avtcore
Internet-Draft
Intended status: Standards Track
Expires: 5 September 2024

HS Yang
X. de Foy
InterDigital

4 March 2024

RTP Payload for Haptics
draft-hsyang-avtcore-rtp-haptics-02

Abstract

This memo describes an RTP payload format for the MPEG-I haptic data. A haptic media stream is composed of MIHS units including a MIHS(MPEG-I Haptic Stream) unit header and zero or more MIHS packets. The RTP payload header format allows for packetization of a MIHS unit in an RTP packet payload as well as fragmentation of a MIHS unit into multiple RTP packets.

Table of Contents

1. Introduction

Haptics provides users with tactile effects in addition to audio and video, allowing them to experience sensory immersion. Haptic data is mainly transmitted to devices that act as actuators and provides them with information to operate according to the values defined in haptic effects. The IETF is registering haptics as a primary media type akin to audio and video [I-D.ietf-mediaman-haptics].

The MPEG Haptics Coding standard [ISO.IEC.23090-31] defines the data formats, metadata, and codec architecture to encode, decode, synthesize and transmit haptic signals. It defines the "MIHS unit" as a unit of packetization suitable for streaming, and similar in essence to the NAL unit defined in some video specifications. This document describes how haptic data (MIHS units) can be transmitted using the RTP protocol. This document followed recommendations in [RFC8088] and [RFC2736] for RTP payload format writers.

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 here.

3. Definition

This document uses the definitions of the MPEG Haptics Coding standard [ISO.IEC.23090-31]. Some of these terms are provided here for convenience.

Actuator: component of a device for rendering haptic sensations.

Avatar: body (or part of body) representation.

Band: component in a channel for containing effects for a specific range of frequencies.

Channel: component in a perception containing one or more bands rendered on a device at a specific body location.

Device: physical system having one or more actuators configured to render a haptic sensation corresponding with a given signal.

Effect: component of a band for defining a signal, consisting of a haptic waveform or one or more haptic keyframes.

Experience: top level haptic component containing perceptions and metadata.

Haptics: tactile sensations.

Keyframe: component of an effect mapping a position in time or space to an effect parameter such as amplitude or frequency.

Metadata: global information about an experience, perception, channel, or band.

MIHS unit: unit of packetization of the MPEG-I Haptic Stream format, which is used as unit of payload in the format described in this memo. See Section 4 for details.

Modality: type of haptics, such as vibration, force, pressure, position, velocity, or temperature.

Perception: haptic perception containing channels of a specific modality.

Signal: representation of the haptics associated with a specific modality to be rendered on a device.

Hmpg format: hmpg is a binary compressed format for haptics data. Information is stored in a binary form and data compression is applied on data at the band level. The haptics/hmpg media subtype is registered in [I-D.ietf-mediaman-haptics] and updated by this memo.

Independent unit: a MIHS unit is independent if it can be decoded independently from earlier units. Independent units contain timing information and are also called "sync units" in [ISO.IEC.23090-31].

Dependent unit: a MIHS unit is dependent if it requires earlier units for decoding. Dependent units do not contain timing information and are also called "non-sync units" in [ISO.IEC.23090-31].

Time-independent effect: a haptic effect that occurs regardless of time. The tactile feedback of a texture is a representative example. Time-independent effects are encoded in spatial MIHS units, defined in Section 4.2.

Time-dependent effect: a haptic effect that varies over time. For example, tactile feedback for vibration and force are time-dependent effects, and are encoded in temporal MIHS units, defined in Section 4.2.

4. Haptic Format Description

4.1. Overview of Haptic Coding

The MPEG Haptics Coding standard specifies methods for