 40 bytes longer than with traditional RFC 4103, and at each pause in transmission five extra packets with only redundant data will be sent compared to two extra packets for the traditional RFC 4103 case.

A new text media subtype "text/t140e" needs to be registered.

The processing time in standard organisation will be long.

4.1.1.9. RTP Mixer indicating participants by a control code in the stream

Text from all participants except the receiving one is transmitted from the media mixer in the same RTP session and stream, thus all using the same destination address/port combination, the same RTP SSRC and , one sequence number series as described in Section 7.1 and 7.3 of RTP RFC 3550 [RFC3550] about the Mixer function. The sources of the text in each RTP packet are identified by a new defined T.140 control code "c" followed by a unique identification of the source in UTF-8 string format.

The receiver can use the string for presenting the source of text. This method is on the RTP level described in RFC 7667, section 3.6.1 Media mixing mixer [RFC7667].

The inline coding of the source of text is applied in the data stream itself, and an RTP mixer function is used for coordinating the sources of text into one RTP stream.

Information uniquely identifying each user in the multi-party session is placed as the parameter value "n" in the T.140 application protocol function with the function code "c". The identifier shall thus be formatted like this: SOS c n ST, where SOS and ST are coded as specified in ITU-T T.140 [T140]. The "c" is the letter "c". The n parameter value is a string uniquely identifying the source. This parameter shall be kept short so that it can be repeated in the transmission without concerns for network load.

A receiving endpoint is supposed to separate text items from the different sources and identify and display them accordingly.

The conference server need to be allowed to decrypt/encrypt the packet payload in order to check the source and repack the text.

Pros:

If loss of packets occur, it is possible to recover text from redundancy at loss of up to the number of redundancy levels carried in the RFC 4103 [RFC4103]stream. (normally primary and two redundant levels.

This method can be implemented with most RTP implementations.

The method can also be used with other transports than RTP

Cons:

The method implies a moderate load by the need to insert the source often in the stream.

If more consecutive packet loss than the number of generations of redundant data appears, it is not possible to deduce the source of the totally lost data.

The mixer needs to be able to generate suitable and unique source identifications which are suitable as labels for the sources.

Requires an extension on the ITU-T T.140 standard, best made by the ITU.

There is a risk that the control code indicating the change of source is lost and the result is false source indication of text.

The conference server need to be allowed to decrypt/encrypt the packet payload.

4.1.1.10. Mixing for multi-party unaware user agents

Multi-party real-time text contents can be transmitted to multi-party unaware user agents if source labelling and formatting of the text is performed by a mixer. This method has the limitations that the layout of the presentation and the format of source identification is purely controlled by the mixer, and that only one source at a time is allowed to present in real-time. Other sources need to be stored temporarily waiting for an appropriate moment to switch the source of transmitted text. The mixer controls the switching of sources and inserts a source identifier in text format at the beginning of text after switch of source. The logic of the mixer to detect when a switch is appropriate should detect a number of places in text where a switch can be allowed, including new line, end of sentence, end of phrase, a period of inactivity, and a word separator after a long time of active transmission.

This method MAY be used when no support for multi-party awareness is detected in the receiving endpoint. The base for this method is described in RFC 7667, section 3.6.1 Media mixing mixer [RFC7667].

See [I-D.ietf-avtcore-multi-party-rtt-mix] for a procedure for mixing RTT for a conference-unaware endpoint.

Pros:

Can be transmitted to conference-unaware endpoints.

Can be used with other transports than RTP

Cons:

Does not allow full real-time presentation of more than one source at a time. Text from other sources will be delayed.

The only realistic presentation format is a style with the text from the different sources presented with a text label indicating source, and the text collected in a chat style presentation but with more frequent turn-taking.

Endpoints often have their own system for adding labels to the RTT presentation. In that case there will be two levels of labels in the presentation, one for the mixer and one for the sources.

If loss of more packets than can be recovered by the redundancy appears, it is not possible to detect which source was struck by the loss. It is also possible that a source switch occurred during the loss, and therefore a false indication of the source of text can be provided to the user after such loss.

Because of all these cons, this method is not recommended be used as the main method, but only as fallback and the last resort for backwards interoperability with multi-party unaware endpoints.

The conference server need to be allowed to decrypt/encrypt the packet payload.

4.1.2. RTP-based bridging with minor RTT media contents reformatting by the bridge

It may be desirable to send text in a multi-party setting in a way that allows the text stream contents to be distributed without being dealt with in detail in any central server. A number of such methods are described. However, when writing this specification, no one of these methods have a specified way of establishing the session by sdp.

4.1.2.1. RTP Translator sending one RTT stream per participant

Within the RTP session, text from each participant is transmitted from the RTP media translator (bridge) in a separate RTP stream, thus using the same destination address/port combination, the same payload type number (PT) but separate RTP SSRC parameters and sequence number series as described in Section 7.1 and 7.2 of RTP RFC 3550 [RFC3550] about the Translator function. The source of the text in each RTP packet is identified by the SSRC parameter in the RTP packets, containing the SSRC of the initial source of text.

A receiving and presenting endpoint is supposed to separate text items from the different sources and identify and display them in a suitable way.

This method is described in RFC 7667, section 3.5.1 Relay-transport translator or 3.5.2 Media translator [RFC7667].

The identification of the source is made through the SSRC. The translation to a readable label can be done by mapping to information from the RTCP SDES CNAME and NAME packets as described in RTP[RFC3550], and also through information in the text media member in the conference notification described in RFC 4575 [RFC4575].

The sdp exchange for establishing this mixing type can be equal to what is used for basic two-party use of RFC 4103 with just an added attribute for indicating multi-party capability.

```
m=text 49170 RTP/AVP 98 103
a=rtpmap:98 red/1000
a=fmtp:98 103/103/103
a=rtpmap:103 t140/1000
a=fmtp:103 cps=150
a=RTT-mixing:RTP-translator
```

A similar answer including the same RTT-mixing attribute would indicate that multi-party coding can begin. An answer without the same RTT-mixing attribute could result in diversion to use of the mixing method for multi-party unaware endpoints Section 4.1.1.10 if more than two parties are involved in the session.

The bridge can add new sources in the communication to a participant by first sending a conference notification according to RFC 4575 [RFC4575] with the SSRC of the new source included in the corresponding "text" media member, or by sending an RTCP message with the new SSRC in an SDES packet.

A receiver should be prepared to receive such indications of new streams being added to the multi-party session, so that the new SSRC is not taken for a change in SSRC value for an already established RTP stream.

Transmission, reception, packet loss recovery and text loss indication is performed per source in the separate RTP streams in the same way as in two-party sessions with RFC 4103 [RFC4575].

Text is recommended to be sent by the bridge as soon as it is available for transmission, but not less than 250 ms after a previous transmission. This will in many cases result in close to 0 added delay by the bridge, because most RTT senders use a 300 ms transmission interval.

It is sometimes said that this configuration is not supported by current media declarations in sdp. RFC 3264 [RFC3264] specifies in some places that one media description is supposed to describe just one RTP media stream. However this is not directly referencing an RTP stream, and use of multiple RTP streams in the same RTP session is recommended in many other RFCs.

This confusion is clarified in RFC 5576 [RFC5576] section 3 by the following statements:

"The term "media stream" does not appear in the SDP specification itself, but is used by a number of SDP extensions, for instance, Interactive Connectivity Establishment (ICE) [ICE], to denote the object described by an SDP media description. This term is unfortunately rather confusing, as the RTP specification [RFC3550] uses the term "media stream" to refer to an individual media source or RTP packet stream, identified by an SSRC, whereas an SDP media stream describes an entire RTP session, which can contain any number of RTP sources."

In most cases, it will be sufficient that new sources are introduced with a conference notification or RTCP message. However, RFC 5576 [RFC5576] specifies attributes which may be used to more explicitly announce new sources or restart of earlier established RTP streams.

This method is encouraged by draft-ietf-avtcore-multiplex-guidelines [I-D.ietf-avtcore-multiplex-guidelines] section 5.2.

Normal operation will be that the bridge receives text packets from the source and handles any text recovery and indication of loss needed before queueing the resulting clean text for transmission from the bridge to the receivers.

It may however also be possible for the bridge to just convey the packet contents as received from the sources, with minor adjustments, and let the receiving endpoint handle all aspects of recovery and indication of loss, even for the source to bridge path. In that case also the sequence number must be maintained as it was at reception in the bridge. This mode needs further study before application.

Pros:

This method is the natural way to do multi-party bridging with RFC 4103 based RTT. Only a small addition is included in the session establishment to verify capability by the parties because many implementations are done without multi-party capability.

This method has moderate overhead in terms of work for the mixer, but high in terms of packet transmission rate. Five sources sending simultaneously cause the bridge to send 15 packets per second to each receiver.

When loss of packets occur, it is possible to recover text from redundancy at loss of up to the number of redundancy levels carried in the RFC 4103 [RFC4103] stream (normally primary and two redundant levels).

More loss than what can be recovered, can be detected and the marker for text loss can be inserted in the correct stream.

It may be possible in some scenarios to keep the text encrypted through the Translator.

Minimal delay. The delay can often be kept close to 0 with at least 5 simultaneous sending participants.

Cons:

There are RTP implementations not supporting the Translator model. They will need to use the fall-back to multi-party-unaware mixing. An investigation about how common this is is needed before the method is used.

The processing time in standard organisation will be long.

With many simultaneous sending sources, the total rate of packets will be high, and can cause congestion. The requirement to handle 3 simultaneous sources in this specification will cause 10 packets per second that is manageable in most cases, e.g. considering that audio usually use 50 packets per second.

4.1.2.2. Distributing packets in an end-to-end encryption structure

In order to achieve end-to-end encryption, it is possible to let the packets from the sources just pass through a central distributor, and handle the security agreements between the participants. Specifications exist for a framework with this functionality for application on RTP based conferences in [I-D.ietf-perc-private-media-framework]. The RTP flow and mixing characteristics has similarities with the method described under "RTP

Translator sending one RTT stream per participant" above. RFC 4103 RTP streams [RFC4103] would fit into the structure and it would provide a base for end-to-end encrypted rtt multi-party conferencing.

Pros:

Good security

Straightforward multi-party handling.

Cons:

Does not operate under the usual SIP central conferencing architecture.

Requires the participants to perform a lot of key handling.

Is work in progress when this is written.

4.1.2.3. Mesh of RTP endpoints

Text from all participants are transmitted directly to all others in one RTP session, without a central bridge. The sources of the text in each RTP packet are identified by the source network address and the SSRC.

This method is described in RFC 7667, section 3.4 Point to multi-point using mesh [RFC7667].

Pros:

When loss of packets occur, it is possible to recover text from redundancy at loss of up to the number of redundancy levels carried in the RFC 4103 [RFC4103] stream. (normally primary and two redundant levels.

This method can be implemented with most RTP implementations.

Transmitted text can also be used with other transports than RTP

Cons:

This model is not described in IMS, NENA and EENA specifications, and does therefore not meet the requirements.

Requires a drastically increasing number of connections when the number of participants increase.

4.1.2.4. Multiple RTP sessions, one for each participant

Text from all participants are transmitted directly to all others in one RTP session each, without a central bridge. Each session is established with a separate media description in SDP. The sources of the text in each RTP packet are identified by the source network address and the SSRC.

Pros:

When loss of packets occur, it is possible to recover text from redundancy at loss of up to the number of redundancy levels carried in the RFC 4103 [RFC4103] stream. (normally primary and two redundant levels.

Complete loss of text can be indicated in the received stream.

This method can be implemented with most RTP implementations.

End-to-end encryption is achievable.

Cons:

This method is not described in IMS, NENA and ETSI specifications and does therefore not meet the requirements.

A lot of network resources are spent on setting up separate sessions for each participant.

5. Preferred RTP-based multi-party RTT transport method

For RTP transport of RTT using RTP-mixer technology, one method for multi-party mixing and transport stand out as fulfilling the goals best and is therefore recommended. That is: "RTP Mixer interleaving packets, receiver using timestamp to recover from loss"
Section 4.1.1.4

For RTP transport in separate streams or sessions, no current recommendation can be made. A bridging method in the process of standardisation with interesting characteristics is the end-to-end encryption model "perc" Section 4.1.2.2.

6. Session control of RTP-based multi-party RTT sessions

General session control aspects for multi-party sessions are described in RFC 4575 [RFC4575] A Session Initiation Protocol (SIP) Event Package for Conference State, and RFC 4579 [RFC4579] Session Initiation Protocol (SIP) Call Control - Conferencing for User Agents. The nomenclature of these specifications are used here.

The procedures for a multi-party aware model for RTT-transmission shall only be applied if a capability exchange for multi-party aware real-time text transmission has been completed and a supported method for multi-party real-time text transmission can be negotiated.

A method for detection of conference-awareness for centralized SIP conferencing in general is specified in RFC 4579 [RFC4579]. The focus sends the "isfocus" feature tag in a SIP Contact header. This causes the conference-aware endpoint to subscribe to conference notifications from the focus. The focus then sends notifications to the endpoint about entering and disappearing conference participants and their media capabilities. The information is carried XML-formatted in a 'conference-info' block in the notification according to RFC 4575 [RFC4575]. The mechanism is described in detail in RFC 4575 [RFC4575].

Before a conference media server starts sending multi-party RTT to an endpoint, a verification of its ability to handle multi-party RTT must be made. A decision on which mechanism to use for identifying text from the different participants must also be taken, implicitly or explicitly. These verifications and decisions can be done in a number of ways. The most apparent ways are specified here and their pros and cons described. One of the methods is selected to be the one to be used by implementations of the centralized conference model according to this specification.

6.1. Implicit RTT multi-party capability indication

Capability for RTT multi-party handling can be decided to be implicitly indicated by session control items.

The focus may implicitly indicate multi-party RTT capability by including the media child with value "text" in the RFC 4575 [RFC4575] conference-info provided in conference notifications.

An endpoint may implicitly indicate multi-party RTT capability by including the text media in the SDP in the session control transactions with the conference focus after the subscription to the conference has taken place.

The implicit RTT capability indication means for the focus that it can handle multi-party RTT according to the preferred method indicated in the RTT multi-party methods section above.

The implicit RTT capability indication means for the endpoint that it can handle multi-party RTT according to the preferred method indicated in the RTT multi-party methods section above.

If the focus detects that an endpoint implicitly declared RTT multi-party capability, it SHALL provide RTT according to the preferred method.

If the focus detects that the endpoint does not indicate any RTT multi-party capability, then it shall either provide RTT multi-party text in the way specified for conference-unaware endpoint above, or refuse to set up the session.

If the endpoint detects that the focus has implicitly declared RTT multi-party capability, it shall be prepared to present RTT in a multi-party fashion according to the preferred method.

Pros:

Acceptance of implicit multi-party capability implies that no standardisation of explicit RTT multi-party capability exchange is required.

Cons:

If other methods for multi-party RTT are to be used in the same implementation environment as the preferred ones, then capability exchange needs to be defined for them.

Cannot be used outside a strictly applied SIP central conference model.

6.2. RTT multi-party capability declared by SIP media-tags

Specifications for RTT multi-party capability declarations can be agreed for use as SIP media feature tags, to be exchanged during SIP call control operation according to the mechanisms in RFC 3840 [RFC3840] and RFC 3841 [RFC3841]. Capability for the RTT Multi-party capability is then indicated by the media feature tag "rtt-mix", with a set of possible values for the different possible methods.

The possible values in the list may for example be:

rtp-mixer

perc

rtp-mixer indicates capability for using the RTP-mixer based presentation of multi-party text.

perc indicates capability for using the perc based transmission of multi-party text.

Example: Contact: <sip:a2@beco.example.com>

```
;methods="INVITE,ACK,OPTIONS,BYE,CANCEL"
```

```
;+sip.rtt-mix="rtp-mixer"
```

If, after evaluation of the alternatives in this specification, only one mixing method is selected to be brought to implementation, then the media tag can be reduced to a single tag with no list of values.

An offer-answer exchange should take place and the common method selected by the answering party shall be used in the session with that UA.

When no common method is declared, then only the fallback method for multi-party unaware participants can be used, or the session dropped.

If more than one text media section is included in SDP, all must be capable of using the declared RTT multi-party method.

Pros:

Provides a clear decision method.

Can be extended with new mixing methods.

Can guide call routing to a suitable capable focus.

Cons:

Requires standardization and IANA registration.

Is not stream specific. If more than one text stream is specified, all must have the same type of multi-party capability.

Cannot be used in the WebRTC environment.

6.3. SDP media attribute for RTT multi-party capability indication

An attribute can be specified on media level, to be used in text media SDP declarations for negotiating RTT multi-party capabilities. The attribute can have the name "rtt-mixing".

More than one attribute can be included in one media description.

The attribute can have a value. The value can for example be:

rtp-mixer

rtp-translator

perc

rtp-mixer indicates capability for using the RTP-mixer and CSRC-list based mixing of multi-party text.

rtp-translator indicates capability for using the RTP-translator based mixing

perc indicates capability for using the perc based transmission of multi-party text.

An offer-answer exchange should take place and the common method selected by the answering party shall be used in the session with that endpoint.

When no common method is declared, then only the fallback method for multi-party unaware endpoints can be used.

Example: a=rtt-mixing:rtp-mixer

If, after evaluation of the alternatives in this specification, only one mixing method is selected to be brought to implementation, then the attribute can be reduced to a single attribute with no list of values.

Pros:

Provides a clear decision method.

Can be extended with new mixing methods.

Can be used on specific text media.

Can be used also for SDP-controlled WebRTC sessions with multiple streams in the same data channel.

Cons:

Requires standardization and IANA registration.

Cannot guide SIP routing.

6.4. Simplified SDP media attribute for RTT multi-party capability indication

An attribute can be specified on media level, to be used in text media SDP declarations for negotiating RTT multi-party capabilities. The attribute can have a name suitable for the selected method and no value. It would be selected and used if only one method for multi-party rtt is brought forward from this specification, and the other left unspecified for now or found to be possible to negotiate in another way.

An offer-answer exchange should take place and if both parties specify rtt-mixing capability with the same attribute, the selected mixing method shall be used.

When no common method is declared, then only the fallback method for multi-party unaware endpoints can be used, or the session not accepted for multi-party use.

Example: a=rtt-mix

Pros:

Provides a clear decision method.

Very simple syntax and semantics.

Can be used on specific text media.

Cons:

Requires standardization and IANA registration.

If another RTT mixing method is also specified in the future, then that method may also need to specify and register its own attribute, instead of if an attribute with a parameter value is used, when only an addition of a new possible value is needed.

Cannot guide SIP routing.

6.5. SDP format parameter for RTT multi-party capability indication

An FMTP format parameter can be specified for the RFC 4103 [RFC4103]media, to be used in text media SDP declarations for negotiating RTT multi-party capabilities. The parameter can have the name "rtt-mixing", with one or more of its possible values.

The possible values in the list are:

rtp-mixer

perc

rtp-mixer indicates capability for using the RTP-mixer based mixing and presentation of multi-party text using the CSRC-list.

perc indicates capability for using the perc based transmission of multi-party text.

Example: a=fmtp 96 98/98/98 rtt-mixing=rtp-mixer

If, after evaluation of the alternatives in this specification, only one mixing method is selected to be brought to implementation, then the parameter can be reduced to a single parameter with no list of values.

An offer-answer exchange should take place and the common method selected by the answering party shall be used in the session with that UA.

When no common method is declared, then only the fallback method can be used, or the session denied.

Pros:

Provides a clear decision method.

Can be extended with new mixing methods.

Can be used on specific text media.

Can be used also for SDP-controlled WebRTC sessions with multiple streams in the same data channel.

Cons:

Requires standardization and IANA registration.

May cause interop problems with current RFC4103 [RFC4103] implementations not expecting a new fntp-parameter.

Cannot guide SIP routing.

6.6. A text media subtype for support of multi-party rtt

Indicating a specific text media subtype in SDP is a straightforward way for negotiating multi-party capability. Especially if there are format differences from the "text/red" and "text/t140" formats of RFC4103 [RFC4103], then this is a natural way to do the negotiation for multi-party rtt.

Pros:

No extra efforts if a new format is needed anyway.

Cons:

None specific to using the format indication for negotiation of multi-party capability. But only feasible if a new format is needed anyway.

6.7. Preferred capability declaration method for RTP-based transport.

If the preferred transport method is one with a specific media subtype in sdp, then specification by media subtype is preferred.

If this would not be the case, then the preferred capability declaration method would be the one with a specific SDP attribute for the selected mixing method Section 6.4 because it is straightforward.

6.8. Identification of the source of text for RTP-based solutions

The main way to identify the source of text in the RTP based solution is by the SSRC of the sending participant. In the RTP-mixer solution, this SSRC is included in the CSRC list of the transmitted packets. Further identification that may be needed for better labelling of received text may be achieved from a number of sources. It may be the RTCP SDES CNAME and NAME reports, and in the conference notification data (RFC 4575) [RFC4575].

As soon as a new member is added to the RTP session, its characteristics should be transmitted in RTCP SDES CNAME and NAME reports according to section 6.5 in RFC 3550 [RFC3550]. The information about the participant should also be included in the conference data including the text media member in a notification according to RFC 4575 [RFC4575].

The RTCP SDES report, SHOULD contain identification of the source represented by the SSRC/CSRC identifier. This identification MUST contain the CNAME field and MAY contain the NAME field and other defined fields of the SDES report.

A focus UA SHOULD primarily convey SDES information received from the sources of the session members. When such information is not available, the focus UA SHOULD compose SSRC/CSRC, CNAME and NAME information from available information from the SIP session with the participant.

Provision of detailed information in the NAME field has security implications, especially if provided without encryption.

7. RTT bridging in WebRTC

Within WebRTC, real-time text is specified to be carried in WebRTC data channels as specified in [I-D.ietf-mmusic-t140-usage-data-channel]. A few ways to handle multi-party RTT are mentioned briefly. They are repeated below.

7.1. RTT bridging in WebRTC with one data channel per source

A straightforward way to handle multi-party RTT is for the bridge to open one T.140 data channel per source towards the receiving participants.

The stream-id forms a unique stream identification.

The identification of the source is made through the Label property of the channel, and session information belonging to the source. The endpoint can compose a readable label for the presentation from this information.

Pros:

This is a straightforward solution.

The load per source is low.

Cons:

With a high number of participants, the overhead of establishing and maintaining the high number of data channels required may be high, even if the load per channel is low.

7.2. RTT bridging in WebRTC with one common data channel

A way to handle multi-party RTT in WebRTC is for the bridge combine text from all sources into one data channel and insert the sources in the stream by a T.140 control code for source.

This method is described in a corresponding section for RTP transmission above in Section 4.1.1.9.

The identification of the source is made through insertion in the beginning of each text transmission from a source of a control code extension "c" followed by a string representing the source, framed by the control code start and end flags SOS and ST (See ITU-T T.140 [T140]).

A receiving endpoint is supposed to separate text items from the different sources and identify and display them in a suitable way.

The endpoint does not always display the source identification in the received text at the place where it is received, but has the information as a guide for planning the presentation of received text. A label corresponding to the source identification is presented when needed depending on the selected presentation style.

Pros:

This solution has relatively low overhead on session and network level

Cons:

This solution has higher overhead on the media contents level than the WebRTC solution above.

Standardisation of the new control code "c" in ITU-T T.140 [T140] is required.

The conference server need to be allowed to decrypt/encrypt the data channel contents.

7.3. Preferred rtt multi-party method for WebRTC

For WebRTC, one method is to prefer because of the simplicity. So, for WebRTC, the method to implement for multi-party RTT with multi-party aware parties when no other method is explicitly agreed between implementing parties is: "RTT bridging in WebRTC with one data channel per source" Section 7.1.

8. Presentation of multi-party text

All session participants with RTP based transport MUST observe the SSRC/CSRC field of incoming text RTP packets, and make note of which source they came from in order to be able to present text in a way that makes it easy to read text from each participant in a session, and get information about the source of the text.

In the WebRTC case, the Label parameter and other provided endpoint information should be used for the same purpose.

8.1. Associating identities with text streams

A source identity SHOULD be composed from available information sources and displayed together with the text as indicated in ITU-T T.140 Appendix[T140].

The source identity should primarily be the NAME field from incoming SDES packets. If this information is not available, and the session is a two-party session, then the T.140 source identity SHOULD be composed from the SIP session participant information. For multi-party sessions the source identity may be composed by local information if sufficient information is not available in the session.

Applications may abbreviate the presented source identity to a suitable form for the available display.

Applications may also replace received source information with internally used nicknames.

8.2. Presentation details for multi-party aware endpoints.

The multi-party aware endpoint should after any action for recovery of data from lost packets, separate the incoming streams and present them according to the style that the receiving application supports and the user has selected. The decisions taken for presentation of the multi-party interchange shall be purely on the receiving side. The sending application must not insert any item in the stream to influence presentation that is not requested by the sending participant.

8.2.1. Bubble style presentation

One often used style is to present real-time text in chunks in readable bubbles identified by labels containing names of sources. Bubbles are placed in one column in the presentation area and are closed and moved upwards in the presentation area after certain items or events, when there is also newer text from another source that would go into a new bubble. The text items that allows bubble closing are any character closing a phrase or sentence followed by a space or a timeout of a suitable time (about 10 seconds).

Real-time active text sent from the local user should be presented in a separate area. When there is a reason to close a bubble from the local user, the bubble should be placed above all real-time active bubbles, so that the time order that real-time text entries were completed is visible.

Scrolling is usually provided for viewing of recent or older text. When scrolling is done to an earlier point in the text, the presentation shall not move the scroll position by new received text. It must be the decision of the local user to return to automatic viewing of latest text actions. It may be useful with an indication that there is new text to read after scrolling to an earlier position has been activated.

The presentation area may become too small to present all text in all real-time active bubbles. Various techniques can be applied to provide a good overview and good reading opportunity even in such situations. The active real-time bubble may have a limited number of lines and if their contents need more lines, then a scrolling opportunity within the real-time active bubble is provided. Another method can be to only show the label and the last line of the active real-time bubble contents, and make it possible to expand or compress the bubble presentation between full view and one line view.

Erasures require special consideration. Erasure within a real-time active bubble is straightforward. But if erasure from one participant affects the last character before a bubble, the whole previous bubble becomes the actual bubble for real-time action by that participant and is placed below all other bubbles in the presentation area. If the border between bubbles was caused by the CRLF characters (instead of the normal "Line Separator"), only one erasure action is required to erase this bubble border. When a bubble is closed, it is moved up, above all real-time active bubbles.

A three-party view is shown in this example .

	^
	-
[Alice] Hi, Alice here.	
[Bob] Bob as well.	
[Eve] Hi, this is Eve, calling from Paris. I thought you should be here.	
[Alice] I am coming on Thursday, my performance is not until Friday morning.	
[Bob] And I on Wednesday evening.	
[Alice] Can we meet on Thursday evening?	
[Eve] Yes, definitely. How about 7pm. at the entrance of the restaurant Le Lion Blanc?	
[Eve] we can have dinner and then take a walk	
<Eve-typing> But I need to be back to the hotel by 11 because I need	
<Bob-typing> I wou	-
of course, I underst	v

Figure 1: Three-party call with bubble style.

Figure 1: Example of a three-party call presented in the bubble style.

8.2.2. Other presentation styles

Other presentation styles than the bubble style may be arranged and appreciated by the users. In a video conference one way may be to have a real-time text area below the video view of each participant. Another view may be to provide one column in a presentation area for each participant and place the text entries in a relative vertical position corresponding to when text entry in them was completed. The labels can then be placed in the column header. The considerations for ending and moving and erasure of entered text discussed above for the bubble style are valid also for these styles.

This figure shows how a coordinated column view MAY be presented.

Bob	Eve	Alice
My flight is to Orly		I will arrive by TGV. Convenient to the main station.
Eve, will you do your presentation on Friday?	Hi all, can we plan for the seminar?	
Fine, wo	Yes, Friday at 10.	We need to meet befo

Figure 2: A coordinated column-view of a three-party session with entries ordered in approximate time-order.

9. Presentation details for multi-party unaware endpoints.

Multi-party unaware endpoints are prepared only for presentation of two sources of text, the local user and a remote user. If mixing for multi-party unaware endpoints is to be supported, in order to enable some multi-party communication with such endpoint, the mixer need to plan the presentation and insert labels and line breaks before lables. Many limitations appear for this presentation mode, and it must be seen as a fallback and a last resort.

A procedure for presenting RTT to a conference-unaware endpoint is included in [I-D.ietf-avtcore-multi-party-rtt-mix]

10. Security Considerations

The security considerations valid for RFC 4103 [RFC4103] and RFC 3550 [RFC3550] are valid also for the multi-party sessions with text.

11. IANA Considerations

The items for indication and negotiation of capability for multi-party rtt should be registered with IANA in the specifications where they are specified in detail.

12. Congestion considerations

The congestion considerations described in RFC 4103 [RFC4103] are valid also for the recommended RTP-based multi-party use of the real-time text transport. A risk for congestion may appear if a number of conference participants are active transmitting text simultaneously, because the recommended RTP-based multi-party transmission method does not allow multiple sources of text to contribute to the same packet.

In situations of risk for congestion, the Focus UA MAY combine packets from the same source to increase the transmission interval per source up to one second. Local conference policy in the Focus UA may be used to decide which streams shall be selected for such transmission frequency reduction.

13. Acknowledgements

Arnoud van Wijk for contributions to an earlier, expired draft of this memo.

14. Change history

14.1. Changes to draft-hellstrom-avtcore-multi-party-rtt-solutions-05

Modify the solution changing source in every packet in the RTP-mixer solution, and base recovery on analyzing timestamp and make it the recommended one. Aligned with the recommendation in draft-ietf-avtcore-multi-party-rtt-mix-10.

14.2. Changes to draft-hellstrom-avtcore-multi-party-rtt-solutions-04

Change name of simplified sdp attribute to "rtt-mix" to match a change in the draft draft-ietf-avtcore-multi-party-rtt-mix-09.

14.3. Changes to draft-hellstrom-avtcore-multi-party-rtt-solutions-03

Modified info on the method with RFC 4103 format and sdp attribute "rtt-mix-rtp-mixer".

Increased the performance requirements section.

Inserted recommendations, with emphasis on ease of implementation and ease of standardisation.

14.4. Changes to draft-hellstrom-avtcore-multi-party-rtt-solutions-02

Added detail in the section on RTP translator model alternative 4.1.2.1.

14.5. Changes to draft-hellstrom-avtcore-multi-party-rtt-solutions-01

Added three more methods for RTP-mixer mixing. Two RFC 5109 FEC based and another with modified data header to detect source of completely lost text.

Separated RTP-based and WebRTC based solutions.

Deleted the multi-party-unaware mixing procedure appendix. It is now included in the draft draft-ietf-avtcore-multi-party-rtt-mix. Kept a section with a reference to the new place.

14.6. Changes from draft-hellstrom-mmusic-multi-party-rtt-02 to draft-hellstrom-avtcore-multi-party-rtt-solutions-00

Add discussion about switching performance, as discussed in avtcore on March 13.

Added that a decrease of transmission interval to 100 ms increases switching performance by a factor 3, but still not sufficient.

Added that the CSRC-list method also uses 100 milliseconds transmission interval.

Added the method with multiple primary text in each packet.

Added the timestamp-based method for rtp-mixing proposed by James Hamlin on March 14.

Corrected the chat style presentation example picture. Delete a few "[mix]".

14.7. Changes from version draft-hellstrom-mmusic-multi-party-rtt-01 to -02

Change from a general overview to overview with clear recommendations.

Splits text coordination methods in three groups.

Recommends rtt-mixer with sources in CSRC-list but refers to its spec for details.

Shortened Appendix with conference-unaware example.

Cleaned up preferences.

Inserted pictures of screen-views.

15. References

15.1. 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>>.

15.2. Informative References

[EN301549] ETSI, "EN 301 549. Accessibility requirements for ICT products and services", November 2019, <https://www.etsi.org/deliver/etsi_en/301500_301599/301549/03.01.01_60/en_301549v030101p.pdf>.

[I-D.ietf-avtcore-multi-party-rtt-mix]
Hellstrom, G., "RTP-mixer formatting of multi-party Real-time text", Work in Progress, Internet-Draft, draft-ietf-avtcore-multi-party-rtt-mix-10, 18 November 2020, <<https://tools.ietf.org/html/draft-ietf-avtcore-multi-party-rtt-mix-10>>.

[I-D.ietf-avtcore-multiplex-guidelines]
Westerlund, M., Burman, B., Perkins, C., Alvestrand, H., and R. Even, "Guidelines for using the Multiplexing Features of RTP to Support Multiple Media Streams", Work in Progress, Internet-Draft, draft-ietf-avtcore-multiplex-guidelines-12, 16 June 2020, <<https://tools.ietf.org/html/draft-ietf-avtcore-multiplex-guidelines-12>>.

[I-D.ietf-mmusic-t140-usage-data-channel]
Holmberg, C. and G. Hellstrom, "T.140 Real-time Text Conversation over WebRTC Data Channels", Work in Progress, Internet-Draft, draft-ietf-mmusic-t140-usage-data-channel-14, 10 April 2020, <<https://tools.ietf.org/html/draft-ietf-mmusic-t140-usage-data-channel-14>>.

- [I-D.ietf-perc-private-media-framework]
Jones, P., Benham, D., and C. Groves, "A Solution Framework for Private Media in Privacy Enhanced RTP Conferencing (PERC)", Work in Progress, Internet-Draft, draft-ietf-perc-private-media-framework-12, 5 June 2019, <<https://tools.ietf.org/html/draft-ietf-perc-private-media-framework-12>>.
- [NENAI3] NENA, "NENA-STA-010.2-2016. Detailed Functional and Interface Standards for the NENA i3 Solution", October 2016, <https://www.nena.org/page/i3_Stage3>.
- [RFC2198] Perkins, C., Kouvelas, I., Hodson, O., Hardman, V., Handley, M., Bolot, J.C., 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>>.
- [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>>.
- [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>>.
- [RFC3840] Rosenberg, J., Schulzrinne, H., and P. Kyzivat, "Indicating User Agent Capabilities in the Session Initiation Protocol (SIP)", RFC 3840, DOI 10.17487/RFC3840, August 2004, <<https://www.rfc-editor.org/info/rfc3840>>.
- [RFC3841] Rosenberg, J., Schulzrinne, H., and P. Kyzivat, "Caller Preferences for the Session Initiation Protocol (SIP)", RFC 3841, DOI 10.17487/RFC3841, August 2004, <<https://www.rfc-editor.org/info/rfc3841>>.
- [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>>.

- [RFC4353] Rosenberg, J., "A Framework for Conferencing with the Session Initiation Protocol (SIP)", RFC 4353, DOI 10.17487/RFC4353, February 2006, <<https://www.rfc-editor.org/info/rfc4353>>.
- [RFC4575] Rosenberg, J., Schulzrinne, H., and O. Levin, Ed., "A Session Initiation Protocol (SIP) Event Package for Conference State", RFC 4575, DOI 10.17487/RFC4575, August 2006, <<https://www.rfc-editor.org/info/rfc4575>>.
- [RFC4579] Johnston, A. and O. Levin, "Session Initiation Protocol (SIP) Call Control - Conferencing for User Agents", BCP 119, RFC 4579, DOI 10.17487/RFC4579, August 2006, <<https://www.rfc-editor.org/info/rfc4579>>.
- [RFC4597] Even, R. and N. Ismail, "Conferencing Scenarios", RFC 4597, DOI 10.17487/RFC4597, August 2006, <<https://www.rfc-editor.org/info/rfc4597>>.
- [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>>.
- [RFC5194] van Wijk, A., Ed. and G. Gybels, Ed., "Framework for Real-Time Text over IP Using the Session Initiation Protocol (SIP)", RFC 5194, DOI 10.17487/RFC5194, June 2008, <<https://www.rfc-editor.org/info/rfc5194>>.
- [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>>.
- [RFC6443] Rosen, B., Schulzrinne, H., Polk, J., and A. Newton, "Framework for Emergency Calling Using Internet Multimedia", RFC 6443, DOI 10.17487/RFC6443, December 2011, <<https://www.rfc-editor.org/info/rfc6443>>.
- [RFC6881] Rosen, B. and J. Polk, "Best Current Practice for Communications Services in Support of Emergency Calling", BCP 181, RFC 6881, DOI 10.17487/RFC6881, March 2013, <<https://www.rfc-editor.org/info/rfc6881>>.
- [RFC7667] Westerlund, M. and S. Wenger, "RTP Topologies", RFC 7667, DOI 10.17487/RFC7667, November 2015, <<https://www.rfc-editor.org/info/rfc7667>>.

- [T140] ITU-T, "Recommendation ITU-T T.140 (02/1998), Protocol for multimedia application text conversation", February 1998, <<https://www.itu.int/rec/T-REC-T.140-199802-I/en>>.
- [T140ad1] ITU-T, "Recommendation ITU-T.140 Addendum 1 - (02/2000), Protocol for multimedia application text conversation", February 2000, <<https://www.itu.int/rec/T-REC-T.140-200002-I!Add1/en>>.
- [TS103479] ETSI, "TS 103 479. Emergency communications (EMTEL); Core elements for network independent access to emergency services", December 2019, <https://www.etsi.org/deliver/etsi_ts/103400_103499/103479/01.01.01_60/ts_103479v010101p.pdf>.
- [TS22173] 3GPP, "IP Multimedia Core Network Subsystem (IMS) Multimedia Telephony Service and supplementary services; Stage 1", 3GPP TS 22.173 17.1.0, 20 December 2019, <<http://www.3gpp.org/ftp/Specs/html-info/22173.htm>>.
- [TS24147] 3GPP, "Conferencing using the IP Multimedia (IM) Core Network (CN) subsystem; Stage 3", 3GPP TS 24.147 16.0.0, 19 December 2019, <<http://www.3gpp.org/ftp/Specs/html-info/24147.htm>>.

Author's Address

Gunnar Hellstrom
Gunnar Hellstrom Accessible Communication
Esplanaden 30
SE-136 70 Vendelso
Sweden

Phone: +46 708 204 288
Email: gunnar.hellstrom@ghaccess.se

Network Working Group
Internet-Draft
Intended status: Informational
Expires: 3 May 2021

S. Hurst
BBC Research & Development
30 October 2020

QRT: QUIC RTP Tunnelling
draft-hurst-quic-rtp-tunnelling-00

Abstract

QUIC is a UDP-based transport protocol for stream-orientated, congestion-controlled, secure, multiplexed data transfer. RTP carries real-time data between endpoints, and the accompanying control protocol RTCP allows monitoring and control of the transfer of such data. With RTP and RTCP being agnostic to the underlying transport protocol, it is possible to multiplex both the RTP and associated RTCP flows into a single QUIC connection to take advantage of QUIC features such as low-latency setup and strong TLS-based security.

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 3 May 2021.

Copyright Notice

Copyright (c) 2020 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	2
1.1.	Conventions and Definitions	3
1.2.	Definitions	3
2.	Use Cases for an RTP Mapping over QUIC	4
2.1.	Live Event Contribution Feed	4
2.2.	Audio and Video Conference via a Central Server	5
3.	QRT Sessions	5
4.	RTP Sessions	5
4.1.	QRT Flow Identifier	6
4.2.	RTCP Mapping	7
4.2.1.	Restricted RTCP Packet Types	7
5.	Loss Recovery and Retransmission	8
6.	Using the Session Description Protocol to Advertise QRT Sessions	8
6.1.	Using the Session Description Protocol to Advertise QRT Sessions using RTP Retransmission	9
7.	Exposing Round-Trip Time to RTP applications	10
8.	Protocol Identifier	10
8.1.	Draft Version Identification	10
9.	Security Considerations	11
10.	IANA Considerations	11
10.1.	Registration of Protocol Identification String	11
10.2.	Registration of SDP Protocol Identifier	11
10.3.	Registration of SDP Attribute Field	11
11.	References	12
11.1.	Normative References	12
11.2.	Informative References	13
	Acknowledgments	14
	Author's Address	14

1. Introduction

The Real-time Transport Protocol (RTP) [RFC3550] provides end-to-end network transport functions suitable for applications transmitting data, such as audio and video, over multicast or unicast network services for the purposes of telephony, video streaming, conferencing and other real-time applications.

The QUIC transport protocol is a UDP-based stream-orientated and encrypted transport protocol aimed at offering improvements over the common combination of TCP and TLS for web applications. Compared with TCP+TLS, QUIC offers much reduced connection set-up times,

improved stream multiplexing aware congestion control, and the ability to perform connection migration. QUIC offers two modes of data transfer:

- * Reliable transfer using STREAM frames, as specified in [QUIC-TRANSPORT], [QUIC-RECOVERY], etc.
- * Unreliable transfer using DATAGRAM extension frames, as specified in [QUIC-DATAGRAM].

RTP has traditionally been run over UDP or DTLS to achieve timely but unreliable data transfer. For use cases such as real-time audio and video transmission, the underlying media codecs can be considered in part fault-tolerant to an unreliable transport mechanism, with missing data from the stream resulting in glitches in the media presentation, such as missing video frames or gaps in audio playback. By purposely using an unreliable transport mechanism, applications can minimise the added latency that would otherwise result from managing the large packet reception buffers needed to account for network reordering or transport protocol retransmission.

1.1. Conventions and Definitions

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 BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

Packet and frame diagrams in this document use the format described in [QUIC-TRANSPORT].

1.2. Definitions

- * Endpoint: host capable of being a participant in a QRT session.
- * QRT session: A QUIC connection carrying one or more RTP sessions, each with or without an accompanying RTCP channel.
- * Client: The endpoint which initiates the QUIC connection.
- * Server: The endpoint which accepts the incoming QUIC connection.

2. Use Cases for an RTP Mapping over QUIC

The following sections describe some possible use cases for an RTP and RTCP mapping over QUIC, hereafter QRT. The examples were chosen to illustrate some basic concepts, and are neither an exhaustive list of possible use cases nor a limitation on what QRT may be used for.

2.1. Live Event Contribution Feed

A news organisation wishes to provide a two-way link to a live event for distribution as part of an item in a news programme hosted in a studio with a news anchor. The single camera remote production crew will include a camera operator, sound technician and the reporter. In order to deliver this experience, the following media flows are required:

- * A high-quality video feed from the remote camera to the news organisation's gallery;
- * One or more audio feeds for microphones at the event, including an ambient microphone attached to the camera, a lapel microphone for the reporter, and a handheld microphone to conduct interviews, all synchronized;
- * A video feed of the programme output from the gallery, after mixing for local monitoring and for use on a comfort monitor;
- * An audio feed from the anchor in the studio to the reporter;
- * A two-way audio feed from the gallery to the remote production crew for talkback communication;
- * A tally light feed for the remote camera.

These media flows may be realised as a group of RTP sessions, some of which must be synchronised together. The talkback streams do not require any tight synchronisation with other streams in the group, whereas the camera video feed and various microphone feeds need to be tightly synchronised together.

At the event, a production machine running a software package that includes a QRT client has two connections to the Internet; a high-speed fibre link and a bonded cellular network link for backup.

In order to prevent a bad actor on the network path being able to tamper with the contribution, all communication between the news organisation's gallery and the remote production need to be encrypted. Because all the data is flowing between the same two

endpoints, only a single QRT session is required, and the various RTP sessions that are encapsulated by the QRT session are (de)multiplexed at each end.

During the live contribution, an accident cuts the fibre connection to the remote production crew. Using the QUIC connection migration mechanism presented in Section 9 of [QUIC-TRANSPORT], the QRT session migrates from the fibre link onto the backup cellular link. This preserves the state of the RTP sessions across a network migration event, and all sessions continue.

2.2. Audio and Video Conference via a Central Server

A teleconference is taking place across multiple sites using a centralised server. All participants connect to this single server, and the server acts as an RTP mixer to reduce the number of RTP sessions being sent to all participants, as well as re-encoding the streams for efficiency reasons.

One participant of this conference has connected via mobile phone. However, when the participant enters the range of a previously-associated WiFi network, the mobile phone switches its network connection across to this new network. The QRT session can then migrate across, and the participant is able to continue the call with minimal interruption.

3. QRT Sessions

A QRT session is defined as a QUIC connection which carries one or more RTP sessions (including any associated RTCP flows) using "DATAGRAM" frames, as specified in Section 4. Those RTP sessions may be part of one or more RTP multimedia sessions, and a multimedia session may be comprised of RTP sessions carried in one or more QRT sessions.

A QRT session inherits the standard QUIC handshake as specified in [QUIC-TRANSPORT], and all communications between endpoints are secured as specified in [QUIC-TLS].

4. RTP Sessions

QRT allows multiple RTP sessions to be carried in a single QRT session. Each RTP session is operated independently of all the others, and individually discriminated by an QRT Flow Identifier, as described below in Section 4.1.

RTP and RTCP packets are carried in QUIC "DATAGRAM" frames, as described in [QUIC-DATAGRAM]. QUIC allows multiple QUIC frames to be carried within a single QUIC packet, so multiple RTP/RTCP packets for one (or more) RTP sessions may therefore be carried in a single QUIC packet, subject to the network path MTU. If multiple RTP packets are to be carried within a single QUIC packet, then all but the final "DATAGRAM" frame must specify the length of the datagram, since the RTP packet header does not provide its own length field. [QUIC-DATAGRAM] specifies that if a "DATAGRAM" frame is received without a Length field, then this "DATAGRAM" frame extends to the end of the QUIC packet.

4.1. QRT Flow Identifier

[RFC3550] specifies that RTP sessions are distinguished by pairs of transport addresses. However, since QUIC allows for connections to migrate between transport address associations, and because we wish to multiplex multiple RTP session flows over a single QRT session, this profile of RTP amends this statement and instead introduces a flow identifier to distinguish between RTP sessions. The QRT Flow Identifier is a 62-bit unsigned integer between 0 and $2^{62} - 1$.

This specification does not mandate a means by which QRT Flow Identifiers are allocated for use within QRT sessions. An example mapping for this is discussed in Section 6 below. Implementations SHOULD allocate flow identifiers that make the most efficient use of the variable length integer packing mechanism, by not using flow identifiers greater than can be expressed in the smallest variable length integer field until all available flow identifiers have been used.

The flow of packets belonging to an RTP session is identified using an RTP Session Flow Identifier header carried in the "DATAGRAM" frame payload before each RTP/RTCP packet. This flow identifier is encoded as a variable-length integer, as defined in [QUIC-TRANSPORT].

```
QRT Datagram Payload {  
    QRT Flow Identifier (i),  
    RTP/RTCP Packet (...)  
}
```

Figure 1: QRT Datagram Payload

Similar to QUIC stream IDs, the least significant bit (0x1) of the QRT Flow Identifier distinguishes between an RTP and an RTCP packet flow. "DATAGRAM" frames which carry RTP packet flows set this bit to 0, and "DATAGRAM" frames which carry RTCP packet flows set this bit to 1. As a consequence, RTP packet flows have even numbered QRT Flow

Identifiers, and RTCP packet flows have odd-numbered QRT Flow Identifiers. Carriage of RTCP packets is discussed further in Section 4.2.

Least significant bit	Flow identifier category
0x0	RTP packet flow for an RTP session
0x1	RTCP packet flow for an RTP session

Table 1: RTP session flow identifier categories

Author's Note: The author welcomes comments on whether a state model of RTP session flows would be beneficial. Currently, once an RTP session has been used by an endpoint, it is then considered an extant RTP session and implementations would have to keep any resources allocated to that RTP session until the QRT session is complete. In addition, how should endpoints react to receiving packets for unknown QRT flow identifiers?

4.2. RTCP Mapping

An RTP session may have RTCP packet flows associated with it. These flows are carried with different QRT Flow Identifiers, as described in Section 4.1. The QRT Flow Identifier of the RTCP packet flow is always the value of the RTP packet flow QRT Flow Identifier + 1. For example, for an RTP packet flow using flow identifier 18, the RTCP packet flow would use flow identifier 19.

Since RTCP packets contain a length field in their header, implementations MAY combine several RTCP packets pertaining to the same RTP session into a single "DATAGRAM" frame. Alternatively, implementations MAY choose to carry these RTCP packets each in their own "DATAGRAM" frame.

4.2.1. Restricted RTCP Packet Types

Author's Note: I have specifically avoided calling this section "Prohibited RTCP packet types" for the time being, so as to not unnecessarily exclude the carriage of these packet types for the purposes of experimentation. Similarly, most statements below use SHOULD NOT instead of MUST NOT. The author welcomes comments on whether the document should prohibit the sending of some or all of these packet types.

In order to reduce duplication, the following RTCP packet types SHOULD NOT be sent in a QRT session:

- * The "Generic NACK" packet. [RFC4585] states that Generic NACK feedback SHOULD NOT be used if the underlying transport protocol is capable of providing similar feedback information to the sender. Since all "DATAGRAM" frames are ACK-eliciting, QUIC already fulfills this requirement.
- * The "Loss RLE" Extended Report (XR) packet defined in [RFC3611] contains information that should already be known to both ends of the QUIC connection by means of the loss detection mechanism specified in [QUIC-RECOVERY].
- * The "Port Mapping" packet type defined in [RFC6284] is used to negotiate UDP port pairs for the carriage of RTP and RTCP packets to peers. This does not apply in a QRT session, because the QUIC endpoints manage the UDP port association(s) for the QUIC connection as a whole.

5. Loss Recovery and Retransmission

**Author's Note:* Do we want to mandate (make a MUST) doing session-multiplexing instead of SSRC-multiplexing for RTP retransmission?

[RFC4588] specifies two schemes to support retransmission in the case of RTP packet loss. Since QRT natively supports RTP session multiplexing on a single QUIC connection, endpoints choosing to implement retransmission SHOULD do so using the session-multiplexing scheme.

The selection of a new QRT Flow Identifier to use for the retransmission RTP session is implementation-specific. Section 6.1 specifies how the mapping between original and retransmission RTP sessions is expressed using the Session Description Protocol (SDP).

6. Using the Session Description Protocol to Advertise QRT Sessions

[RFC4566] describes a format for advertising multimedia sessions, which is used by protocols such as [RFC3261].

This specification introduces a new SDP value attribute "qrtflow" as a means of assigning QRT Flow Identifiers to RTP and RTCP packet flows. Its formatting in SDP is described by the following ABNF [RFC5234]:

```
qrtflow-attribute = "a=qrtflow:" qrt-flow-id
qrt-flow-id       = 1*DIGIT ; unsigned 62-bit integer
```

Per Section 4.1 the value of the "qrt-flow-id" is required to be an even number. (The odd-numbered RTCP flow associated with the RTP session is not explicitly signalled in the SDP object.)

The example in Figure 2 below shows a hypothetical QRT server advertising an endpoint to use for live contribution. It instructs a prospective client to send a VC2-encoded video stream and a Vorbis-encoded audio stream on two separate RTP sessions. In addition, it uses the SDP grouping framework described in [RFC5888] to ensure lip synchronisation between both of those RTP sessions.

```
v=0
o=gfreeman 1594130940 1594135167 IN IP6 qrt.example.org
s=Live Event Contribution
c=IN IP6 2001:db8::7361:6d68
t=1594130980 1594388466
a=group:LS 1 2
m=video 443 RTP/QRT 96
a=qrtflow:0
a=rtpmap:96 vc2
a=mid:1
a=sendonly
m=audio 443 RTP/QRT 97
a=qrtflow:2
a=rtpmap:97 vorbis
a=mid:2
a=sendonly
```

Figure 2: SDP object describing a QRT session

Since the value of a QRT Flow Identifier for an associated RTCP flow is specified in Section 4.2, SDP advertisements containing the "a=qrtflow:" attribute MUST NOT contain an instance of the "a=rtcp:" attribute as defined in [RFC3605].

6.1. Using the Session Description Protocol to Advertise QRT Sessions using RTP Retransmission

The example in Figure 3 below shows a hypothetical QRT session advertisement for a bidirectional RTP session carrying an MPEG-2 Transport Stream in each direction on QRT Flow Identifier 0, and a corresponding pair of retransmission flows on QRT Flow Identifier 2.

```
v=0
o=gfreeman 1594130940 1594135167 IN IP6 qrt.example.org
s=Live Event Contribution
c=IN IP6 2001:db8::4242:4351:5254
t=1594130980 1594388466
m=video 443 RTP/QRT 33
a=qrtflow:0
m=video 443 RTP/QRT 96
a=rtpmap:97 rtx/90000
a=fmtp:96 apt=33;rtx-time=4000
a=qrtflow:2
```

Figure 3: SDP object describing a QRT session with RTP retransmission

7. Exposing Round-Trip Time to RTP applications

Section 5 of [QUIC-RECOVERY] specifies a mechanism for QUIC endpoints to estimate the round-trip time (RTT) of a connection. QRT implementations SHOULD expose the values of "min_rtt", "smoothed_rtt" and "rttvar" for each network path to the RTP layer, and they MAY use these values either alone or in combination with RTCP messages to discern the round-trip time of the QRT session.

Author's Note: The author welcomes comments on how appropriate these QUIC RTT measurements are to the RTP layer.

8. Protocol Identifier

The QRT protocol specified in this document is identified by the Application-Layer Protocol Negotiation (ALPN) [RFC7301] identifier "qrt".

8.1. Draft Version Identification

RFC Editor's Note: Please remove this section prior to publication of a final version of this document.

Only implementations of the final, published RFC can identify themselves as "qrt". Until such an RFC exists, implementations MUST NOT identify themselves using this string. Implementations of draft versions of the protocol MUST add the string "-h" and the corresponding draft number to the identifier. For example, draft-hurst-quick-rtp-tunnelling-00 is identified using the string "qrt-h00".

Non-compatible experiments that are based on these draft versions MUST append the string "-" and an experiment name to the identifier. For example, an experimental implementation based on draft-hurst-

quic-rtp-tunnelling-00 which uses extension features not registered with the appropriate IANA registry might identify itself as "qrt-h00-extension-foo". Note that any label MUST conform to the "token" syntax defined in Section 5.7.2 of [HTTP-SEMANTICS]. Experimenters are encouraged to coordinate their experiments.

9. Security Considerations

Implementations of the protocol defined in this specification are subject to the security considerations discussed in [QUIC-TRANSPORT] and [QUIC-TLS].

10. IANA Considerations

10.1. Registration of Protocol Identification String

This document creates a new registration for the identification of the QUIC RTP Tunnelling protocol in the "Application-Layer Protocol Negotiation (ALPN) Protocol IDs" registry established by [RFC7301].

The "qrt" string identifies RTP sessions multiplexed and carried over a QUIC transport layer:

Protocol: QUIC RTP Tunnelling

Identification Sequence: 0x71 0x72 0x74 ("qrt")

Specification: This document, Section 8

10.2. Registration of SDP Protocol Identifier

This document creates a new registration for the SDP Protocol Identifier ("proto") "RTP/QRT" in the SDP Protocol Identifiers ("proto") registry established by [RFC4566].

The "RTP/QRT" string identifies a profile of RTP where sessions are multiplexed and carried over a QUIC transport layer:

SDP Protocol Name: RTP/QRT

Reference: This document, Section 6

10.3. Registration of SDP Attribute Field

This document creates a new registration for the SDP Attribute Field ("att-field") "qrtflow" in the SDP Attribute Field registry established by [RFC4566].

SDP Attribute Field: "qrtflow"

Reference: This document, Section 6

11. References

11.1. Normative References

[HTTP-SEMANTICS]

Fielding, R., Ed., Nottingham, M., Ed., and J. Reschke, Ed., "HTTP Semantics", Work in Progress, Internet-Draft, draft-ietf-httpbis-semantic-12, <<https://tools.ietf.org/html/draft-ietf-httpbis-semantic-12>>.

[QUIC-DATAGRAM]

Pauly, T., Ed., Kinnear, E., Ed., and D. Schinazi, Ed., "An Unreliable Datagram Extension to QUIC", Work in Progress, Internet-Draft, draft-ietf-quick-datagram-01, <<https://tools.ietf.org/html/draft-ietf-quick-datagram-01>>.

[QUIC-RECOVERY]

Iyengar, J., Ed. and I. Swett, Ed., "QUIC Loss Detection and Congestion Control", Work in Progress, Internet-Draft, draft-ietf-quick-recovery-32, <<https://tools.ietf.org/html/draft-ietf-quick-recovery-32>>.

[QUIC-TRANSPORT]

Iyengar, J., Ed. and M. Thomson, Ed., "QUIC: A UDP-Based Multiplexed and Secure Transport", Work in Progress, Internet-Draft, draft-ietf-quick-transport-32, <<https://tools.ietf.org/html/draft-ietf-quick-transport-32>>.

[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>>.

[RFC3605] Huitema, C., "Real Time Control Protocol (RTCP) attribute in Session Description Protocol (SDP)", RFC 3605, DOI 10.17487/RFC3605, October 2003, <<https://www.rfc-editor.org/info/rfc3605>>.

- [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>>.
- [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>>.
- [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>>.
- [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>>.
- [RFC7301] Friedl, S., Popov, A., Langley, A., and E. Stephan, "Transport Layer Security (TLS) Application-Layer Protocol Negotiation Extension", RFC 7301, DOI 10.17487/RFC7301, July 2014, <<https://www.rfc-editor.org/info/rfc7301>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.

11.2. Informative References

- [QUIC-TLS] Thomson, M., Ed. and S. Turner, Ed., "Using Transport Layer Security (TLS) to Secure QUIC", Work in Progress, Internet-Draft, draft-ietf-quic-tls-32, <<https://tools.ietf.org/html/draft-ietf-quic-tls-32>>.
- [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>>.
- [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>>.
- [RFC6284] Begen, A., Wing, D., and T. Van Caenegem, "Port Mapping between Unicast and Multicast RTP Sessions", RFC 6284, DOI 10.17487/RFC6284, June 2011, <<https://www.rfc-editor.org/info/rfc6284>>.

Acknowledgments

The author would like to thank Richard Bradbury, David Waring, Colin Perkins, Joerg Ott, and Lucas Pardue for their helpful comments on both the design and review of this document.

Author's Address

Sam Hurst
BBC Research & Development

Email: sam.hurst@bbc.co.uk

IETF RMCAT Working Group
Internet-Draft
Intended status: Standards Track
Expires: March 6, 2021

Z. Sarker
Ericsson AB
C. Perkins
University of Glasgow
V. Singh
callstats.io
M. Ramalho
September 2, 2020

RTP Control Protocol (RTCP) Feedback for Congestion Control
draft-ietf-avtcore-cc-feedback-message-08

Abstract

This document describes an RTCP feedback message intended to enable congestion control for interactive real-time traffic using RTP. The feedback message is designed for use with a sender-based congestion control algorithm, in which the receiver of an RTP flow sends RTCP feedback packets to the sender containing the information the sender needs to perform congestion control.

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 March 6, 2021.

Copyright Notice

Copyright (c) 2020 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	2
2. Terminology	3
3. RTCP Feedback for Congestion Control	3
3.1. RTCP Congestion Control Feedback Report	4
4. Feedback Frequency and Overhead	7
5. Response to Loss of Feedback Packets	8
6. SDP Signalling	8
7. Relation to RFC 6679	9
8. Design Rationale	9
9. Acknowledgements	11
10. IANA Considerations	11
11. Security Considerations	12
12. References	12
12.1. Normative References	12
12.2. Informative References	14
Authors' Addresses	15

1. Introduction

For interactive real-time traffic, such as video conferencing flows, the typical protocol choice is the Real-time Transport Protocol (RTP) [RFC3550] running over the User Datagram Protocol (UDP). RTP does not provide any guarantee of Quality of Service (QoS), reliability, or timely delivery, and expects the underlying transport protocol to do so. UDP alone certainly does not meet that expectation. However, the RTP Control Protocol (RTCP) [RFC3550] provides a mechanism by which the receiver of an RTP flow can periodically send transport and media quality metrics to the sender of that RTP flow. This information can be used by the sender to perform congestion control. In the absence of standardized messages for this purpose, designers of congestion control algorithms have developed proprietary RTCP messages that convey only those parameters needed for their respective designs. As a direct result, the different congestion control 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 packet format that can be

used by NADA [RFC8698], SCReAM [RFC8298], Google Congestion Control [I-D.ietf-rmcat-gcc] and Shared Bottleneck Detection [RFC8382], and hopefully also by future RTP congestion control algorithms.

2. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

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

3. RTCP Feedback for Congestion Control

Based on an analysis of NADA [RFC8698], SCReAM [RFC8298], Google Congestion Control [I-D.ietf-rmcat-gcc] and Shared Bottleneck Detection [RFC8382], the following per-RTP packet congestion control feedback information has been determined to be necessary:

- o RTP sequence number: The receiver of an RTP flow needs to feed the sequence numbers of the received RTP packets back to the sender, so the sender can determine which packets were received and which were lost. Packet loss is used as an indication of congestion by many congestion control algorithms.
- o Packet Arrival Time: The receiver of an RTP flow needs to feed the arrival time of each RTP packet back to the sender. Packet delay and/or delay variation (jitter) is used as a congestion signal by some congestion control algorithms.
- o Packet Explicit Congestion Notification (ECN) Marking: If ECN [RFC3168], [RFC6679] is used, it is necessary to feed back the 2-bit ECN mark in received RTP packets, indicating for each RTP packet whether it is marked not-ECT, ECT(0), ECT(1), or ECN-CE. If the path used by the RTP traffic is ECN capable the sender can use Congestion Experienced (ECN-CE) marking information as a congestion control signal.

Every RTP flow is identified by its Synchronization Source (SSRC) identifier. Accordingly, the RTCP feedback format needs to group its reports by SSRC, sending one report block per received SSRC.

As a practical matter, we note that host operating system (OS) process interruptions can occur at inopportune times. Accordingly, recording RTP packet send times at the sender, and the corresponding

RTP packet arrival times at the receiver, needs to be done 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 needs to be recorded at the last opportunity prior to transmitting the RTP packet at the sender, and the arrival time at the receiver needs to be recorded at the earliest available opportunity.

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 shown in Figure 1:

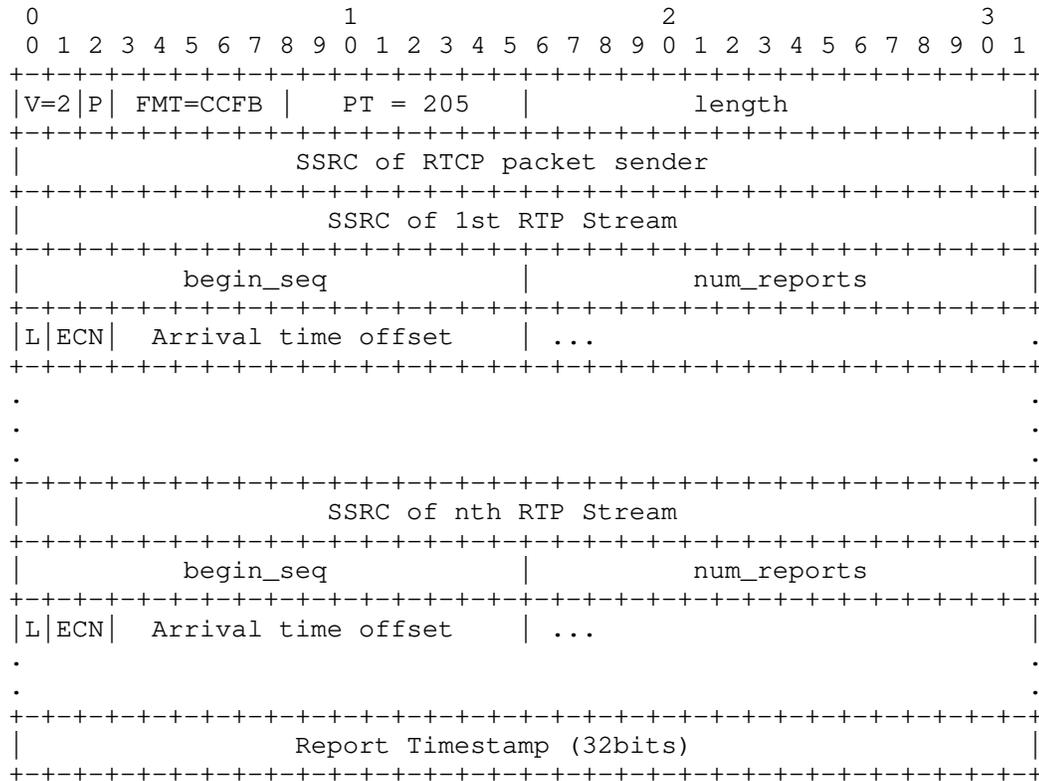


Figure 1: RTCP Congestion Control Feedback Packet Format

The first eight octets comprise a standard RTCP header, with PT=205 and FMT=CCFB indicating that this is a congestion control feedback packet, and with the SSRC set to that of the sender of the RTCP packet. (NOTE TO RFC EDITOR: please replace CCFB here and in the above diagram with the IANA assigned RTCP feedback packet type, and remove this note)

Section 6.1 of [RFC4585] requires the RTCP header to be followed by the SSRC of the RTP flow being reported upon. Accordingly, the RTCP header is followed by a report block for each SSRC from which RTP packets have been received, followed by a Report Timestamp.

Each report block begins with the SSRC of the received RTP Stream on which it is reporting. Following this, the report block contains a 16-bit packet metric block for each RTP packet with sequence number in the range begin_seq to begin_seq+num_reports inclusive (calculated using arithmetic modulo 65536 to account for possible sequence number wrap-around). If the number of 16-bit packet metric blocks included

in the report block is not a multiple of two, then 16 bits of zero padding MUST be added after the last packet metric block, to align the end of the packet metric blocks with the next 32 bit boundary. The value of num_reports MAY be zero, indicating that there are no packet metric blocks included for that SSRC. Each report block MUST NOT include more than 16384 packet metric blocks (i.e., it MUST NOT report on more than one quarter of the sequence number space in a single report).

The contents of each 16-bit packet metric block comprises the L, ECN, and ATO fields as follows:

- o 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 represents that the packet was received and the subsequent bits in the block need to be parsed.
- o 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.
- o Arrival time offset (ATO, 13 bits): is the arrival time of the RTP packet at the receiver, as an offset before the time represented by the Report Timestamp (RTS) field of this RTCP congestion control feedback report. The ATO field is in units of 1/1024 seconds (this unit is chosen to give exact offsets from the RTS field) so, for example, an ATO value of 512 indicates that the corresponding RTP packet arrived exactly half a second before the time instant represented by the RTS field. If the measured value is greater than 8189/1024 seconds (the value that would be coded as 0x1FFD), the value 0x1FFE MUST be reported to indicate an over-range measurement. If the measurement is unavailable, or if the arrival time of the RTP packet is after the time represented by the RTS field, then an ATO value of 0x1FFF MUST be reported for the packet.

The RTCP congestion control feedback report packet concludes with the Report Timestamp field (RTS, 32 bits). This denotes the time instant on which this packet is reporting, and is the instant from which the arrival time offset values are calculated. The value of RTS field is derived from the same clock used to generate the NTP timestamp field in RTCP Sender Report (SR) packets. It is formatted as the middle 32 bits of an NTP format timestamp, as described in Section 4 of [RFC3550].

RTCP congestion control feedback packets SHOULD include a report block for every active SSRC. The sequence number ranges reported on in consecutive reports for a given SSRC will generally be contiguous,

but overlapping reports MAY be sent (and need to be sent in cases where RTP packet reordering occurs across the boundary between consecutive reports). If reports covering overlapping sequence number ranges are sent, information in later reports updates that in sent previous reports for RTP packets included in both reports. If an RTP packet was reported as received in one report, that packet MUST also be reported as received in any overlapping reports sent later that cover its sequence number range.

RTCP congestion control feedback packets can be large if they are sent infrequently relative to the number of RTP data packets. If an RTCP congestion control feedback packet is too large to fit within the path MTU, its sender SHOULD split it into multiple feedback packets. The RTCP reporting interval SHOULD be chosen such that feedback packets are sent often enough that they are small enough to fit within the path MTU ([I-D.ietf-rmcat-rtp-cc-feedback] discusses how to choose the reporting interval; specifications for RTP congestion control algorithms can also provide guidance).

If duplicate copies of a particular RTP packet are received, then the arrival time of the first copy to arrive MUST be reported. If any of the copies of the duplicated packet are ECN-CE marked, then an ECN-CE mark MUST be reported that for packet; otherwise the ECN mark of the first copy to arrive is reported.

If no packets are received from an SSRC in a reporting interval, a report block MAY be sent with `begin_seq` set to the highest sequence number previously received from that SSRC and `num_reports` set to zero (or, the report can simply be omitted). The corresponding SR/RR packet will have a non-increased extended highest sequence number received field that will inform the sender that no packets have been received, but it can ease processing to have that information available in the congestion control feedback reports too.

A report block indicating that certain RTP packets were lost is not to be interpreted as a request to retransmit the lost packets. The receiver of such a report might choose to retransmit such packets, provided a retransmission payload format has been negotiated, but there is no requirement that it do so.

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, suggests desirable RTCP feedback rates, and provides guidance on how to configure the RTCP bandwidth fraction, etc., to make appropriate use of the reporting block described in this memo.

Specifications for RTP congestion control algorithms can also provide guidance.

It is generally understood that congestion control algorithms work better with more frequent feedback. However, RTCP bandwidth and transmission rules put some upper limits on how frequently the RTCP feedback messages can be sent from an RTP receiver to the RTP sender. In many cases, sending feedback once per frame is an upper bound before the reporting overhead becomes excessive, although this will depend on the media rate and more frequent feedback might be needed with high-rate media flows [I-D.ietf-rmcat-rtp-cc-feedback]. Analysis [feedback-requirements] has also shown that some candidate congestion control algorithms can operate with less frequent feedback, using a feedback interval range of 50-200ms. Applications need to negotiate an appropriate congestion control feedback interval at session setup time, based on the choice of congestion control algorithm, the expected media bit rate, and the acceptable feedback overhead.

5. Response to Loss of Feedback Packets

Like all RTCP packets, RTCP congestion control feedback packets might be lost. All RTP congestion control algorithms MUST specify how they respond to the loss of feedback packets.

If only a single congestion control feedback packet is lost, an appropriate response is to assume that the level of congestion has remained roughly the same as the previous report. However, if multiple consecutive congestion control feedback packets are lost, then the sender SHOULD rapidly reduce its sending rate as this likely indicates a path failure. The RTP circuit breaker [RFC8083] provides further guidance.

6. SDP Signalling

A new "ack" feedback parameter, "ccfb", is defined for use with the "a=rtcp-fb:" SDP extension to indicate the use of the RTP Congestion Control feedback packet format defined in Section 3. The ABNF definition of this SDP parameter extension is:

```
rtcp-fb-ack-param = <See Section 4.2 of [RFC4585]>
rtcp-fb-ack-param =/ ccfb-par
ccfb-par           = SP "ccfb"
```

The payload type used with "ccfb" feedback MUST be the wildcard type ("*"). This implies that the congestion control feedback is sent for all payload types in use in the session, including any FEC and

retransmission payload types. An example of the resulting SDP attribute is:

```
a=rtcp-fb:* ack ccfb
```

The offer/answer rules for these SDP feedback parameters are specified in Section 4.2 of the RTP/AVPF profile [RFC4585].

An SDP offer might indicate support for both the congestion control feedback mechanism specified in this memo and one or more alternative congestion control feedback mechanisms that offer substantially the same semantics. In this case, the answering party SHOULD include only one of the offered congestion control feedback mechanisms in its answer. If a re-invite offering the same set of congestion control feedback mechanisms is received, the generated answer SHOULD choose the same congestion control feedback mechanism as in the original answer where possible.

When the SDP BUNDLE extension [I-D.ietf-mmusic-sdp-bundle-negotiation] is used for multiplexing, the "a=rtcp-fb:" attribute has multiplexing category IDENTICAL-PER-PT [I-D.ietf-mmusic-sdp-mux-attributes].

7. Relation to RFC 6679

Use of Explicit Congestion Notification (ECN) with RTP is described in [RFC6679]. That specifies how to negotiate the use of ECN with RTP, and defines an RTCP ECN Feedback Packet to carry ECN feedback reports. It uses an SDP "a=ecn-capable-rtp:" attribute to negotiate use of ECN, and the "a=rtcp-fb:" attributes with the "nack" parameter "ecn" to negotiate the use of RTCP ECN Feedback Packets.

The RTCP ECN Feedback Packet is not useful when ECN is used with the RTP Congestion Control Feedback Packet defined in this memo since it provides duplicate information. Accordingly, when congestion control feedback is to be used with RTP and ECN, the SDP offer generated MUST include an "a=ecn-capable-rtp:" attribute to negotiate ECN support, along with an "a=rtcp-fb:" attribute with the "ack" parameter "ccfb" to indicate that the RTP Congestion Control Feedback Packet is to be used for feedback. The "a=rtcp-fb:" attribute MUST NOT include the "nack" parameter "ecn", so the RTCP ECN Feedback Packet will not be used.

8. Design Rationale

The primary function of RTCP SR/RR packets is to report statistics on the reception of RTP packets. The reception report blocks sent in these packets contain information about observed jitter, fractional

packet loss, and cumulative packet loss. It was intended that this information could be used to support congestion control algorithms, but experience has shown that it is not sufficient for that purpose. An efficient congestion control algorithm requires more fine-grained information on per-packet reception quality than is provided by SR/RR packets to react effectively. The feedback format defined in this memo provides such fine-grained feedback.

Several other RTCP extensions also provide more detailed feedback than SR/RR packets:

TMMBR: The Codec Control Messages for the RTP/AVPF profile [RFC5104] include a Temporary Maximum Media Bit Rate (TMMBR) message. This is used to convey a temporary maximum bit rate limitation from a receiver of RTP packets to their sender. Even though it was 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 when used with non-compound RTCP packets [RFC5506]. This approach requires the receiver of the RTP packets to monitor their reception, determine the level of congestion, and recommend a maximum bit rate suitable for current available bandwidth on the path; it also assumes that the RTP sender can/will respect that bit rate. This is the opposite of the sender-based congestion control approach suggested in this memo, so TMMBR cannot be used to convey the information needed for a sender-based congestion control. TMMBR could, however, be viewed a complementary mechanism that can inform the sender of the receiver's current view of acceptable maximum bit rate. The Received Estimated Maximum Bit-rate (REMB) mechanism [I-D.alvestrand-remcat-remb] provides similar feedback.

RTCP Extended Reports (XR): Numerous RTCP extended report (XR) blocks have been defined to report details of packet loss, arrival times [RFC3611], delay [RFC6843], and ECN marking [RFC6679]. It is possible to combine several such XR blocks into a compound RTCP packet, to report the detailed loss, arrival time, and ECN marking information needed for effective sender-based congestion control. However, the result has high overhead both in terms of bandwidth and complexity, due to the need to stack multiple reports.

Transport-wide Congestion Control: The format defined in this memo provides individual feedback on each SSRC. An alternative is to add a header extension to each RTP packet, containing a single, transport-wide, packet sequence number, then have the receiver send RTCP reports giving feedback on these additional sequence numbers [I-D.holmer-remcat-transport-wide-cc-extensions]. Such an

approach adds the per-packet overhead of the header extension (8 octets per packet in the referenced format), but reduces the size of the feedback packets, and can simplify the rate calculation at the sender if it maintains a single rate limit that applies to all RTP packets sent irrespective of their SSRC. Equally, the use of transport-wide feedback makes it more difficult to adapt the sending rate, or respond to lost packets, based on the reception and/or loss patterns observed on a per-SSRC basis (for example, to perform differential rate control and repair for audio and video flows, based on knowledge of what packets from each flow were lost). Transport-wide feedback is also a less natural fit with the wider RTP framework, which makes extensive use of per-SSRC sequence numbers and feedback.

Considering these issues, we believe it appropriate to design a new RTCP feedback mechanism to convey information for sender-based congestion control algorithms. The new congestion control feedback RTCP packet described in Section 3 provides such a mechanism.

9. Acknowledgements

This document is based on the outcome of a design team discussion in the RTP Media Congestion Avoidance Techniques (RMCAT) working group. The authors would like to thank David Hayes, Stefan Holmer, Randell Jesup, Ingemar Johansson, Jonathan Lennox, Sergio Mena, Nils Ohlmeier, Magnus Westerlund, and Xiaoqing Zhu for their valuable feedback.

10. IANA Considerations

The IANA is requested to register one new RTP/AVPF Transport-Layer Feedback Message in the table for FMT values for RTPFB Payload Types [RFC4585] as defined in Section 3.1:

Name: CCFB
Long name: RTP Congestion Control Feedback
Value: (to be assigned by IANA)
Reference: (RFC number of this document, when published)

The IANA is also requested to register one new SDP "rtcp-fb" attribute "ack" parameter, "ccfb", in the SDP ("ack" and "nack" Attribute Values) registry:

Value name: ccfb
Long name: Congestion Control Feedback
Usable with: ack
Reference: (RFC number of this document, when published)

11. Security Considerations

The security considerations of the RTP specification [RFC3550], the applicable RTP profile (e.g., [RFC3551], [RFC3711], or [RFC4585]), and the RTP congestion control algorithm that is in use (e.g., [RFC8698], [RFC8298], [I-D.ietf-rmcat-gcc], or [RFC8382]) apply.

A receiver that intentionally generates inaccurate RTCP congestion control feedback reports might be able to trick the sender into sending at a greater rate than the path can support, thereby causing congestion on the path. This will negatively impact the quality of experience of that receiver. Since RTP is an unreliable transport, a sender can intentionally leave a gap in the RTP sequence number space without causing harm, to check that the receiver is correctly reporting losses.

An on-path attacker that can modify RTCP congestion control feedback packets can change the reports to trick the sender into sending at either an excessively high or excessively low rate, leading to denial of service. The secure RTCP profile [RFC3711] can be used to authenticate RTCP packets to protect against this attack.

12. References

12.1. Normative 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-54 (work in progress), December 2018.
- [I-D.ietf-mmusic-sdp-mux-attributes]
Nandakumar, S., "A Framework for SDP Attributes when
Multiplexing", draft-ietf-mmusic-sdp-mux-attributes-19
(work in progress), August 2020.
- [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>>.
- [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>>.
- [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>>.
- [RFC8083] Perkins, C. and V. Singh, "Multimedia Congestion Control: Circuit Breakers for Unicast RTP Sessions", RFC 8083, DOI 10.17487/RFC8083, March 2017, <<https://www.rfc-editor.org/info/rfc8083>>.

- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.

12.2. Informative References

- [feedback-requirements]
"RMCAT Feedback Requirements",
<[://www.ietf.org/proceedings/95/slides/slides-95-rmcat-1.pdf](http://www.ietf.org/proceedings/95/slides/slides-95-rmcat-1.pdf)>.
- [I-D.alvestrand-rmcat-remb]
Alvestrand, H., "RTCP message for Receiver Estimated Maximum Bitrate", draft-alvestrand-rmcat-remb-03 (work in progress), October 2013.
- [I-D.holmer-rmcat-transport-wide-cc-extensions]
Holmer, S., Flodman, M., and E. Sprang, "RTP Extensions for Transport-wide Congestion Control", draft-holmer-rmcat-transport-wide-cc-extensions-01 (work in progress), October 2015.
- [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-rtp-cc-feedback]
Perkins, C., "RTP Control Protocol (RTCP) Feedback for Congestion Control in Interactive Multimedia Conferences", draft-ietf-rmcat-rtp-cc-feedback-05 (work in progress), November 2019.
- [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>>.
- [RFC6843] Clark, A., Gross, K., and Q. Wu, "RTP Control Protocol (RTCP) Extended Report (XR) Block for Delay Metric Reporting", RFC 6843, DOI 10.17487/RFC6843, January 2013, <<https://www.rfc-editor.org/info/rfc6843>>.
- [RFC8298] Johansson, I. and Z. Sarker, "Self-Clocked Rate Adaptation for Multimedia", RFC 8298, DOI 10.17487/RFC8298, December 2017, <<https://www.rfc-editor.org/info/rfc8298>>.

- [RFC8382] Hayes, D., Ed., Ferlin, S., Welzl, M., and K. Hiorth, "Shared Bottleneck Detection for Coupled Congestion Control for RTP Media", RFC 8382, DOI 10.17487/RFC8382, June 2018, <<https://www.rfc-editor.org/info/rfc8382>>.
- [RFC8698] Zhu, X., Pan, R., Ramalho, M., and S. Mena, "Network-Assisted Dynamic Adaptation (NADA): A Unified Congestion Control Scheme for Real-Time Media", RFC 8698, DOI 10.17487/RFC8698, February 2020, <<https://www.rfc-editor.org/info/rfc8698>>.

Authors' Addresses

Zaheduzzaman Sarker
Ericsson AB
Torshamnsgatan 21
Stockholm 164 40
Sweden

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

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

Email: csp@csp Perkins.org

Varun Singh
CALLSTATS I/O Oy
Annankatu 31-33 C 42
Helsinki 00100
Finland

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

Michael A. Ramalho
6310 Watercrest Way Unit 203
Lakewood Ranch, FL 34202-5122
USA

Phone: +1 732 832 9723
Email: mar42@cornell.edu
URI: <http://ramalho.webhop.info/>

IETF RMCAT Working Group
Internet-Draft
Intended status: Standards Track
Expires: May 6, 2021

Z. Sarker
Ericsson AB
C. Perkins
University of Glasgow
V. Singh
callstats.io
M. Ramalho
November 2, 2020

RTP Control Protocol (RTCP) Feedback for Congestion Control
draft-ietf-avtc core-cc-feedback-message-09

Abstract

An effective RTP congestion control algorithm requires more fine-grained feedback on packet loss, timing, and ECN marks than is provided by the standard RTP Control Protocol (RTCP) Sender Report (SR) and Receiver Report (RR) packets. This document describes an RTCP feedback message intended to enable congestion control for interactive real-time traffic using RTP. The feedback message is designed for use with a sender-based congestion control algorithm, in which the receiver of an RTP flow sends RTCP feedback packets to the sender containing the information the sender needs to perform congestion control.

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 6, 2021.

Copyright Notice

Copyright (c) 2020 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	2
2. Terminology	3
3. RTCP Feedback for Congestion Control	3
3.1. RTCP Congestion Control Feedback Report	4
4. Feedback Frequency and Overhead	7
5. Response to Loss of Feedback Packets	8
6. SDP Signalling	8
7. Relation to RFC 6679	9
8. Design Rationale	10
9. Acknowledgements	11
10. IANA Considerations	11
11. Security Considerations	12
12. References	13
12.1. Normative References	13
12.2. Informative References	14
Authors' Addresses	15

1. Introduction

For interactive real-time traffic, such as video conferencing flows, the typical protocol choice is the Real-time Transport Protocol (RTP) [RFC3550] running over the User Datagram Protocol (UDP). RTP does not provide any guarantee of Quality of Service (QoS), reliability, or timely delivery, and expects the underlying transport protocol to do so. UDP alone certainly does not meet that expectation. However, the RTP Control Protocol (RTCP) [RFC3550] provides a mechanism by which the receiver of an RTP flow can periodically send transport and media quality metrics to the sender of that RTP flow. This information can be used by the sender to perform congestion control. In the absence of standardized messages for this purpose, designers of congestion control algorithms have developed proprietary RTCP messages that convey only those parameters needed for their respective designs. As a direct result, the different congestion control 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 a generic congestion control feedback format.

To help achieve interoperability for unicast RTP congestion control, this memo proposes a common RTCP feedback packet format that can be used by NADA [RFC8698], SCReAM [RFC8298], Google Congestion Control [I-D.ietf-rmcat-gcc] and Shared Bottleneck Detection [RFC8382], and hopefully also by future RTP congestion control algorithms.

2. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

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

3. RTCP Feedback for Congestion Control

Based on an analysis of NADA [RFC8698], SCReAM [RFC8298], Google Congestion Control [I-D.ietf-rmcat-gcc] and Shared Bottleneck Detection [RFC8382], the following per-RTP packet congestion control feedback information has been determined to be necessary:

- o RTP sequence number: The receiver of an RTP flow needs to feed the sequence numbers of the received RTP packets back to the sender, so the sender can determine which packets were received and which were lost. Packet loss is used as an indication of congestion by many congestion control algorithms.
- o Packet Arrival Time: The receiver of an RTP flow needs to feed the arrival time of each RTP packet back to the sender. Packet delay and/or delay variation (jitter) is used as a congestion signal by some congestion control algorithms.
- o Packet Explicit Congestion Notification (ECN) Marking: If ECN [RFC3168], [RFC6679] is used, it is necessary to feed back the 2-bit ECN mark in received RTP packets, indicating for each RTP packet whether it is marked not-ECT, ECT(0), ECT(1), or ECN-CE. If the path used by the RTP traffic is ECN capable the sender can use Congestion Experienced (ECN-CE) marking information as a congestion control signal.

Every RTP flow is identified by its Synchronization Source (SSRC) identifier. Accordingly, the RTCP feedback format needs to group its reports by SSRC, sending one report block per received SSRC.

As a practical matter, we note that host operating system (OS) process interruptions can occur at inopportune times. Accordingly, recording RTP packet send times at the sender, and the corresponding RTP packet arrival times at the receiver, needs to be done 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 needs to be recorded at the last opportunity prior to transmitting the RTP packet at the sender, and the arrival time at the receiver needs to be recorded at the earliest available opportunity.

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 shown in Figure 1:

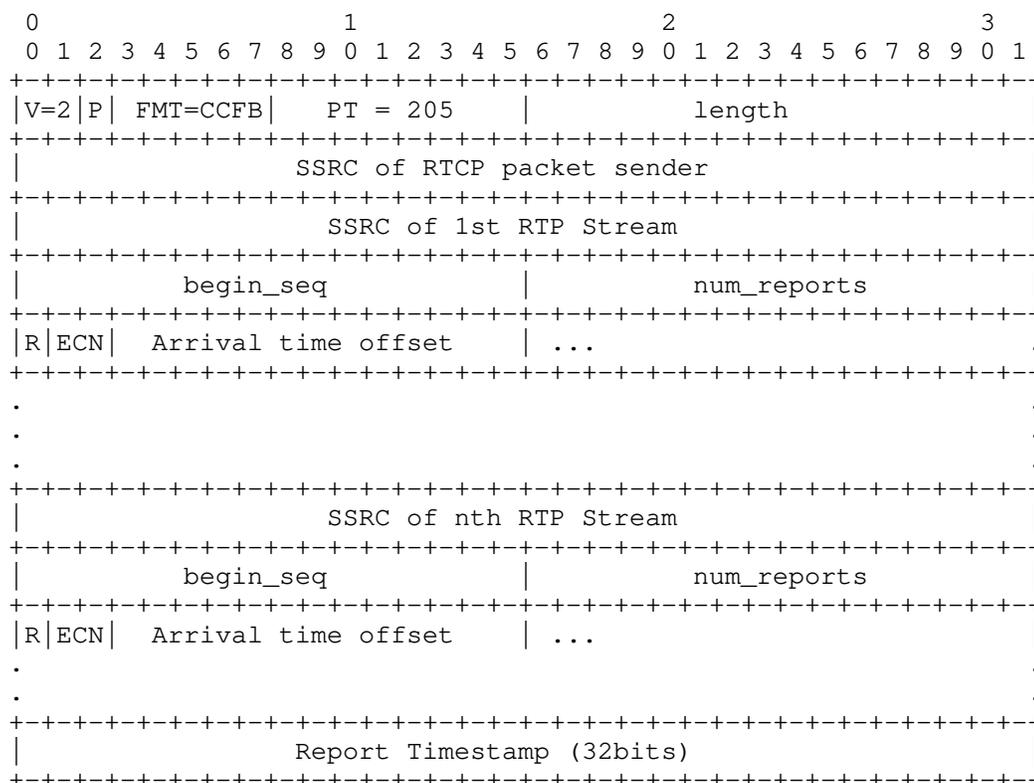


Figure 1: RTCP Congestion Control Feedback Packet Format

The first eight octets comprise a standard RTCP header, with PT=205 and FMT=CCFB indicating that this is a congestion control feedback packet, and with the SSRC set to that of the sender of the RTCP packet. (NOTE TO RFC EDITOR: please replace CCFB here and in the above diagram with the IANA assigned RTCP feedback packet type, and remove this note)

Section 6.1 of [RFC4585] requires the RTCP header to be followed by the SSRC of the RTP flow being reported upon. Accordingly, the RTCP header is followed by a report block for each SSRC from which RTP packets have been received, followed by a Report Timestamp.

Each report block begins with the SSRC of the received RTP Stream on which it is reporting. Following this, the report block contains a 16-bit packet metric block for each RTP packet with sequence number in the range begin_seq to begin_seq+num_reports inclusive (calculated using arithmetic modulo 65536 to account for possible sequence number wrap-around). If the number of 16-bit packet metric blocks included

in the report block is not a multiple of two, then 16 bits of zero padding MUST be added after the last packet metric block, to align the end of the packet metric blocks with the next 32 bit boundary. The value of num_reports MAY be zero, indicating that there are no packet metric blocks included for that SSRC. Each report block MUST NOT include more than 16384 packet metric blocks (i.e., it MUST NOT report on more than one quarter of the sequence number space in a single report).

The contents of each 16-bit packet metric block comprises the R, ECN, and ATO fields as follows:

- o Received (R, 1 bit): is a boolean to indicate if the packet was received. 0 represents that the packet was not yet received and the subsequent 15-bits (ECN and ATO) in this 16-bit packet metric block are also set to 0 and MUST be ignored. 1 represents that the packet was received and the subsequent bits in the block need to be parsed.
- o 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.
- o Arrival time offset (ATO, 13 bits): is the arrival time of the RTP packet at the receiver, as an offset before the time represented by the Report Timestamp (RTS) field of this RTCP congestion control feedback report. The ATO field is in units of 1/1024 seconds (this unit is chosen to give exact offsets from the RTS field) so, for example, an ATO value of 512 indicates that the corresponding RTP packet arrived exactly half a second before the time instant represented by the RTS field. If the measured value is greater than 8189/1024 seconds (the value that would be coded as 0x1FFD), the value 0x1FFE MUST be reported to indicate an over-range measurement. If the measurement is unavailable, or if the arrival time of the RTP packet is after the time represented by the RTS field, then an ATO value of 0x1FFF MUST be reported for the packet.

The RTCP congestion control feedback report packet concludes with the Report Timestamp field (RTS, 32 bits). This denotes the time instant on which this packet is reporting, and is the instant from which the arrival time offset values are calculated. The value of RTS field is derived from the same clock used to generate the NTP timestamp field in RTCP Sender Report (SR) packets. It is formatted as the middle 32 bits of an NTP format timestamp, as described in Section 4 of [RFC3550].

RTCP congestion control feedback packets SHOULD include a report block for every active SSRC. The sequence number ranges reported on

in consecutive reports for a given SSRC will generally be contiguous, but overlapping reports MAY be sent (and need to be sent in cases where RTP packet reordering occurs across the boundary between consecutive reports). If an RTP packet was reported as received in one report, that packet MUST also be reported as received in any overlapping reports sent later that cover its sequence number range. If reports covering overlapping sequence number ranges are sent, information in later reports updates that sent in previous reports for RTP packets included in both reports.

RTCP congestion control feedback packets can be large if they are sent infrequently relative to the number of RTP data packets. If an RTCP congestion control feedback packet is too large to fit within the path MTU, its sender SHOULD split it into multiple feedback packets. The RTCP reporting interval SHOULD be chosen such that feedback packets are sent often enough that they are small enough to fit within the path MTU ([I-D.ietf-rmcat-rtp-cc-feedback] discusses how to choose the reporting interval; specifications for RTP congestion control algorithms can also provide guidance).

If duplicate copies of a particular RTP packet are received, then the arrival time of the first copy to arrive MUST be reported. If any of the copies of the duplicated packet are ECN-CE marked, then an ECN-CE mark MUST be reported that for packet; otherwise the ECN mark of the first copy to arrive is reported.

If no packets are received from an SSRC in a reporting interval, a report block MAY be sent with `begin_seq` set to the highest sequence number previously received from that SSRC and `num_reports` set to zero (or, the report can simply be omitted). The corresponding SR/RR packet will have a non-increased extended highest sequence number received field that will inform the sender that no packets have been received, but it can ease processing to have that information available in the congestion control feedback reports too.

A report block indicating that certain RTP packets were lost is not to be interpreted as a request to retransmit the lost packets. The receiver of such a report might choose to retransmit such packets, provided a retransmission payload format has been negotiated, but there is no requirement that it do so.

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, suggests desirable RTCP feedback rates, and provides guidance on how to configure the RTCP bandwidth fraction, etc., to make appropriate use of the reporting block described in this memo.

Specifications for RTP congestion control algorithms can also provide guidance.

It is generally understood that congestion control algorithms work better with more frequent feedback. However, RTCP bandwidth and transmission rules put some upper limits on how frequently the RTCP feedback messages can be sent from an RTP receiver to the RTP sender. In many cases, sending feedback once per frame is an upper bound before the reporting overhead becomes excessive, although this will depend on the media rate and more frequent feedback might be needed with high-rate media flows [I-D.ietf-rmcat-rtcp-cc-feedback]. Analysis [feedback-requirements] has also shown that some candidate congestion control algorithms can operate with less frequent feedback, using a feedback interval range of 50-200ms. Applications need to negotiate an appropriate congestion control feedback interval at session setup time, based on the choice of congestion control algorithm, the expected media bit rate, and the acceptable feedback overhead.

5. Response to Loss of Feedback Packets

Like all RTCP packets, RTCP congestion control feedback packets might be lost. All RTP congestion control algorithms MUST specify how they respond to the loss of feedback packets.

RTCP packets do not contain a sequence number, so loss of feedback packets has to be inferred based on the time since the last feedback packet. If only a single congestion control feedback packet is lost, an appropriate response is to assume that the level of congestion has remained roughly the same as the previous report. However, if multiple consecutive congestion control feedback packets are lost, then the media sender SHOULD rapidly reduce its sending rate as this likely indicates a path failure. The RTP circuit breaker [RFC8083] provides further guidance.

6. SDP Signalling

A new "ack" feedback parameter, "ccfb", is defined for use with the "a=rtcp-fb:" SDP extension to indicate the use of the RTP Congestion Control feedback packet format defined in Section 3. The ABNF definition of this SDP parameter extension is:

```
rtcp-fb-ack-param = <See Section 4.2 of [RFC4585]>
rtcp-fb-ack-param =/ ccfb-par
ccfb-par          = SP "ccfb"
```

The payload type used with "ccfb" feedback MUST be the wildcard type ("*"). This implies that the congestion control feedback is sent for

all payload types in use in the session, including any FEC and retransmission payload types. An example of the resulting SDP attribute is:

```
a=rtcp-fb:* ack ccfb
```

The offer/answer rules for these SDP feedback parameters are specified in Section 4.2 of the RTP/AVPF profile [RFC4585].

An SDP offer might indicate support for both the congestion control feedback mechanism specified in this memo and one or more alternative congestion control feedback mechanisms that offer substantially the same semantics. In this case, the answering party SHOULD include only one of the offered congestion control feedback mechanisms in its answer. If a re-invite offering the same set of congestion control feedback mechanisms is received, the generated answer SHOULD choose the same congestion control feedback mechanism as in the original answer where possible.

When the SDP BUNDLE extension [I-D.ietf-mmusic-sdp-bundle-negotiation] is used for multiplexing, the "a=rtcp-fb:" attribute has multiplexing category IDENTICAL-PER-PT [I-D.ietf-mmusic-sdp-mux-attributes].

7. Relation to RFC 6679

Use of Explicit Congestion Notification (ECN) with RTP is described in [RFC6679]. That specifies how to negotiate the use of ECN with RTP, and defines an RTCP ECN Feedback Packet to carry ECN feedback reports. It uses an SDP "a=ecn-capable-rtp:" attribute to negotiate use of ECN, and the "a=rtcp-fb:" attributes with the "nack" parameter "ecn" to negotiate the use of RTCP ECN Feedback Packets.

The RTCP ECN Feedback Packet is not useful when ECN is used with the RTP Congestion Control Feedback Packet defined in this memo since it provides duplicate information. When congestion control feedback is to be used with RTP and ECN, the SDP offer generated MUST include an "a=ecn-capable-rtp:" attribute to negotiate ECN support, along with an "a=rtcp-fb:" attribute with the "ack" parameter "ccfb" to indicate that the RTP Congestion Control Feedback Packet can be used. The "a=rtcp-fb:" attribute MAY also include the "nack" parameter "ecn", to indicate that the RTCP ECN Feedback Packet is also supported. If an SDP offer signals support for both RTP Congestion Control Feedback Packets and the RTCP ECN Feedback Packet, the answering party SHOULD signal support for one, but not both, formats in its SDP answer to avoid sending duplicate feedback.

When using ECN with RTP, the guidelines in Section 7.2 of [RFC6679] MUST be followed to initiate the use of ECN in an RTP session. The guidelines in Section 7.3 of [RFC6679] MUST also be followed about ongoing use of ECN within an RTP session, with the exception that feedback is sent using the RTCP Congestion Control Feedback Packets described in this memo rather than using RTP ECN Feedback Packets. Similarly, the guidance in Section 7.4 of [RFC6679] around detecting failures MUST be followed, with the exception that the necessary information is retrieved from the RTCP Congestion Control Feedback Packets rather than from RTP ECN Feedback Packets.

8. Design Rationale

The primary function of RTCP SR/RR packets is to report statistics on the reception of RTP packets. The reception report blocks sent in these packets contain information about observed jitter, fractional packet loss, and cumulative packet loss. It was intended that this information could be used to support congestion control algorithms, but experience has shown that it is not sufficient for that purpose. An efficient congestion control algorithm requires more fine-grained information on per-packet reception quality than is provided by SR/RR packets to react effectively. The feedback format defined in this memo provides such fine-grained feedback.

Several other RTCP extensions also provide more detailed feedback than SR/RR packets:

TMMBR: The Codec Control Messages for the RTP/AVPF profile [RFC5104] include a Temporary Maximum Media Bit Rate (TMMBR) message. This is used to convey a temporary maximum bit rate limitation from a receiver of RTP packets to their sender. Even though it was 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 when used with non-compound RTCP packets [RFC5506]. This approach requires the receiver of the RTP packets to monitor their reception, determine the level of congestion, and recommend a maximum bit rate suitable for current available bandwidth on the path; it also assumes that the RTP sender can/will respect that bit rate. This is the opposite of the sender-based congestion control approach suggested in this memo, so TMMBR cannot be used to convey the information needed for a sender-based congestion control. TMMBR could, however, be viewed a complementary mechanism that can inform the sender of the receiver's current view of acceptable maximum bit rate. Mechanisms that convey the receiver's estimate of the maximum available bit-rate provide similar feedback.

RTCP Extended Reports (XR): Numerous RTCP extended report (XR) blocks have been defined to report details of packet loss, arrival times [RFC3611], delay [RFC6843], and ECN marking [RFC6679]. It is possible to combine several such XR blocks into a compound RTCP packet, to report the detailed loss, arrival time, and ECN marking information needed for effective sender-based congestion control. However, the result has high overhead both in terms of bandwidth and complexity, due to the need to stack multiple reports.

Transport-wide Congestion Control: The format defined in this memo provides individual feedback on each SSRC. An alternative is to add a header extension to each RTP packet, containing a single, transport-wide, packet sequence number, then have the receiver send RTCP reports giving feedback on these additional sequence numbers [I-D.holmer-rmcat-transport-wide-cc-extensions]. Such an approach adds the per-packet overhead of the header extension (8 octets per packet in the referenced format), but reduces the size of the feedback packets, and can simplify the rate calculation at the sender if it maintains a single rate limit that applies to all RTP packets sent irrespective of their SSRC. Equally, the use of transport-wide feedback makes it more difficult to adapt the sending rate, or respond to lost packets, based on the reception and/or loss patterns observed on a per-SSRC basis (for example, to perform differential rate control and repair for audio and video flows, based on knowledge of what packets from each flow were lost). Transport-wide feedback is also a less natural fit with the wider RTP framework, which makes extensive use of per-SSRC sequence numbers and feedback.

Considering these issues, we believe it appropriate to design a new RTCP feedback mechanism to convey information for sender-based congestion control algorithms. The new congestion control feedback RTCP packet described in Section 3 provides such a mechanism.

9. Acknowledgements

This document is based on the outcome of a design team discussion in the RTP Media Congestion Avoidance Techniques (RMCA) working group. The authors would like to thank David Hayes, Stefan Holmer, Randell Jesup, Ingemar Johansson, Jonathan Lennox, Sergio Mena, Nils Ohlmeier, Magnus Westerlund, and Xiaoqing Zhu for their valuable feedback.

10. IANA Considerations

The IANA is requested to register one new RTP/AVPF Transport-Layer Feedback Message in the table for FMT values for RTPFB Payload Types [RFC4585] as defined in Section 3.1:

Name: CCFB
Long name: RTP Congestion Control Feedback
Value: (to be assigned by IANA)
Reference: (RFC number of this document, when published)

The IANA is also requested to register one new SDP "rtcp-fb" attribute "ack" parameter, "ccfb", in the SDP ("ack" and "nack" Attribute Values) registry:

Value name: ccfb
Long name: Congestion Control Feedback
Usable with: ack
Mux: IDENTICAL-PER-PT
Reference: (RFC number of this document, when published)

11. Security Considerations

The security considerations of the RTP specification [RFC3550], the applicable RTP profile (e.g., [RFC3551], [RFC3711], or [RFC4585]), and the RTP congestion control algorithm that is in use (e.g., [RFC8698], [RFC8298], [I-D.ietf-rmcat-gcc], or [RFC8382]) apply.

A receiver that intentionally generates inaccurate RTCP congestion control feedback reports might be able to trick the sender into sending at a greater rate than the path can support, thereby causing congestion on the path. This will negatively impact the quality of experience of that receiver, and potentially cause denial of service to other traffic sharing the path and excessive resource usage at the media sender. Since RTP is an unreliable transport, a sender can intentionally drop a packet, leaving a gap in the RTP sequence number space without causing serious harm, to check that the receiver is correctly reporting losses (this needs to be done with care and some awareness of the media data being sent, to limit impact on the user experience).

An on-path attacker that can modify RTCP congestion control feedback packets can change the reports to trick the sender into sending at either an excessively high or excessively low rate, leading to denial of service. The secure RTCP profile [RFC3711] can be used to authenticate RTCP packets to protect against this attack.

An off-path attacker that can spoof RTCP congestion control feedback packets can similarly trick a sender into sending at an incorrect rate, leading to denial of service. This attack is difficult, since the attacker needs to guess the SSRC and sequence number in addition to the destination transport address. As with on-path attacks, the secure RTCP profile [RFC3711] can be used to authenticate RTCP packets to protect against this attack.

12. References

12.1. Normative 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-54 (work in progress), December 2018.
- [I-D.ietf-mmusic-sdp-mux-attributes]
Nandakumar, S., "A Framework for SDP Attributes when
Multiplexing", draft-ietf-mmusic-sdp-mux-attributes-19
(work in progress), August 2020.
- [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>>.
- [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>>.
- [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>>.
- [RFC8083] Perkins, C. and V. Singh, "Multimedia Congestion Control: Circuit Breakers for Unicast RTP Sessions", RFC 8083, DOI 10.17487/RFC8083, March 2017, <<https://www.rfc-editor.org/info/rfc8083>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.

12.2. Informative References

- [feedback-requirements]
"RMCAT Feedback Requirements",
<[://www.ietf.org/proceedings/95/slides/slides-95-rmcat-1.pdf](http://www.ietf.org/proceedings/95/slides/slides-95-rmcat-1.pdf)>.
- [I-D.alvestrand-rmcat-remb]
Alvestrand, H., "RTCP message for Receiver Estimated Maximum Bitrate", draft-alvestrand-rmcat-remb-03 (work in progress), October 2013.
- [I-D.holmer-rmcat-transport-wide-cc-extensions]
Holmer, S., Flodman, M., and E. Sprang, "RTP Extensions for Transport-wide Congestion Control", draft-holmer-rmcat-transport-wide-cc-extensions-01 (work in progress), October 2015.
- [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-rtp-cc-feedback]
Perkins, C., "RTP Control Protocol (RTCP) Feedback for Congestion Control in Interactive Multimedia Conferences", draft-ietf-rmcat-rtp-cc-feedback-05 (work in progress), November 2019.
- [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>>.
- [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>>.
- [RFC6843] Clark, A., Gross, K., and Q. Wu, "RTP Control Protocol (RTCP) Extended Report (XR) Block for Delay Metric Reporting", RFC 6843, DOI 10.17487/RFC6843, January 2013, <<https://www.rfc-editor.org/info/rfc6843>>.
- [RFC8298] Johansson, I. and Z. Sarker, "Self-Clocked Rate Adaptation for Multimedia", RFC 8298, DOI 10.17487/RFC8298, December 2017, <<https://www.rfc-editor.org/info/rfc8298>>.
- [RFC8382] Hayes, D., Ed., Ferlin, S., Welzl, M., and K. Hiorth, "Shared Bottleneck Detection for Coupled Congestion Control for RTP Media", RFC 8382, DOI 10.17487/RFC8382, June 2018, <<https://www.rfc-editor.org/info/rfc8382>>.
- [RFC8698] Zhu, X., Pan, R., Ramalho, M., and S. Mena, "Network-Assisted Dynamic Adaptation (NADA): A Unified Congestion Control Scheme for Real-Time Media", RFC 8698, DOI 10.17487/RFC8698, February 2020, <<https://www.rfc-editor.org/info/rfc8698>>.

Authors' Addresses

Zaheduzzaman Sarker
Ericsson AB
Torshamnsgatan 21
Stockholm 164 40
Sweden

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

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

Email: csp@csp Perkins.org

Varun Singh
CALLSTATS I/O Oy
Annankatu 31-33 C 42
Helsinki 00100
Finland

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

Michael A. Ramalho
6310 Watercrest Way Unit 203
Lakewood Ranch, FL 34202-5122
USA

Phone: +1 732 832 9723
Email: mar42@cornell.edu
URI: <http://ramalho.webhop.info/>

AVTCore
Internet-Draft
Updates: RFC 4103 (if approved)
Intended status: Standards Track
Expires: 26 May 2021

G. Hellstrom
Gunnar Hellstrom Accessible Communication
22 November 2020

RTP-mixer formatting of multi-party Real-time text
draft-ietf-avtc core-multi-party-rtt-mix-11

Abstract

Real-time text mixers for multi-party sessions need to identify the source of each transmitted group of text so that the text can be presented by endpoints in suitable grouping with other text from the same source, while new text from other sources is also presented in readable grouping as received interleaved in real-time.

Regional regulatory requirements specify provision of real-time text in multi-party calls. RFC 4103 mixer implementations can use traditional RTP functions for source identification, but the mixer source switching performance is limited when using the default transmission characteristics with redundancy.

Enhancements for RFC 4103 real-time text mixing is provided in this document, suitable for a centralized conference model that enables source identification and source switching. The intended use is for real-time text mixers and multi-party-aware participant endpoints. The specified mechanism build on the standard use of the CSRC list in the RTP packet for source identification. The method makes use of the same "text/t140" and "text/red" formats as for two-party sessions.

A capability exchange is specified so that it can be verified that a participant can handle the multi-party coded real-time text stream. The capability is indicated by use of a media attribute "rtt-mixer".

The document updates RFC 4103[RFC4103]

A specifications of how a mixer can format text for the case when the endpoint is not multi-party aware is also provided.

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 26 May 2021.

Copyright Notice

Copyright (c) 2020 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 4
 - 1.1. Selected solution and considered alternative 5
 - 1.2. Nomenclature 7
 - 1.3. Intended application 8
- 2. Overview over the two specified solutions 9
 - 2.1. Negotiated use of the RFC 4103 format for multi-party transmission in a single RTP stream 9
 - 2.2. Mixing for multi-party unaware endpoints 9
- 3. Details for the multi-party aware mixing case 9
 - 3.1. Offer/answer considerations 10
 - 3.2. Actions depending on capability negotiation result . . . 10
 - 3.3. Use of fields in the RTP packets 10
 - 3.4. Initial transmission of a BOM character 11
 - 3.5. Keep-alive 11
 - 3.6. Transmission interval 11
 - 3.7. Only one source per packet 11
 - 3.8. Do not send received text to the originating source . . . 12
 - 3.9. Clean incoming text 12
 - 3.10. Redundant transmission principles 12
 - 3.11. Interleaving text from different sources 12

3.12.	Text placement in packets	12
3.13.	Empty T140blocks	13
3.14.	Creation of the redundancy	13
3.15.	Timer offset fields	14
3.16.	Other RTP header fields	14
3.17.	Pause in transmission	14
3.18.	RTCP considerations	15
3.19.	Reception of multi-party contents	15
3.20.	Performance considerations	17
3.21.	Security for session control and media	18
3.22.	SDP offer/answer examples	18
3.23.	Packet sequence example from interleaved transmission	19
3.24.	Maximum character rate "CPS"	22
4.	Presentation level considerations	22
4.1.	Presentation by multi-party aware endpoints	23
4.2.	Multi-party mixing for multi-party unaware endpoints	25
5.	Relation to Conference Control	30
5.1.	Use with SIP centralized conferencing framework	31
5.2.	Conference control	31
6.	Gateway Considerations	31
6.1.	Gateway considerations with Textphones (e.g. TTYs).	31
6.2.	Gateway considerations with WebRTC.	32
7.	Updates to RFC 4103	32
8.	Congestion considerations	32
9.	Acknowledgements	33
10.	IANA Considerations	33
10.1.	Registration of the "rtt-mixer" sdp media attribute	33
11.	Security Considerations	34
12.	Change history	34
12.1.	Changes included in draft-ietf-avtcore-multi-party-rtt-mix-11	34
12.2.	Changes included in draft-ietf-avtcore-multi-party-rtt-mix-10	34
12.3.	Changes included in draft-ietf-avtcore-multi-party-rtt-mix-09	34
12.4.	Changes included in draft-ietf-avtcore-multi-party-rtt-mix-08	35
12.5.	Changes included in draft-ietf-avtcore-multi-party-rtt-mix-07	35
12.6.	Changes included in draft-ietf-avtcore-multi-party-rtt-mix-06	35
12.7.	Changes included in draft-ietf-avtcore-multi-party-rtt-mix-05	35
12.8.	Changes included in draft-ietf-avtcore-multi-party-rtt-mix-04	36
12.9.	Changes included in draft-ietf-avtcore-multi-party-rtt-mix-03	36

- 12.10. Changes included in
 draft-ietf-avtcore-multi-party-rtt-mix-02 37
- 12.11. Changes to draft-ietf-avtcore-multi-party-rtt-mix-01 . . 37
- 12.12. Changes from
 draft-hellstrom-avtcore-multi-party-rtt-source-03 to
 draft-ietf-avtcore-multi-party-rtt-mix-00 37
- 12.13. Changes from
 draft-hellstrom-avtcore-multi-party-rtt-source-02 to
 -03 38
- 12.14. Changes from
 draft-hellstrom-avtcore-multi-party-rtt-source-01 to
 -02 38
- 12.15. Changes from
 draft-hellstrom-avtcore-multi-party-rtt-source-00 to
 -01 39
- 13. References 39
 - 13.1. Normative References 39
 - 13.2. Informative References 40
- Author's Address 41

1. Introduction

RFC 4103[RFC4103] specifies use of RFC 3550 RTP [RFC3550] for transmission of real-time text (RTT) and the "text/t140" format. It also specifies a redundancy format "text/red" for increased robustness. RFC 4102 [RFC4102] registers the "text/red" format. Regional regulatory requirements specify provision of real-time text in multi-party calls.

Real-time text is usually provided together with audio and sometimes with video in conversational sessions.

A requirement related to multi-party sessions from the presentation level standard T.140 for real-time text is: "The display of text from the members of the conversation should be arranged so that the text from each participant is clearly readable, and its source and the relative timing of entered text is visualized in the display."

Another requirement is that the mixing procedure must not introduce delays in the text streams that are experienced disturbing the real-time experience of the receiving users.

The redundancy scheme of RFC 4103 [RFC4103] enables efficient transmission of redundant text in packets together with new text. However the redundancy header format has no source indicators for the redundant transmissions. The redundant parts in a packet must therefore be from the same source as the new text. The recommended transmission is one new and two redundant generations of text (T140blocks) in each packet and the recommended transmission interval for two-party use is 300 ms.

Real-time text mixers for multi-party sessions therefore need to insert the source of each transmitted group of text from a conference participant so that the text can be transmitted interleaved with text groups from different sources in the rate they are created. This enables the text groups to be presented by endpoints in suitable grouping with other text from the same source. The presentation can then be arranged so that text from different sources can be presented in real-time and easily read while it is possible for a reading user to also perceive approximately when the text was created in real time by the different parties. The transmission and mixing is intended to be done in a general way so that presentation can be arranged in a layout decided by the endpoint.

There are existing implementations of RFC 4103 without the updates from this document. These will not be able to receive and present real-time text mixed for multi-party aware endpoints.

A negotiation mechanism is therefore needed for verification if the parties are able to handle a multi-party coded stream and agreeing on using that method.

A fall-back mixing procedure is also needed for cases when the negotiation result indicates that a receiving endpoint is not capable of handling the mixed format. This method is called the mixing procedure for multi-party unaware endpoints. The fall-back method is naturally not expected to meet all performance requirements placed on the mixing procedure for multi-party aware endpoints.

The document updates RFC 4103[RFC4103] by introducing an attribute for indicating capability for the multi-party mixing case and rules for source indications and source switching.

1.1. Selected solution and considered alternative

A number of alternatives were considered when searching an efficient and easily implemented multi-party method for real-time text. This section explains a few of them briefly.

One RTP stream per source, sent in the same RTP session with "text/red" format.

From some points of view, use of multiple RTP streams, one for each source, sent in the same RTP session, called the RTP translator model in [RFC3550], would be efficient, and use exactly the same packet format as [RFC4103], the same payload type and a simple SDP declaration. However, the RTP implementation in both mixers and endpoints need to support multiple streams in the same RTP session in order to use this mechanism. For best deployment opportunity, it should be possible to upgrade existing endpoint solutions to be multi-party aware with a reasonable effort. There is currently a lack of support for multi-stream RTP in certain implementation technologies. This fact made this solution not selected for inclusion in this document.

The "text/red" format in RFC 4103 with shorter transmission interval, and indicating source in CSRC.

The "text/red" format with "text/t140" payload in a single RTP stream can be sent with 100 ms packet intervals instead of the regular 300 ms. The source is indicated in the CSRC field. Source switching can then be done every 100 ms while simultaneous transmission occurs. With five participants sending text simultaneously, the switching and transmission performance is good. With more simultaneously sending participants, there will be a noticeable jerkiness in text presentation. The jerkiness will be more expressed the more participants who send text simultaneously. With ten sending participants, the jerkiness will be about one second. Text sent from a source at the end of the period its text is sent by the mixer will have close to zero extra delay. Recent text will be presented with no or low delay. The one second jerkiness will be noticeable and slightly unpleasant, but corresponds in time to what typing humans often cause by hesitation or changing position while typing. A benefit of this method is that no new packet format needs to be introduced and implemented. Since simultaneous typing by more than two parties is very rare, and in most applications also more than three parties in a call is rare, this method can be used successfully with good performance. Recovery of text in case of packet loss is based on analysis of timestamps of received redundancy versus earlier received text. Negotiation is based on a new sdp media attribute "rtt-mixer". This method is selected to be the main one specified in this document.

A new "text" media subtype with up to 15 sources in each packet.

The mechanism makes use of the RTP mixer model specified in RFC3550[RFC3550]. Text from up to 15 sources can be included in each packet. Packets are normally sent every 300 ms. The mean delay will be 150 ms. The sources are indicated in strict order

in the CSRC list of the RTP packets. A new redundancy packet format is specified. This method would result in good performance, but would require standardisation and implementation of new releases in the target technologies that would take more time than desirable to complete. It was therefore not selected to be included in this document.

The presentation planned by the mixer for multi-party unaware endpoints.

It is desirable to have a method that does not require any modifications in existing user devices implementing RFC 4103 for RTT without explicit support of multi-party sessions. This is possible by having the mixer insert a new line and a text formatted source label before each switch of text source in the stream. Switch of source can only be done in places in the text where it does not disturb the perception of the contents. Text from only one source can be presented in real time at a time. The delay will therefore be varying. The method has also other limitations, but is included in this document as a fallback method. In calls where parties take turns properly by ending their entries with a new line, the limitations will have limited influence on the user experience. While only two parties send text, these two will see the text in real time with no delay. This method is specified as a fallback method in this document.

RTT transport in WebRTC

Transport of real-time text in the WebRTC technology is specified to use the WebRTC data channel in [I-D.ietf-mmusic-t140-usage-data-channel]. That specification contains a section briefly describing its use in multi-party sessions. The focus of this document is RTP transport. Therefore, even if the WebRTC transport provides good multi-party performance, it is just mentioned in this document in relation to providing gateways with multi-party capabilities between RTP and WebRTC technologies.

1.2. Nomenclature

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].

The terms SDES, CNAME, NAME, SSRC, CSRC, CSRC list, CC, RTCP, RTP-mixer, RTP-translator are explained in [RFC3550]

The term "T140block" is defined in RFC 4103 [RFC4103] to contain one or more T.140 code elements.

"TTY" stands for a text telephone type used in North America.

"WebRTC" stands for web based communication specified by W3C and IETF.

"DTLS-SRTP" stands for security specified in RFC 5764 [RFC5764].

"multi-party aware" stands for an endpoint receiving real-time text from multiple sources through a common conference mixer being able to present the text in real-time separated by source and presented so that a user can get an impression of the approximate relative timing of text from different parties.

"multi-party unaware" stands for an endpoint not itself being able to separate text from different sources when received through a common conference mixer.

1.3. Intended application

The method for multi-party real-time text specified in this document is primarily intended for use in transmission between mixers and endpoints in centralised mixing configurations. It is also applicable between mixers. An often mentioned application is for emergency service calls with real-time text and voice, where a calltaker want to make an attended handover of a call to another agent, and stay observing the session. Multimedia conference sessions with support for participants to contribute in text is another application. Conferences with central support for speech-to-text conversion is yet another mentioned application.

In all these applications, normally only one participant at a time will send long text utterances. In some cases, one other participant will occasionally contribute with a longer comment simultaneously. That may also happen in some rare cases when text is interpreted to text in another language in a conference. Apart from these cases, other participants are only expected to contribute with very brief utterings while others are sending text.

Text is supposed to be human generated, by some text input means, such as typing on a keyboard or using speech-to-text technology. Occasional small cut-and-paste operations may appear even if that is not the initial purpose of real-time text.

The real-time characteristics of real-time text is essential for the participants to be able to contribute to a conversation. If the text is too much delayed from typing a letter to its presentation, then, in some conference situations, the opportunity to comment will be gone and someone else will grab the turn. A delay of more than one second in such situations is an obstacle for good conversation.

2. Overview over the two specified solutions

This section contains a brief introduction of the two methods specified in this document.

2.1. Negotiated use of the RFC 4103 format for multi-party transmission in a single RTP stream

The main purpose of this document is to specify a method for true multi-party real-time text mixing for multi-party aware endpoints. The method makes use of the current format for real-time text in [RFC4103]. It is an update of RFC 4103 by a clarification on one way to use it in the multi-party situation. It is done by completing a negotiation for this kind of multi-party capability and by indicating source in the CSRC element in the RTP packets. Specific considerations are made to be able to recover text after packet loss.

The detailed procedures for the multi-party aware case are specified in Section 3

Please use [RFC4103] as reference when reading the specification.

2.2. Mixing for multi-party unaware endpoints

A method is also specified in this document for cases when the endpoint participating in a multi-party call does not itself implement any solution for multi-party handling of real-time text. The solution requires the mixer to insert text dividers and readable labels and only send text from one source at a time until a suitable point appears for source change. This solution is a fallback method with functional limitations. It acts on the presentation level.

A party performing as a mixer, which has not negotiated the "rtt-mixer" sdp media attribute, but negotiated a "text/red" or "text/t140" format in a session with a participant SHOULD, if nothing else is specified for the application, format transmitted text to that participant to be suitable to present on a multi-party unaware endpoint as further specified in Section 4.2.

3. Details for the multi-party aware mixing case

3.1. Offer/answer considerations

RFC 4103[RFC4103] specifies use of RFC 3550 RTP[RFC3550], and a redundancy format "text/red" for increased robustness of real-time text transmission. This document updates RFC 4103[RFC4103] by introducing a capability negotiation for handling multi-party real-time text, a way to indicate the source of transmitted text, and rules for efficient timing of the transmissions interleaved from different sources.

The capability negotiation is based on use of the sdp media attribute "rtt-mixer".

Both parties shall indicate their capability in a session setup or modification, and evaluate the capability of the counterpart.

The syntax is as follows:

```
"a=rtt-mixer"
```

3.2. Actions depending on capability negotiation result

A transmitting party SHALL send text according to the multi-party format only when the negotiation for this method was successful and when the CC field in the RTP packet is set to 1. In all other cases, the packets SHALL be populated and interpreted as for a two-party session.

A party which has negotiated the "rtt-mixer" sdp media attribute MUST populate the CSRC-list and format the packets according to Section 3 if it acts as an rtp-mixer and sends multi-party text.

A party which has negotiated the "rtt-mixer" sdp media attribute MUST interpret the contents of the "CC" field the CSRC-list and the packets according to Section 3 in received rtp packets in the corresponding RTP stream.

A party not performing as a mixer MUST not include the CSRC list.

3.3. Use of fields in the RTP packets

The CC field SHALL show the number of members in the CSRC list, which SHALL be one (1) in transmissions from a mixer involved in a multi-party session, and otherwise 0.

When transmitted from a mixer during a multi-party session, a CSRC list SHALL be included in the packet. The single member in the CSRC-list SHALL contain the SSRC of the source of the T140blocks in the packet. When redundancy is used, the recommended level of redundancy

is to use one primary and two redundant generations of T140blocks. In some cases, a primary or redundant T140block is empty, but is still represented by a member in the redundancy header.

From other aspects, the contents of the RTP packets are equal to what is specified in [RFC4103].

3.4. Initial transmission of a BOM character

As soon as a participant is known to participate in a session with another entity and being available for text reception, a Unicode BOM character SHALL be sent to it by the other entity according to the procedures in this section. If the transmitter is a mixer, then the source of this character SHALL be indicated to be the mixer itself.

Note that the BOM character SHALL be transmitted with the same redundancy procedures as any other text.

3.5. Keep-alive

After that, the transmitter SHALL send keep-alive traffic to the receiver(s) at regular intervals when no other traffic has occurred during that interval, if that is decided for the actual connection. Recommendations for keep-alive can be found in [RFC6263].

3.6. Transmission interval

A "text/red" or "text/t140" transmitter in a mixer SHOULD send packets distributed in time as long as there is something (new or redundant T140blocks) to transmit. The maximum transmission interval SHOULD then be 330 ms. It is RECOMMENDED to send next packet to a receiver as soon as new text to that receiver is available, as long as the time after the latest sent packet to the same receiver is more than or equal to 100 ms, and also the maximum character rate ("CPS") to the receiver is not exceeded. The intention is to keep the latency low and network load limited while keeping a good protection against text loss in bursty packet loss conditions.

For a transmitter not acting in a mixer, the same transmission interval principles apply, but the maximum transmission interval SHOULD be 300 ms.

3.7. Only one source per packet

New and redundant text from one source SHALL be transmitted in the same packet if available for transmission at the same time. Text from different sources MUST NOT be transmitted in the same packet.

3.8. Do not send received text to the originating source

Text received to a mixer from a participant SHOULD NOT be included in transmission from the mixer to that participant.

3.9. Clean incoming text

A mixer SHALL handle reception, recovery from packet loss, deletion of superfluous redundancy, marking of possible text loss and deletion of 'BOM' characters from each participant before queueing received text for transmission to receiving participants.

3.10. Redundant transmission principles

A transmitting party using redundancy SHALL send redundant repetitions of T140blocks already transmitted in earlier packets.

The number of redundant generations of T140blocks to include in transmitted packets SHALL be deduced from the SDP negotiation. It SHOULD be set to the minimum of the number declared by the two parties negotiating a connection. It is RECOMMENDED to declare and transmit one original and two redundant generations of the T140blocks.

3.11. Interleaving text from different sources

When text from more than one source is available for transmission from a mixer, the mixer SHALL let the sources take turns in having their text transmitted.

The source with the oldest text received in the mixer or oldest redundant text SHOULD be next in turn to get all its available unsent text transmitted. The age of redundant text SHOULD then be considered to be 330 ms after its previous transmission.

3.12. Text placement in packets

The mixer SHOULD compose and transmit an RTP packet to a receiver when one of the following conditions has occurred:

- * 100 ms has passed since the latest transmission to that receiver, and there is unsent text available for transmission.
- * New text has arrived and more than 100 ms has passed since latest transmission to that receiver.

- * 330 ms has passed since already transmitted text was queued for transmission as redundant text, and more than 100 ms has passed since the latest transmission to that receiver, and the redundant text is still not sent.

At time of transmission, the mixer SHALL populate the RTP packet with all T140blocks queued for transmission originating from the source in turn for transmission as long as this is not in conflict with the allowed number of characters per second ("CPS") or the maximum packet size. In this way, the latency of the latest received text is kept low even in moments of simultaneous transmission from many sources.

Redundant text SHALL also be included. See Section 3.14

The SSRC of the source shall be placed as the only member in the CSRC-list.

Note: The CSRC-list in an RTP packet only includes the participant who's text is included in text blocks. It is not the same as the total list of participants in a conference. With audio and video media, the CSRC-list would often contain all participants who are not muted whereas text participants that don't type are completely silent and thus are not represented in RTP packet CSRC-lists.

3.13. Empty T140blocks

If no unsent T140blocks were available for a source at the time of populating a packet, but T140blocks are available which have not yet been sent the full intended number of redundant transmissions, then the primary T140block for that source is composed of an empty T140block, and populated (without taking up any length) in a packet for transmission. The corresponding SSRC SHALL be placed as usual in its place in the CSRC-list.

The first packet in the session, the first after a source switch and the first after a pause SHALL be populated with the available T140blocks for the source in turn to be sent as primary, and empty T140blocks for the agreed number of redundancy generations.

3.14. Creation of the redundancy

The primary T140block from a source in the latest transmitted packet is saved for populating the first redundant T140block for that source in next transmission of text from that source. The first redundant T140block for that source from the latest transmission is saved for populating the second redundant T140block in next transmission of text from that source.

Usually this is the level of redundancy used. If a higher number of redundancy is negotiated, then the procedure SHALL be maintained until all available redundant levels of T140blocks are placed in the packet. If a receiver has negotiated a lower number of "text/red" generations, then that level shall be the maximum used by the transmitter.

The T140blocks saved for transmission as redundant data are assigned a planned transmission time 330 ms after the current time, but SHOULD be transmitted earlier if new text for the same source gets in turn for transmission before that time.

3.15. Timer offset fields

The timestamp offset values are inserted in the redundancy header, with the time offset from the RTP timestamp in the packet when the corresponding T140block was sent as primary.

The timestamp offsets are expressed in the same clock tick units as the RTP timestamp.

The timestamp offset values for empty T140blocks have no relevance but SHOULD be assigned realistic values.

3.16. Other RTP header fields

The number of members in the CSRC list (0 or 1) shall be placed in the "CC" header field. Only mixers place value 1 in the "CC" field. A value of "0" indicates that the source is the transmitting device itself and that the source is indicated by the SSRC field. This value is used by endpoints, and by mixers sending data that it is source of itself.

The current time SHALL be inserted in the timestamp.

The SSRC of the mixer for the RTT session SHALL be inserted in the SSRC field of the RTP header.

The M-bit shall be handled as specified in [RFC4103].

3.17. Pause in transmission

When there is no new T140block to transmit, and no redundant T140block that has not been retransmitted the intended number of times from any source, the transmission process can stop until either new T140blocks arrive, or a keep-alive method calls for transmission of keep-alive packets.

3.18. RTCP considerations

A mixer SHALL send RTCP reports with SDES, CNAME and NAME information about the sources in the multi-party call. This makes it possible for participants to compose a suitable label for text from each source.

Integrity considerations SHALL be considered when composing these fields.

3.19. Reception of multi-party contents

The "text/red" receiver included in an endpoint with presentation functions will receive RTP packets in the single stream from the mixer, and SHALL distribute the T140blocks for presentation in presentation areas for each source. Other receiver roles, such as gateways or chained mixers are also feasible, and requires consideration if the stream shall just be forwarded, or distributed based on the different sources.

3.19.1. Multi-party vs two-party use

If the "CC" field value of a received packet is 1, it indicates that multi-party transmission is active, and the receiver MUST be prepared to act on the source according to its role. If the CC value is 0, the connection is point-to-point.

3.19.2. Level of redundancy

The used level of redundancy generations SHALL be evaluated from the received packet contents. The number of generations (including the primary) is equal to the number of members in the redundancy header.

3.19.3. Empty T140blocks

Empty T140blocks are included as fillers for unused primary or redundancy levels in the packets. They just do not provide any contents.

3.19.4. Detection and indication of possible text loss

The RTP sequence numbers of the received packets SHALL be monitored for gaps and packets out of order. If a sequence number gap appears and still exists after some defined short time for jitter resolution, the packets in the gap SHALL be regarded lost.

If it is known that only one source is active in the RTP session, then it is likely that a gap equal to or larger than the agreed number of redundancy generations (including the primary) causes text loss. In that case a t140block SHALL be created with a marker for possible text loss [T140ad1] and assigned to the source and inserted in the reception buffer for that source.

If it is known that more than one source is active in the RTP session, then it is not possible in general to evaluate if text was lost when packets were lost. With two active sources and the recommended number of redundancy generations (3), it can take a gap of five consecutive lost packets until any text may be lost, but text loss can also appear if three non-consecutive packets are lost when they contained consecutive data from the same source. A simple method to decide when there is risk for resulting text loss is to evaluate if three or more packets were lost within one second. Then a t140block SHALL be created with a marker for possible text loss [T140ad1] and assigned to the SSRC of the transmitter as a general input from the mixer.

Implementations MAY apply more refined methods for more reliable detection of if text was lost or not. Any refined method SHOULD rather falsely mark possible loss when there was no loss instead of not marking possible loss when there was loss.

3.19.5. Extracting text and handling recovery

When applying the following procedures, the effects MUST be considered of possible timestamp wrap around and the RTP session possibly changing SSRC.

When a packet is received in an RTP session using the packetization for multi-party aware endpoints, its T140blocks SHALL be extracted in the following way. The description is adapted to the default redundancy case using the original and two redundant generations.

The source SHALL be extracted from the CSRC-list if available, otherwise from the SSRC.

If the received packet is the first packet received from the source, then all T140blocks in the packet SHALL be retrieved and assigned to a receive buffer for the source beginning with the second generation redundancy, continuing with the first generation redundancy and finally the primary.

Note: The normal case is that in the first packet, only the primary data has contents. The redundant data has contents in the first received packet from a source only after initial packet loss.

If the packet is not the first packet from a source, then if the second generation redundant data is available, its timestamp SHALL be created by subtracting its timestamp offset from the RTP timestamp. If the resulting timestamp is later than the latest retrieved data from the same source, then the redundant data SHALL be retrieved and appended to the receive buffer. The process SHALL be continued in the same way for the first generation redundant data. After that, the primary data SHALL be retrieved from the packet and appended to the receive buffer for the source.

3.19.6. Delete 'BOM'

Unicode character 'BOM' is used as a start indication and sometimes used as a filler or keep alive by transmission implementations. These SHALL be deleted after extraction from received packets.

3.20. Performance considerations

This solution has good performance for up to five participants simultaneously sending text. At higher numbers of participants simultaneously sending text, a jerkiness is visible in the presentation of text. With ten participants simultaneously transmitting text, the jerkiness is about one second. Even so, the transmission of text catches up, so there is limited delay of new text. The solution is therefore suitable for emergency service use, relay service use, and small or well-managed larger multimedia conferences. Only in large unmanaged conferences with a high number of participants there may on very rare occasions appear situations when many participants happen to send text simultaneously, resulting in unpleasantly jerky presentation of text from each sending participant. It should be noted that it is only the number of users sending text within the same moment that causes jerkiness, not the total number of users with RTT capability.

3.21. Security for session control and media

Security SHOULD be applied on both session control and media. In applications where legacy endpoints without security may exist, a negotiation SHOULD be performed to decide if security by encryption will be applied. If no other security solution is mandated for the application, then RFC 8643 OSRTP [RFC8643] SHOULD be applied to negotiate SRTP media security with DTLS. Most SDP examples below are for simplicity expressed without the security additions. The principles (but not all details) for applying DTLS-SRTP [RFC5764] security is shown in a couple of the following examples.

3.22. SDP offer/answer examples

This sections shows some examples of SDP for session negotiation of the real-time text media in SIP sessions. Audio is usually provided in the same session, and sometimes also video. The examples only show the part of importance for the real-time text media.

Offer example for "text/red" format and multi-party support:

```
m=text 11000 RTP/AVP 100 98
a=rtpmap:98 t140/1000
a=rtpmap:100 red/1000
a=fmtp:100 98/98/98
a=rtt-mixer
```

Answer example from a multi-party capable device

```
m=text 14000 RTP/AVP 100 98
a=rtpmap:98 t140/1000
a=rtpmap:100 red/1000
a=fmtp:100 98/98/98
a=rtt-mixer
```

Offer example for "text/red" format including multi-party and security:

```
a=fingerprint: (fingerprint1)
m=text 11000 RTP/AVP 100 98
a=rtpmap:98 t140/1000
a=rtpmap:100 red/1000
a=fmtp:100 98/98/98
a=rtt-mixer
```

The "fingerprint" is sufficient to offer DTLS-SRTP, with the media line still indicating RTP/AVP.

Note: For brevity, the entire value of the SDP fingerprint attribute is not shown in this and the following example.

```

Answer example from a multi-party capable device with security
a=fingerprint: (fingerprint2)
m=text 16000 RTP/AVP 100 98
a=rtpmap:98 t140/1000
a=rtpmap:100 red/1000
a=fmtp:100 98/98/98
a=rtt-mixer

```

With the "fingerprint" the device acknowledges use of SRTP/DTLS.

Answer example from a multi-party unaware device that also does not support security:

```

m=text 12000 RTP/AVP 100 98
a=rtpmap:98 t140/1000
a=rtpmap:100 red/1000
a=fmtp:100 98/98/98

```

3.23. Packet sequence example from interleaved transmission

This example shows a symbolic flow of packets from a mixer including loss and recovery. The sequence includes interleaved transmission of text from two RTT sources A and B. P indicates primary data. R1 is first redundant generation data and R2 is the second redundant generation data. A1, B1, A2 etc are text chunks (T140blocks) received from the respective sources and sent on to the receiver by the mixer. X indicates dropped packet between the mixer and a receiver. The session is assumed to use original and two redundant generations of RTT.

```

-----
Seq no 101, Time=20400
CC=1
CSRC list A
R2: A1, Offset=600
R1: A2, Offset=300
P:  A3
-----

```

Assuming that earlier packets (with text A1 and A2) were received in sequence, text A3 is received from packet 101 and assigned to reception area A. The mixer is now assumed to have received text from source B and will send that text 100 ms after packet 101. Transmission of A2 and A3 as redundancy is planned for 330 ms after packet 101 if no new text from A is ready to be sent before that.

```
-----|
Seq no 102, Time=20500
CC=1
CSRC list B
R2 Empty, Offset=600
R1: Empty, Offset=300
P: B1
-----|
```

Packet 102 is received.

B1 is retrieved from this packet. Redundant transmission of B1 is planned 330 ms after packet 102.

```
X-----|
X Seq no 103, Timer=20730
X CC=1
X CSRC list A
X R2: A2, Offset=630
X R1: A3, Offset=330
X P: Empty
X-----|
```

Packet 103 is assumed to be dropped in network problems. It contains redundancy for A. Sending A3 as second level redundancy is planned for 330 ms after packet 104.

```
X-----|
X Seq no 104, Timer=20820
X CC=1
X CSRC list B
X R2: Empty, Offset=600
X R1: B1, Offset=300
X P: B2
X-----|
```

Packet 104 contains text from B, including new B2 and redundant B1. It is assumed dropped in network problems.

The mixer has A3 redundancy to send but no new text appears from A and therefore the redundancy is sent 330 ms after the previous packet with text from A.

```
-----  
Seq no 105, Timer=21060  
CC=1  
CSRC list A  
R2: A3, Offset=660  
R1: Empty, Offset=330  
P: Empty  
-----
```

Packet 105 is received.

A gap for lost 103, and 104 is detected.

Assume that no other loss was detected the last second.

Then it can be concluded that nothing was totally lost.

R2 is checked. Its original time was $21040-660=20400$.

A packet with text from A was received with that timestamp, so nothing needs to be recovered.

B1 and B2 still needs to be transmitted as redundancy.

This is planned 330 ms after packet 105. But that would be at 21150 which is only 90 ms after the latest packet. It is instead transmitted at time 21160.

```
-----  
Seq no 106, Timer=21160  
CC=1  
CSRC list B  
R2: B1, Offset=660  
R1: B2, Offset=340  
P: Empty  
-----
```

Packet 106 is received.

The second level redundancy in packet 106 is B1 and has timestamp offset 660 ms. The timestamp of packet 106 minus 660 is 20500 which is the timestamp of packet 101 THAT was received. So B1 does not need to be retrieved. The first level redundancy in packet 106 has offset 340. The timestamp of packet 106 minus 340 is 20820. That is later than the latest received packet with source B. Therefore B2 is retrieved and assigned to the input buffer for source B. No primary is available in packet 106

After this sequence, A3 and B1 and B2 have been received. In this case no text was lost.

3.24. Maximum character rate "CPS"

The default maximum rate of reception of "text/t140" real-time text is in RFC 4103 [RFC4103] specified to be 30 characters per second. The value MAY be modified in the CPS parameter of the FMTP attribute in the media section for the "text/t140" media. A mixer combining real-time text from a number of sources may occasionally have a higher combined flow of text coming from the sources. Endpoints SHOULD therefore specify a suitable higher value for the CPS parameter, corresponding to its real reception capability. A value for "CPS" of 90 is the default for the "text/t140" stream in the "text/red" format when multi-party real-time text is negotiated. See RFC 4103 [RFC4103] for the format and use of the CPS parameter. The same rules apply for the multi-party case except for the default value.

4. Presentation level considerations

ITU-T T.140 [T140] provides the presentation level requirements for the RFC 4103 [RFC4103] transport. T.140 [T140] has functions for erasure and other formatting functions and has the following general statement for the presentation:

"The display of text from the members of the conversation should be arranged so that the text from each participant is clearly readable, and its source and the relative timing of entered text is visualized in the display. Mechanisms for looking back in the contents from the current session should be provided. The text should be displayed as soon as it is received."

Strict application of T.140 [T140] is of essence for the interoperability of real-time text implementations and to fulfill the intention that the session participants have the same information of the text contents of the conversation without necessarily having the exact same layout of the conversation.

T.140 [T140] specifies a set of presentation control codes to include in the stream. Some of them are optional. Implementations MUST be able to ignore optional control codes that they do not support.

There is no strict "message" concept in real-time text. The Unicode Line Separator character SHALL be used as a separator allowing a part of received text to be grouped in presentation. The characters "CRLF" may be used by other implementations as replacement for Line Separator. The "CRLF" combination SHALL be erased by just one erasing action, just as the Line Separator. Presentation functions are allowed to group text for presentation in smaller groups than the line separators imply and present such groups with source indication

together with text groups from other sources (see the following presentation examples). Erasure has no specific limit by any delimiter in the text stream.

4.1. Presentation by multi-party aware endpoints

A multi-party aware receiving party, presenting real-time text MUST separate text from different sources and present them in separate presentation fields. The receiving party MAY separate presentation of parts of text from a source in readable groups based on other criteria than line separator and merge these groups in the presentation area when it benefits the user to most easily find and read text from the different participants. The criteria MAY e.g. be a received comma, full stop, or other phrase delimiters, or a long pause.

When text is received from multiple original sources, the presentation SHOULD provide a view where text is added in multiple presentation fields.

If the presentation presents text from different sources in one common area, the presenting endpoint SHOULD insert text from the local user ended at suitable points merged with received text to indicate the relative timing for when the text groups were completed. In this presentation mode, the receiving endpoint SHALL present the source of the different groups of text. This presentation style is called the "chat" style here.

A view of a three-party RTT call in chat style is shown in this example .

[Alice] Hi, Alice here.	^
[Bob] Bob as well.	-
[Eve] Hi, this is Eve, calling from Paris. I thought you should be here.	
[Alice] I am coming on Thursday, my performance is not until Friday morning.	
[Bob] And I on Wednesday evening.	
[Alice] Can we meet on Thursday evening?	
[Eve] Yes, definitely. How about 7pm. at the entrance of the restaurant Le Lion Blanc?	
[Eve] we can have dinner and then take a walk	-
<Eve-typing> But I need to be back to the hotel by 11 because I need	v
<Bob-typing> I wou	^
of course, I underst	-
	v

Figure 3: Example of a three-party RTT call presented in chat style seen at participant 'Alice's endpoint.

Other presentation styles than the chat style may be arranged.

This figure shows how a coordinated column view MAY be presented.

Bob	Eve	Alice
My flight is to Orly		I will arrive by TGV. Convenient to the main station.
Eve, will you do your presentation on Friday?	Hi all, can we plan for the seminar?	
Fine, wo	Yes, Friday at 10.	We need to meet befo

Figure 4: An example of a coordinated column-view of a three-party session with entries ordered vertically in approximate time-order.

4.2. Multi-party mixing for multi-party unaware endpoints

When the mixer has indicated multi-party capability by the "rtt-mixer" sdp attribute in an SDP negotiation, but the multi-party capability negotiation fails with an endpoint, then the agreed "text/red" or "text/t140" format SHALL be used and the mixer SHOULD compose a best-effort presentation of multi-party real-time text in one stream intended to be presented by an endpoint with no multi-party awareness.

This presentation format has functional limitations and SHOULD be used only to enable participation in multi-party calls by legacy deployed endpoints implementing only RFC 4103 without any multi-party extensions specified in this document.

The principles and procedures below do not specify any new protocol elements. They are instead composed from the information in ITU-T T.140 [T140] and an ambition to provide a best effort presentation on an endpoint which has functions only for two-party calls.

The mixer mixing for multi-party unaware endpoints SHALL compose a simulated limited multi-party RTT view suitable for presentation in one presentation area. The mixer SHALL group text in suitable groups and prepare for presentation of them by inserting a new line between them if the transmitted text did not already end with a new line. A presentable label SHOULD be composed and sent for the source initially in the session and after each source switch. With this procedure the time for source switching is depending on the actions of the users. In order to expedite source switch, a user can for example end its turn with a new line.

4.2.1. Actions by the mixer at reception from the call participants

When text is received by the mixer from the different participants, the mixer SHALL recover text from redundancy if any packets are lost. The mark for lost text [T140ad1] SHOULD be inserted in the stream if unrecoverable loss appears. Any Unicode "BOM" characters, possibly used for keep-alive shall be deleted. The time of creation of text (retrieved from the RTP timestamp) SHALL be stored together with the received text from each source in queues for transmission to the recipients.

4.2.2. Actions by the mixer for transmission to the recipients

The following procedure SHOULD be applied for each recipient of multi-part text from the mixer.

The text for transmission SHOULD be formatted by the mixer for each receiving user for presentation in one single presentation area. Text received from a participant SHOULD NOT be included in transmission to that participant. When there is text available for transmission from the mixer to a receiving party from more than one participant, the mixer SHOULD switch between transmission of text from the different sources at suitable points in the transmitted stream.

When switching source, the mixer SHOULD insert a line separator if the already transmitted text did not end with a new line (line separator or CRLF). A label SHOULD be composed from information in the CNAME and NAME fields in RTCP reports from the participant to have its text transmitted, or from other session information for that user. The label SHOULD be delimited by suitable characters (e.g. '[']') and transmitted. The CSRC SHOULD indicate the selected source. Then text from that selected participant SHOULD be transmitted until a new suitable point for switching source is reached.

Integrity considerations SHALL be taken when composing the label.

Seeking a suitable point for switching source SHOULD be done when there is older text waiting for transmission from any party than the age of the last transmitted text. Suitable points for switching are:

- * A completed phrase ended by comma
- * A completed sentence
- * A new line (line separator or CRLF)
- * A long pause (e.g. > 10 seconds) in received text from the currently transmitted source
- * If text from one participant has been transmitted with text from other sources waiting for transmission for a long time (e.g. > 1 minute) and none of the other suitable points for switching has occurred, a source switch MAY be forced by the mixer at next word delimiter, and also if even a word delimiter does not occur within a time (e.g. 15 seconds) after the scan for word delimiter started.

When switching source, the source which has the oldest text in queue SHOULD be selected to be transmitted. A character display count SHOULD be maintained for the currently transmitted source, starting at zero after the label is transmitted for the currently transmitted source.

The status SHOULD be maintained for the latest control code for Select Graphic Rendition (SGR) from each source. If there is an SGR code stored as the status for the current source before the source switch is done, a reset of SGR shall be sent by the sequence SGR 0 [009B 0000 006D] after the new line and before the new label during a source switch. See SGR below for an explanation. This transmission does not influence the display count.

If there is an SGR code stored for the new source after the source switch, that SGR code SHOULD be transmitted to the recipient before the label. This transmission does not influence the display count.

4.2.3. Actions on transmission of text

Text from a source sent to the recipient SHOULD increase the display count by one per transmitted character.

4.2.4. Actions on transmission of control codes

The following control codes specified by T.140 require specific actions. They SHOULD cause specific considerations in the mixer. Note that the codes presented here are expressed in UCS-16, while transmission is made in UTF-8 transform of these codes.

BEL 0007 Bell Alert in session, provides for alerting during an active session. The display count SHOULD not be altered.

NEW LINE 2028 Line separator. Check and perform a source switch if appropriate. Increase display count by 1.

CR LF 000D 000A A supported, but not preferred way of requesting a new line. Check and perform a source switch if appropriate. Increase display count by 1.

INT ESC 0061 Interrupt (used to initiate mode negotiation procedure). The display count SHOULD not be altered.

SGR 009B Ps 006D Select graphic rendition. Ps is rendition parameters specified in ISO 6429. The display count SHOULD not be altered. The SGR code SHOULD be stored for the current source.

SOS 0098 Start of string, used as a general protocol element

introducer, followed by a maximum 256 bytes string and the ST. The display count SHOULD not be altered.

ST 009C String terminator, end of SOS string. The display count SHOULD not be altered.

ESC 001B Escape - used in control strings. The display count SHOULD not be altered for the complete escape code.

Byte order mark "BOM" (U+FEFF) "Zero width, no break space", used for synchronization and keep-alive. SHOULD be deleted from incoming streams. Shall be sent first after session establishment to the recipient. The display count shall not be altered.

Missing text mark (U+FFFD) "Replacement character", represented as a question mark in a rhombus, or if that is not feasible, replaced by an apostrophe ', marks place in stream of possible text loss. SHOULD be inserted by the reception procedure in case of unrecoverable loss of packets. The display count SHOULD be increased by one when sent as for any other character.

SGR If a control code for selecting graphic rendition (SGR), other than reset of the graphic rendition (SGR 0) is sent to a recipient, that control code shall also be stored as status for the source in the storage for SGR status. If a reset graphic rendition (SGR 0) originated from a source is sent, then the SGR status storage for that source shall be cleared. The display count shall not be increased.

BS (U+0008) Back Space, intended to erase the last entered character by a source. Erasure by backspace cannot always be performed as the erasing party intended. If an erasing action erases all text up to the end of the leading label after a source switch, then the mixer must not transmit more backspaces. Instead it is RECOMMENDED that a letter "X" is inserted in the text stream for each backspace as an indication of the intent to erase more. A new line is usually coded by a Line Separator, but the character combination "CRLF" MAY be used instead. Erasure of a new line is in both cases done by just one erasing action (Backspace). If the display count has a positive value it is decreased by one when the BS is sent. If the display count is at zero, it is not altered.

4.2.5. Packet transmission

A mixer transmitting to a multi-party unaware terminal SHOULD send primary data only from one source per packet. The SSRC SHOULD be the SSRC of the mixer. The CSRC list SHOULD contain one member and be the SSRC of the source of the primary data.

4.2.6. Functional limitations

When a multi-party unaware endpoint presents a conversation in one display area in a chat style, it inserts source indications for remote text and local user text as they are merged in completed text groups. When an endpoint using this layout receives and presents text mixed for multi-party unaware endpoints, there will be two levels of source indicators for the received text; one generated by the mixer and inserted in a label after each source switch, and another generated by the receiving endpoint and inserted after each switch between local and remote source in the presentation area. This will waste display space and look inconsistent to the reader.

New text can be presented only from one source at a time. Switch of source to be presented takes place at suitable places in the text, such as end of phrase, end of sentence, line separator and inactivity. Therefore the time to switch to present waiting text from other sources may become long and will vary and depend on the actions of the currently presented source.

Erasure can only be done up to the latest source switch. If a user tries to erase more text, the erasing actions will be presented as letter X after the label.

Text loss because of network errors may hit the label between entries from different parties, causing risk for misunderstanding from which source a piece of text is.

These facts makes it strongly RECOMMENDED to implement multi-party awareness in RTT endpoints. The use of the mixing method for multi-party-unaware endpoints should be left for use with endpoints which are impossible to upgrade to become multi-party aware.

4.2.7. Example views of presentation on multi-party unaware endpoints

The following pictures are examples of the view on a participant's display for the multi-party-unaware case.

Conference	Alice
[Bob]:My flight is to Orly.	I will arrive by TGV. Convenient to the main station.
[Eve]:Hi all, can we plan for the seminar.	
[Bob]:Eve, will you do your presentation on Friday?	
[Eve]:Yes, Friday at 10.	
[Bob]: Fine, wo	We need to meet befo

Figure 5: Alice who has a conference-unaware client is receiving the multi-party real-time text in a single-stream. This figure shows how a coordinated column view MAY be presented on Alice's device.

[Alice] Hi, Alice here.	^
[mix][Bob] Bob as well.	-
[Eve] Hi, this is Eve, calling from Paris I thought you should be here.	
[Alice] I am coming on Thursday, my performance is not until Friday morning.	
[mix][Bob] And I on Wednesday evening.	
[Eve] we can have dinner and then walk	
[Eve] But I need to be back to the hotel by 11 because I need	
	-
of course, I underst	v

Figure 6: An example of a view of the multi-party unaware presentation in chat style. Alice is the local user.

5. Relation to Conference Control

5.1. Use with SIP centralized conferencing framework

The SIP conferencing framework, mainly specified in RFC 4353[RFC4353], RFC 4579[RFC4579] and RFC 4575[RFC4575] is suitable for coordinating sessions including multi-party RTT. The RTT stream between the mixer and a participant is one and the same during the conference. Participants get announced by notifications when participants are joining or leaving, and further user information may be provided. The SSRC of the text to expect from joined users MAY be included in a notification. The notifications MAY be used both for security purposes and for translation to a label for presentation to other users.

5.2. Conference control

In managed conferences, control of the real-time text media SHOULD be provided in the same way as other for media, e.g. for muting and unmuting by the direction attributes in SDP [RFC4566].

Note that floor control functions may be of value for RTT users as well as for users of other media in a conference.

6. Gateway Considerations

6.1. Gateway considerations with Textphones (e.g. TTYs).

Multi-party RTT sessions may involve gateways of different kinds. Gateways involved in setting up sessions SHALL correctly reflect the multi-party capability or unawareness of the combination of the gateway and the remote endpoint beyond the gateway.

One case that may occur is a gateway to PSTN for communication with textphones (e.g. TTYs). Textphones are limited devices with no multi-party awareness, and it SHOULD therefore be suitable for the gateway to not indicate multi-party awareness for that case. Another solution is that the gateway indicates multi-party capability towards the mixer, and includes the multi-party mixer function for multi-party unaware endpoints itself. This solution makes it possible to make adaptations for the functional limitations of the textphone (TTY).

More information on gateways to textphones (TTYs) is found in RFC 5194[RFC5194]

6.2. Gateway considerations with WebRTC.

Gateway operation to real-time text in WebRTC may also be required. In WebRTC, RTT is specified in [I-D.ietf-mmusic-t140-usage-data-channel].

A multi-party bridge may have functionality for communicating by RTT both in RTP streams with RTT and WebRTC t140 data channels. Other configurations may consist of a multi-party bridge with either technology for RTT transport and a separate gateway for conversion of the text communication streams between RTP and t140 data channel.

In WebRTC, it is assumed that for a multi-party session, one t140 data channel is established for each source from a gateway or bridge to each participant. Each participant also has a data channel with two-way connection with the gateway or bridge.

The t140 channel used both ways is for text from the WebRTC user and from the bridge or gateway itself to the WebRTC user. The label parameter of this t140 channel is used as NAME field in RTCP to participants on the RTP side. The other t140 channels are only for text from other participants to the WebRTC user.

When a new participant has entered the session with RTP transport of rtt, a new t140 channel SHOULD be established to WebRTC users with the label parameter composed from the NAME field in RTCP on the RTP side.

When a new participant has entered the multi-party session with RTT transport in a WebRTC t140 data channel, the new participant SHOULD be announced by a notification to RTP users. The label parameter from the WebRTC side SHOULD be used as the NAME RTCP field on the RTP side, or other available session information.

7. Updates to RFC 4103

This document updates RFC 4103[RFC4103] by introducing an sdp media attribute "rtt-mixer" for negotiation of multi-party mixing capability with the [RFC4103] format, and by specifying the rules for packets when multi-party capability is negotiated and in use.

8. Congestion considerations

The congestion considerations and recommended actions from RFC 4103 [RFC4103] are valid also in multi-party situations.

The first action in case of congestion SHOULD be to temporarily increase the transmission interval up to two seconds.

If the unlikely situation appears that more than 20 participants in a conference send text simultaneously, it will take more than 7 seconds between presentation of text from each of these participants. More time than that can cause confusion in the session. It is therefore RECOMMENDED that the mixer discards such text in excess inserts a general indication of possible text loss [T140ad1] in the session. If the main text contributor is indicated in any way, the mixer MAY avoid deleting text from that participant.

9. Acknowledgements

James Hamlin for format and performance aspects.

10. IANA Considerations

10.1. Registration of the "rtt-mixer" sdp media attribute

[RFC EDITOR NOTE: Please replace all instances of RFCXXXX with the RFC number of this document.]

IANA is asked to register the new sdp attribute "rtt-mixer".

Contact name: IESG

Contact email: iesg@ietf.org

Attribute name: rtt-mixer

Attribute semantics: See RFCXXXX Section 3.1

Attribute value: none

Usage level: media

Purpose: Indicate support by mixer and endpoint of multi-party mixing for real-time text transmission, using a common RTP-stream for transmission of text from a number of sources mixed with one source at a time and the source indicated in a single CSRC-list member.

Charset Dependent: no

O/A procedure: See RFCXXXX Section 3.1

Mux Category: normal

Reference: RFCXXXX

11. Security Considerations

The RTP-mixer model requires the mixer to be allowed to decrypt, pack and encrypt secured text from the conference participants. Therefore the mixer needs to be trusted. This is similar to the situation for central mixers of audio and video.

The requirement to transfer information about the user in RTCP reports in SDES, CNAME and NAME fields, and in conference notifications, for creation of labels may have privacy concerns as already stated in RFC 3550 [RFC3550], and may be restricted of privacy reasons. The receiving user will then get a more symbolic label for the source.

Participants with malicious intentions may appear and e.g. disturb the multi-party session by a continuous flow of text, or masquerading as text from other participants. Counteractions should be to require secure signaling, media and authentication, and to provide higher level conference functions e.g. for blocking and expelling participants.

12. Change history

12.1. Changes included in draft-ietf-avtcore-multi-party-rtt-mix-11

Timestamps and timestamp offsets added to the packet examples in section 3.23, and the description corrected.

A number of minor corrections added in sections 3.10 - 3.23.

12.2. Changes included in draft-ietf-avtcore-multi-party-rtt-mix-10

The packet composition was modified for interleaving packets from different sources.

The packet reception was modified for the new interleaving method.

The packet sequence examples was adjusted for the new interleaving method.

Modifications according to responses to Brian Rosen of 2020-11-03

12.3. Changes included in draft-ietf-avtcore-multi-party-rtt-mix-09

Changed name on the SDP media attribute to "rtt-mixer"

Restructure of section 2 for balance between aware and unaware cases.

Moved conference control to own section.

Improved clarification of recovery and loss in the packet sequence example.

A number of editorial corrections and improvements.

12.4. Changes included in draft-ietf-avtcore-multi-party-rtt-mix-08

Deleted the method requiring a new packet format "text/rex" because of the longer standardization and implementation period it needs.

Focus on use of RFC 4103 text/red format with shorter transmission interval, and source indicated in CSRC.

12.5. Changes included in draft-ietf-avtcore-multi-party-rtt-mix-07

Added a method based on the "text/red" format and single source per packet, negotiated by the "rtt-mixer" sdp attribute.

Added reasoning and recommendation about indication of loss.

The highest number of sources in one packet is 15, not 16. Changed.

Added in information on update to RFC 4103 that RFC 4103 explicitly allows addition of FEC method. The redundancy is a kind of forward error correction..

12.6. Changes included in draft-ietf-avtcore-multi-party-rtt-mix-06

Improved definitions list format.

The format of the media subtype parameters is made to match the requirements.

The mapping of media subtype parameters to sdp is included.

The CPS parameter belongs to the t140 subtype and does not need to be registered here.

12.7. Changes included in draft-ietf-avtcore-multi-party-rtt-mix-05

nomenclature and editorial improvements

"this document" used consistently to refer to this document.

12.8. Changes included in draft-ietf-avtcore-multi-party-rtt-mix-04

'Redundancy header' renamed to 'data header'.

More clarifications added.

Language and figure number corrections.

12.9. Changes included in draft-ietf-avtcore-multi-party-rtt-mix-03

Mention possible need to mute and raise hands as for other media.

---done ----

Make sure that use in two-party calls is also possible and explained.

- may need more wording -

Clarify the RTT is often used together with other media. --done--

Tell that text mixing is N-1. A users own text is not received in the mix. -done-

In 3. correct the interval to: A "text/rex" transmitter SHOULD send packets distributed in time as long as there is something (new or redundant T140blocks) to transmit. The maximum transmission interval SHOULD then be 300 ms. It is RECOMMENDED to send a packet to a receiver as soon as new text to that receiver is available, as long as the time after the latest sent packet to the same receiver is more than 150 ms, and also the maximum character rate to the receiver is not exceeded. The intention is to keep the latency low while keeping a good protection against text loss in bursty packet loss conditions. -done-

In 1.3 say that the format is used both ways. -done-

In 13.1 change presentation area to presentation field so that reader does not think it shall be totally separated. -done-

In Performance and intro, tell the performance in number of simultaneous sending users and introduced delay 16, 150 vs requirements 5 vs 500. -done --

Clarify redundancy level per connection. -done-

Timestamp also for the last data header. To make it possible for all text to have time offset as for transmission from the source. Make that header equal to the others. -done-

Mixer always use the CSRC list, even for its own BOM. -done-

Combine all talk about transmission interval (300 ms vs when text has arrived) in section 3 in one paragraph or close to each other. -done-

Documents the goal of good performance with low delay for 5 simultaneous typers in the introduction. -done-

Describe better that only primary text shall be sent on to receivers. Redundancy and loss must be resolved by the mixer. -done-

12.10. Changes included in draft-ietf-avtcore-multi-party-rtt-mix-02

SDP and better description and visibility of security by OSRTP RFC 8634 needed.

The description of gatewaying to WebRTC extended.

The description of the data header in the packet is improved.

12.11. Changes to draft-ietf-avtcore-multi-party-rtt-mix-01

2,5,6 More efficient format "text/rex" introduced and attribute a=rtt-mix deleted.

3. Brief about use of OSRTP for security included- More needed.

4. Brief motivation for the solution and why not rtp-translator is used added to intro.

7. More limitations for the multi-party unaware mixing method inserted.

8. Updates to RFC 4102 and 4103 more clearly expressed.

9. Gateway to WebRTC started. More needed.

12.12. Changes from draft-hellstrom-avtcore-multi-party-rtt-source-03 to draft-ietf-avtcore-multi-party-rtt-mix-00

Changed file name to draft-ietf-avtcore-multi-party-rtt-mix-00

Replaced CDATA in IANA registration table with better coding.

Converted to xml2rfc version 3.

12.13. Changes from draft-hellstrom-avtcore-multi-party-rtt-source-02 to -03

Changed company and e-mail of the author.

Changed title to "RTP-mixer formatting of multi-party Real-time text" to better match contents.

Check and modification where needed of use of RFC 2119 words SHALL etc.

More about the CC value in sections on transmitters and receivers so that 1-to-1 sessions do not use the mixer format.

Enhanced section on presentation for multi-party-unaware endpoints

A paragraph recommending CPS=150 inserted in the performance section.

12.14. Changes from draft-hellstrom-avtcore-multi-party-rtt-source-01 to -02

In Abstract and 1. Introduction: Introduced wording about regulatory requirements.

In section 5: The transmission interval is decreased to 100 ms when there is text from more than one source to transmit.

In section 11 about SDP negotiation, a SHOULD-requirement is introduced that the mixer should make a mix for multi-party unaware endpoints if the negotiation is not successful. And a reference to a later chapter about it.

The presentation considerations chapter 14 is extended with more information about presentation on multi-party aware endpoints, and a new section on the multi-party unaware mixing with low functionality but SHOULD a be implemented in mixers. Presentation examples are added.

A short chapter 15 on gateway considerations is introduced.

Clarification about the text/t140 format included in chapter 10.

This sentence added to the chapter 10 about use without redundancy. "The text/red format SHOULD be used unless some other protection against packet loss is utilized, for example a reliable network or transport."

Note about deviation from RFC 2198 added in chapter 4.

In chapter 9. "Use with SIP centralized conferencing framework" the following note is inserted: Note: The CSRC-list in an RTP packet only includes participants who's text is included in one or more text blocks. It is not the same as the list of participants in a conference. With audio and video media, the CSRC-list would often contain all participants who are not muted whereas text participants that don't type are completely silent and so don't show up in RTP packet CSRC-lists.

12.15. Changes from draft-hellstrom-avtcore-multi-party-rtt-source-00 to -01

Editorial cleanup.

Changed capability indication from fntp-parameter to SDP attribute "rtt-mix".

Swapped order of redundancy elements in the example to match reality.

Increased the SDP negotiation section

13. References

13.1. Normative References

[I-D.ietf-mmusic-t140-usage-data-channel]

Holmberg, C. and G. Hellstrom, "T.140 Real-time Text Conversation over WebRTC Data Channels", Work in Progress, Internet-Draft, draft-ietf-mmusic-t140-usage-data-channel-14, 10 April 2020, <<https://tools.ietf.org/html/draft-ietf-mmusic-t140-usage-data-channel-14>>.

[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>>.

[RFC4102] Jones, P., "Registration of the text/red MIME Sub-Type", RFC 4102, DOI 10.17487/RFC4102, June 2005, <<https://www.rfc-editor.org/info/rfc4102>>.

- [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>>.
- [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>>.
- [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>>.
- [RFC6263] Marjou, X. and A. Sollaud, "Application Mechanism for Keeping Alive the NAT Mappings Associated with RTP / RTP Control Protocol (RTCP) Flows", RFC 6263, DOI 10.17487/RFC6263, June 2011, <<https://www.rfc-editor.org/info/rfc6263>>.
- [RFC8643] Johnston, A., Aboba, B., Hutton, A., Jesske, R., and T. Stach, "An Opportunistic Approach for Secure Real-time Transport Protocol (OSRTP)", RFC 8643, DOI 10.17487/RFC8643, August 2019, <<https://www.rfc-editor.org/info/rfc8643>>.
- [T140] ITU-T, "Recommendation ITU-T T.140 (02/1998), Protocol for multimedia application text conversation", February 1998, <<https://www.itu.int/rec/T-REC-T.140-199802-I/en>>.
- [T140ad1] ITU-T, "Recommendation ITU-T.140 Addendum 1 - (02/2000), Protocol for multimedia application text conversation", February 2000, <<https://www.itu.int/rec/T-REC-T.140-200002-I!Add1/en>>.

13.2. Informative References

- [RFC4353] Rosenberg, J., "A Framework for Conferencing with the Session Initiation Protocol (SIP)", RFC 4353, DOI 10.17487/RFC4353, February 2006, <<https://www.rfc-editor.org/info/rfc4353>>.
- [RFC4575] Rosenberg, J., Schulzrinne, H., and O. Levin, Ed., "A Session Initiation Protocol (SIP) Event Package for Conference State", RFC 4575, DOI 10.17487/RFC4575, August 2006, <<https://www.rfc-editor.org/info/rfc4575>>.

[RFC4579] Johnston, A. and O. Levin, "Session Initiation Protocol (SIP) Call Control - Conferencing for User Agents", BCP 119, RFC 4579, DOI 10.17487/RFC4579, August 2006, <<https://www.rfc-editor.org/info/rfc4579>>.

[RFC5194] van Wijk, A., Ed. and G. Gybels, Ed., "Framework for Real-Time Text over IP Using the Session Initiation Protocol (SIP)", RFC 5194, DOI 10.17487/RFC5194, June 2008, <<https://www.rfc-editor.org/info/rfc5194>>.

Author's Address

Gunnar Hellstrom
Gunnar Hellstrom Accessible Communication
SE-13670 Vendelso
Sweden

Email: gunnar.hellstrom@ghaccess.se

avtcore
Internet-Draft
Intended status: Standards Track
Expires: 6 May 2021

S. Zhao
S. Wenger
Tencent
Y. Sanchez
Fraunhofer HHI
Y.-K. Wang
Bytedance Inc.
2 November 2020

RTP Payload Format for Versatile Video Coding (VVC)
draft-ietf-avtcore-rtp-vvc-05

Abstract

This memo describes an RTP payload format for the video coding standard ITU-T Recommendation H.266 and ISO/IEC International Standard 23090-3, both also known as Versatile Video Coding (VVC) and developed by the Joint Video Experts Team (JVET). The RTP payload format allows for packetization of one or more Network Abstraction Layer (NAL) units in each RTP packet payload as well as fragmentation of a NAL unit into multiple RTP packets. The payload format has wide applicability in videoconferencing, Internet video streaming, and high-bitrate entertainment-quality video, among other applications.

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 6 May 2021.

Copyright Notice

Copyright (c) 2020 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	3
1.1.	Overview of the VVC Codec	3
1.1.1.	Coding-Tool Features (informative)	4
1.1.2.	Systems and Transport Interfaces (informative)	6
1.1.3.	High-Level Picture Partitioning (informative)	11
1.1.4.	NAL Unit Header	13
1.2.	Overview of the Payload Format	15
2.	Conventions	15
3.	Definitions and Abbreviations	15
3.1.	Definitions	15
3.1.1.	Definitions from the VVC Specification	15
3.1.2.	Definitions Specific to This Memo	18
3.2.	Abbreviations	19
4.	RTP Payload Format	20
4.1.	RTP Header Usage	20
4.2.	Payload Header Usage	22
4.3.	Payload Structures	23
4.3.1.	Single NAL Unit Packets	23
4.3.2.	Aggregation Packets (APs)	24
4.3.3.	Fragmentation Units	28
4.4.	Decoding Order Number	31
5.	Packetization Rules	32
6.	De-packetization Process	33
7.	Payload Format Parameters	35
7.1.	Media Type Registration	35
7.2.	SDP Parameters	35
7.2.1.	Mapping of Payload Type Parameters to SDP	35
7.2.2.	Usage with SDP Offer/Answer Model	50
8.	Use with Feedback Messages	50
8.1.	Picture Loss Indication (PLI)	50
8.2.	Slice Loss Indication (SLI)	51
8.3.	Reference Picture Selection Indication (RPSI)	51
8.4.	Full Intra Request (FIR)	51
9.	Frame Marking	52
9.1.	Frame Marking Short Extension	52
9.2.	Frame Marking Long Extension	53
10.	Security Considerations	54

11. Congestion Control	55
12. IANA Considerations	56
13. Acknowledgements	57
14. References	57
14.1. Normative References	57
14.2. Informative References	58
Appendix A. Change History	60
Authors' Addresses	60

1. Introduction

The Versatile Video Coding [VVC] specification, formally published as both ITU-T Recommendation H.266 and ISO/IEC International Standard 23090-3, is currently in the ITU-T publication process and the ISO/IEC approval process. VVC is reported to provide significant coding efficiency gains over HEVC [HEVC] as known as H.265, and other earlier video codecs.

This memo specifies an RTP payload format for VVC. It shares its basic design with the NAL (Network Abstraction Layer) unit-based RTP payload formats of, H.264 Video Coding [RFC6184], Scalable Video Coding (SVC) [RFC6190], High Efficiency Video Coding (HEVC) [RFC7798] and their respective predecessors. With respect to design philosophy, security, congestion control, and overall implementation complexity, it has similar properties to those earlier payload format specifications. This is a conscious choice, as at least RFC 6184 is widely deployed and generally known in the relevant implementer communities. Certain mechanisms known from [RFC6190] were incorporated in VVC, as VVC version 1 supports temporal, spatial, and signal-to-noise ratio (SNR) scalability.

1.1. Overview of the VVC Codec

VVC and HEVC share a similar hybrid video codec design. In this memo, we provide a very brief overview of those features of VVC that are, in some form, addressed by the payload format specified herein. Implementers have to read, understand, and apply the ITU-T/ISO/IEC specifications pertaining to VVC to arrive at interoperable, well-performing implementations.

Conceptually, both VVC and HEVC include a Video Coding Layer (VCL), which is often used to refer to the coding-tool features, and a NAL, which is often used to refer to the systems and transport interface aspects of the codecs.

1.1.1.1. Coding-Tool Features (informative)

Coding tool features are described below with occasional reference to the coding tool set of HEVC, which is well known in the community.

Similar to earlier hybrid-video-coding-based standards, including HEVC, the following basic video coding design is employed by VVC. A prediction signal is first formed by either intra- or motion-compensated prediction, and the residual (the difference between the original and the prediction) is then coded. The gains in coding efficiency are achieved by redesigning and improving almost all parts of the codec over earlier designs. In addition, VVC includes several tools to make the implementation on parallel architectures easier.

Finally, VVC includes temporal, spatial, and SNR scalability as well as multiview coding support.

Coding blocks and transform structure

Among major coding-tool differences between HEVC and VVC, one of the important improvements is the more flexible coding tree structure in VVC, i.e., multi-type tree. In addition to quadtree, binary and ternary trees are also supported, which contributes significant improvement in coding efficiency. Moreover, the maximum size of coding tree unit (CTU) is increased from 64x64 to 128x128. To improve the coding efficiency of chroma signal, luma chroma separated trees at CTU level may be employed for intra-slices. The square transforms in HEVC are extended to non-square transforms for rectangular blocks resulting from binary and ternary tree splits. Besides, VVC supports multiple transform sets (MTS), including DCT-2, DST-7, and DCT-8 as well as the non-separable secondary transform. The transforms used in VVC can have different sizes with support for larger transform sizes. For DCT-2, the transform sizes range from 2x2 to 64x64, and for DST-7 and DCT-8, the transform sizes range from 4x4 to 32x32. In addition, VVC also support sub-block transform for both intra and inter coded blocks. For intra coded blocks, intra sub-partitioning (ISP) may be used to allow sub-block based intra prediction and transform. For inter blocks, sub-block transform may be used assuming that only a part of an inter-block has non-zero transform coefficients.

Entropy coding

Similar to HEVC, VVC uses a single entropy-coding engine, which is based on context adaptive binary arithmetic coding [CABAC], but with the support of multi-window sizes. The window sizes can be initialized differently for different context models. Due to such a design, it has more efficient adaptation speed and better coding

efficiency. A joint chroma residual coding scheme is applied to further exploit the correlation between the residuals of two color components. In VVC, different residual coding schemes are applied for regular transform coefficients and residual samples generated using transform-skip mode.

In-loop filtering

VVC has more feature support in loop filters than HEVC. The deblocking filter in VVC is similar to HEVC but operates at a smaller grid. After deblocking and sample adaptive offset (SAO), an adaptive loop filter (ALF) may be used. As a Wiener filter, ALF reduces distortion of decoded pictures. Besides, VVC introduces a new module before deblocking called luma mapping with chroma scaling to fully utilize the dynamic range of signal so that rate-distortion performance of both SDR and HDR content is improved.

Motion prediction and coding

Compared to HEVC, VVC introduces several improvements in this area. First, there is the adaptive motion vector resolution (AMVR), which can save bit cost for motion vectors by adaptively signaling motion vector resolution. Then the affine motion compensation is included to capture complicated motion like zooming and rotation. Meanwhile, prediction refinement with the optical flow with affine mode (PROF) is further deployed to mimic affine motion at the pixel level. Thirdly the decoder side motion vector refinement (DMVR) is a method to derive MV vector at decoder side based on block matching so that fewer bits may be spent on motion vectors. Bi-directional optical flow (BDOF) is a similar method to PROF. BDOF adds a sample wise offset at 4x4 sub-block level that is derived with equations based on gradients of the prediction samples and a motion difference relative to CU motion vectors. Furthermore, merge with motion vector difference (MMVD) is a special mode, which further signals a limited set of motion vector differences on top of merge mode. In addition to MMVD, there are another three types of special merge modes, i.e., sub-block merge, triangle, and combined intra-/inter-prediction (CIIP). Sub-block merge list includes one candidate of sub-block temporal motion vector prediction (SbTMVP) and up to four candidates of affine motion vectors. Triangle is based on triangular block motion compensation. CIIP combines intra- and inter- predictions with weighting. Adaptive weighting may be employed with a block-level tool called bi-prediction with CU based weighting (BCW) which provides more flexibility than in HEVC.

Intra prediction and intra-coding

To capture the diversified local image texture directions with finer granularity, VVC supports 65 angular directions instead of 33 directions in HEVC. The intra mode coding is based on a 6-most-probable-mode scheme, and the 6 most probable modes are derived using the neighboring intra prediction directions. In addition, to deal with the different distributions of intra prediction angles for different block aspect ratios, a wide-angle intra prediction (WAIP) scheme is applied in VVC by including intra prediction angles beyond those present in HEVC. Unlike HEVC which only allows using the most adjacent line of reference samples for intra prediction, VVC also allows using two further reference lines, as known as multi-reference-line (MRL) intra prediction. The additional reference lines can be only used for the 6 most probable intra prediction modes. To capture the strong correlation between different colour components, in VVC, a cross-component linear mode (CCLM) is utilized which assumes a linear relationship between the luma sample values and their associated chroma samples. For intra prediction, VVC also applies a position-dependent prediction combination (PDPC) for refining the prediction samples closer to the intra prediction block boundary. Matrix-based intra prediction (MIP) modes are also used in VVC which generates an up to 8x8 intra prediction block using a weighted sum of downsampled neighboring reference samples, and the weights are hardcoded constants.

Other coding-tool feature

VVC introduces dependent quantization (DQ) to reduce quantization error by state-based switching between two quantizers.

1.1.2. Systems and Transport Interfaces (informative)

VVC inherits the basic systems and transport interfaces designs from HEVC and H.264. These include the NAL-unit-based syntax structure, the hierarchical syntax and data unit structure, the supplemental enhancement information (SEI) message mechanism, and the video buffering model based on the hypothetical reference decoder (HRD). The scalability features of VVC are conceptually similar to the scalable variant of HEVC known as SHVC. The hierarchical syntax and data unit structure consists of parameter sets at various levels (decoder, sequence (pertaining to all), sequence (pertaining to a single), picture), picture-level header parameters, slice-level header parameters, and lower-level parameters.

A number of key components that influenced the network abstraction layer design of VVC as well as this memo are described below

Decoding capability information

The decoding capability information includes parameters that stay constant for the lifetime of a Video Bitstream, which in IETF terms can translate to the lifetime of a session. Such information includes profile, level, and sub-profile information to determine a maximum capability interop point that is guaranteed to be never exceeded, even if splicing of video sequences occurs within a session. It further includes constraint fields (most of which are flags), which can optionally be set to indicate that the video bitstream will be constraint in the use of certain features as indicated by the values of those fields. With this, a bitstream can be labelled as not using certain tools, which allows among other things for resource allocation in a decoder implementation.

Video parameter set

The video parameter set (VPS) pertains to a coded video sequences (CVS) of multiple layers covering the same range of access units, and includes, among other information decoding dependency expressed as information for reference picture list construction of enhancement layers. The VPS provides a "big picture" of a scalable sequence, including what types of operation points are provided, the profile, tier, and level of the operation points, and some other high-level properties of the bitstream that can be used as the basis for session negotiation and content selection, etc. One VPS may be referenced by one or more sequence parameter sets.

Sequence parameter set

The sequence parameter set (SPS) contains syntax elements pertaining to a coded layer video sequence (CLVS), which is a group of pictures belonging to the same layer, starting with a random access point, and followed by pictures that may depend on each other, until the next random access point picture. In MPEG-2, the equivalent of a CVS was a group of pictures (GOP), which normally started with an I frame and was followed by P and B frames. While more complex in its options of random access points, VVC retains this basic concept. One remarkable difference of VVC is that a CLVS may start with a Gradual Decoding Refresh (GDR) picture, without requiring presence of traditional random access points in the bitstream, such as instantaneous decoding refresh (IDR) or clean random access (CRA) pictures. In many TV-like applications, a CVS contains a few hundred milliseconds to a few seconds of video. In video conferencing (without switching MCUs involved), a CVS can be as long in duration as the whole session.

Picture and adaptation parameter set

The picture parameter set and the adaptation parameter set (PPS and APS, respectively) carry information pertaining to zero or more pictures and zero or more slices, respectively. The PPS contains information that is likely to stay constant from picture to picture—at least for pictures for a certain type—whereas the APS contains information, such as adaptive loop filter coefficients, that are likely to change from picture to picture or even within a picture. A single APS is referenced by all slices of the same picture if that APS contains information about luma mapping with chroma scaling (LMCS) or scaling list. Different APSs containing ALF parameters can be referenced by slices of the same picture.

Picture header

A Picture Header contains information that is common to all slices that belong to the same picture. Being able to send that information as a separate NAL unit when pictures are split into several slices allows for saving bitrate, compared to repeating the same information in all slices. However, there might be scenarios where low-bitrate video is transmitted using a single slice per picture. Having a separate NAL unit to convey that information incurs in an overhead for such scenarios. For such scenarios, the picture header syntax structure is directly included in the slice header, instead of in its own NAL unit. The mode of the picture header syntax structure being included in its own NAL unit or not can only be switched on/off for an entire CLVS, and can only be switched off when in the entire CLVS each picture contains only one slice.

Profile, tier, and level

The profile, tier and level syntax structures in DCI, VPS and SPS contain profile, tier, level information for all layers that refer to the DCI, for layers associated with one or more output layer sets specified by the VPS, and for any layer that refers to the SPS, respectively.

Sub-profiles

Within the VVC specification, a sub-profile is a 32-bit number, coded according to ITU-T Rec. T.35, that does not carry a semantics. It is carried in the `profile_tier_level` structure and hence (potentially) present in the DCI, VPS, and SPS. External registration bodies can register a T.35 codepoint with ITU-T registration authorities and associate with their registration a description of bitstream restrictions beyond the profiles defined by ITU-T and ISO/IEC. This would allow encoder manufacturers to label the bitstreams generated by their encoder as complying with such sub-profile. It is expected that upstream standardization organizations (such as: DVB and ATSC),

as well as walled-garden video services will take advantage of this labelling system. In contrast to "normal" profiles, it is expected that sub-profiles may indicate encoder choices traditionally left open in the (decoder-centric) video coding specs, such as GOP structures, minimum/maximum QP values, and the mandatory use of certain tools or SEI messages.

General constraint fields

The `profile_tier_level` structure carries a considerable number of constraint fields (most of which are flags), which an encoder can use to indicate to a decoder that it will not use a certain tool or technology. They were included in reaction to a perceived market need for labelling a bitstream as not exercising a certain tool that has become commercially unviable.

Temporal scalability support

VVC includes support of temporal scalability, by inclusion of the signaling of `TemporalId` in the NAL unit header, the restriction that pictures of a particular temporal sublayer cannot be used for inter prediction reference by pictures of a lower temporal sublayer, the sub-bitstream extraction process, and the requirement that each sub-bitstream extraction output be a conforming bitstream. Media-Aware Network Elements (MANEs) can utilize the `TemporalId` in the NAL unit header for stream adaptation purposes based on temporal scalability.

Reference picture resampling (RPR)

In AVC and HEVC, the spatial resolution of pictures cannot change unless a new sequence using a new SPS starts, with an IRAP picture. VVC enables picture resolution change within a sequence at a position without encoding an IRAP picture, which is always intra-coded. This feature is sometimes referred to as reference picture resampling (RPR), as the feature needs resampling of a reference picture used for inter prediction when that reference picture has a different resolution than the current picture being decoded. RPR allows resolution change without the need of coding an IRAP picture, which causes a momentary bit rate spike in streaming or video conferencing scenarios, e.g., to cope with network condition changes. RPR can also be used in application scenarios wherein zooming of the entire video region or some region of interest is needed.

Spatial, SNR, and multiview scalability

VVC includes support for spatial, SNR, and multiview scalability. Scalable video coding is widely considered to have technical benefits and enrich services for various video applications. Until recently,

however, the functionality has not been included in the first version of specifications of the video codecs. In VVC, however, all those forms of scalability are supported in the first version of VVC natively through the signaling of the `layer_id` in the NAL unit header, the VPS which associates layers with given `layer_ids` to each other, reference picture selection, reference picture resampling for spatial scalability, and a number of other mechanisms not relevant for this memo.

Spatial scalability

With the existence of Reference Picture Resampling (RPR), the additional burden for scalability support is just a modification of the high-level syntax (HLS). The inter-layer prediction is employed in a scalable system to improve the coding efficiency of the enhancement layers. In addition to the spatial and temporal motion-compensated predictions that are available in a single-layer codec, the inter-layer prediction in VVC uses the possibly resampled video data of the reconstructed reference picture from a reference layer to predict the current enhancement layer. The resampling process for inter-layer prediction, when used, is performed at the block-level, reusing the existing interpolation process for motion compensation in single-layer coding. It means that no additional resampling process is needed to support spatial scalability.

SNR scalability

SNR scalability is similar to spatial scalability except that the resampling factors are 1:1. In other words, there is no change in resolution, but there is inter-layer prediction.

Multiview scalability

The first version of VVC also supports multiview scalability, wherein a multi-layer bitstream carries layers representing multiple views, and one or more of the represented views can be output at the same time.

SEI messages

Supplementary enhancement information (SEI) messages are information in the bitstream that do not influence the decoding process as specified in the VVC spec, but address issues of representation/rendering of the decoded bitstream, label the bitstream for certain applications, among other, similar tasks. The overall concept of SEI messages and many of the messages themselves has been inherited from

the H.264 and HEVC specs. Except for the SEI messages that affect the specification of the hypothetical reference decoder (HRD), other SEI messages for use in the VVC environment, which are generally useful also in other video coding technologies, are not included in the main VVC specification but in a companion specification [VSEI].

1.1.3. High-Level Picture Partitioning (informative)

VVC inherited the concept of tiles and wavefront parallel processing (WPP) from HEVC, with some minor to moderate differences. The basic concept of slices was kept in VVC but designed in an essentially different form. VVC is the first video coding standard that includes subpictures as a feature, which provides the same functionality as HEVC motion-constrained tile sets (MCTSs) but designed differently to have better coding efficiency and to be friendlier for usage in application systems. More details of these differences are described below.

Tiles and WPP

Same as in HEVC, a picture can be split into tile rows and tile columns in VVC, in-picture prediction across tile boundaries is disallowed, etc. However, the syntax for signaling of tile partitioning has been simplified, by using a unified syntax design for both the uniform and the non-uniform mode. In addition, signaling of entry point offsets for tiles in the slice header is optional in VVC while it is mandatory in HEVC. The WPP design in VVC has two differences compared to HEVC: i) The CTU row delay is reduced from two CTUs to one CTU; ii) Signaling of entry point offsets for WPP in the slice header is optional in VVC while it is mandatory in HEVC.

Slices

In VVC, the conventional slices based on CTUs (as in HEVC) or macroblocks (as in AVC) have been removed. The main reasoning behind this architectural change is as follows. The advances in video coding since 2003 (the publication year of AVC v1) have been such that slice-based error concealment has become practically impossible, due to the ever-increasing number and efficiency of in-picture and inter-picture prediction mechanisms. An error-concealed picture is the decoding result of a transmitted coded picture for which there is some data loss (e.g., loss of some slices) of the coded picture or a reference picture for at least some part of the coded picture is not error-free (e.g., that reference picture was an error-concealed picture). For example, when one of the multiple slices of a picture is lost, it may be error-concealed using an interpolation of the neighboring slices. While advanced video coding prediction

mechanisms provide significantly higher coding efficiency, they also make it harder for machines to estimate the quality of an error-concealed picture, which was already a hard problem with the use of simpler prediction mechanisms. Advanced in-picture prediction mechanisms also cause the coding efficiency loss due to splitting a picture into multiple slices to be more significant. Furthermore, network conditions become significantly better while at the same time techniques for dealing with packet losses have become significantly improved. As a result, very few implementations have recently used slices for maximum transmission unit size matching. Instead, substantially all applications where low-delay error resilience is required (e.g., video telephony and video conferencing) rely on system/transport-level error resilience (e.g., retransmission, forward error correction) and/or picture-based error resilience tools (feedback-based error resilience, insertion of IRAPs, scalability with higher protection level of the base layer, and so on). Considering all the above, nowadays it is very rare that a picture that cannot be correctly decoded is passed to the decoder, and when such a rare case occurs, the system can afford to wait for an error-free picture to be decoded and available for display without resulting in frequent and long periods of picture freezing seen by end users.

Slices in VVC have two modes: rectangular slices and raster-scan slices. The rectangular slice, as indicated by its name, covers a rectangular region of the picture. Typically, a rectangular slice consists of several complete tiles. However, it is also possible that a rectangular slice is a subset of a tile and consists of one or more consecutive, complete CTU rows within a tile. A raster-scan slice consists of one or more complete tiles in a tile raster scan order, hence the region covered by a raster-scan slices need not but could have a non-rectangular shape, but it may also happen to have the shape of a rectangle. The concept of slices in VVC is therefore strongly linked to or based on tiles instead of CTUs (as in HEVC) or macroblocks (as in AVC).

Subpictures

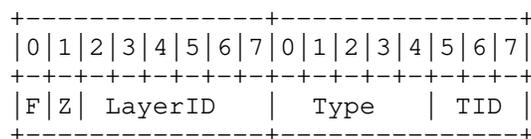
VVC is the first video coding standard that includes the support of subpictures as a feature. Each subpicture consists of one or more complete rectangular slices that collectively cover a rectangular region of the picture. A subpicture may be either specified to be extractable (i.e., coded independently of other subpictures of the same picture and of earlier pictures in decoding order) or not extractable. Regardless of whether a subpicture is extractable or not, the encoder can control whether in-loop filtering (including deblocking, SAO, and ALF) is applied across the subpicture boundaries individually for each subpicture.

Functionally, subpictures are similar to the motion-constrained tile sets (MCTSs) in HEVC. They both allow independent coding and extraction of a rectangular subset of a sequence of coded pictures, for use cases like viewport-dependent 360o video streaming optimization and region of interest (ROI) applications.

There are several important design differences between subpictures and MCTSs. First, the subpictures feature in VVC allows motion vectors of a coding block pointing outside of the subpicture even when the subpicture is extractable by applying sample padding at subpicture boundaries in this case, similarly as at picture boundaries. Second, additional changes were introduced for the selection and derivation of motion vectors in the merge mode and in the decoder side motion vector refinement process of VVC. This allows higher coding efficiency compared to the non-normative motion constraints applied at the encoder-side for MCTSs. Third, rewriting of SHs (and PH NAL units, when present) is not needed when extracting one or more extractable subpictures from a sequence of pictures to create a sub-bitstream that is a conforming bitstream. In sub-bitstream extractions based on HEVC MCTSs, rewriting of SHs is needed. Note that in both HEVC MCTSs extraction and VVC subpictures extraction, rewriting of SPSs and PPSs is needed. However, typically there are only a few parameter sets in a bitstream, while each picture has at least one slice, therefore rewriting of SHs can be a significant burden for application systems. Fourth, slices of different subpictures within a picture are allowed to have different NAL unit types. Fifth, VVC specifies HRD and level definitions for subpicture sequences, thus the conformance of the sub-bitstream of each extractable subpicture sequence can be ensured by encoders.

1.1.4. NAL Unit Header

VVC maintains the NAL unit concept of HEVC with modifications. VVC uses a two-byte NAL unit header, as shown in Figure 1. The payload of a NAL unit refers to the NAL unit excluding the NAL unit header.



The Structure of the VVC NAL Unit Header.

Figure 1

The semantics of the fields in the NAL unit header are as specified in VVC and described briefly below for convenience. In addition to the name and size of each field, the corresponding syntax element name in VVC is also provided.

F: 1 bit

forbidden_zero_bit. Required to be zero in VVC. Note that the inclusion of this bit in the NAL unit header was to enable transport of VVC video over MPEG-2 transport systems (avoidance of start code emulations) [MPEG2S]. In the context of this memo the value 1 may be used to indicate a syntax violation, e.g., for a NAL unit resulted from aggregating a number of fragmented units of a NAL unit but missing the last fragment, as described in Section TBD.

Z: 1 bit

nuh_reserved_zero_bit. Required to be zero in VVC, and reserved for future extensions by ITU-T and ISO/IEC.

This memo does not overload the "Z" bit for local extensions, as a) overloading the "F" bit is sufficient and b) to preserve the usefulness of this memo to possible future versions of [VVC].

LayerId: 6 bits

nuh_layer_id. Identifies the layer a NAL unit belongs to, wherein a layer may be, e.g., a spatial scalable layer, a quality scalable layer .

Type: 5 bits

nal_unit_type. This field specifies the NAL unit type as defined in Table 7-1 of VVC. For a reference of all currently defined NAL unit types and their semantics, please refer to Section 7.4.2.2 in VVC.

TID: 3 bits

nuh_temporal_id_plus1. This field specifies the temporal identifier of the NAL unit plus 1. The value of TemporalId is equal to TID minus 1. A TID value of 0 is illegal to ensure that there is at least one bit in the NAL unit header equal to 1, so to enable independent considerations of start code emulations in the NAL unit header and in the NAL unit payload data.

1.2. Overview of the Payload Format

This payload format defines the following processes required for transport of VVC coded data over RTP [RFC3550]:

- * Usage of RTP header with this payload format
- * Packetization of VVC coded NAL units into RTP packets using three types of payload structures: a single NAL unit packet, aggregation packet, and fragment unit
- * Transmission of VVC NAL units of the same bitstream within a single RTP stream.
- * Media type parameters to be used with the Session Description Protocol (SDP) [RFC4566]
- * Frame-marking mapping [FrameMarking]

2. Conventions

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 BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown above.

3. Definitions and Abbreviations

3.1. Definitions

This document uses the terms and definitions of VVC. Section 3.1.1 lists relevant definitions from [VVC] for convenience. Section 3.1.2 provides definitions specific to this memo.

3.1.1. Definitions from the VVC Specification

Access unit (AU): A set of PUs that belong to different layers and contain coded pictures associated with the same time for output from the DPB.

Adaptation parameter set (APS): A syntax structure containing syntax elements that apply to zero or more slices as determined by zero or more syntax elements found in slice headers.

Bitstream: A sequence of bits, in the form of a NAL unit stream or a byte stream, that forms the representation of a sequence of AUs forming one or more coded video sequences (CVSs).

Coded picture: A coded representation of a picture comprising VCL NAL units with a particular value of `nuh_layer_id` within an AU and containing all CTUs of the picture.

Clean random access (CRA) PU: A PU in which the coded picture is a CRA picture.

Clean random access (CRA) picture: An IRAP picture for which each VCL NAL unit has `nal_unit_type` equal to `CRA_NUT`.

Coded video sequence (CVS): A sequence of AUs that consists, in decoding order, of a CVSS AU, followed by zero or more AUs that are not CVSS AUs, including all subsequent AUs up to but not including any subsequent AU that is a CVSS AU.

Coded video sequence start (CVSS) AU: An AU in which there is a PU for each layer in the CVS and the coded picture in each PU is a CLVSS picture.

Coded layer video sequence (CLVS): A sequence of PUs with the same value of `nuh_layer_id` that consists, in decoding order, of a CLVSS PU, followed by zero or more PUs that are not CLVSS PUs, including all subsequent PUs up to but not including any subsequent PU that is a CLVSS PU.

Coded layer video sequence start (CLVSS) PU: A PU in which the coded picture is a CLVSS picture.

Coded layer video sequence start (CLVSS) picture: A coded picture that is an IRAP picture with `NoOutputBeforeRecoveryFlag` equal to 1 or a GDR picture with `NoOutputBeforeRecoveryFlag` equal to 1.

Coding tree unit (CTU): A CTB of luma samples, two corresponding CTBs of chroma samples of a picture that has three sample arrays, or a CTB of samples of a monochrome picture or a picture that is coded using three separate colour planes and syntax structures used to code the samples.

Decoding Capability Information (DCI): A syntax structure containing syntax elements that apply to the entire bitstream.

Decoded picture buffer (DPB): A buffer holding decoded pictures for reference, output reordering, or output delay specified for the hypothetical reference decoder.

Gradual decoding refresh (GDR) picture: A picture for which each VCL NAL unit has `nal_unit_type` equal to `GDR_NUT`.

Instantaneous decoding refresh (IDR) PU: A PU in which the coded picture is an IDR picture.

Instantaneous decoding refresh (IDR) picture: An IRAP picture for which each VCL NAL unit has `nal_unit_type` equal to `IDR_W_RADL` or `IDR_N_LP`.

Intra random access point (IRAP) AU: An AU in which there is a PU for each layer in the CVS and the coded picture in each PU is an IRAP picture.

Intra random access point (IRAP) PU: A PU in which the coded picture is an IRAP picture.

Intra random access point (IRAP) picture: A coded picture for which all VCL NAL units have the same value of `nal_unit_type` in the range of `IDR_W_RADL` to `CRA_NUT`, inclusive.

Layer: A set of VCL NAL units that all have a particular value of `nuh_layer_id` and the associated non-VCL NAL units.

Network abstraction layer (NAL) unit: A syntax structure containing an indication of the type of data to follow and bytes containing that data in the form of an RBSP interspersed as necessary with emulation prevention bytes.

Network abstraction layer (NAL) unit stream: A sequence of NAL units.

Operation point (OP): A temporal subset of an OLS, identified by an OLS index and a highest value of `TemporalId`.

Picture parameter set (PPS): A syntax structure containing syntax elements that apply to zero or more entire coded pictures as determined by a syntax element found in each slice header.

Picture unit (PU): A set of NAL units that are associated with each other according to a specified classification rule, are consecutive in decoding order, and contain exactly one coded picture.

Random access: The act of starting the decoding process for a bitstream at a point other than the beginning of the stream.

Sequence parameter set (SPS): A syntax structure containing syntax elements that apply to zero or more entire CLVSs as determined by the content of a syntax element found in the PPS referred to by a syntax element found in each picture header.

Slice: An integer number of complete tiles or an integer number of consecutive complete CTU rows within a tile of a picture that are exclusively contained in a single NAL unit.

Slice header (SH): A part of a coded slice containing the data elements pertaining to all tiles or CTU rows within a tile represented in the slice.

Sublayer: A temporal scalable layer of a temporal scalable bitstream consisting of VCL NAL units with a particular value of the TemporalId variable, and the associated non-VCL NAL units.

Subpicture: A rectangular region of one or more slices within a picture.

Sublayer representation: A subset of the bitstream consisting of NAL units of a particular sublayer and the lower sublayers.

Tile: A rectangular region of CTUs within a particular tile column and a particular tile row in a picture.

Tile column: A rectangular region of CTUs having a height equal to the height of the picture and a width specified by syntax elements in the picture parameter set.

Tile row: A rectangular region of CTUs having a height specified by syntax elements in the picture parameter set and a width equal to the width of the picture.

Video coding layer (VCL) NAL unit: A collective term for coded slice NAL units and the subset of NAL units that have reserved values of nal_unit_type that are classified as VCL NAL units in this Specification.

3.1.2. Definitions Specific to This Memo

Media-Aware Network Element (MANE): A network element, such as a middlebox, selective forwarding unit, or application-layer gateway that is capable of parsing certain aspects of the RTP payload headers or the RTP payload and reacting to their contents.

Editor Notes: the following informative needs to be updated along with frame marking update

Informative note: The concept of a MANE goes beyond normal routers or gateways in that a MANE has to be aware of the signaling (e.g., to learn about the payload type mappings of the media streams), and in that it has to be trusted when working with Secure RTP

(SRTP). The advantage of using MANEs is that they allow packets to be dropped according to the needs of the media coding. For example, if a MANE has to drop packets due to congestion on a certain link, it can identify and remove those packets whose elimination produces the least adverse effect on the user experience. After dropping packets, MANEs must rewrite RTCP packets to match the changes to the RTP stream, as specified in Section 7 of [RFC3550].

NAL unit decoding order: A NAL unit order that conforms to the constraints on NAL unit order given in Section 7.4.2.4 in [VVC], follow the Order of NAL units in the bitstream.

NAL unit output order: A NAL unit order in which NAL units of different access units are in the output order of the decoded pictures corresponding to the access units, as specified in [VVC], and in which NAL units within an access unit are in their decoding order.

RTP stream: See [RFC7656]. Within the scope of this memo, one RTP stream is utilized to transport one or more temporal sublayers.

Transmission order: The order of packets in ascending RTP sequence number order (in modulo arithmetic). Within an aggregation packet, the NAL unit transmission order is the same as the order of appearance of NAL units in the packet.

3.2. Abbreviations

AU	Access Unit
AP	Aggregation Packet
CTU	Coding Tree Unit
CVS	Coded Video Sequence
DPB	Decoded Picture Buffer
DCI	Decoding capability information
DON	Decoding Order Number
FIR	Full Intra Request
FU	Fragmentation Unit
HRD	Hypothetical Reference Decoder

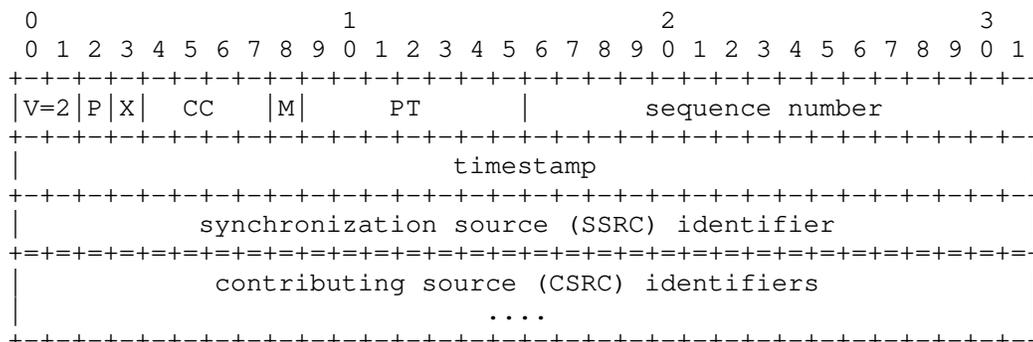
IDR	Instantaneous Decoding Refresh
MANE	Media-Aware Network Element
MTU	Maximum Transfer Unit
NAL	Network Abstraction Layer
NALU	Network Abstraction Layer Unit
PLI	Picture Loss Indication
PPS	Picture Parameter Set
RPS	Reference Picture Set
RPSI	Reference Picture Selection Indication
SEI	Supplemental Enhancement Information
SLI	Slice Loss Indication
SPS	Sequence Parameter Set
VCL	Video Coding Layer
VPS	Video Parameter Set

4. RTP Payload Format

4.1. RTP Header Usage

The format of the RTP header is specified in [RFC3550] (reprinted as Figure 2 for convenience). This payload format uses the fields of the header in a manner consistent with that specification.

The RTP payload (and the settings for some RTP header bits) for aggregation packets and fragmentation units are specified in Section 4.3.2 and Section 4.3.3, respectively.



RTP Header According to {{RFC3550}}

Figure 2

The RTP header information to be set according to this RTP payload format is set as follows:

Marker bit (M): 1 bit

Set for the last packet of the access unit, carried in the current RTP stream. This is in line with the normal use of the M bit in video formats to allow an efficient playout buffer handling.

Editor notes: The informative note below needs updating once the NAL unit type table is stable in the [VVC] spec.

Informative note: The content of a NAL unit does not tell whether or not the NAL unit is the last NAL unit, in decoding order, of an access unit. An RTP sender implementation may obtain this information from the video encoder. If, however, the implementation cannot obtain this information directly from the encoder, e.g., when the bitstream was pre-encoded, and also there is no timestamp allocated for each NAL unit, then the sender implementation can inspect subsequent NAL units in decoding order to determine whether or not the NAL unit is the last NAL unit of an access unit as follows. A NAL unit is determined to be the last NAL unit of an access unit if it is the last NAL unit of the bitstream. A NAL unit nal_uX is also determined to be the last NAL unit of an access unit if both the following conditions are true: 1) the next VCL NAL unit nal_uY in decoding order has the high-order bit of the first byte after its NAL unit header equal to 1 or nal_unit_type equal to 19, and 2) all NAL units between nal_uX and nal_uY, when present, have nal_unit_type in the range of 13 to 17, inclusive, equal to 20, equal to 23 or equal to 26.

Payload Type (PT): 7 bits

The assignment of an RTP payload type for this new packet format is outside the scope of this document and will not be specified here. The assignment of a payload type has to be performed either through the profile used or in a dynamic way.

Sequence Number (SN): 16 bits

Set and used in accordance with [RFC3550].

Timestamp: 32 bits

The RTP timestamp is set to the sampling timestamp of the content. A 90 kHz clock rate **MUST** be used. If the NAL unit has no timing properties of its own (e.g., parameter set and SEI NAL units), the RTP timestamp **MUST** be set to the RTP timestamp of the coded picture of the access unit in which the NAL unit (according to Annex D of VVC) is included. Receivers **MUST** use the RTP timestamp for the display process, even when the bitstream contains picture timing SEI messages or decoding unit information SEI messages as specified in VVC.

Synchronization source (SSRC): 32 bits

Used to identify the source of the RTP packets. A single SSRC is used for all parts of a single bitstream.

4.2. Payload Header Usage

The first two bytes of the payload of an RTP packet are referred to as the payload header. The payload header consists of the same fields (F, Z, LayerId, Type, and TID) as the NAL unit header as shown in Section 1.1.4, irrespective of the type of the payload structure.

The TID value indicates (among other things) the relative importance of an RTP packet, for example, because NAL units belonging to higher temporal sublayers are not used for the decoding of lower temporal sublayers. A lower value of TID indicates a higher importance. More-important NAL units **MAY** be better protected against transmission losses than less-important NAL units.

For Discussion: quite possibly something similar can be said for the Layer_id in layered coding, but perhaps not in multiview coding. (The relevant part of the spec is relatively new, therefore the soft language). However, for serious layer pruning, interpretation of the VPS is required. We can add language about the need for stateful interpretation of LayerID vis-a-vis stateless interpretation of TID later.

4.3. Payload Structures

Three different types of RTP packet payload structures are specified. A receiver can identify the type of an RTP packet payload through the Type field in the payload header.

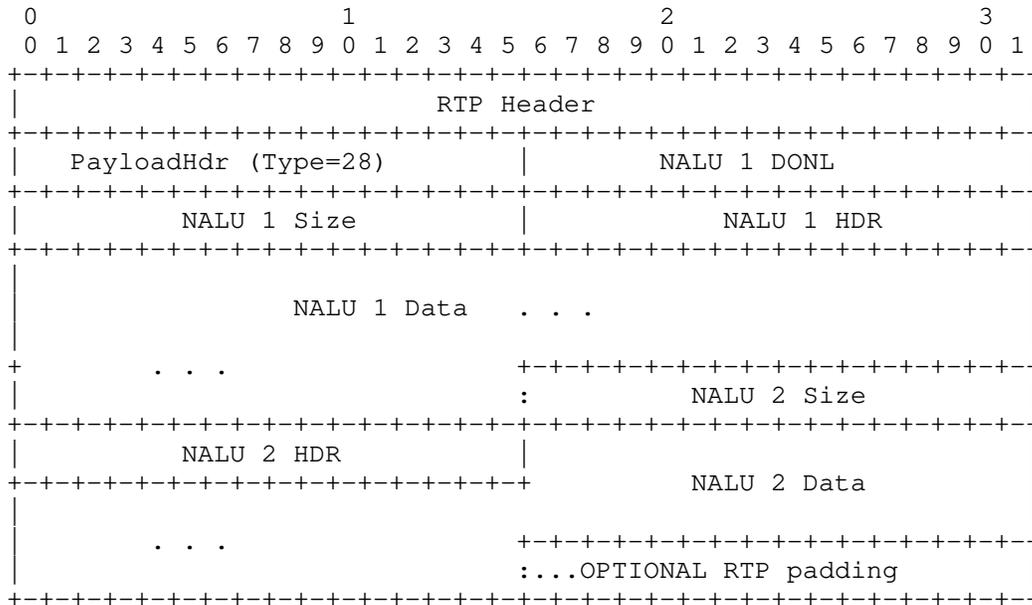
The three different payload structures are as follows:

- * Single NAL unit packet: Contains a single NAL unit in the payload, and the NAL unit header of the NAL unit also serves as the payload header. This payload structure is specified in Section 4.4.1.
- * Aggregation Packet (AP): Contains more than one NAL unit within one access unit. This payload structure is specified in Section 4.3.2.
- * Fragmentation Unit (FU): Contains a subset of a single NAL unit. This payload structure is specified in Section 4.3.3.

4.3.1. Single NAL Unit Packets

Editor notes: its better to add a section to describe DONL and sprop-max_don_diff. sprop-max_don_diff is used but not specified as parameters in section 7 are not yet specified. A value of sprop-max_don_diff greater than 0 indicates that the transmission order may not correspond to the decoding order and that the DON is included in the payload header.

A single NAL unit packet contains exactly one NAL unit, and consists of a payload header (denoted as PayloadHdr), a conditional 16-bit DONL field (in network byte order), and the NAL unit payload data (the NAL unit excluding its NAL unit header) of the contained NAL unit, as shown in Figure 3.



An Example of an AP Containing
Two Aggregation Units with the DONL Field

Figure 8

4.3.3. Fragmentation Units

Fragmentation Units (FUs) are introduced to enable fragmenting a single NAL unit into multiple RTP packets, possibly without cooperation or knowledge of the [VVC] encoder. A fragment of a NAL unit consists of an integer number of consecutive octets of that NAL unit. Fragments of the same NAL unit MUST be sent in consecutive order with ascending RTP sequence numbers (with no other RTP packets within the same RTP stream being sent between the first and last fragment).

When a NAL unit is fragmented and conveyed within FUs, it is referred to as a fragmented NAL unit. APs MUST NOT be fragmented. FUs MUST NOT be nested; i.e., an FU can not contain a subset of another FU.

The RTP timestamp of an RTP packet carrying an FU is set to the NALU-time of the fragmented NAL unit.

An FU consists of a payload header (denoted as PayloadHdr), an FU header of one octet, a conditional 16-bit DONL field (in network byte order), and an FU payload, as shown in Figure 9.

When set to 1, the E bit indicates the end of a fragmented NAL unit, i.e., the last byte of the payload is also the last byte of the fragmented NAL unit. When the FU payload is not the last fragment of a fragmented NAL unit, the E bit MUST be set to 0.

Reserved: 1 bit

Placeholder

FuType: 5 bits

The field FuType MUST be equal to the field Type of the fragmented NAL unit.

The DONL field, when present, specifies the value of the 16 least significant bits of the decoding order number of the fragmented NAL unit.

If sprop-max-don-diff is greater than 0, and the S bit is equal to 1, the DONL field MUST be present in the FU, and the variable DON for the fragmented NAL unit is derived as equal to the value of the DONL field. Otherwise (sprop-max-don-diff is equal to 0, or the S bit is equal to 0), the DONL field MUST NOT be present in the FU.

A non-fragmented NAL unit MUST NOT be transmitted in one FU; i.e., the Start bit and End bit must not both be set to 1 in the same FU header.

The FU payload consists of fragments of the payload of the fragmented NAL unit so that if the FU payloads of consecutive FUs, starting with an FU with the S bit equal to 1 and ending with an FU with the E bit equal to 1, are sequentially concatenated, the payload of the fragmented NAL unit can be reconstructed. The NAL unit header of the fragmented NAL unit is not included as such in the FU payload, but rather the information of the NAL unit header of the fragmented NAL unit is conveyed in F, LayerId, and TID fields of the FU payload headers of the FUs and the FuType field of the FU header of the FUs. An FU payload MUST NOT be empty.

If an FU is lost, the receiver SHOULD discard all following fragmentation units in transmission order corresponding to the same fragmented NAL unit, unless the decoder in the receiver is known to be prepared to gracefully handle incomplete NAL units.

A receiver in an endpoint or in a MANE MAY aggregate the first $n-1$ fragments of a NAL unit to an (incomplete) NAL unit, even if fragment n of that NAL unit is not received. In this case, the `forbidden_zero_bit` of the NAL unit MUST be set to 1 to indicate a syntax violation.

4.4. Decoding Order Number

For each NAL unit, the variable `AbsDon` is derived, representing the decoding order number that is indicative of the NAL unit decoding order.

Let NAL unit n be the n -th NAL unit in transmission order within an RTP stream.

If `sprop-max-don-diff` is equal to 0, `AbsDon[n]`, the value of `AbsDon` for NAL unit n , is derived as equal to n .

Otherwise (`sprop-max-don-diff` is greater than 0), `AbsDon[n]` is derived as follows, where `DON[n]` is the value of the variable `DON` for NAL unit n :

* If n is equal to 0 (i.e., NAL unit n is the very first NAL unit in transmission order), `AbsDon[0]` is set equal to `DON[0]`.

* Otherwise (n is greater than 0), the following applies for derivation of `AbsDon[n]`:

If `DON[n] == DON[n-1]`,
`AbsDon[n] = AbsDon[n-1]`

If (`DON[n] > DON[n-1]` and `DON[n] - DON[n-1] < 32768`),
`AbsDon[n] = AbsDon[n-1] + DON[n] - DON[n-1]`

If (`DON[n] < DON[n-1]` and `DON[n-1] - DON[n] >= 32768`),
`AbsDon[n] = AbsDon[n-1] + 65536 - DON[n-1] + DON[n]`

If (`DON[n] > DON[n-1]` and `DON[n] - DON[n-1] >= 32768`),
`AbsDon[n] = AbsDon[n-1] - (DON[n-1] + 65536 - DON[n])`

If (`DON[n] < DON[n-1]` and `DON[n-1] - DON[n] < 32768`),
`AbsDon[n] = AbsDon[n-1] - (DON[n-1] - DON[n])`

For any two NAL units m and n , the following applies:

* `AbsDon[n]` greater than `AbsDon[m]` indicates that NAL unit n follows NAL unit m in NAL unit decoding order.

- * When AbsDon[n] is equal to AbsDon[m], the NAL unit decoding order of the two NAL units can be in either order.
- * AbsDon[n] less than AbsDon[m] indicates that NAL unit n precedes NAL unit m in decoding order.

Informative note: When two consecutive NAL units in the NAL unit decoding order have different values of AbsDon, the absolute difference between the two AbsDon values may be greater than or equal to 1.

Informative note: There are multiple reasons to allow for the absolute difference of the values of AbsDon for two consecutive NAL units in the NAL unit decoding order to be greater than one. An increment by one is not required, as at the time of associating values of AbsDon to NAL units, it may not be known whether all NAL units are to be delivered to the receiver. For example, a gateway might not forward VCL NAL units of higher sublayers or some SEI NAL units when there is congestion in the network.

In another example, the first intra-coded picture of a pre-encoded clip is transmitted in advance to ensure that it is readily available in the receiver, and when transmitting the first intra-coded picture, the originator does not exactly know how many NAL units will be encoded before the first intra-coded picture of the pre-encoded clip follows in decoding order. Thus, the values of AbsDon for the NAL units of the first intra-coded picture of the pre-encoded clip have to be estimated when they are transmitted, and gaps in values of AbsDon may occur.

5. Packetization Rules

The following packetization rules apply:

- * If sprop-max-don-diff is greater than 0, the transmission order of NAL units carried in the RTP stream MAY be different than the NAL unit decoding order and the NAL unit output order.
- * A NAL unit of a small size SHOULD be encapsulated in an aggregation packet together one or more other NAL units in order to avoid the unnecessary packetization overhead for small NAL units. For example, non-VCL NAL units such as access unit delimiters, parameter sets, or SEI NAL units are typically small and can often be aggregated with VCL NAL units without violating MTU size constraints.

- * Each non-VCL NAL unit SHOULD, when possible from an MTU size match viewpoint, be encapsulated in an aggregation packet together with its associated VCL NAL unit, as typically a non-VCL NAL unit would be meaningless without the associated VCL NAL unit being available.
- * For carrying exactly one NAL unit in an RTP packet, a single NAL unit packet MUST be used.

6. De-packetization Process

The general concept behind de-packetization is to get the NAL units out of the RTP packets in an RTP stream and pass them to the decoder in the NAL unit decoding order.

The de-packetization process is implementation dependent. Therefore, the following description should be seen as an example of a suitable implementation. Other schemes may be used as well, as long as the output for the same input is the same as the process described below. The output is the same when the set of output NAL units and their order are both identical. Optimizations relative to the described algorithms are possible.

All normal RTP mechanisms related to buffer management apply. In particular, duplicated or outdated RTP packets (as indicated by the RTP sequence number and the RTP timestamp) are removed. To determine the exact time for decoding, factors such as a possible intentional delay to allow for proper inter-stream synchronization MUST be factored in.

NAL units with NAL unit type values in the range of 0 to 27, inclusive, may be passed to the decoder. NAL-unit-like structures with NAL unit type values in the range of 28 to 31, inclusive, MUST NOT be passed to the decoder.

The receiver includes a receiver buffer, which is used to compensate for transmission delay jitter within individual RTP streams and across RTP streams, to reorder NAL units from transmission order to the NAL unit decoding order. In this section, the receiver operation is described under the assumption that there is no transmission delay jitter within an RTP stream and across RTP streams. To make a difference from a practical receiver buffer that is also used for compensation of transmission delay jitter, the receiver buffer is hereafter called the de-packetization buffer in this section. Receivers should also prepare for transmission delay jitter; that is, either reserve separate buffers for transmission delay jitter buffering and de-packetization buffering or use a receiver buffer for both transmission delay jitter and de-packetization. Moreover,

receivers should take transmission delay jitter into account in the buffering operation, e.g., by additional initial buffering before starting of decoding and playback.

When `sprop-max-don-diff` is equal to 0, the de-packetization buffer size is zero bytes, and the process described in the remainder of this paragraph applies. The NAL units carried in the single RTP stream are directly passed to the decoder in their transmission order, which is identical to their decoding order. When there are several NAL units of the same RTP stream with the same NTP timestamp, the order to pass them to the decoder is their transmission order.

Informative note: The mapping between RTP and NTP timestamps is conveyed in RTCP SR packets. In addition, the mechanisms for faster media timestamp synchronization discussed in [RFC6051] may be used to speed up the acquisition of the RTP-to-wall-clock mapping.

When `sprop-max-don-diff` is greater than 0, the process described in the remainder of this section applies.

There are two buffering states in the receiver: initial buffering and buffering while playing. Initial buffering starts when the reception is initialized. After initial buffering, decoding and playback are started, and the buffering-while-playing mode is used.

Regardless of the buffering state, the receiver stores incoming NAL units, in reception order, into the de-packetization buffer. NAL units carried in RTP packets are stored in the de-packetization buffer individually, and the value of `AbsDon` is calculated and stored for each NAL unit.

Initial buffering lasts until condition A (the difference between the greatest and smallest `AbsDon` values of the NAL units in the de-packetization buffer is greater than or equal to the value of `sprop-max-don-diff`) or condition B (the number of NAL units in the de-packetization buffer is greater than the value of `sprop-depack-buf-nalus`) is true.

After initial buffering, whenever condition A or condition B is true, the following operation is repeatedly applied until both condition A and condition B become false:

- * The NAL unit in the de-packetization buffer with the smallest value of `AbsDon` is removed from the de-packetization buffer and passed to the decoder.

When no more NAL units are flowing into the de-packetization buffer, all NAL units remaining in the de-packetization buffer are removed from the buffer and passed to the decoder in the order of increasing AbsDon values.

7. Payload Format Parameters

This section specifies the optional parameters. A mapping of the parameters with Session Description Protocol (SDP) [RFC4556] is also provided for applications that use SDP.

7.1. Media Type Registration

The receiver MUST ignore any parameter unspecified in this memo.

Type name: Video

Subtype name: H266

Required parameters: none

Optional parameters:

Editor's notes: To be added

7.2. SDP Parameters

The receiver MUST ignore any parameter unspecified in this memo.

7.2.1. Mapping of Payload Type Parameters to SDP

The media type video/H266 string is mapped to fields in the Session Description Protocol (SDP) [RFC4566] as follows:

- * The media name in the "m=" line of SDP MUST be video.
- * The encoding name in the "a=rtpmap" line of SDP MUST be H266 (the media subtype).
- * The clock rate in the "a=rtpmap" line MUST be 90000.
- * OPTIONAL PARAMETERS:

Editor's notes: To be dicussed here

profile-id, tier-flag, sub-profile-id, interop-constraints, and level-id:

These parameters indicate the profile, tier, default level, sub-profile, and some constraints of the bitstream carried by the RTP stream, or a specific set of the profile, tier, default level, sub-profile and some constraints the receiver supports.

The subset of coding tools that may have been used to generate the bitstream or that the receiver supports, as well as, some additional constraints are indicated collectively by profile-id, sub-profile-id, and interop-constraints.

Informative note: There are 128 values of profile-id. The subset of coding tools identified by the profile-id can be further constrained with up to 255 sub-profile-ids. In addition, 68 bits included in interop-constraints, which can be extended up to 324 bits provide means to further restrict tools from existing profiles. To be able to support this fine-granular signalling of coding tool subsets with profile-id, sub-profile-id and interop-constraints, it would be safe to require symmetric use of these parameters in SDP offer/answer unless recv-ols-id or sprop-opi is included in the SDP answer for choosing one of the layers offered.

Editor's notes: confirm when decided whether we use recv-ols-id or sprop-opi

The tier is indicated by tier-flag. The default level is indicated by level-id. The tier and the default level specify the limits on values of syntax elements or arithmetic combinations of values of syntax elements that are followed when generating the bitstream or that the receiver supports.

In SDP offer/answer, when the SDP answer does not include either the recv-ols-id parameter that is less than the sprop-ols-id parameter in the SDP offer or the sprop-opi, the following applies:

Editor's notes: confirm when decided whether we use recv-ols-id or sprop-opi for profile asymmetry - sub-layers cannot

- o The tier-flag, profile-id, sub-profile-id, and interop-constraints parameters MUST be used symmetrically, i.e., the value of each of these parameters in the offer MUST be the same as that in the answer, either explicitly signaled or implicitly inferred.
- o The level-id parameter is changeable as long as the highest level indicated by the answer is either equal to or lower than that in the offer. Note that the highest level is

indicated by level-id and max-recv-level-id together and a higher level than that in the offer can be included as max-recv-level-id.

In SDP offer/answer, when the SDP answer does include the recv-ols-id parameter that is less than the sprop-ols-id parameter in the SDP offer or includes the sprop-opi, the set of tier-flag, profile-id, sub-profile-id, interop-constraints, and level-id parameters included in the answer MUST be consistent with that for the chosen output layer set as indicated in the SDP offer, with the exception that the level-id parameter in the SDP answer is changeable as long as the highest level indicated by the answer is either lower than or equal to that in the offer.

Editor's notes: confirm when decided whether we use recv-ols-id or sprop-opi for profile asymmetry - sub-layers cannot. The consistency of profiles is not yet in the text. I think this parts needs a bit of discussion

More specifications of these parameters, including how they relate syntax elements specified in [VVC] are provided below.

profile-id:

When profile-id is not present, a value of 1 (i.e., the Main 10 profile) MUST be inferred.

When used to indicate properties of a bitstream, profile-id is derived from the general_profile_idc syntax element in the profile_tier_level() syntax structure in SPS, VPS or DCI NAL units as specified in [VVC]. When a VPS contains several profile_tier_level() syntax structures, the syntax structure corresponding to the OLS to which the bitstream applies is used.

Editor's notes: What if the DCI contains several profile_tier_level() syntax structures and they are not onion shell?

tier-flag, level-id:

The value of tier-flag MUST be in the range of 0 to 1, inclusive. The value of level-id MUST be in the range of 0 to 255, inclusive.

If the tier-flag and level-id parameters are used to indicate properties of a bitstream, they indicate the tier and the highest level the bitstream complies with.

If the tier-flag and level-id parameters are used for capability exchange, the following applies. If max-recv-level-id is not present, the default level defined by level-id indicates the highest level the codec wishes to support. Otherwise, max-recv-level-id indicates the highest level the codec supports for receiving. For either receiving or sending, all levels that are lower than the highest level supported MUST also be supported.

If no tier-flag is present, a value of 0 MUST be inferred; if no level-id is present, a value of 51 (i.e., level 3.1) MUST be inferred.

Informative note: The level numbers currently defined in the VVC specification are in the form of "majorNum.minorNum", and the value of the level-id for each of the levels is equal to $\text{majorNum} * 16 + \text{minorNum} * 3$. It is expected that if any level are defined in the future, the same convention will be used, but this cannot be guaranteed.

Editor's notes: double check this informative note

When used to indicate properties of a bitstream, the tier-flag and level-id parameters are derived from the profile_tier_level() syntax structure in SPS, VPS or DCI NAL units as specified in [VVC] as follows.

If the tier-flag and level-id are derived from the profile_tier_level() syntax structure in the DCI NAL unit, the following applies:

- o tier-flag = general_tier_flag
- o level-id = general_level_idc

Otherwise, if the tier-flag and level-id are derived from the profile_tier_level() syntax structure in the SPS or VPS NAL unit, and the bitstream contains the highest sub-layer representation in the OLS corresponding to the bitstream, the following applies:

- o tier-flag = general_tier_flag
- o level-id = general_level_idc

Otherwise, if the tier-flag and level-id are derived from the profile_tier_level() syntax structure in the SPS or VPS NAL unit, and the bitstream does not contains the highest sub-layer

representation in the OLS corresponding to the bitstream, the following applies, with *j* being the value of the `sprop-sub-layer-id` parameter or the sub-layer representation indicated in the `sprop-opi` parameter:

- o `tier-flag = general_tier_flag`
- o `level-id = sub_layer_level_idc[j]`

Editor's notes: double check this part above inherited from HEVC. What if more than one SPS, VPS and they have different `general_level_idcs` or `tier_flags`? We would say it applies to all of them, i.e. to the highest one.

`sub-profile-id`:

The value of the parameter is a comma-separated (',') list of values.

Editor's notes: What is the value? integer, base32?

When used to indicate properties of a bitstream, `sub-profile-id` is derived from each of the `ptl_num_sub_profiles` `general_sub_profile_idc[i]` syntax elements in the `profile_tier_level()` syntax structure in SPS, VPS or DCI NAL units as specified in [VVC]. When a VPS contains several `profile_tier_level()` syntax structures, the syntax structure corresponding to the OLS to which the bitstream applies is used.

Editor's notes: What if the DCI contains several `profile_tier_level()` syntax structures and they are not onion shell?

`interop-constraints`:

A base16 [RFC4648] (hexadecimal) representation of the data in the `profile_tier_level()` syntax structure in SPS, VPS or DCI NAL units as specified in [VVC], that include the syntax elements `ptl_frame_only_constraint_flag` and `ptl_multilayer_enabled_flag` and, when present, the `general_constraints_info()` syntax structure. When a VPS contains several `profile_tier_level()` syntax structures, the syntax structure corresponding to the OLS to which the bitstream applies is used.

Editor's notes: What if the DCI contains several `profile_tier_level()` syntax structures and they are not equal?

If the `interop-constraints` parameter is not present, the following MUST be inferred:

- o `ptl_frame_only_constraint_flag` = 0
- o `ptl_multilayer_enabled_flag` = 1
- o `gci_present_flag` in the `general_constraints_info()` syntax structure = 1

Editor's notes: Double check the default values. Currently, no constraints, but actually, with the Main 10 profile as default multi-layer not possible.

Using `interop-constraints` for capability exchange results in a requirement on any bitstream to be compliant with the `interop-constraints`.

`sprop-sub-layer-id`:

This parameter MAY be used to indicate the highest allowed value of TID in the bitstream. When not present, the value of `sprop-sub-layer-id` is inferred to be equal to 6.

The value of `sprop-sub-layer-id` MUST be in the range of 0 to 6, inclusive.

`sprop-ols-id`:

This parameter MAY be used to indicate the OLS that the bitstream applies to. When not present, the value of `sprop-ols-id` is inferred to be equal to `TargetOlsIdx` as specified in 8.1.1 in [VVC].

The value of `sprop-ols-id` MUST be in the range of 0 to 257, inclusive.

Editor's notes: Confirm this value

`recv-sub-layer-id`:

This parameter MAY be used to signal a receiver's choice of the offered or declared sub-layer representations in the `sprop-vps` and `sprop-sps`. The value of `recv-sub-layer-id` indicates the TID of the highest sub-layer of the bitstream that a receiver supports. When not present, the value of `recv-sub-layer-id` is inferred to be equal to the value of the `sprop-sub-layer-id` parameter in the SDP offer.

The value of `recv-sub-layer-id` MUST be in the range of 0 to 6, inclusive.

`recv-ols-id`:

This parameter MAY be used to signal a receiver's choice of the offered or declared output layer sets in the `sprop-vps`. The value of `recv-ols-id` indicates the OLS index of the bitstream that a receiver supports. When not present, the value of `recv-ols-id` is inferred to be equal to the value of the `sprop-ols-id` parameter in the SDP offer.

The value of `recv-ols-id` MUST be in the range of 0 to 257, inclusive.

Editor's notes: Confirm this value

`max-recv-level-id`:

This parameter MAY be used to indicate the highest level a receiver supports.

The value of `max-recv-level-id` MUST be in the range of 0 to 255, inclusive.

When `max-recv-level-id` is not present, the value is inferred to be equal to `level-id`.

`max-recv-level-id` MUST NOT be present when the highest level the receiver supports is not higher than the default level.

`sprop-dci`:

This parameter MAY be used to convey a decoding capability information NAL unit of the bitstream for out-of-band transmission. The parameter MAY also be used for capability exchange. The value of the parameter a base64 [RFC4648] representations of the decoding capability information NAL unit as specified in Section 7.3.2.1 of [VVC].

`sprop-vps`:

This parameter MAY be used to convey any video parameter set NAL unit of the bitstream for out-of-band transmission of video parameter sets. The parameter MAY also be used for capability exchange and to indicate sub-stream characteristics (i.e., properties of output layer sets and sublayer representations as defined in [VVC]). The value of the parameter is a comma-

separated (',') list of base64 [RFC4648] representations of the video parameter set NAL units as specified in Section 7.3.2.3 of [VVC].

The sprop-vps parameter MAY contain one or more than one video parameter set NAL unit. However, all other video parameter sets contained in the sprop-vps parameter MUST be consistent with the first video parameter set in the sprop-vps parameter. A video parameter set vpsB is said to be consistent with another video parameter set vpsA if any decoder that conforms to the profile, tier, level, and constraints indicated by the 12 bytes of data starting from the syntax element general_profile_space to the syntax element general_level_idc, inclusive, in the first profile_tier_level() syntax structure in vpsA can decode any bitstream that conforms to the profile, tier, level, and constraints indicated by the 12 bytes of data starting from the syntax element general_profile_space to the syntax element general_level_idc, inclusive, in the first profile_tier_level() syntax structure in vpsB.

sprop-sei:

This parameter MAY be used to convey one or more SEI messages that describe bitstream characteristics. When present, a decoder can rely on the bitstream characteristics that are described in the SEI messages for the entire duration of the session, independently from the persistence scopes of the SEI messages as specified in [VSEI].

The value of the parameter is a comma-separated (',') list of base64 [RFC4648] representations of SEI NAL units as specified in [VSEI].

Informative note: Intentionally, no list of applicable or inapplicable SEI messages is specified here. Conveying certain SEI messages in sprop-sei may be sensible in some application scenarios and meaningless in others. However, a few examples are described below:

1) In an environment where the bitstream was created from film-based source material, and no splicing is going to occur during the lifetime of the session, the film grain characteristics SEI message is likely meaningful, and sending it in sprop-sei rather than in the bitstream at each entry point may help with saving bits and allows one to configure the renderer only once, avoiding unwanted artifacts.

2) Examples for SEI messages that would be meaningless to be conveyed in sprop-sei include the decoded picture hash SEI message (it is close to impossible that all decoded pictures have the same hashtag), the display orientation SEI message when the device is a handheld device (as the display orientation may change when the handheld device is turned around), or the filler payload SEI message (as there is no point in just having more bits in SDP).

sprop-opi:

Editor's notes: VVC does not envision to provide the OPI by external means but this should not be a problem

This parameter MAY be used to convey an operating point information NAL unit of the bitstream for out-of-band transmission. The value of the parameter is a base64 [RFC4648] representations of the operating point information NAL unit as specified in Section 7.3.2.2 of [VVC].

max-lsr, max-lps, max-cpb, max-dpb, max-br, max-tr, max-tc:

These parameters MAY be used to signal the capabilities of a receiver implementation. These parameters MUST NOT be used for any other purpose. The highest level (specified by max-recv-level-id) MUST be the highest that the receiver is fully capable of supporting. max-lsr, max-lps, max-cpb, max-dpb, max-br, max-tr, and max-tc MAY be used to indicate capabilities of the receiver that extend the required capabilities of the highest level, as specified below.

When more than one parameter from the set (max-lsr, max-lps, max-cpb, max-dpb, max-br, max-tr, max-tc) is present, the receiver MUST support all signaled capabilities simultaneously. For example, if both max-lsr and max-br are present, the highest level with the extension of both the picture rate and bitrate is supported. That is, the receiver is able to decode bitstreams in which the luma sample rate is up to max-lsr (inclusive), the bitrate is up to max-br (inclusive), the coded picture buffer size is derived as specified in the semantics of the max-br parameter below, and the other properties comply with the highest level specified by max-recv-level-id.

Informative note: When the OPTIONAL media type parameters are used to signal the properties of a bitstream, and max-lsr, max-lps, max-cpb, max-dpb, max-br, max-tr, and max-tc are not present, the values of tier-flag, profile-id, sub-profile-id, interop-constraints, and level-id must always be such that the bitstream complies fully with the specified profile, tier, and level.

max-lsr:

The value of max-lsr is an integer indicating the maximum processing rate in units of luma samples per second. The max-lsr parameter signals that the receiver is capable of decoding video at a higher rate than is required by the highest level.

When max-lsr is signaled, the receiver MUST be able to decode bitstreams that conform to the highest level, with the exception that the MaxLumaSr value in Table 136 of [VVC] for the highest level is replaced with the value of max-lsr. Senders MAY use this knowledge to send pictures of a given size at a higher picture rate than is indicated in the highest level.

When not present, the value of max-lsr is inferred to be equal to the value of MaxLumaSr given in Table 136 of [VVC] for the highest level.

The value of max-lsr MUST be in the range of MaxLumaSr to $16 * \text{MaxLumaSr}$, inclusive, where MaxLumaSr is given in Table 136 of [VVC] for the highest level.

max-lps:

The value of max-lps is an integer indicating the maximum picture size in units of luma samples. The max-lps parameter signals that the receiver is capable of decoding larger picture sizes than are required by the highest level. When max-lps is signaled, the receiver MUST be able to decode bitstreams that conform to the highest level, with the exception that the MaxLumaPs value in Table 135 of [VVC] for the highest level is replaced with the value of max-lps. Senders MAY use this knowledge to send larger pictures at a proportionally lower picture rate than is possible for the largest picture size for the highest level.

When not present, the value of max-lps is inferred to be equal to the value of MaxLumaPs given in Table 135 of [VVC] for the highest level.

The value of max-lps MUST be in the range of MaxLumaPs to $16 * \text{MaxLumaPs}$, inclusive, where MaxLumaPs is given in Table 135 of [VVC] for the highest level.

max-cpb:

The value of max-cpb is an integer indicating the maximum coded picture buffer size in units of CpbVclFactor bits for the VCL HRD parameters and in units of CpbNalFactor bits for the NAL HRD parameters, where CpbVclFactor and CpbNalFactor are defined in Table 137 of [VVC]. The max-cpb parameter signals that the receiver has more memory than the minimum amount of coded picture buffer memory required by the highest level. When max-cpb is signaled, the receiver MUST be able to decode bitstreams that conform to the highest level, with the exception that the MaxCPB value in Table 135 of [VVC] for the highest level is replaced with the value of max-cpb. Senders MAY use this knowledge to construct coded bitstreams with greater variation of bitrate than can be achieved with the MaxCPB value in Table 135 of [VVC].

When not present, the value of max-cpb is inferred to be equal to the value of MaxCPB given in Table 135 of [VVC] for the highest level.

The value of max-cpb MUST be in the range of MaxCPB to $16 * \text{MaxCPB}$, inclusive, where MaxCPB is given in Table 135 of [VVC] for the highest level.

Informative note: The coded picture buffer is used in the hypothetical reference decoder (Annex C of [VVC]). The use of the hypothetical reference decoder is recommended in VVC encoders to verify that the produced bitstream conforms to the standard and to control the output bitrate. Thus, the coded picture buffer is conceptually independent of any other potential buffers in the receiver, including de-packetization and de-jitter buffers. The coded picture buffer need not be implemented in decoders as specified in Annex C of [VVC], but rather standard-compliant decoders can have any buffering arrangements provided that they can decode standard-compliant bitstreams. Thus, in practice, the input buffer for a video decoder can be integrated with de-packetization and de-jitter buffers of the receiver.

max-dpb:

The value of max-dpb is an integer indicating the maximum decoded picture buffer size in units decoded pictures at the MaxLumaPs for the highest level, i.e., the number of decoded pictures at the maximum picture size defined by the highest level. The value of max-dpb MUST be in the range of 1 to 16, respectively. The max-dpb parameter signals that the receiver has more memory than the minimum amount of decoded picture buffer memory required by default, which is maxDpbPicBuf as defined in [VVC] (equal to 8). When max-dpb is signaled, the receiver MUST be able to decode bitstreams that conform to the highest level, with the exception that the maxDpbPicBuf value defined in [VVC] as 8 is replaced with the value of max-dpb. Consequently, a receiver that signals max-dpb MUST be capable of storing the following number of decoded pictures (MaxDpbSize) in its decoded picture buffer:

```
if( 2 \* PicSizeMaxInSamplesY <= ( MaxLumaPs >> 2 ) )
    MaxDpbSize = 2 \* max-dpb
else if( 3 \* PicSizeMaxInSamplesY <= 2 \* MaxLumaPs )
    MaxDpbSize = 3 \* max-dpb / 2
else
    MaxDpbSize = max-dpb
```

Wherein MaxLumaPs given in Table 135 of [VVC] for the highest level and PicSizeMaxInSamplesY is the maximum allowed picture size in units of luma samples as defined in [VVC].

Editor's notes: I think that max-lps needs to be accounted for here.

The value of max-dpb MUST be greater than or equal to the value of maxDpbPicBuf (i.e., 8) as defined in [VVC]. Senders MAY use this knowledge to construct coded bitstreams with improved compression.

When not present, the value of max-dpb is inferred to be equal to the value of maxDpbPicBuf (i.e., 8) as defined in [VVC].

Informative note: This parameter was added primarily to complement a similar codepoint in the ITU-T Recommendation H.245, so as to facilitate signaling gateway designs. The decoded picture buffer stores reconstructed samples. There is no relationship between the size of the decoded picture buffer and the buffers used in RTP, especially de-packetization and de-jitter buffers.

max-br:

The value of max-br is an integer indicating the maximum video bitrate in units of BrVclFactor bits per second for the VCL HRD parameters and in units of BrNalFactor bits per second for the NAL HRD parameters, where BrVclFactor and BrNalFactor are defined in Section A.4 of [VVC].

The max-br parameter signals that the video decoder of the receiver is capable of decoding video at a higher bitrate than is required by the highest level.

When max-br is signaled, the video codec of the receiver MUST be able to decode bitstreams that conform to the highest level, with the following exceptions in the limits specified by the highest level:

- o The value of max-br replaces the MaxBR value in Table 136 of [VVC] for the highest level.
- o When the max-cpb parameter is not present, the result of the following formula replaces the value of MaxCPB in Table 135 of [VVC]:

$(\text{MaxCPB of the highest level}) * \text{max-br} / (\text{MaxBR of the highest level})$

For example, if a receiver signals capability for Main 10 profile Level 2 with max-br equal to 2000, this indicates a maximum video bitrate of 2000 kbits/sec for VCL HRD parameters, a maximum video bitrate of 2200 kbits/sec for NAL HRD parameters, and a CPB size for VCL HRD of 2000000 bits ($1500000 * 2000000 / 1500000$).

Senders MAY use this knowledge to send higher bitrate video as allowed in the level definition of Annex A of [VVC] to achieve improved video quality.

When not present, the value of max-br is inferred to be equal to the value of MaxBR given in Table 136 of [VVC] for the highest level.

The value of max-br MUST be in the range of MaxBR to $16 * \text{MaxBR}$, inclusive, where MaxBR is given in Table 136 of [VVC] for the highest level.

Informative note: This parameter was added primarily to complement a similar codepoint in the ITU-T Recommendation H.245, so as to facilitate signaling gateway designs. The assumption that the network is capable of handling such

bitrates at any given time cannot be made from the value of this parameter. In particular, no conclusion can be drawn that the signaled bitrate is possible under congestion control constraints.

max-fps:

The value of max-fps is an integer indicating the maximum picture rate in units of pictures per 100 seconds that can be effectively processed by the receiver. The max-fps parameter MAY be used to signal that the receiver has a constraint in that it is not capable of processing video effectively at the full picture rate that is implied by the highest level and, when present, one or more of the parameters max-lsr, max-lps, and max-br.

The value of max-fps is not necessarily the picture rate at which the maximum picture size can be sent, it constitutes a constraint on maximum picture rate for all resolutions.

Informative note: The max-fps parameter is semantically different from max-lsr, max-lps, max-cpb, max-dpb, max-br, max-tr, and max-tc in that max-fps is used to signal a constraint, lowering the maximum picture rate from what is implied by other parameters.

The encoder MUST use a picture rate equal to or less than this value. In cases where the max-fps parameter is absent, the encoder is free to choose any picture rate according to the highest level and any signaled optional parameters.

The value of max-fps MUST be smaller than or equal to the full picture rate that is implied by the highest level and, when present, one or more of the parameters max-lsr, max-lps, and max-br.

sprop-max-don-diff:

If there is no NAL unit naluA that is followed in transmission order by any NAL unit preceding naluA in decoding order (i.e., the transmission order of the NAL units is the same as the decoding order), the value of this parameter MUST be equal to 0.

Otherwise, this parameter specifies the maximum absolute difference between the decoding order number (i.e., AbsDon) values of any two NAL units naluA and naluB, where naluA follows naluB in decoding order and precedes naluB in transmission order.

The value of sprop-max-don-diff MUST be an integer in the range of 0 to 32767, inclusive.

When not present, the value of sprop-max-don-diff is inferred to be equal to 0.

sprop-depack-buf-bytes:

This parameter signals the required size of the de-packetization buffer in units of bytes. The value of the parameter MUST be greater than or equal to the maximum buffer occupancy (in units of bytes) of the de-packetization buffer as specified in Section 6.

The value of sprop-depack-buf-bytes MUST be an integer in the range of 0 to 4294967295, inclusive.

When sprop-max-don-diff is present and greater than 0, this parameter MUST be present and the value MUST be greater than 0. When not present, the value of sprop-depack-buf-bytes is inferred to be equal to 0.

Informative note: The value of sprop-depack-buf-bytes indicates the required size of the de-packetization buffer only. When network jitter can occur, an appropriately sized jitter buffer has to be available as well.

depack-buf-cap:

This parameter signals the capabilities of a receiver implementation and indicates the amount of de-packetization buffer space in units of bytes that the receiver has available for reconstructing the NAL unit decoding order from NAL units carried in the RTP stream. A receiver is able to handle any RTP stream for which the value of the sprop-depack-buf-bytes parameter is smaller than or equal to this parameter.

When not present, the value of depack-buf-cap is inferred to be equal to 4294967295. The value of depack-buf-cap MUST be an integer in the range of 1 to 4294967295, inclusive.

Informative note: `depack-buf-cap` indicates the maximum possible size of the de-packetization buffer of the receiver only, without allowing for network jitter.

Editor's notes: `sprop-depack-buf-nalus` not included but mentioned in section 6 for startup in de-packetization process. We should decide on whether it needs to be included or not.

7.2.1.1. SDP Example

An example of media representation in SDP is as follows:

```
m=video 49170 RTP/AVP 98
a=rtpmap:98 H266/90000
a=fmtp:98 profile-id=1; sprop-vps=<video parameter sets data>
```

7.2.2. Usage with SDP Offer/Answer Model

When [VVC] is offered over RTP using SDP in an offer/answer model [RFC3264] for negotiation for unicast usage, the following limitations and rules apply:

Placeholder: To add limitations and considerations.

8. Use with Feedback Messages

The following subsections define the use of the Picture Loss Indication (PLI), Slice Lost Indication (SLI), Reference Picture Selection Indication (RPSI), and Full Intra Request (FIR) feedback messages with HEVC. The PLI, SLI, and RPSI messages are defined in [RFC4585], and the FIR message is defined in [RFC5104].

8.1. Picture Loss Indication (PLI)

As specified in RFC 4585, Section 6.3.1, the reception of a PLI by a media sender indicates "the loss of an undefined amount of coded video data belonging to one or more pictures". Without having any specific knowledge of the setup of the bitstream (such as use and location of in-band parameter sets, non-IRAP decoder refresh points, picture structures, and so forth), a reaction to the reception of an PLI by a [VVC] sender SHOULD be to send an IRAP picture and relevant parameter sets; potentially with sufficient redundancy so to ensure correct reception. However, sometimes information about the bitstream structure is known. For example, state could have been established outside of the mechanisms defined in this document that parameter sets are conveyed out of band only, and stay static for the duration of the session. In that case, it is obviously unnecessary to send them in-band as a result of the reception of a PLI. Other

examples could be devised based on a priori knowledge of different aspects of the bitstream structure. In all cases, the timing and congestion control mechanisms of RFC 4585 MUST be observed.

8.2. Slice Loss Indication (SLI)

For further study. Maybe remove as there are no known implementations of SDLI in [HEVC] based systems

8.3. Reference Picture Selection Indication (RPSI)

Feedback-based reference picture selection has been shown as a powerful tool to stop temporal error propagation for improved error resilience [Girod99] [Wang05]. In one approach, the decoder side tracks errors in the decoded pictures and informs the encoder side that a particular picture that has been decoded relatively earlier is correct and still present in the decoded picture buffer; it requests the encoder to use that correct picture-availability information when encoding the next picture, so to stop further temporal error propagation. For this approach, the decoder side should use the RPSI feedback message.

Encoders can encode some long-term reference pictures as specified in [VVC] for purposes described in the previous paragraph without the need of a huge decoded picture buffer. As shown in [Wang05], with a flexible reference picture management scheme, as in VVC, even a decoded picture buffer size of two picture storage buffers would work for the approach described in the previous paragraph.

The text above is copy-paste from RFC 7798. If we keep the RPSI message, it needs adaptation to the [VVC] syntax. Doing so shouldn't be too hard as the [VVC] reference picture mechanism is not too different from the [HEVC] one.

8.4. Full Intra Request (FIR)

The purpose of the FIR message is to force an encoder to send an independent decoder refresh point as soon as possible, while observing applicable congestion-control-related constraints, such as those set out in [RFC8082]).

Upon reception of a FIR, a sender MUST send an IDR picture. Parameter sets MUST also be sent, except when there is a priori knowledge that the parameter sets have been correctly established. A typical example for that is an understanding between sender and receiver, established by means outside this document, that parameter sets are exclusively sent out-of-band.

- * The RTP packet is a FU with its S bit equal to 1 and the FU payload contains a fragment of the first VCL NAL unit, in decoding order, of a picture.

The E bit MUST be set to 1 if any of the following conditions is true and MUST be set to 0 otherwise:

- * The RTP packet is a single NAL unit packet and it is the last VCL NAL unit, in decoding order, of a picture.
- * The RTP packet is an AP and the NAL unit in the last contained aggregation unit is the last VCL NAL unit, in decoding order, of a picture.
- * The RTP packet is a FU with its E bit equal to 1 and the FU payload contains a fragment of the last VCL NAL unit, in decoding order, of a picture.

10. Security Considerations

The scope of this Security Considerations section is limited to the payload format itself and to one feature of [VVC] that may pose a particularly serious security risk if implemented naively. The payload format, in isolation, does not form a complete system. Implementers are advised to read and understand relevant security-related documents, especially those pertaining to RTP (see the Security Considerations section in [RFC3550]), and the security of the call-control stack chosen (that may make use of the media type registration of this memo). Implementers should also consider known security vulnerabilities of video coding and decoding implementations in general and avoid those.

Within this RTP payload format, and with the exception of the user data SEI message as described below, no security threats other than those common to RTP payload formats are known. In other words, neither the various media-plane-based mechanisms, nor the signaling part of this memo, seems to pose a security risk beyond those common to all RTP-based systems.

RTP packets using the payload format defined in this specification are subject to the security considerations discussed in the RTP specification [RFC3550] , and in any applicable RTP profile such as RTP/AVP [RFC3551] , RTP/AVPF [RFC4585] , RTP/SAVP [RFC3711] , or RTP/SAVPF [RFC5124] . However, as "Securing the RTP Framework: Why RTP Does Not Mandate a Single Media Security Solution" [RFC7202] discusses, it is not an RTP payload format's responsibility to discuss or mandate what solutions are used to meet the basic security goals like confidentiality, integrity and source authenticity for RTP

in general. This responsibility lays on anyone using RTP in an application. They can find guidance on available security mechanisms and important considerations in "Options for Securing RTP Sessions" [RFC7201] . The rest of this section discusses the security impacting properties of the payload format itself.

Because the data compression used with this payload format is applied end-to-end, any encryption needs to be performed after compression. A potential denial-of-service threat exists for data encodings using compression techniques that have non-uniform receiver-end computational load. The attacker can inject pathological datagrams into the bitstream that are complex to decode and that cause the receiver to be overloaded. [VVC] is particularly vulnerable to such attacks, as it is extremely simple to generate datagrams containing NAL units that affect the decoding process of many future NAL units. Therefore, the usage of data origin authentication and data integrity protection of at least the RTP packet is RECOMMENDED, for example, with SRTP [RFC3711] .

Like HEVC [RFC7798], [VVC] includes a user data Supplemental Enhancement Information (SEI) message. This SEI message allows inclusion of an arbitrary bitstring into the video bitstream. Such a bitstring could include JavaScript, machine code, and other active content. [VVC] leaves the handling of this SEI message to the receiving system. In order to avoid harmful side effects the user data SEI message, decoder implementations cannot naively trust its content. For example, it would be a bad and insecure implementation practice to forward any JavaScript a decoder implementation detects to a web browser. The safest way to deal with user data SEI messages is to simply discard them, but that can have negative side effects on the quality of experience by the user.

End-to-end security with authentication, integrity, or confidentiality protection will prevent a MANE from performing media-aware operations other than discarding complete packets. In the case of confidentiality protection, it will even be prevented from discarding packets in a media-aware way. To be allowed to perform such operations, a MANE is required to be a trusted entity that is included in the security context establishment.

11. Congestion Control

Congestion control for RTP SHALL be used in accordance with RTP [RFC3550] and with any applicable RTP profile, e.g., AVP [RFC3551]. If best-effort service is being used, an additional requirement is that users of this payload format MUST monitor packet loss to ensure that the packet loss rate is within an acceptable range. Packet loss is considered acceptable if a TCP flow across the same network path,

and experiencing the same network conditions, would achieve an average throughput, measured on a reasonable timescale, that is not less than all RTP streams combined are achieving. This condition can be satisfied by implementing congestion-control mechanisms to adapt the transmission rate, the number of layers subscribed for a layered multicast session, or by arranging for a receiver to leave the session if the loss rate is unacceptably high.

The bitrate adaptation necessary for obeying the congestion control principle is easily achievable when real-time encoding is used, for example, by adequately tuning the quantization parameter. However, when pre-encoded content is being transmitted, bandwidth adaptation requires the pre-coded bitstream to be tailored for such adaptivity. The key mechanisms available in [VVC] are temporal scalability, and spatial/SNR scalability. A media sender can remove NAL units belonging to higher temporal sublayers (i.e., those NAL units with a high value of TID) or higher spatio-SNR layers (as indicated by interpreting the VPS) until the sending bitrate drops to an acceptable range.

The mechanisms mentioned above generally work within a defined profile and level and, therefore, no renegotiation of the channel is required. Only when non-downgradable parameters (such as profile) are required to be changed does it become necessary to terminate and restart the RTP stream(s). This may be accomplished by using different RTP payload types.

MANEs MAY remove certain unusable packets from the RTP stream when that RTP stream was damaged due to previous packet losses. This can help reduce the network load in certain special cases. For example, MANEs can remove those FUs where the leading FUs belonging to the same NAL unit have been lost or those dependent slice segments when the leading slice segments belonging to the same slice have been lost, because the trailing FUs or dependent slice segments are meaningless to most decoders. MANEs can also remove higher temporal scalable layers if the outbound transmission (from the MANE's viewpoint) experiences congestion.

12. IANA Considerations

Placeholder

13. Acknowledgements

Dr. Byeongdoo Choi is thanked for the video codec related technical discussion and other aspects in this memo. Xin Zhao and Dr. Xiang Li are thanked for their contributions on [VVC] specification descriptive content. Spencer Dawkins is thanked for his valuable review comments that led to great improvements of this memo. Some parts of this specification share text with the RTP payload format for HEVC [RFC7798]. We thank the authors of that specification for their excellent work.

14. References

14.1. 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>>.
- [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>>.
- [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>>.
- [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>>.
- [RFC4556] Zhu, L. and B. Tung, "Public Key Cryptography for Initial Authentication in Kerberos (PKINIT)", RFC 4556, DOI 10.17487/RFC4556, June 2006, <<https://www.rfc-editor.org/info/rfc4556>>.
- [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>>.

- [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>>.
- [RFC4648] Josefsson, S., "The Base16, Base32, and Base64 Data Encodings", RFC 4648, DOI 10.17487/RFC4648, October 2006, <<https://www.rfc-editor.org/info/rfc4648>>.
- [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>>.
- [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>>.
- [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>>.
- [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, <<https://www.rfc-editor.org/info/rfc8082>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.
- [VSEI] "ISO/IEC 23002-7 (ITU-T H.274) Versatile supplemental enhancement information messages for coded video bitstreams", 2020, <<https://www.iso.org/standard/79112.html>>.
- [VVC] "ISO/IEC FDIS 23090-3 Information technology --- Coded representation of immersive media --- Part 3 - Versatile video coding", 2020, <<https://www.iso.org/standard/73022.html>>.

14.2. Informative References

- [CABAC] Sole, J, . and . et al, "Transform coefficient coding in HEVC, IEEE Transactions on Circuits and Systems for Video Technology", DOI 10.1109/TCSVT.2012.2223055, December 2012, <<https://doi.org/10.1109/TCSVT.2012.2223055>>.
- [FrameMarking] Berger, E, ., Nandakumar, S, ., and . Zanaty M, "Frame Marking RTP Header Extension", Work in Progress draft-berger-avtext-framemarking , 2015.
- [Girod99] Girod, B, . and . et al, "Feedback-based error control for mobile video transmission, Proceedings of the IEEE", DOI 110.1109/5.790632, October 1999, <<https://doi.org/110.1109/5.790632>>.
- [HEVC] "High efficiency video coding, ITU-T Recommendation H.265", April 2013.
- [MPEG2S] ISO/IEC, ., "Information technology - Generic coding of moving pictures and associated audio information - Part 1: Systems, ISO International Standard 13818-1", 2013.
- [RFC6051] Perkins, C. and T. Schierl, "Rapid Synchronisation of RTP Flows", RFC 6051, DOI 10.17487/RFC6051, November 2010, <<https://www.rfc-editor.org/info/rfc6051>>.
- [RFC6184] Wang, Y.-K., 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.-K., 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>>.
- [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>>.

- [RFC7798] Wang, Y.-K., Sanchez, Y., Schierl, T., Wenger, S., and M. 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>>.
- [Wang05] Wang, YK, ., Zhu, C, ., and . Li, H, "Error resilient video coding using flexible reference frames", Visual Communications and Image Processing 2005 (VCIP 2005) , July 2005.

Appendix A. Change History

draft-zhao-payload-rtp-vvc-00 initial version

draft-zhao-payload-rtp-vvc-01 editorial clarifications and corrections

draft-ietf-payload-rtp-vvc-00 initial WG draft

draft-ietf-payload-rtp-vvc-01 VVC specification update

draft-ietf-payload-rtp-vvc-02 VVC specification update

draft-ietf-payload-rtp-vvc-03 VVC coding tool introduction update

draft-ietf-payload-rtp-vvc-04 VVC coding tool introduction update

Authors' Addresses

Shuai Zhao
Tencent
2747 Park Blvd
Palo Alto, 94588
United States of America

Email: shuai.zhao@ieee.org

Stephan Wenger
Tencent
2747 Park Blvd
Palo Alto, 94588
United States of America

Email: stewe@stewe.org

Yago Sanchez
Fraunhofer HHI
Einsteinufer 37
10587 Berlin
Germany

Email: yago.sanchez@hhi.fraunhofer.de

Ye-Kui Wang
Bytedance Inc.
8910 University Center Lane
San Diego, 92122
United States of America

Email: yekui.wang@bytedance.com

avtcore
Internet-Draft
Intended status: Standards Track
Expires: May 31, 2021

S. Lugan
intoPIX
A. Descampe
UCL
C. Damman
intoPIX
T. Richter
IIS
T. Bruylants
intoPIX
November 27, 2020

RTP Payload Format for ISO/IEC 21122 (JPEG XS)
draft-ietf-payload-rtp-jpegxs-07

Abstract

This document specifies a Real-Time Transport Protocol (RTP) payload format to be used for transporting JPEG XS (ISO/IEC 21122) encoded video. JPEG XS is a low-latency, lightweight image coding system. Compared to an uncompressed video use case, it allows higher resolutions and frame rates, while offering visually lossless quality, reduced power consumption, and end-to-end latency confined to a fraction of a frame.

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 31, 2021.

Copyright Notice

Copyright (c) 2020 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	2
2. Conventions, Definitions, and Abbreviations	3
3. Media Format Description	5
3.1. Image Data Structures	5
3.2. Codestream	5
3.3. Video support box and colour specification box	5
3.4. JPEG XS Frame	6
4. RTP Payload Format	6
4.1. RTP packetization	6
4.2. RTP Header Usage	9
4.3. Payload Header Usage	10
4.4. Payload Data	12
4.5. Traffic Shaping and Delivery Timing	17
5. Congestion Control Considerations	18
6. Payload Format Parameters	18
6.1. Media Type Definition	18
6.2. Mapping to SDP	22
6.2.1. General	22
6.2.2. Media type and subtype	22
6.2.3. Traffic shaping	23
6.2.4. Offer/Answer Considerations	23
7. IANA Considerations	23
8. Security Considerations	23
9. Acknowledgments	24
10. RFC Editor Considerations	24
11. References	25
11.1. Normative References	25
11.2. Informative References	26
11.3. URIs	27
Authors' Addresses	27

1. Introduction

This document specifies a payload format for packetization of JPEG XS [ISO21122-1] encoded video signals into the Real-time Transport Protocol (RTP) [RFC3550].

The JPEG XS coding system offers compression and recompression of image sequences with very moderate computational resources while remaining robust under multiple compression and decompression cycles and mixing of content sources, e.g. embedding of subtitles, overlays or logos. Typical target compression ratios ensuring visually lossless quality are in the range of 2:1 to 10:1, depending on the nature of the source material. The end-to-end latency can be confined to a fraction of a frame, typically between a small number of lines down to below a single line.

2. Conventions, Definitions, and Abbreviations

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].

Application Data Unit (ADU)

The unit of source data provided as payload to the transport layer, and corresponding, in this RTP payload definition, to a single JPEG XS frame.

Colour specification box (CS box)

A ISO colour specification box defined in ISO/IEC 21122-3 [ISO21122-3] that includes colour-related metadata required to correctly display JPEG XS frames, such as colour primaries, transfer characteristics and matrix coefficients.

EOC marker

A marker that consists of the two bytes 0xff11 indicating the end of a JPEG XS codestream.

JPEG XS codestream

A sequence of bytes representing a compressed image formatted according to JPEG XS Part-1 [ISO21122-1].

JPEG XS codestream header

A sequence of bytes, starting with a SOC marker, at the beginning of each JPEG XS codestream encoded in multiple markers and marker segments that does not carry entropy coded data, but metadata such as the frame dimension and component precision.

JPEG XS frame

A JPEG XS picture segment in the case of a progressive frame, or, in the case of an interlaced frame, the concatenation of two JPEG XS picture segments.

JPEG XS header segment

The concatenation of a video support box, as defined in ISO/IEC 21122-3 [ISO21122-3], a colour specification box, as defined in ISO/IEC 21122-3 as well [ISO21122-3] and a JPEG XS codestream header.

JPEG XS picture segment

The concatenation of a video support box, as defined in ISO/IEC 21122-3 [ISO21122-3], a colour specification box, as defined in ISO/IEC 21122-3 as well [ISO21122-3] and a JPEG XS codestream.

JPEG XS stream

A sequence of JPEG XS frames.

Marker

A two-byte functional sequence that is part of a JPEG XS codestream starting with a 0xff byte and a subsequent byte defining its function.

Marker segment

A marker along with a 16-bit marker size and payload data following the size.

Packetization unit

A portion of an Application Data Unit whose boundaries coincide with boundaries of RTP packet payloads (excluding payload header), i.e. the first (resp. last) byte of a packetization unit is the first (resp. last) byte of a RTP packet payload (excluding its payload header).

Slice

The smallest independently decodable unit of a JPEG XS codestream, bearing in mind that it decodes to wavelet coefficients which still require inverse wavelet filtering to give an image.

SOC marker

A marker that consists of the two bytes 0xff10 indicating the start of a JPEG XS codestream.

Video support box (VS box)

A ISO video support box defined in ISO/IEC 21122-3 [ISO21122-3] that includes metadata required to play back a JPEG XS stream, such as its maximum bitrate, its subsampling structure, its buffer model and its frame rate.

3. Media Format Description

3.1. Image Data Structures

JPEG XS is a low-latency lightweight image coding system for coding continuous-tone grayscale or continuous-tone colour digital images.

This coding system provides an efficient representation of image signals through the mathematical tool of wavelet analysis. The wavelet filter process separates each component into multiple bands, where each band consists of multiple coefficients describing the image signal of a given component within a frequency domain specific to the wavelet filter type, i.e. the particular filter corresponding to the band.

Wavelet coefficients are grouped into precincts, where each precinct includes all coefficients over all bands that contribute to a spatial region of the image.

One or multiple precincts are furthermore combined into slices consisting of an integer number of precincts. Precincts do not cross slice boundaries, and wavelet coefficients in precincts that are part of different slices can be decoded independently from each other. Note, however, that the wavelet transformation runs across slice boundaries. A slice always extends over the full width of the image, but may only cover parts of its height.

3.2. Codestream

A JPEG XS codestream header, followed by several slices, and terminated by an EOC marker form a JPEG XS codestream.

The overall codestream format, including the definition of all markers, is further defined in ISO/IEC 21122-1 [ISO21122-1]. It represents sample values of a single image, bare any interpretation relative to a colour space.

3.3. Video support box and colour specification box

While the information defined in the codestream is sufficient to reconstruct the sample values of one image, the interpretation of the samples remains undefined by the codestream itself. This interpretation is given by the video support box and the colour specification box which contain significant information to correctly play the JPEG XS stream. The layout and syntax of these boxes, together with their content, are defined in ISO/IEC 21122-3 [ISO21122-3]. The video support box provides information on the maximum bitrate, the frame rate, the frame mode (progressive or

interlaced), the subsampling image format, the timecode of the current JPEG XS frame, the profile, level and sublevel used (as defined in ISO/IEC 21122-2 [ISO21122-2]), and optionally on the buffer model and the mastering display metadata. The colour specification box indicates the colour primaries, transfer characteristics, matrix coefficients and video full range flag needed to specify the colour space of the video stream.

3.4. JPEG XS Frame

The concatenation of a video support box, a colour specification box, and a JPEG XS codestream forms a JPEG XS picture segment.

In the case of a progressive video stream, each JPEG XS frame consists of one single JPEG XS picture segment.

In the case of an interlaced video stream, each JPEG XS frame is made of two concatenated JPEG XS picture segments. The codestream of each segment corresponds exclusively to one of the two fields of the interlaced frame. Both picture segments SHALL contain identical boxes (i.e. concatenation of the video support box and the colour specification box is byte exact the same for both picture segments).

Note that the interlaced mode as signaled by the `frat` field in the video support box indicates either progressive, interlaced top-field first, or interlaced bottom-field first mode. Thus, its value too SHALL be identical in both picture segments.

4. RTP Payload Format

This section specifies the payload format for JPEG XS streams over the Real-time Transport Protocol (RTP) [RFC3550].

In order to be transported over RTP, each JPEG XS stream is transported in a distinct RTP stream, identified by a distinct SSRC.

A JPEG XS stream is divided into Application Data Units (ADUs), each ADU corresponding to a single JPEG XS frame.

4.1. RTP packetization

An ADU is made of several packetization units. If a packetization unit is bigger than the maximum size of a RTP packet payload, the unit is split into multiple RTP packet payloads, as illustrated in Figure 1. As seen there, each packet SHALL contain (part of) one and only one packetization unit. A packetization unit may extend over multiple packets. The payload of every packet SHALL have the same size (based e.g. on the Maximum Transfer Unit of the network), except

(possibly) the last packet of a packetization unit. The boundaries of a packetization unit SHALL coincide with the boundaries of the payload of a packet (excluding the payload header), i.e. the first (resp. last) byte of the packetization unit SHALL be the first (resp. last) byte of the payload (excluding its header).

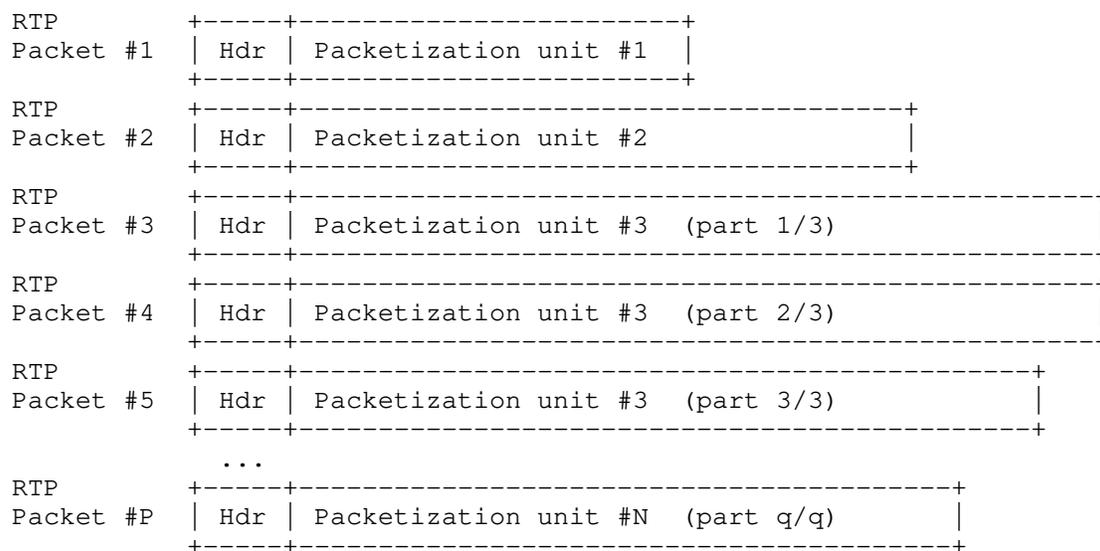


Figure 1: Example of ADU packetization

There are two different packetization modes defined for this RTP payload format.

1. Codestream packetization mode: in this mode, the packetization unit SHALL be the entire JPEG XS picture segment (i.e. codestream preceded by boxes). This means that a progressive frame will have a single packetization unit, while an interlaced frame will have two. The progressive case is illustrated in Figure 2.
2. Slice packetization mode: in this mode, the packetization unit SHALL be the slice, i.e. there SHALL be data from no more than one slice per RTP packet. The first packetization unit SHALL be made of the JPEG XS header segment (i.e. the concatenation of the VS box, the CS box and the JPEG XS codestream header). This first unit is then followed by successive units, each containing one and only one slice. The packetization unit containing the last slice of a JPEG XS codestream SHALL also contain the EOC marker immediately following this last slice. This is illustrated in Figure 3. In the case of an interlaced frame, the

JPEG XS header segment of the second field SHALL be in its own packetization unit.

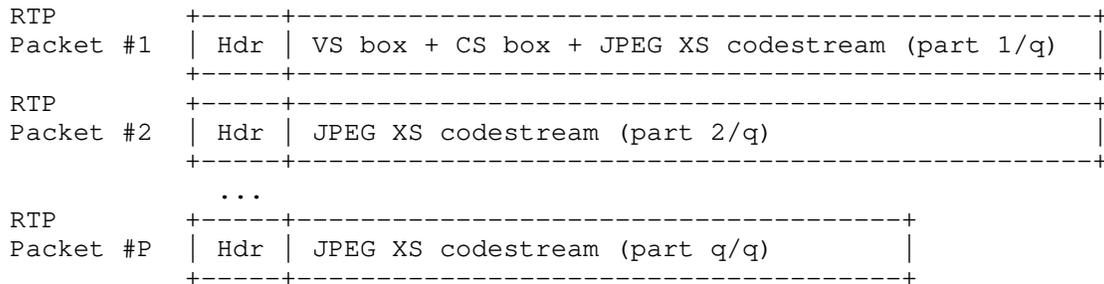


Figure 2: Example of codestream packetization mode

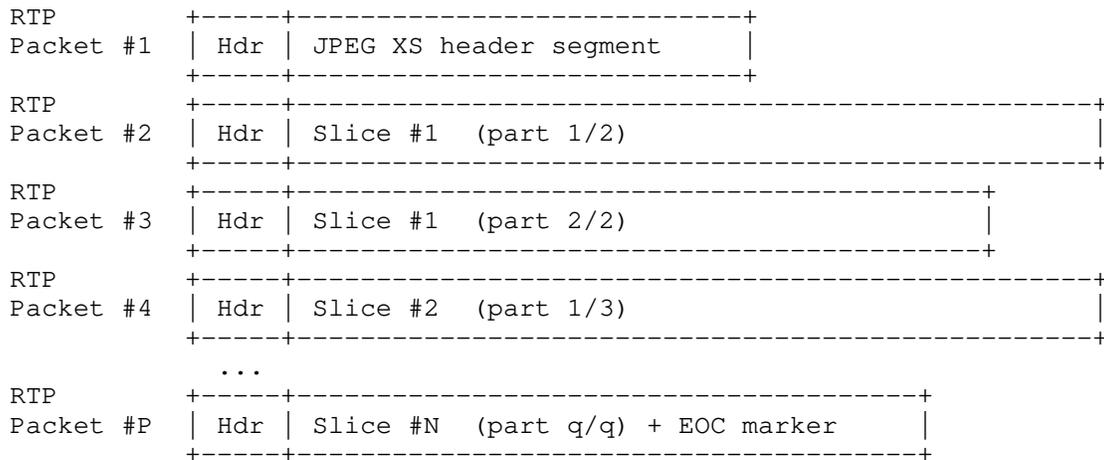


Figure 3: Example of slice packetization mode

Due to the constant bit-rate of JPEG XS, the codestream packetization mode guarantees that a JPEG XS RTP stream will produce a constant number of bytes per frame, and a constant number of RTP packets per frame. To reach the same guarantee with the slice packetization mode, an additional mechanism is required. This can involve a constraint at the rate allocation stage in the JPEG XS encoder to impose a constant bit-rate at the slice level, the usage of padding data, or the insertion of empty RTP packets (i.e. a RTP packet whose payload data is empty).

4.2. RTP Header Usage

The format of the RTP header is specified in RFC 3550 [RFC3550] and reprinted in Figure 4 for convenience. This RTP payload format uses the fields of the header in a manner consistent with that specification.

The RTP payload (and the settings for some RTP header bits) for packetization units are specified in Section 4.3.

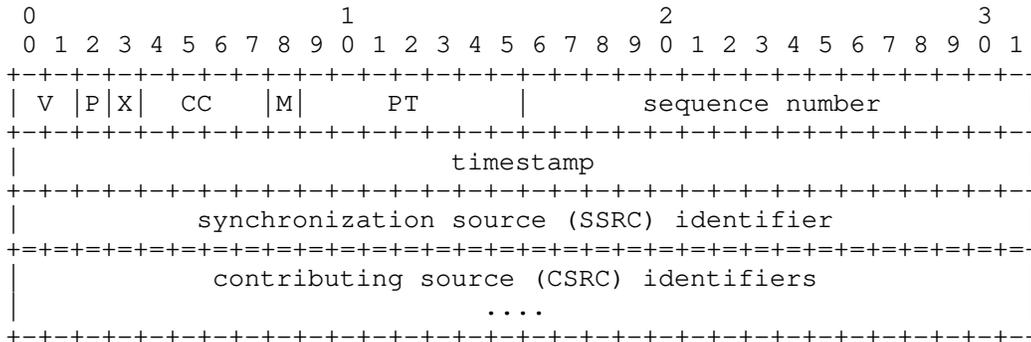


Figure 4: RTP header according to RFC 3550

The version (V), padding (P), extension (X), CSRC count (CC), sequence number, synchronization source (SSRC) and contributing source (CSRC) fields follow their respective definitions in RFC 3550 [RFC3550].

The remaining RTP header information to be set according to this RTP payload format is set as follows:

Marker (M) [1 bit]:

If progressive scan video is being transmitted, the marker bit denotes the end of a video frame. If interlaced video is being transmitted, it denotes the end of the field. The marker bit SHALL be set to 1 for the last packet of the video frame/field. It SHALL be set to 0 for all other packets.

Payload Type (PT) [7 bits]:

A dynamically allocated payload type field that designates the payload as JPEG XS video.

Timestamp [32 bits]:

The RTP timestamp is set to the sampling timestamp of the content. A 90 kHz clock rate SHOULD be used.

As per specified in RFC 3550 [RFC3550] and RFC 4175 [RFC4175], the RTP timestamp designates the sampling instant of the first octet of the frame to which the RTP packet belongs. Packets SHALL not include data from multiple frames, and all packets belonging to the same frame SHALL have the same timestamp. Several successive RTP packets will consequently have equal timestamps if they belong to the same frame (that is until the marker bit is set to 1, marking the last packet of the frame), and the timestamp is only increased when a new frame begins.

If the sampling instant does not correspond to an integer value of the clock, the value SHALL be truncated to the next lowest integer, with no ambiguity.

4.3. Payload Header Usage

The first four bytes of the payload of an RTP packet in this RTP payload format are referred to as the payload header. Figure 5 illustrates the structure of this payload header.

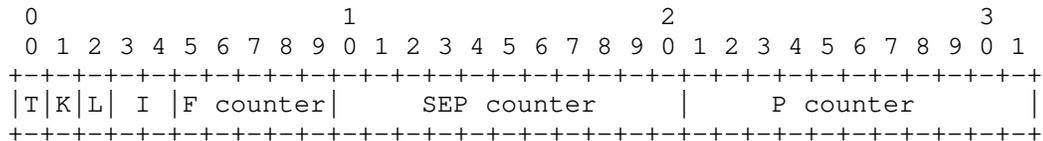


Figure 5: Payload header

The payload header consists of the following fields:

Transmission mode (T) [1 bit]:

The T bit is set to indicate that packets are sent sequentially by the transmitter. This information allows a receiver to dimension its input buffer(s) accordingly. If T=0, nothing can be assumed about the transmission order and packets may be sent out-of-order by the transmitter. If T=1, packets SHALL be sent sequentially by the transmitter.

packEtization mode (K) [1 bit]:

The K bit is set to indicate which packetization mode is used. K=0 indicates codestream packetization mode, while K=1 indicates slice packetization mode. In the case that the Transmission mode

(T) is set to 0, the slice packetization mode SHALL be used and K SHALL be set to 1.

Last (L) [1 bit]:

The L bit is set to indicate the last packet of a packetization unit. As the end of the frame also ends the packet containing the last unit of the frame, the L bit is set whenever the M bit is set. If codestream packetization mode is used, L bit and M bit are equivalent.

Interlaced information (I) [2 bit]:

These 2 bits are used to indicate how the JPEG XS frame is scanned (progressive or interlaced). In case of an interlaced frame, they also indicate which JPEG XS picture segment the payload is part of (first or second).

00: The payload is progressively scanned.

01: Reserved for future use.

10: The payload is part of the first JPEG XS picture segment of an interlaced video frame. The height specified in the included JPEG XS codestream header is half of the height of the entire displayed image.

11: The payload is part of the second JPEG XS picture segment of an interlaced video frame. The height specified in the included JPEG XS codestream header is half of the height of the entire displayed image.

F counter [5 bits]:

The frame (F) counter identifies the frame number modulo 32 to which a packet belongs. Frame numbers are incremented by 1 for each frame transmitted. The frame number, in addition to the timestamp, may help the decoder manage its input buffer and bring packets back into their natural order.

SEP counter [11 bits]:

The Slice and Extended Packet (SEP) counter is used differently depending on the packetization mode.

* In the case of codestream packetization mode (K=0), this counter resets whenever the Packet counter resets (see

hereunder), and increments by 1 whenever the Packet counter overruns.

- * In the case of slice packetization mode ($K=1$), this counter identifies the slice modulo 2047 to which the packet contributes. If the data belongs to the JPEG XS header segment, this field SHALL have its maximal value, namely $2047 = 0x07ff$. Otherwise, it is the slice index modulo 2047. Slice indices are counted from 0 (corresponding to the top of the frame).

P counter [11 bits]:

The packet (P) counter identifies the packet number modulo 2048 within the current packetization unit. It is set to 0 at the start of the packetization unit and incremented by 1 for every subsequent packet (if any) belonging to the same unit. Practically, if codestream packetization mode is enabled, this field counts the packets within a JPEG XS picture segment and is extended by the SEP counter when it overruns. If slice packetization mode is enabled, this field counts the packets within a slice or within the JPEG XS header segment.

4.4. Payload Data

The payload data of a JPEG XS RTP stream consists of a concatenation of multiple JPEG XS frames. In the RTP stream, all the JPEG XS frames SHALL contain identical boxes in their picture segment(s) (i.e. concatenation of the video support box and the colour specification box of any picture segment is always byte exact the same for any of the frames).

Each JPEG XS frame is the concatenation of one or more packetization unit(s), as explained in Section 4.1. Figure 6 depicts this layout for a progressive frame in the codestream packetization mode, Figure 7 depicts this layout for an interlaced frame in the codestream packetization mode, Figure 8 depicts this layout for a progressive frame in the slice packetization mode and Figure 9 depicts this layout for an interlaced frame in the slice packetization mode. The Frame counter value is not indicated because the value is constant for all packetization units of a given frame.

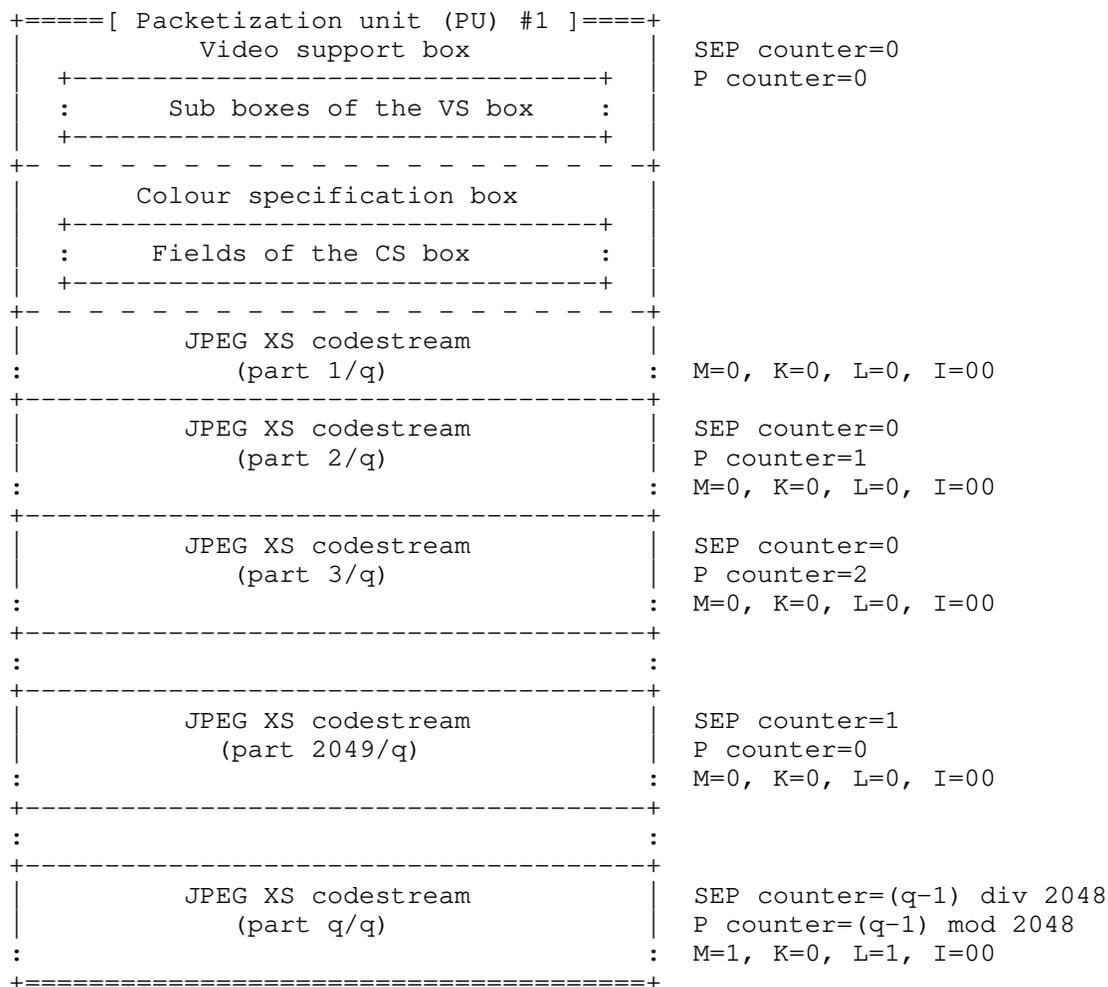


Figure 6: Example of JPEG XS Payload Data (codestream packetization mode, progressive frame)

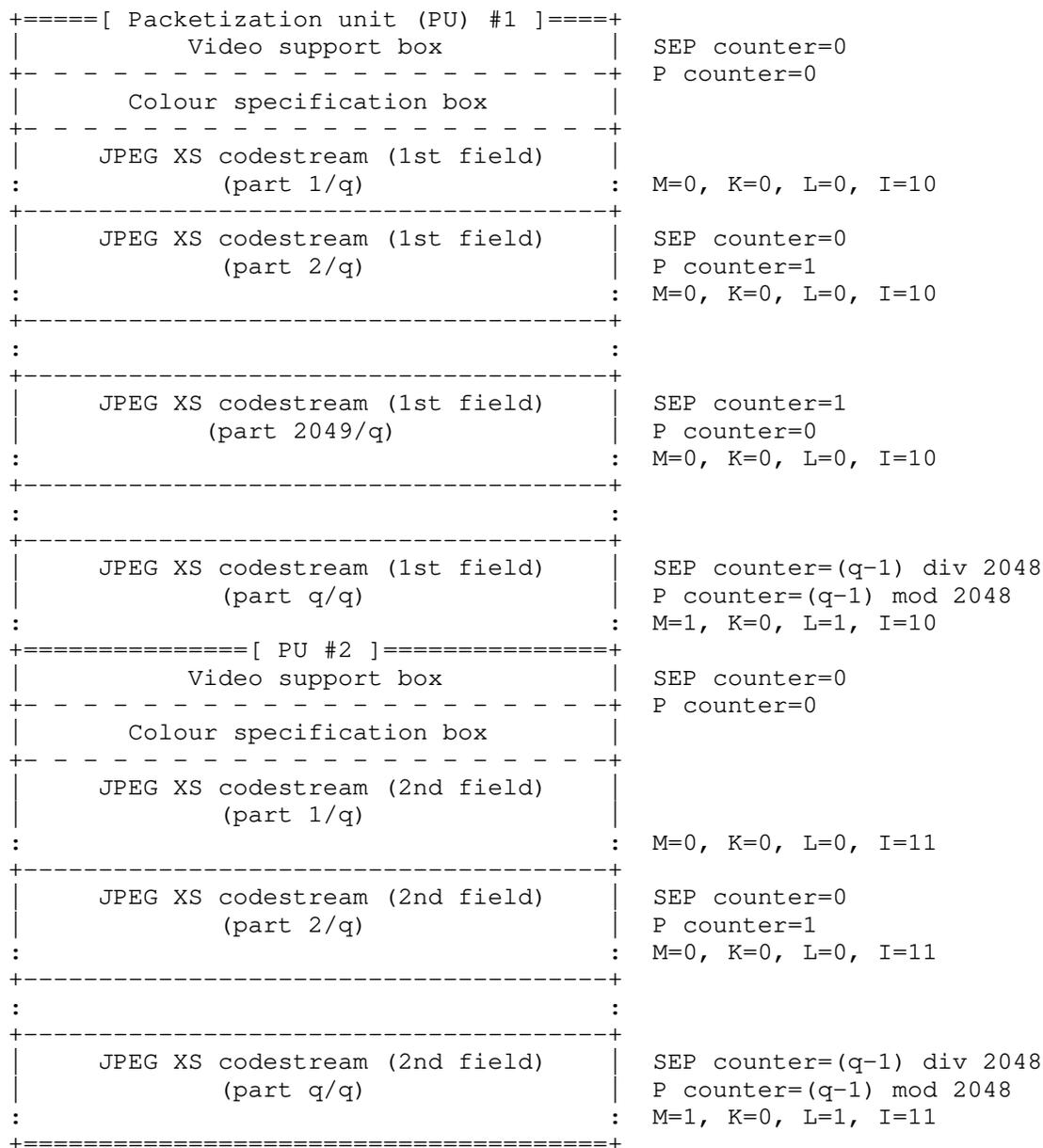


Figure 7: Example of JPEG XS Payload Data (codestream packetization mode, interlaced frame)

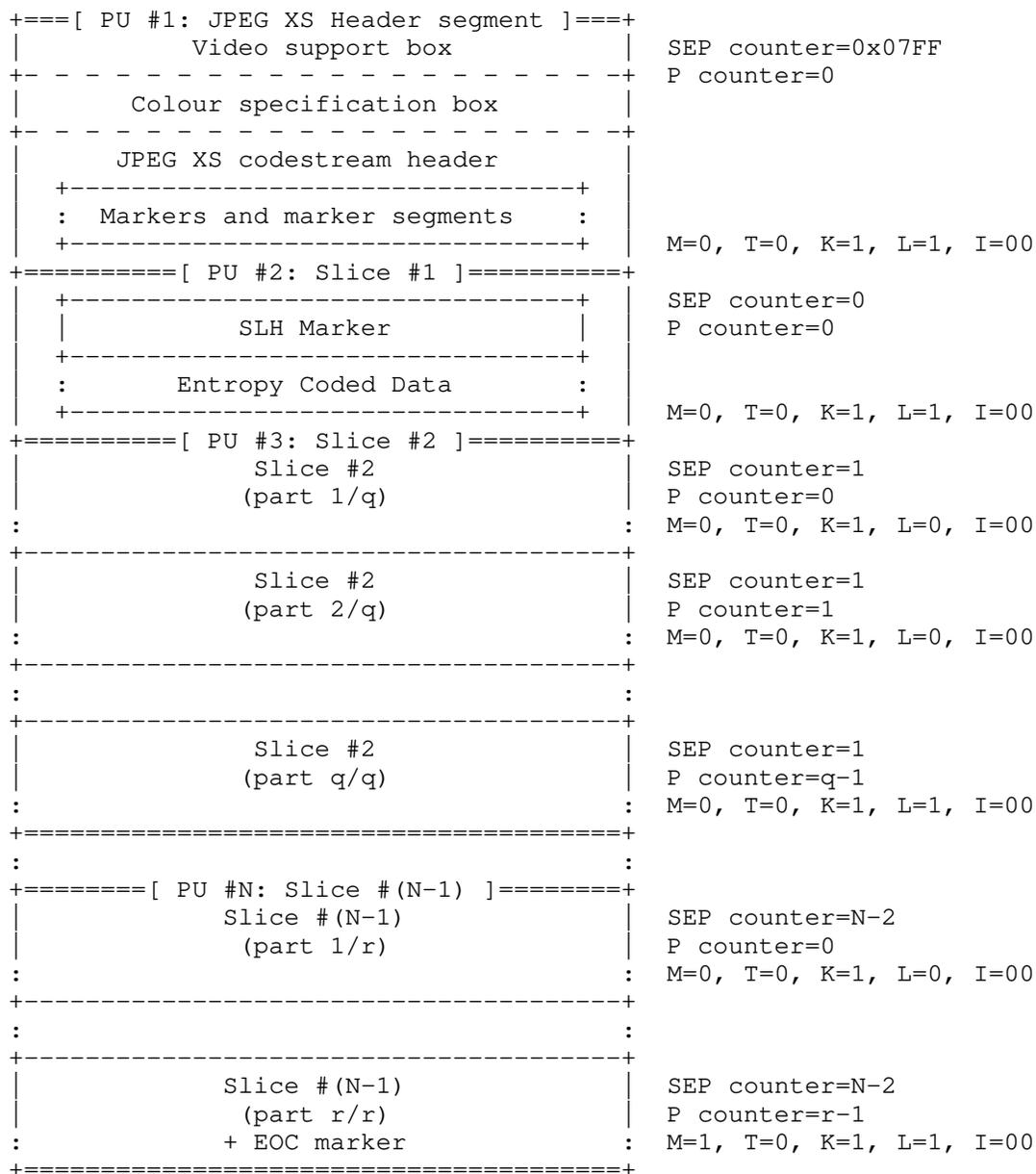


Figure 8: Example of JPEG XS Payload Data (slice packetization mode, progressive frame)

```

+====[ PU #1: JPEG XS Hdr segment 1 ]====+
|           Video support box           | SEP counter=0x07FF
+-----+                               | P counter=0
|           Colour specification box     |
+-----+                               |
|           JPEG XS codestream header 1  |
|           +-----+                   |
|           : Markers and marker segments : |
|           +-----+                   | M=0, T=0, K=1, L=1, I=10
+====[ PU #2: Slice #1 (1st field) ]====+
|           +-----+                   | SEP counter=0
|           | SLH Marker                  | | P counter=0
|           +-----+                   |
|           : Entropy Coded Data         : |
|           +-----+                   | M=0, T=0, K=1, L=1, I=10
+====[ PU #3: Slice #2 (1st field) ]====+
|           Slice #2                     | SEP counter=1
|           (part 1/q)                   | P counter=0
|           :                             : | M=0, T=0, K=1, L=0, I=10
+-----+                               |
|           Slice #2                     | SEP counter=1
|           (part 2/q)                   | P counter=1
|           :                             : | M=0, T=0, K=1, L=0, I=10
+-----+                               |
|           :                             : |
+-----+                               |
|           Slice #2                     | SEP counter=1
|           (part q/q)                   | P counter=q-1
|           :                             : | M=0, T=0, K=1, L=1, I=10
+-----+                               |
|           :                             : |
+====[ PU #N: Slice #(N-1) (1st field) ]====+
|           Slice #(N-1)                 | SEP counter=N-2
|           (part 1/r)                   | P counter=0
|           :                             : | M=0, T=0, K=1, L=0, I=10
+-----+                               |
|           :                             : |
+-----+                               |
|           Slice #(N-1)                 | SEP counter=N-2
|           (part r/r)                   | P counter=r-1
|           : + EOC marker                : | M=1, T=0, K=1, L=1, I=10
+-----+                               |
+====[ PU #N+1: JPEG XS Hdr segment 2 ]====+
|           Video support box           | SEP counter=0x07FF
+-----+                               | P counter=0
|           Colour specification box     |
+-----+                               |
|           JPEG XS codestream header 2  |

```

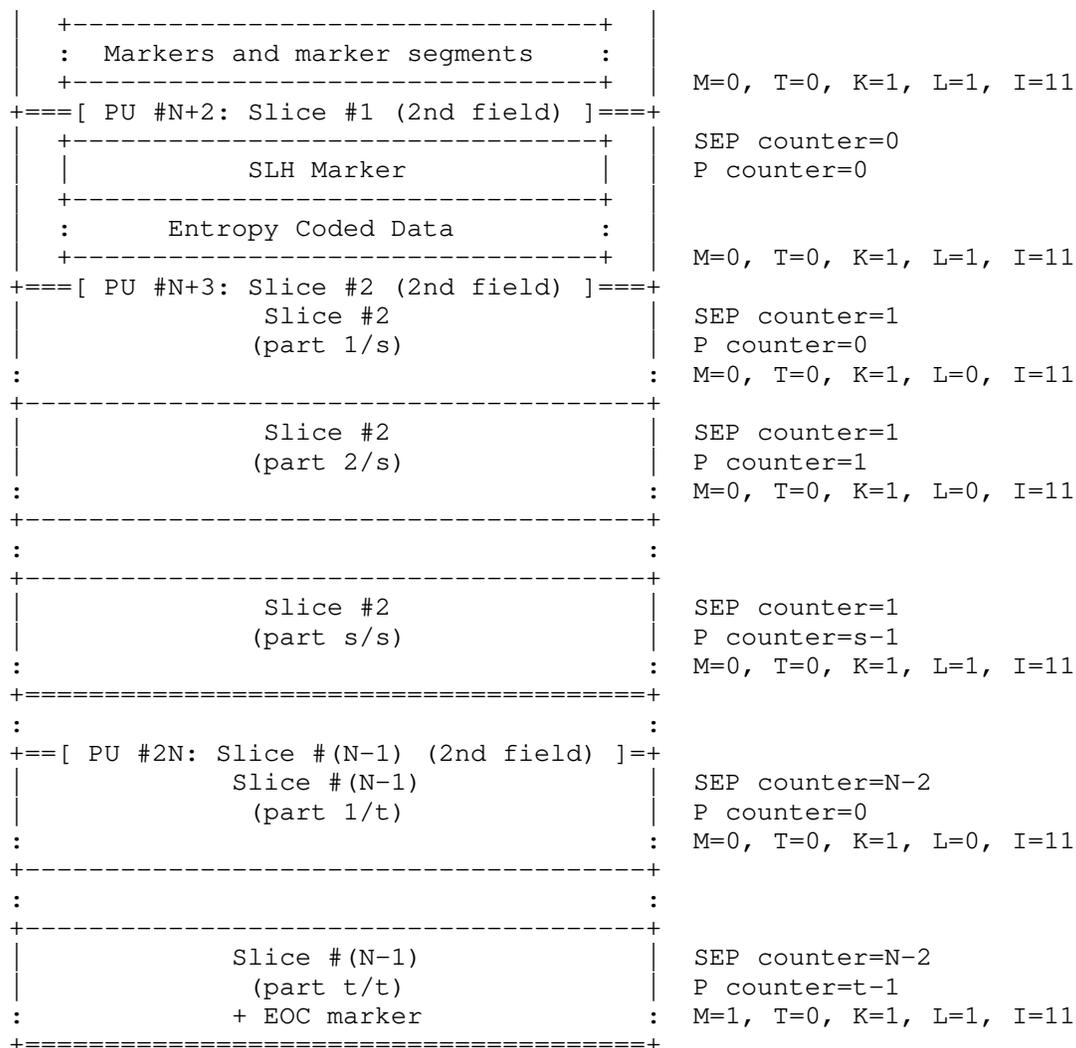


Figure 9: Example of JPEG XS Payload Data (slice packetization mode, interlaced frame)

4.5. Traffic Shaping and Delivery Timing

The traffic shaping and delivery timing SHALL be in accordance with the Network Compatibility Model compliance definitions specified in SMPTE ST 2110-21 [SMPTE-ST2110-21] for either Narrow Linear Senders (Type NL) or Wide Senders (Type W). The session description SHALL include a format-specific parameter of either TP=2110TPNL or

TP=2110TPW to indicate compliance with Type NL or Type W respectively.

NOTE: The Virtual Receiver Buffer Model compliance definitions of ST 2110-21 do not apply.

5. Congestion Control Considerations

Congestion control for RTP SHALL be used in accordance with RFC 3550 [RFC3550], and with any applicable RTP profile: e.g., RFC 3551 [RFC3551]. An additional requirement if best-effort service is being used is users of this payload format SHALL monitor packet loss to ensure that the packet loss rate is within acceptable parameters. Circuit Breakers [RFC8083] is an update to RTP [RFC3550] that defines criteria for when one is required to stop sending RTP Packet Streams and applications implementing this standard SHALL comply with it. RFC 8085 [RFC8085] provides additional information on the best practices for applying congestion control to UDP streams.

6. Payload Format Parameters

6.1. Media Type Definition

Type name: video

Subtype name: jxsv

Required parameters:

rate: The RTP timestamp clock rate. Applications using this payload format SHOULD use a value of 90000.

transmode: This parameter specifies the configured transmission mode as defined by the Transmission mode (T) bit in the payload header of Section 4.3. This value SHALL be equal to the T bit value configured in the RTP stream (i.e. 0 for out-of-order-allowed or 1 for sequential).

Optional parameters:

packetmode: This parameter specifies the configured packetization mode as defined by the packetization mode (K) bit in the payload header of Section 4.3. If specified, this value SHALL be equal to the K bit value configured in the RTP stream (i.e. 0 for codestream or 1 for slice).

profile: The JPEG XS profile in use, as defined in ISO/IEC 21122-2 (JPEG XS Part 2) [ISO21122-2]. Any white space in the

profile name SHALL be replaced by a dash (-). Examples are 'Main-444.12' or 'High-444.12'.

level: The JPEG XS level in use, as defined in ISO/IEC 21122-2 (JPEG XS Part 2) [ISO21122-2]. Any white space in the level name SHALL be replaced by a dash (-). Examples are '2k-1' or '4k-2'.

sublevel: The JPEG XS sublevel in use, as defined in ISO/IEC 21122-2 (JPEG XS Part 2) [ISO21122-2]. Any white space in the sublevel name SHALL be replaced by a dash (-). Examples are 'Sublev3bpp' or 'Sublev6bpp'.

depth: Determines the number of bits per sample. This is an integer with typical values including 8, 10, 12, and 16.

width: Determines the number of pixels per line. This is an integer between 1 and 32767.

height: Determines the number of lines per frame. This is an integer between 1 and 32767.

exactframerate: Signals the frame rate in frames per second. Integer frame rates SHALL be signaled as a single decimal number (e.g. "25") whilst non-integer frame rates SHALL be signaled as a ratio of two integer decimal numbers separated by a "forward-slash" character (e.g. "30000/1001"), utilizing the numerically smallest numerator value possible.

interlaced: If this parameter name is present, it indicates that the video is interlaced, or that the video is Progressive segmented Frame (PsF). If this parameter name is not present, the progressive video format SHALL be assumed.

segmented: If this parameter name is present, and the interlace parameter name is also present, then the video is a Progressive segmented Frame (PsF). Signaling of this parameter without the interlace parameter is forbidden.

sampling: Signals the colour difference signal sub-sampling structure.

Signals utilizing the non-constant luminance Y'C'B C'R signal format of Recommendation ITU-R BT.601-7, Recommendation ITU-R BT.709-6, Recommendation ITU-R BT.2020-2, or Recommendation ITU-R BT.2100 SHALL use the appropriate one of the following values for the Media Type Parameter "sampling":

YCbCr-4:4:4 (4:4:4 sampling)
YCbCr-4:2:2 (4:2:2 sampling)
YCbCr-4:2:0 (4:2:0 sampling)

Signals utilizing the Constant Luminance Y'C C'BC C'RC signal format of Recommendation ITU-R BT.2020-2 SHALL use the appropriate one of the following values for the Media Type Parameter "sampling":

CLYCbCr-4:4:4 (4:4:4 sampling)
CLYCbCr-4:2:2 (4:2:2 sampling)
CLYCbCr-4:2:0 (4:2:0 sampling)

Signals utilizing the constant intensity I CT CP signal format of Recommendation ITU-R BT.2100 SHALL use the appropriate one of the following values for the Media Type Parameter "sampling":

ICtCp-4:4:4 (4:4:4 sampling)
ICtCp-4:2:2 (4:2:2 sampling)
ICtCp-4:2:0 (4:2:0 sampling)

Signals utilizing the 4:4:4 R' G' B' or RGB signal format (such as that of Recommendation ITU-R BT.601, Recommendation ITU-R BT.709, Recommendation ITU-R BT.2020, Recommendation ITU-R BT.2100, SMPTE ST 2065-1 or ST 2065-3) SHALL use the following value for the Media Type Parameter sampling.

RGB (RGB or R' G' B' samples)

Signals utilizing the 4:4:4 X' Y' Z' signal format (such as defined in SMPTE ST 428-1) SHALL use the following value for the Media Type Parameter sampling.

XYZ (X' Y' Z' samples)

Key signals as defined in SMPTE RP 157 SHALL use the value key for the Media Type Parameter sampling. The Key signal is represented as a single component.

KEY (Samples of the key signal)

Signals utilizing a colour sub-sampling other than what is defined here SHALL use the following value for the Media Type Parameter sampling.

UNSPECIFIED (Sampling signaled by the payload.)

colorimetry: Specifies the system colorimetry used by the image samples. Valid values and their specification are:

BT601-5	ITU-R Recommendation BT.601-5.
BT709-2	ITU-R Recommendation BT.709-2.
SMPTE240M	SMPTE ST 240M.
BT601	ITU-R Recommendation BT.601-7.
BT709	ITU-R Recommendation BT.709-6.
BT2020	ITU-R Recommendation BT.2020-2.
BT2100	ITU-R Recommendation BT.2100
	Table 2 titled "System colorimetry".
ST2065-1	SMPTE ST 2065-1 Academy Color Encoding Specification (ACES).
ST2065-3	SMPTE ST 2065-3 Academy Density Exchange Encoding (ADX).
XYZ	ISO/IEC 11664-1, section titled "1931 Observer".
UNSPECIFIED	Colorimetry is signaled in the payload by the Color Specification Box of ISO/IEC 21122-3, or it must be manually coordinated between sender and receiver.

Signals utilizing the Recommendation ITU-R BT.2100 colorimetry SHOULD also signal the representational range using the optional parameter RANGE defined below. Signals utilizing the UNSPECIFIED colorimetry might require manual coordination between the sender and the receiver.

TCS: Transfer Characteristic System. This parameter specifies the transfer characteristic system of the image samples. Valid values and their specification are:

SDR	Standard Dynamic Range video streams that utilize the OETF of ITU-R Recommendation BT.709 or ITU-R Recommendation BT.2020. Such streams SHALL be assumed to target the EOTF specified in ITU-R Recommendation BT.1886.
PQ	High dynamic range video streams that utilize the Perceptual Quantization system of ITU-R Recommendation BT.2100.
HLG	High dynamic range video streams that utilize the Hybrid Log-Gamma system of ITU-R Recommendation BT.2100.
UNSPECIFIED	Video streams whose transfer characteristics are signaled by the payload as specified in ISO/IEC 21122-3, or must be manually coordinated between sender and receiver.

RANGE: This parameter SHOULD be used to signal the encoding range of the sample values within the stream. When paired with ITU Rec BT.2100 colorimetry, this parameter has two allowed values NARROW and FULL, corresponding to the ranges specified in table 9 of ITU Rec BT.2100. In any other context, this parameter has three allowed values: NARROW, FULLPROTECT, and FULL, which correspond to the ranges specified in SMPTE RP 2077. In the absence of this parameter, and for all but the UNSPECIFIED colometries, NARROW SHALL be the assumed value. When paired with the UNSPECIFIED colometry, FULL SHALL be the default assumed value.

Encoding considerations:

This media type is framed and binary; see Section 4.8 in RFC 6838 [RFC6838].

Security considerations:

Please see the Security Considerations section in RFC XXXX

6.2. Mapping to SDP

6.2.1. General

A Session Description Protocol (SDP) [RFC4566] object SHALL be created for each RTP stream and it SHALL be in accordance with the provisions of SMPTE ST 2110-10 [SMPTE-ST2110-10].

The information carried in the media type specification has a specific mapping to the SDP fields, used to describe RTP sessions. This information is redundant with the information found in the payload data (namely, in the JPEG XS header segment) and SHALL be consistent with it. In case of discrepancy between parameters values found in the payload data and in the SDP fields, the values from the payload data SHALL prevail.

6.2.2. Media type and subtype

The media type ("video") goes in SDP "m=" as the media name.

The media subtype ("jxsv") goes in SDP "a=rtpmap" as the encoding name, followed by a slash ("/") and the required parameter "rate" corresponding to the RTP timestamp clock rate (which for the payload format defined in this document SHOULD be 90000). The required parameter "transmode" and the additional optional parameters go in the SDP "a=fmtp" attribute by copying them directly from the MIME

media type string as a semicolon-separated list of parameter=value pairs.

A sample SDP mapping for JPEG XS video is as follows:

```
m=video 30000 RTP/AVP 112
a=rtpmap:112 jxsv/90000
a=fmtp:112 transmode=1;sampling=YCbCr-4:2:2;width=1920;
          height=1080;depth=10;colorimetry=BT709;TCS=SDR;
          RANGE=FULL;TP=2110TPNL
```

In this example, a JPEG XS RTP stream is being sent to UDP destination port 30000, with an RTP dynamic payload type of 112 and a media clock rate of 90000 Hz. Note that the "a=fmtp:" line has been wrapped to fit this page, and will be a single long line in the SDP file.

6.2.3. Traffic shaping

The SDP object SHALL include the TP parameter (either 2110TPNL or 2110TPW as specified in Section 4.5) and may include the CMAX parameter as specified in SMPTE ST 2110-21 [SMPTE-ST2110-21].

6.2.4. Offer/Answer Considerations

All parameters are declarative.

7. IANA Considerations

This memo requests that IANA registers video/jxsv as specified in Section 6.1. The media type is also requested to be added to the IANA registry for "RTP Payload Format MIME types" [1].

8. Security Considerations

RTP packets using the payload format defined in this specification are subject to the security considerations discussed in the RTP specification [RFC3550] and in any applicable RTP profile such as RTP/AVP [RFC3551], RTP/AVPF [RFC4585], RTP/SAVP [RFC3711], or RTP/SAVPF [RFC5124]. This implies that confidentiality of the media streams is achieved by encryption.

However, as "Securing the RTP Framework: Why RTP Does Not Mandate a Single Media Security Solution" [RFC7202] discusses, it is not an RTP payload format's responsibility to discuss or mandate what solutions are used to meet the basic security goals like confidentiality, integrity, and source authenticity for RTP in general. This responsibility lies on anyone using RTP in an application. They can

find guidance on available security mechanisms and important considerations in "Options for Securing RTP Sessions" [RFC7201]. Applications SHOULD use one or more appropriate strong security mechanisms.

This payload format and the JPEG XS encoding do not exhibit any substantial non-uniformity, either in output or in complexity to perform the decoding operation and thus are unlikely to pose a denial-of-service threat due to the receipt of pathological datagrams.

It is important to note that HD or UHDTV JPEG XS-encoded video can have significant bandwidth requirements (typically more than 1 Gbps for ultra high-definition video, especially if using high framerate). This is sufficient to cause potential for denial-of-service if transmitted onto most currently available Internet paths.

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

This payload format may also be used in networks that provide quality-of-service guarantees. If enhanced service is being used, receivers SHOULD monitor packet loss to ensure that the service that was requested is actually being delivered. If it is not, then they SHOULD assume that they are receiving best-effort service and behave accordingly.

9. Acknowledgments

The authors would like to thank the following people for their valuable contributions to this specification: Alexandre Willeme, Gael Rouvroy, and Jean-Baptiste Lorent.

10. RFC Editor Considerations

Note to RFC Editor: This section may be removed after carrying out all the instructions of this section.

RFC XXXX is to be replaced by the RFC number this specification receives when published.

11. References

11.1. Normative References

[ISO21122-1]

International Organization for Standardization (ISO) -
International Electrotechnical Commission (IEC),
"Information technology - JPEG XS low-latency lightweight
image coding system - Part 1: Core coding system", ISO/
IEC IS 21122-1, 2019,
<<https://www.iso.org/standard/74535.html>>.

[ISO21122-2]

International Organization for Standardization (ISO) -
International Electrotechnical Commission (IEC),
"Information technology - JPEG XS low-latency lightweight
image coding system - Part 2: Profiles and buffer models",
ISO/IEC IS 21122-2, 2019,
<<https://www.iso.org/standard/74536.html>>.

[ISO21122-3]

International Organization for Standardization (ISO) -
International Electrotechnical Commission (IEC),
"Information technology - JPEG XS low-latency lightweight
image coding system - Part 3: Transport and container
formats", ISO/IEC IS 21122-3, 2019,
<<https://www.iso.org/standard/74537.html>>.

[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>>.

[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>>.
- [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>>.
- [RFC6838] Freed, N., Klensin, J., and T. Hansen, "Media Type Specifications and Registration Procedures", BCP 13, RFC 6838, DOI 10.17487/RFC6838, January 2013, <<https://www.rfc-editor.org/info/rfc6838>>.
- [RFC8083] Perkins, C. and V. Singh, "Multimedia Congestion Control: Circuit Breakers for Unicast RTP Sessions", RFC 8083, DOI 10.17487/RFC8083, March 2017, <<https://www.rfc-editor.org/info/rfc8083>>.
- [RFC8085] Eggert, L., Fairhurst, G., and G. Shepherd, "UDP Usage Guidelines", BCP 145, RFC 8085, DOI 10.17487/RFC8085, March 2017, <<https://www.rfc-editor.org/info/rfc8085>>.
- [SMPTE-ST2110-10] Society of Motion Picture and Television Engineers, "SMPTE Standard - Professional Media Over Managed IP Networks: System Timing and Definitions", SMPTE ST 2110-10:2017, 2017, <<https://doi.org/10.5594/SMPTE.ST2110-10.2017>>.
- [SMPTE-ST2110-21] Society of Motion Picture and Television Engineers, "SMPTE Standard - Professional Media Over Managed IP Networks: Traffic Shaping and Delivery Timing for Video", SMPTE ST 2110-21:2017, 2017, <<https://doi.org/10.5594/SMPTE.ST2110-21.2017>>.

11.2. Informative References

- [RFC4175] Gharai, L. and C. Perkins, "RTP Payload Format for Uncompressed Video", RFC 4175, DOI 10.17487/RFC4175, September 2005, <<https://www.rfc-editor.org/info/rfc4175>>.
- [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>>.
- [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>>.

11.3. URIs

- [1] <http://www.iana.org/assignments/rtp-parameters>

Authors' Addresses

Sebastien Lukan
intoPIX S.A.
Rue Emile Francqui, 9
1435 Mont-Saint-Guibert
Belgium

Phone: +32 10 23 84 70
Email: rtp@intopix.com
URI: <https://www.intopix.com/>

Antonin Descampe
Universite catholique de Louvain
Place du Levant, 3 - bte L5.03.02
1348 Louvain-la-Neuve
Belgium

Phone: +32 10 47 25 97
Email: antonin.descampe@uclouvain.be
URI: <https://uclouvain.be/en/research-institutes/icteam>

Corentin Damman
intoPIX S.A.
Rue Emile Francqui, 9
1435 Mont-Saint-Guibert
Belgium

Phone: +32 10 23 84 70
Email: c.damman@intopix.com
URI: <https://www.intopix.com/>

Thomas Richter
Fraunhofer IIS
Am Wolfsmantel 33
91048 Erlangen
Germany

Phone: +49 9131 776 5126
Email: thomas.richter@iis.fraunhofer.de
URI: <https://www.iis.fraunhofer.de/>

Tim Bruylants
intoPIX S.A.
Rue Emile Francqui, 9
1435 Mont-Saint-Guibert
Belgium

Phone: +32 10 23 84 70
Email: t.bruylants@intopix.com
URI: <https://www.intopix.com/>

Network Working Group
Internet-Draft
Intended status: Informational
Expires: May 20, 2021

E. Omara
J. Uberti
Google
A. GOUAILLARD
S. Murillo
CoSMo Software
November 16, 2020

Secure Frame (SFrame)
draft-omara-sframe-01

Abstract

This document describes the Secure Frame (SFrame) end-to-end encryption and authentication mechanism for media frames in a multiparty conference call, in which central media servers (SFUs) can access the media metadata needed to make forwarding decisions without having access to the actual media. The proposed mechanism differs from other approaches through its use of media frames as the encryptable unit, instead of individual RTP packets, which makes it more bandwidth efficient and also allows use with non-RTP transports.

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 20, 2021.

Copyright Notice

Copyright (c) 2020 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	3
2. Terminology	4
3. Goals	4
4. SFrame	5
4.1. SFrame Format	7
4.2. SFrame Header	7
4.3. Encryption Schema	8
4.3.1. Key Selection	9
4.3.2. Key Derivation	9
4.3.3. Encryption	10
4.3.4. Decryption	12
4.3.5. Duplicate Frames	12
4.4. Authentication	12
4.5. Ciphersuites	14
4.5.1. AES-CM with SHA2	15
5. Key Management	16
5.1. Sender Keys	16
5.2. MLS	17
6. Media Considerations	18
6.1. SFU	18
6.1.1. LastN and RTP stream reuse	19
6.1.2. Simulcast	19
6.1.3. SVC	19
6.2. Video Key Frames	20
6.3. Partial Decoding	20
7. Overhead	20
7.1. Audio	20
7.2. Video	21
7.3. SFrame vs PERC-lite	21
7.3.1. Audio	22
7.3.2. Video	22
8. Security Considerations	22
8.1. Key Management	22
8.2. Authentication tag length	22
9. IANA Considerations	23
10. References	23
10.1. Normative References	23
10.2. Informative References	23
Authors' Addresses	24

1. Introduction

Modern multi-party video call systems use Selective Forwarding Unit (SFU) servers to efficiently route RTP streams to call endpoints based on factors such as available bandwidth, desired video size, codec support, and other factors. In order for the SFU to work properly though, it needs to be able to access RTP metadata and RTCP feedback messages, which is not possible if all RTP/RTCP traffic is end-to-end encrypted.

As such, two layers of encryptions and authentication are required:

1. Hop-by-hop (HBH) encryption of media, metadata, and feedback messages between the the endpoints and SFU
2. End-to-end (E2E) encryption of media between the endpoints

While DTLS-SRTP can be used as an efficient HBH mechanism, it is inherently point-to-point and therefore not suitable for a SFU context. In addition, given the various scenarios in which video calling occurs, minimizing the bandwidth overhead of end-to-end encryption is also an important goal.

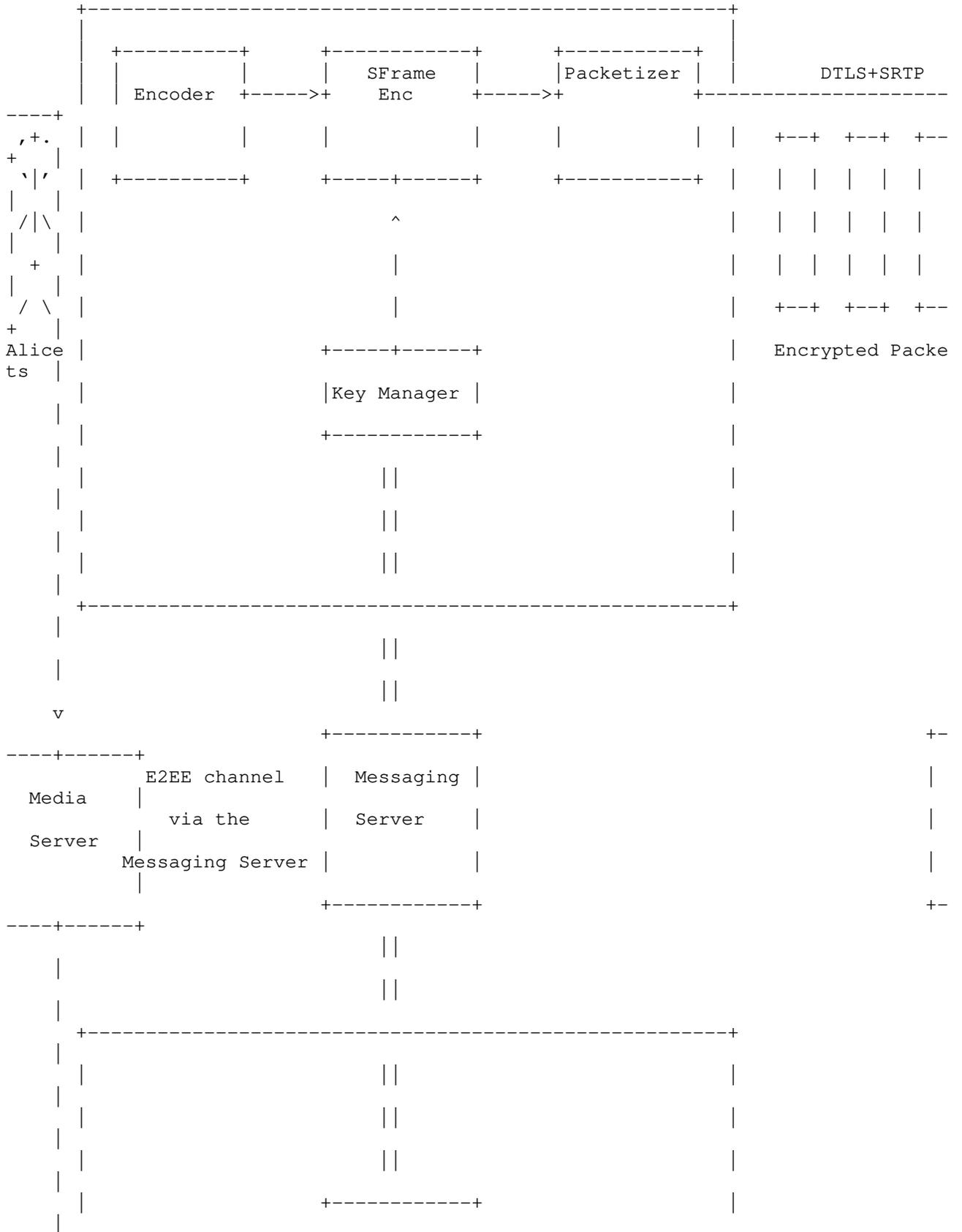
This document proposes a new end-to-end encryption mechanism known as SFrame, specifically designed to work in group conference calls with SFUs.

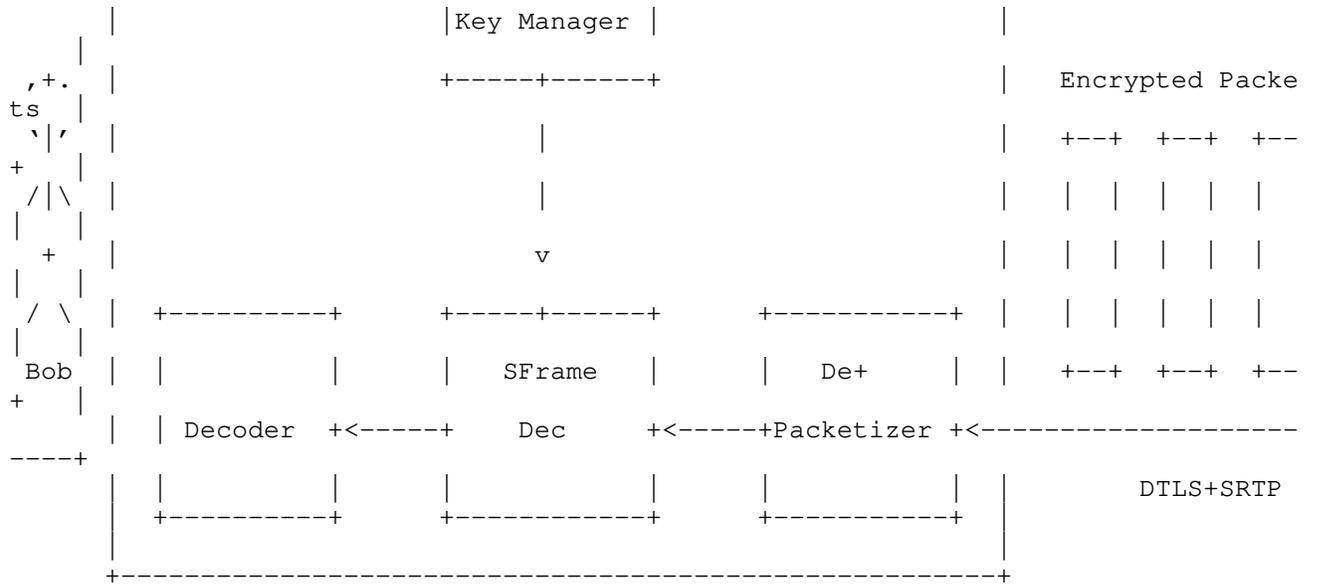
1. Provide an secure E2EE mechanism for audio and video in conference calls that can be used with arbitrary SFU servers.
 2. Decouple media encryption from key management to allow SFrame to be used with an arbitrary KMS.
 3. Minimize packet expansion to allow successful conferencing in as many network conditions as possible.
 4. Independence from the underlying transport, including use in non-RTP transports, e.g., WebTransport.
 5. When used with RTP and its associated error resilience mechanisms, i.e., RTX and FEC, require no special handling for RTX and FEC packets.
 6. Minimize the changes needed in SFU servers.
 7. Minimize the changes needed in endpoints.
 8. Work with the most popular audio and video codecs used in conferencing scenarios.
4. SFrame

We propose a frame level encryption mechanism that provides effective end-to-end encryption, is simple to implement, has no dependencies on RTP, and minimizes encryption bandwidth overhead. Because SFrame encrypts the full frame, rather than individual packets, bandwidth overhead is reduced by having a single IV and authentication tag for each media frame.

Also, because media is encrypted prior to packetization, the encrypted frame is packetized using a generic RTP packetizer instead of codec-dependent packetization mechanisms. With this move to a generic packetizer, media metadata is moved from codec-specific mechanisms to a generic frame RTP header extension which, while visible to the SFU, is authenticated end-to-end. This extension includes metadata needed for SFU routing such as resolution, frame beginning and end markers, etc.

The generic packetizer splits the E2E encrypted media frame into one or more RTP packets and adds the SFrame header to the beginning of the first packet and an auth tag to the end of the last packet.

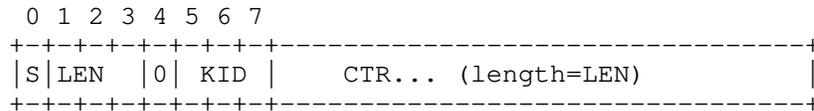




The E2EE keys used to encrypt the frame are exchanged out of band using a secure E2EE channel.

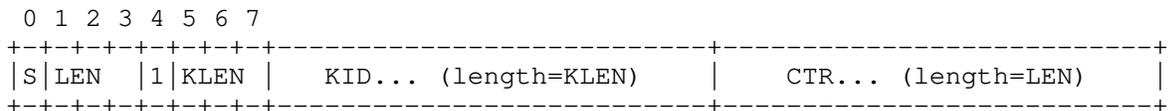
Key or Key Length: 3 bits This field contains the key id (KID) if the X flag is set to 0, or the key length (KLEN) if set to 1.

If X flag is 0 then the KID is in the range of 0-7 and the frame counter (CTR) is found in the next LEN bytes:



Key id (KID): 3 bits The key id (0-7). Frame counter (CTR): (Variable length) Frame counter value up to 8 bytes long.

if X flag is 1 then KLEN is the length of the key (KID), that is found after the SFrame header metadata byte. After the key id (KID), the frame counter (CTR) will be found in the next LEN bytes:



Key length (KLEN): 3 bits The key length in bytes. Key id (KID): (Variable length) The key id value up to 8 bytes long. Frame counter (CTR): (Variable length) Frame counter value up to 8 bytes long.

4.3. Encryption Schema

SFrame encryption uses an AEAD encryption algorithm and hash function defined by the ciphersuite in use (see Section 4.5). We will refer to the following aspects of the AEAD algorithm below:

- o "AEAD.Encrypt" and "AEAD.Decrypt" - The encryption and decryption functions for the AEAD. We follow the convention of RFC 5116 [RFC5116] and consider the authentication tag part of the ciphertext produced by "AEAD.Encrypt" (as opposed to a separate field as in SRTP [RFC3711]).
- o "AEAD.Nk" - The size of a key for the encryption algorithm, in bytes
- o "AEAD.Nn" - The size of a nonce for the encryption algorithm, in bytes

4.3.1. Key Selection

Each SFrame encryption or decryption operation is premised on a single secret "base_key", which is labeled with an integer KID value signaled in the SFrame header.

The sender and receivers need to agree on which key should be used for a given KID. The process for provisioning keys and their KID values is beyond the scope of this specification, but its security properties will bound the assurances that SFrame provides. For example, if SFrame is used to provide E2E security against intermediary media nodes, then SFrame keys MUST be negotiated in a way that does not make them accessible to these intermediaries.

For each known KID value, the client stores the corresponding symmetric key "base_key". For keys that can be used for encryption, the client also stores the next counter value CTR to be used when encrypting (initially 0).

When encrypting a frame, the application specifies which KID is to be used, and the counter is incremented after successful encryption. When decrypting, the "base_key" for decryption is selected from the available keys using the KID value in the SFrame Header.

A given key MUST NOT be used for encryption by multiple senders. Such reuse would result in multiple encrypted frames being generated with the same (key, nonce) pair, which harms the protections provided by many AEAD algorithms. Implementations SHOULD mark each key as usable for encryption or decryption, never both.

Note that the set of available keys might change over the lifetime of a real-time session. In such cases, the client will need to manage key usage to avoid media loss due to a key being used to encrypt before all receivers are able to use it to decrypt. For example, an application may make decryption-only keys available immediately, but delay the use of encryption-only keys until (a) all receivers have acknowledged receipt of the new key or (b) a timeout expires.

4.3.2. Key Derivation

SFrame encryption and decryption use a key and salt derived from the "base_key" associated to a KID. Given a "base_key" value, the key and salt are derived using HKDF [RFC5869] as follows:

```
sframe_secret = HKDF-Extract(K, 'SFrame10')
sframe_key = HKDF-Expand(sframe_secret, 'key', AEAD.Nk)
sframe_salt = HKDF-Expand(sframe_secret, 'salt', AEAD.Nn)
```

The hash function used for HKDF is determined by the ciphersuite in use.

4.3.3. Encryption

After encoding the frame and before packetizing it, the necessary media metadata will be moved out of the encoded frame buffer, to be used later in the RTP generic frame header extension. The encoded frame, the metadata buffer and the frame counter are passed to SFrame encryptor.

SFrame encryption uses the AEAD encryption algorithm for the ciphersuite in use. The key for the encryption is the "sframe_key" and the nonce is formed by XORing the "sframe_salt" with the current counter, encoded as a big-endian integer of length "AEAD.Nn".

The encryptor forms an SFrame header using the S, CTR, and KID values provided. The encoded header is provided as AAD to the AEAD encryption operation, with any frame metadata appended.

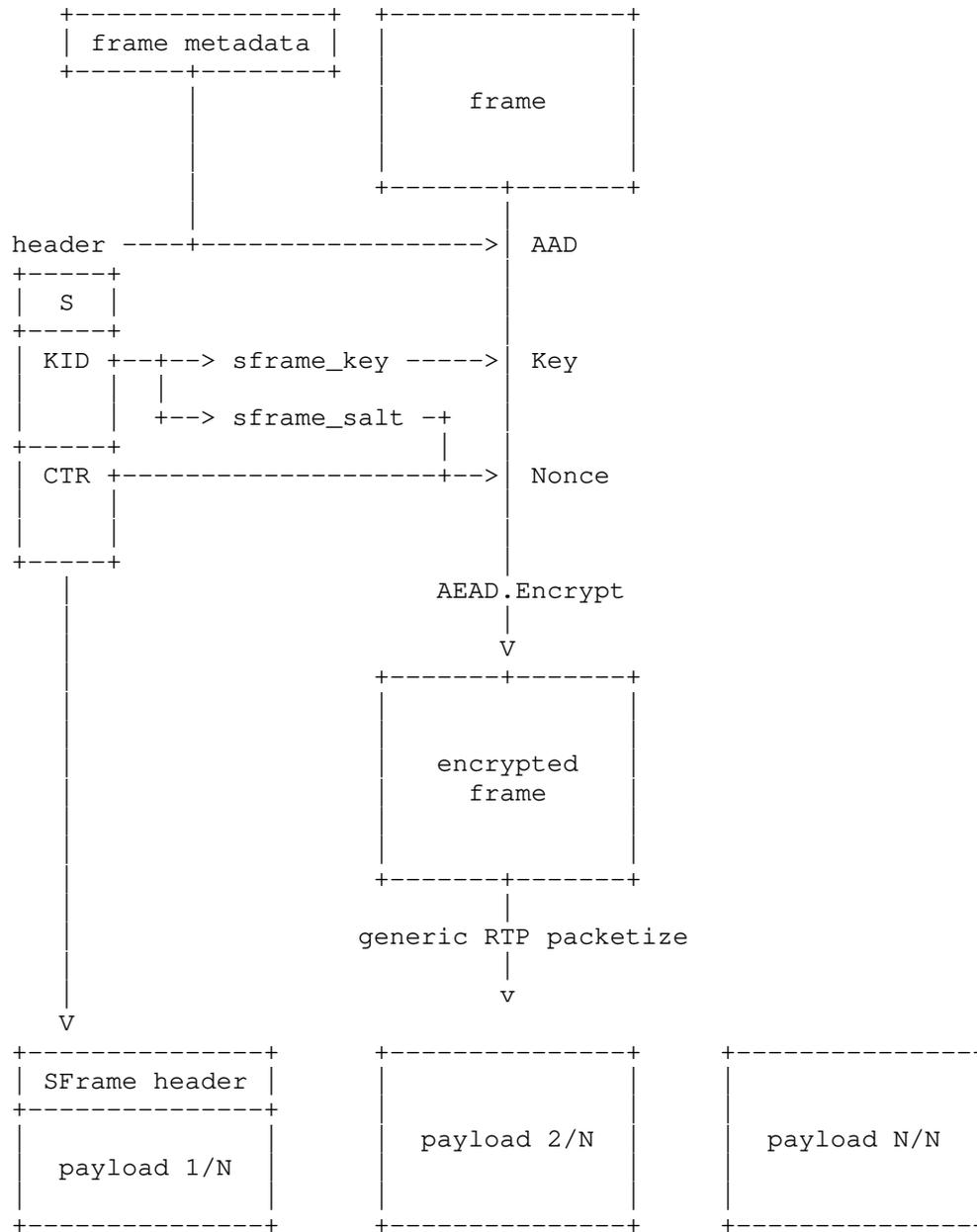
```
def encrypt(S, CTR, KID, frame_metadata, frame):
    sframe_key, sframe_salt = key_store[KID]

    frame_ctr = encode_big_endian(CTR, AEAD.Nn)
    frame_nonce = xor(sframe_salt, frame_ctr)

    header = encode_sframe_header(S, CTR, KID)
    frame_aad = header + frame_metadata

    encrypted_frame = AEAD.Encrypt(sframe_key, frame_nonce, frame_aad, frame)
    return header + encrypted_frame
```

The encrypted payload is then passed to a generic RTP packetized to construct the RTP packets and encrypt it using SRTP keys for the HBH encryption to the media server.



Encryption flow

4.3.4. Decryption

The receiving clients buffer all packets that belongs to the same frame using the frame beginning and ending marks in the generic RTP frame header extension, and once all packets are available, it passes it to SFrame for decryption. The KID field in the SFrame header is used to find the right key for the encrypted frame.

```
def decrypt(frame_metadata, sframe):
    header, encrypted_frame = split_header(sframe)
    S, CTR, KID = parse_header(header)

    sframe_key, sframe_salt = key_store[KID]

    frame_ctr = encode_big_endian(CTR, AEAD.Nn)
    frame_nonce = xor(sframe_salt, frame_ctr)
    frame_aad = header + frame_metadata

    return AEAD.Decrypt(sframe_key, frame_nonce, frame_aad, encrypted_frame)
```

For frames that are failed to decrypt because there is key available for the KID in the SFrame header, the client MAY buffer the frame and retry decryption once a key with that KID is received.

4.3.5. Duplicate Frames

Unlike messaging application, in video calls, receiving a duplicate frame doesn't necessary mean the client is under a replay attack, there are other reasons that might cause this, for example the sender might just be sending them in case of packet loss. SFrame decryptors use the highest received frame counter to protect against this. It allows only older frame pithing a short interval to support out of order delivery.

4.4. Authentication

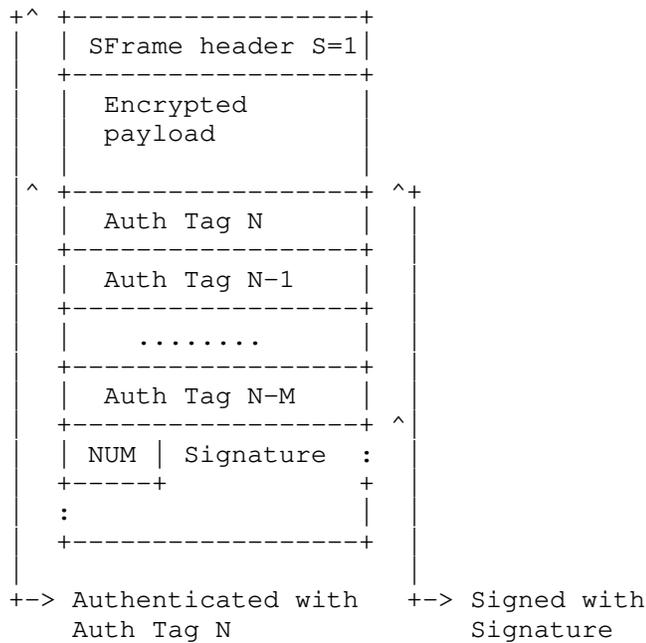
Every client in the call knows the secret key for all other clients so it can decrypt their traffic, it also means a malicious client can impersonate any other client in the call by using the victim key to encrypt their traffic. This might not be a problem for consumer application where the number of clients in the call is small and users know each others, however for enterprise use case where large conference calls are common, an authentication mechanism is needed to protect against malicious users. This authentication will come with extra cost.

Adding a digital signature to each encrypted frame will be an overkill, instead we propose adding signature over multiple frames.

The signature is calculated by concatenating the authentication tags of the frames that the sender wants to authenticate (in reverse sent order) and signing it with the signature key. Signature keys are exchanged out of band along the encryption keys.

$$\text{Signature} = \text{Sign}(\text{Key}, \text{AuthTag}(\text{Frame } N) \parallel \text{AuthTag}(\text{Frame } N-1) \parallel \dots \parallel \text{AuthTag}(\text{Frame } N-M))$$

The authentication tags for the previous frames covered by the signature and the signature itself will be appended at end of the frame, after the current frame authentication tag, in the same order that the signature was calculated, and the SFrame header metadata signature bit (S) will be set to 1.



Encrypted Frame with Signature

Note that the authentication tag for the current frame will only authenticate the SFrame header and the encrypted payload, and not the signature nor the previous frames's authentication tags (N-1 to N-M) used to calculate the signature.

The last byte (NUM) after the authentication tag list and before the signature indicates the number of the authentication tags from previous frames present in the current frame. All the authentications tags MUST have the same size, which MUST be equal to

the authentication tag size of the current frame. The signature is fixed size depending on the signature algorithm used (for example, 64 bytes for Ed25519).

The receiver has to keep track of all the frames received but yet not verified, by storing the authentication tags of each received frame. When a signature is received, the receiver will verify it with the signature key associated to the key id of the frame the signature was sent in. If the verification is successful, the received will mark the frames as authenticated and remove them from the list of the not verified frames. It is up to the application to decide what to do when signature verification fails.

When using SVC, the hash will be calculated over all the frames of the different spatial layers within the same superframe/picture. However the SFU will be able to drop frames within the same stream (either spatial or temporal) to match target bitrate.

If the signature is sent on a frame which layer that is dropped by the SFU, the receiver will not receive it and will not be able to perform the signature of the other received layers.

An easy way of solving the issue would be to perform signature only on the base layer or take into consideration the frame dependency graph and send multiple signatures in parallel (each for a branch of the dependency graph).

In case of simulcast or K-SVC, each spatial layer should be authenticated with different signatures to prevent the SFU to discard frames with the signature info.

In any case, it is possible that the frame with the signature is lost or the SFU drops it, so the receiver MUST be prepared to not receive a signature for a frame and remove it from the pending to be verified list after a timeout.

4.5. Ciphersuites

Each SFrame session uses a single ciphersuite that specifies the following primitives:

- o A hash function used for key derivation and hashing signature inputs
- o An AEAD encryption algorithm [RFC5116] used for frame encryption, optionally with a truncated authentication tag
- o [Optional] A signature algorithm

This document defines the following ciphersuites:

Value	Name	Nk	Nn	Reference
0x0001	AES_CM_128_HMAC_SHA256_8	16	12	RFC XXXX
0x0002	AES_CM_128_HMAC_SHA256_4	16	12	RFC XXXX
0x0003	AES_GCM_128_SHA256	16	12	RFC XXXX
0x0004	AES_GCM_256_SHA512	32	12	RFC XXXX

In the "AES_CM" suites, the length of the authentication tag is indicated by the last value: "_8" indicates an eight-byte tag and "_4" indicates a four-byte tag.

In a session that uses multiple media streams, different ciphersuites might be configured for different media streams. For example, in order to conserve bandwidth, a session might use a ciphersuite with 80-bit tags for video frames and another ciphersuite with 32-bit tags for audio frames.

4.5.1. AES-CM with SHA2

In order to allow very short tag sizes, we define a synthetic AEAD function using the authenticated counter mode of AES together with HMAC for authentication. We use an encrypt-then-MAC approach as in SRTP [RFC3711].

Before encryption or decryption, encryption and authentication subkeys are derived from the single AEAD key using HKDF. The subkeys are derived as follows, where "Nk" represents the key size for the AES block cipher in use and "Nh" represents the output size of the hash function:

```
def derive_subkeys(key):
    aead_secret = HKDF-Extract(K, 'SFrame10 AES CM AEAD')
    enc_key = HKDF-Expand(aead_secret, 'enc', Nk)
    auth_key = HKDF-Expand(aead_secret, 'auth', Nh)
```

The AEAD encryption and decryption functions are then composed of individual calls to the CM encrypt function and HMAC. The resulting MAC value is truncated to a number of bytes "tag_len" fixed by the ciphersuite.

```
def compute_tag(aad, ct):
    aad_len = encode_big_endian(len(aad), 8)
    auth_data = aad_len + aad + ct
    tag = HMAC(auth_key, auth_data)
    return truncate(tag, tag_len)

def AEAD.Encrypt(key, nonce, aad, pt):
    ct = AES-CM.Encrypt(key, nonce, pt)
    tag = compute_tag(aad, ct)
    return ct + tag

def AEAD.Decrypt(key, nonce, aad, ct):
    inner_ct, tag = split_ct(ct, tag_len)

    candidate_tag = compute_tag(aad, inner_ct)
    if !constant_time_equal(tag, candidate_tag):
        raise Exception("Authentication Failure")

    return AES-CM.Decrypt(key, nonce, inner_ct)
```

5. Key Management

SFrame must be integrated with an E2E key management framework to exchange and rotate the keys used for SFrame encryption and/or signing. The key management framework provides the following functions:

- o Provisioning KID/"base_key" mappings to participating clients
- o (optional) Provisioning clients with a list of trusted signing keys
- o Updating the above data as clients join or leave

It is up to the application to define a rotation schedule for keys. For example, one application might have an ephemeral group for every call and keep rotating key when end points joins or leave the call, while another application could have a persistent group that can be used for multiple calls and simply derives ephemeral symmetric keys for a specific call.

5.1. Sender Keys

If the participants in a call have a pre-existing E2E-secure channel, they can use it to distribute SFrame keys. Each client participating in a call generates a fresh encryption key and optionally a signing key pair. The client then uses the E2E-secure channel to send their encryption key and signing public key to the other participants.

In this scheme, it is assumed that receivers have a signal outside of SFrame for which client has sent a given frame, for example the RTP SSRC. SFrame KID values are then used to distinguish generations of the sender's key. At the beginning of a call, each sender encrypts with KID=0. Thereafter, the sender can ratchet their key forward for forward secrecy:

```
sender_key[i+1] = HKDF-Expand(  
    HKDF-Extract(sender_key[i], 'SFrame10 ratchet'),  
    '', AEAD.Nk)
```

The sender signals such an update by incrementing their KID value. A receiver who receives from a sender with a new KID computes the new key as above. The old key may be kept for some time to allow for out-of-order delivery, but should be deleted promptly.

If a new participant joins mid-call, they will need to receive from each sender (a) the current sender key for that sender, (b) the signing key for the sender, if used, and (c) the current KID value for the sender. Evicting a participant requires each sender to send a fresh sender key to all receivers.

5.2. MLS

The Messaging Layer Security (MLS) protocol provides group authenticated key exchange [I-D.ietf-mls-architecture] [I-D.ietf-mls-protocol]. In principle, it could be used to instantiate the sender key scheme above, but it can also be used more efficiently directly.

MLS creates a linear sequence of keys, each of which is shared among the members of a group at a given point in time. When a member joins or leaves the group, a new key is produced that is known only to the augmented or reduced group. Each step in the lifetime of the group is known as an "epoch", and each member of the group is assigned an "index" that is constant for the time they are in the group.

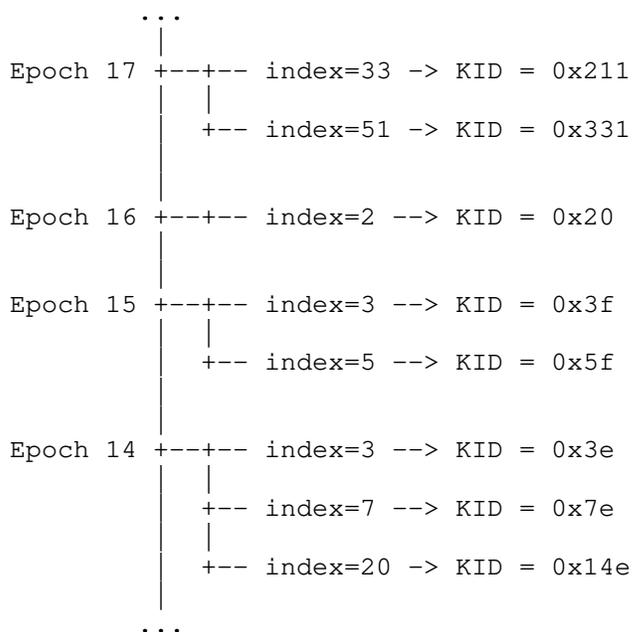
In SFrame, we derive per-sender "base_key" values from the group secret for an epoch, and use the KID field to signal the epoch and sender index. First, we use the MLS exporter to compute a shared SFrame secret for the epoch.

```
sframe_epoch_secret = MLS-Exporter("SFrame 10 MLS", "", AEAD.Nk)  
  
sender_base_key[index] = HKDF-Expand(sframe_epoch_secret,  
    encode_big_endian(index, 4), AEAD.Nk)
```

For compactness, do not send the whole epoch number. Instead, we send only its low-order E bits. Note that E effectively defines a re-ordering window, since no more than 2^E epoch can be active at a given time. Receivers MUST be prepared for the epoch counter to roll over, removing an old epoch when a new epoch with the same E lower bits is introduced. (Sender indices cannot be similarly compressed.)

$$\text{KID} = (\text{sender_index} \ll E) + (\text{epoch} \% (1 \ll E))$$

Once an SFrame stack has been provisioned with the "sframe_epoch_secret" for an epoch, it can compute the required KIDs and "sender_base_key" values on demand, as it needs to encrypt/decrypt for a given member.



MLS also provides an authenticated signing key pair for each participant. When SFrame uses signatures, these are the keys used to generate SFrame signatures.

6. Media Considerations

6.1. SFU

Selective Forwarding Units (SFUs) as described in <https://tools.ietf.org/html/rfc7667#section-3.7> receives the RTP streams from each participant and selects which ones should be

forwarded to each of the other participants. There are several approaches about how to do this stream selection but in general, in order to do so, the SFU needs to access metadata associated to each frame and modify the RTP information of the incoming packets when they are transmitted to the received participants.

This section describes how this normal SFU modes of operation interacts with the E2EE provided by SFrame

6.1.1. LastN and RTP stream reuse

The SFU may choose to send only a certain number of streams based on the voice activity of the participants. To reduce the number of SDP O/A required to establish a new RTP stream, the SFU may decide to reuse previously existing RTP sessions or even pre-allocate a predefined number of RTP streams and choose in each moment in time which participant media will be sending through it. This means that in the same RTP stream (defined by either SSRC or MID) may carry media from different streams of different participants. As different keys are used by each participant for encoding their media, the receiver will be able to verify which is the sender of the media coming within the RTP stream at any given point if time, preventing the SFU trying to impersonate any of the participants with another participant's media. Note that in order to prevent impersonation by a malicious participant (not the SFU) usage of the signature is required. In case of video, the a new signature should be started each time a key frame is sent to allow the receiver to identify the source faster after a switch.

6.1.2. Simulcast

When using simulcast, the same input image will produce N different encoded frames (one per simulcast layer) which would be processed independently by the frame encryptor and assigned an unique counter for each.

6.1.3. SVC

In both temporal and spatial scalability, the SFU may choose to drop layers in order to match a certain bitrate or forward specific media sizes or frames per second. In order to support it, the sender MUST encode each spatial layer of a given picture in a different frame. That is, an RTP frame may contain more than one SFrame encrypted frame with an incrementing frame counter.

6.2. Video Key Frames

Forward and Post-Compromise Security requires that the e2ee keys are updated anytime a participant joins/leave the call.

The key exchange happens async and on a different path than the SFU signaling and media. So it may happen that when a new participant joins the call and the SFU side requests a key frame, the sender generates the e2ee encrypted frame with a key not known by the receiver, so it will be discarded. When the sender updates his sending key with the new key, it will send it in a non-key frame, so the receiver will be able to decrypt it, but not decode it.

Receiver will re-request an key frame then, but due to sender and sfu policies, that new key frame could take some time to be generated.

If the sender sends a key frame when the new e2ee key is in use, the time required for the new participant to display the video is minimized.

6.3. Partial Decoding

Some codes support partial decoding, where it can decrypt individual packets without waiting for the full frame to arrive, with SFrame this won't be possible because the decoder will not access the packets until the entire frame is arrived and decrypted.

7. Overhead

The encryption overhead will vary between audio and video streams, because in audio each packet is considered a separate frame, so it will always have extra MAC and IV, however a video frame usually consists of multiple RTP packets. The number of bytes overhead per frame is calculated as the following $1 + \text{FrameCounter length} + 4$ The constant 1 is the SFrame header byte and 4 bytes for the HBH authentication tag for both audio and video packets.

7.1. Audio

Using three different audio frame durations 20ms (50 packets/s) 40ms (25 packets/s) 100ms (10 packets/s) Up to 3 bytes frame counter (3.8 days of data for 20ms frame duration) and 4 bytes fixed MAC length.

Counter len	Packets	Overhead bps@20ms	Overhead bps@40ms	Overhead bps@100ms
1	0-255	2400	1200	480
2	255 - 65K	2800	1400	560
3	65K - 16M	3200	1600	640

7.2. Video

The per-stream overhead bits per second as calculated for the following video encodings: 30fps@1000Kbps (4 packets per frame) 30fps@512Kbps (2 packets per frame) 15fps@200Kbps (2 packets per frame) 7.5fps@30Kbps (1 packet per frame) Overhead bps = (Counter length + 1 + 4) * 8 * fps

Counter len	Frames	Overhead bps@30fps	Overhead bps@15fps	Overhead bps@7.5fps
1	0-255	1440	1440	720
2	256 - 65K	1680	1680	840
3	56K - 16M	1920	1920	960
4	16M - 4B	2160	2160	1080

7.3. SFrame vs PERC-lite

[RFC8723] has significant overhead over SFrame because the overhead is per packet, not per frame, and OHB (Original Header Block) which duplicates any RTP header/extension field modified by the SFU.

[I-D.murillo-perc-lite] <<https://mailarchive.ietf.org/arch/msg/perc/SB0qMHWz6EsDtz3yIEX0HWp5IEY/>> is slightly better because it doesn't use the OHB anymore, however it still does per packet encryption using SRTP. Below the the overhead in [I-D.murillo-perc-lite] implemented by Cosmos Software which uses extra 11 bytes per packet to preserve the PT, SEQ_NUM, TIME_STAMP and SSRC fields in addition to the extra MAC tag per packet.

OverheadPerPacket = 11 + MAC length Overhead bps = PacketPerSecond * OverHeadPerPacket * 8

Similar to SFrame, we will assume the HBH authentication tag length will always be 4 bytes for audio and video even though it is not the case in this [I-D.murillo-perc-lite] implementation

7.3.1. Audio

Overhead bps@20ms	Overhead bps@40ms	Overhead bps@100ms
6000	3000	1200

7.3.2. Video

Overhead bps@30fps	Overhead bps@15fps	Overhead bps@7.5fps
(4 packets per frame)	(2 packets per frame)	(1 packet per frame)
14400	7200	3600

For a conference with a single incoming audio stream (@ 50 pps) and 4 incoming video streams (@200 Kbps), the savings in overhead is 34800 - 9600 = ~25 Kbps, or ~3%.

8. Security Considerations

8.1. Key Management

Key exchange mechanism is out of scope of this document, however every client MUST change their keys when new clients joins or leaves the call for "Forward Secrecy" and "Post Compromise Security".

8.2. Authentication tag length

The cipher suites defined in this draft use short authentication tags for encryption, however it can easily support other ciphers with full authentication tag if the short ones are proved insecure.

9. IANA Considerations

This document makes no requests of IANA.

10. References

10.1. 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>>.
- [RFC5116] McGrew, D., "An Interface and Algorithms for Authenticated Encryption", RFC 5116, DOI 10.17487/RFC5116, January 2008, <<https://www.rfc-editor.org/info/rfc5116>>.
- [RFC5869] Krawczyk, H. and P. Eronen, "HMAC-based Extract-and-Expand Key Derivation Function (HKDF)", RFC 5869, DOI 10.17487/RFC5869, May 2010, <<https://www.rfc-editor.org/info/rfc5869>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.

10.2. Informative References

- [I-D.ietf-mls-architecture] Omara, E., Beurdouche, B., Rescorla, E., Inguva, S., Kwon, A., and A. Duric, "The Messaging Layer Security (MLS) Architecture", draft-ietf-mls-architecture-05 (work in progress), July 2020.
- [I-D.ietf-mls-protocol] Barnes, R., Beurdouche, B., Millican, J., Omara, E., Cohn-Gordon, K., and R. Robert, "The Messaging Layer Security (MLS) Protocol", draft-ietf-mls-protocol-10 (work in progress), October 2020.
- [I-D.murillo-perc-lite] Murillo, S. and A. Gouaillard, "End to End Media Encryption Procedures", draft-murillo-perc-lite-01 (work in progress), May 2020.

- [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>>.
- [RFC8723] Jennings, C., Jones, P., Barnes, R., and A. Roach, "Double Encryption Procedures for the Secure Real-Time Transport Protocol (SRTP)", RFC 8723, DOI 10.17487/RFC8723, April 2020, <<https://www.rfc-editor.org/info/rfc8723>>.

Authors' Addresses

Emad Omara
Google

Email: emadomara@google.com

Justin Uberti
Google

Email: juberti@google.com

Alexandre GOUAILLARD
CoSMo Software

Email: Alex.GOUAILLARD@cosmosoftware.io

Sergio Garcia Murillo
CoSMo Software

Email: sergio.garcia.murillo@cosmosoftware.io

AVTCORE
Internet-Draft
Intended status: Standards Track
Expires: 6 May 2021

J. Uberti
Google
C. Jennings
Cisco
2 November 2020

Completely Encrypting RTP Header Extensions and Contributing Sources
draft-uberti-avtcore-cryptex-01

Abstract

While the Secure Real-time Transport Protocol (SRTP) provides confidentiality for the contents of a media packet, a significant amount of metadata is left unprotected, including RTP header extensions and contributing sources (CSRCs). However, this data can be moderately sensitive in many applications. While there have been previous attempts to protect this data, they have had limited deployment, due to complexity as well as technical limitations.

This document proposes a new mechanism to completely encrypt header extensions and CSRCs as well a simpler signaling mechanism intended to facilitate deployment.

Discussion Venues

This note is to be removed before publishing as an RFC.

Source for this draft and an issue tracker can be found at <https://github.com/juberti/cryptex>.

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 6 May 2021.

Copyright Notice

Copyright (c) 2020 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 2
 - 1.1. Problem Statement 3
 - 1.2. Previous Solutions 3
 - 1.3. Goals 4
- 2. Terminology 5
- 3. Design 5
- 4. Signaling 5
- 5. RTP Header Processing 5
 - 5.1. Sending 6
 - 5.2. Receiving 6
- 6. Encryption and Decryption 7
 - 6.1. Packet Structure 7
 - 6.2. Encryption Procedure 7
 - 6.3. Decryption Procedure 8
- 7. Backwards Compatibility 8
- 8. Security Considerations 8
- 9. IANA Considerations 9
- 10. Acknowledgements 9
- 11. References 9
 - 11.1. Normative References 9
 - 11.2. Informative References 10
- Authors' Addresses 10

1. Introduction

1.1. Problem Statement

The Secure Real-time Transport Protocol [RFC3711] mechanism provides message authentication for the entire RTP packet, but only encrypts the RTP payload. This has not historically been a problem, as much of the information carried in the header has minimal sensitivity (e.g., RTP timestamp); in addition, certain fields need to remain as cleartext because they are used for key scheduling (e.g., RTP SSRC and sequence number).

However, as noted in [RFC6904], the security requirements can be different for information carried in RTP header extensions, including the per-packet sound levels defined in [RFC6464] and [RFC6465], which are specifically noted as being sensitive in the Security Considerations section of those RFCs.

In addition to the contents of the header extensions, there are now enough header extensions in active use that the header extension identifiers themselves can provide meaningful information in terms of determining the identity of endpoint and/or application. Accordingly, these identifiers can be considered at least slightly sensitive.

Finally, the CSRCs included in RTP packets can also be sensitive, potentially allowing a network eavesdropper to determine who was speaking and when during an otherwise secure conference call.

1.2. Previous Solutions

[RFC6904] was proposed in 2013 as a solution to the problem of unprotected header extension values. However, it has not seen significant adoption, and has a few technical shortcomings.

First, the mechanism is complicated. Since it allows encryption to be negotiated on a per-extension basis, a fair amount of signaling logic is required. And in the SRTP layer, a somewhat complex transform is required to allow only the selected header extension values to be encrypted. One of the most popular SRTP implementations had a significant bug in this area that was not detected for five years.

Second, it only protects the header extension values, and not their ids or lengths. It also does not protect the CSRCs. As noted above, this leaves a fair amount of potentially sensitive information exposed.

Third, it bloats the header extension space. Because each extension must be offered in both unencrypted and encrypted forms, twice as many header extensions must be offered, which will in many cases push implementations past the 14-extension limit for the use of one-byte extension headers defined in [RFC8285]. Accordingly, implementations will need to use two-byte headers in many cases, which are not supported well by some existing implementations.

Finally, the header extension bloat combined with the need for backwards compatibility results in additional wire overhead. Because two-byte extension headers may not be handled well by existing implementations, one-byte extension identifiers will need to be used for the unencrypted (backwards compatible) forms, and two-byte for the encrypted forms. Thus, deployment of [RFC6904] encryption for header extensions will typically result in multiple extra bytes in each RTP packet, compared to the present situation.

1.3. Goals

From this analysis we can state the desired properties of a solution:

- * Build on existing [RFC3711] SRTP framework (simple to understand)
- * Build on existing [RFC8285] header extension framework (simple to implement)
- * Protection of header extension ids, lengths, and values
- * Protection of CSRCs when present
- * Simple signaling
- * Simple crypto transform and SRTP interactions
- * Backward compatible with unencrypted endpoints, if desired
- * Backward compatible with existing RTP tooling

The last point deserves further discussion. While we considered possible solutions that would have encrypted more of the RTP header (e.g., the number of CSRCs), we felt the inability to parse the resultant packets with current tools, as well as additional complexity incurred, outweighed the slight improvement in confidentiality.

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. Design

This specification proposes a mechanism to negotiate encryption of all RTP header extensions (ids, lengths, and values) as well as CSRC values. It reuses the existing SRTP framework, is accordingly simple to implement, and is backward compatible with existing RTP packet parsing code, even when support for this mechanism has been negotiated.

4. Signaling

In order to determine whether this mechanism defined in this specification is supported, this document defines a new "a=extmap-encrypted" Session Description Protocol (SDP) [RFC4566] attribute to indicate support. This attribute takes no value, and can be used at the session level or media level. Offering this attribute indicates that the endpoint is capable of receiving RTP packets encrypted as defined below.

The formal definition of this attribute is:

Name: extmap-encrypted

Value: None

Usage Level: session, media

Charset Dependent: No

Example:

a=extmap-encrypted

When used with BUNDLE, this attribute is specified as the TRANSPORT category. (todo: REF)

5. RTP Header Processing

[RFC8285] defines two values for the "defined by profile" field for carrying one-byte and two-byte header extensions. In order to allow a receiver to determine if an incoming RTP packet is using the encryption scheme in this specification, two new values are defined:

- * 0xC0DE for the encrypted version of the one-byte header extensions (instead of 0xBEDE).
- * 0xC2DE for the encrypted versions of the two-byte header extensions (instead of 0x100).

In the case of using two-byte header extensions, the extension id with value 256 MUST NOT be negotiated, as the value of this id is meant to be contained in the "appbits" of the "defined by profile" field, which are not available when using the values above.

If the "a=extmap-allow-mixed" attribute defined in [RFC8285] is negotiated, either one-byte or two-byte header ids can be used (with the values above), as in [RFC8285].

5.1. Sending

When sending an RTP packet that requires any header extensions to a destination that has negotiated header encryption, the header extensions MUST be formatted as [RFC8285] header extensions, as usual.

If one-byte extension ids are in use, the 16-bit RTP header extension tag MUST be set to 0xC0DE to indicate that the encryption defined in this specification has been applied. If two-byte header extension codes are in use, the 16-bit RTP header extension tag MUST be set to 0xC2DE to indicate the same.

The RTP packet MUST then be encrypted as described in Encryption Procedure.

5.2. Receiving

When receiving an RTP packet that contains header extensions, the "defined by profile" field MUST be checked to ensure the payload is formatted according to this specification. If the field does not match one of the values defined above, the implementation MUST instead handle it according to the specification that defines that value. The implementation MAY stop and report an error if it considers use of this specification mandatory for the RTP stream.

If the RTP packet passes this check, it is then decrypted according to Decryption Procedure, and passed to the the next layer to process the packet and its extensions.

To minimize changes to surrounding code, the encryption mechanism can choose to replace a "defined by profile" field from [RFC8285] with its counterpart defined in RTP Header Processing above and encrypt at the same time.

6.3. Decryption Procedure

The decryption procedure is identical to that of [RFC3711] except for the region to decrypt, which is as shown in the section above.

To minimize changes to surrounding code, the decryption mechanism can choose to replace the "defined by profile" field with its no-encryption counterpart from [RFC8285] and decrypt at the same time.

7. Backwards Compatibility

This specification attempts to encrypt as much as possible without interfering with backwards compatibility for systems that expect a certain structure from an RTPv2 packet, including systems that perform demultiplexing based on packet headers. Accordingly, the first two bytes of the RTP packet are not encrypted.

This specification also attempts to reuse the key scheduling from SRTP, which depends on the RTP packet sequence number and SSRC identifier. Accordingly these values are also not encrypted.

8. Security Considerations

This specification extends SRTP by expanding the portion of the packet that is encrypted, as shown in Packet Structure. It does not change how SRTP authentication works in any way. Given that more of the packet is being encrypted than before, this is necessarily an improvement.

The RTP fields that are left unencrypted (see rationale above) are as follows:

- * RTP version
- * padding bit
- * extension bit
- * number of CSRCs
- * marker bit
- * payload type

- * sequence number
- * timestamp
- * SSRC identifier
- * number of [RFC8285] header extensions

These values contain a fixed set (i.e., one that won't be changed by extensions) of information that, at present, is observed to have low sensitivity. In the event any of these values need to be encrypted, SRTP is likely the wrong protocol to use and a fully-encapsulating protocol such as DTLS is preferred (with its attendant per-packet overhead).

9. IANA Considerations

This document defines two new 'defined by profile' attributes, as noted in RTP Header Processing.

10. Acknowledgements

The authors wish to thank Sergio Murillo, Jonathan Lennox, and Inaki Castillo for their review and text suggestions.

11. References

11.1. 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>>.
- [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>>.
- [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>>.
- [RFC8285] Singer, D., Desineni, H., and R. Even, Ed., "A General Mechanism for RTP Header Extensions", RFC 8285, DOI 10.17487/RFC8285, October 2017, <<https://www.rfc-editor.org/info/rfc8285>>.

11.2. Informative References

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

Authors' Addresses

Justin Uberti
Google

Email: justin@uberti.name

Cullen Jennings
Cisco

Email: fluffy@iii.ca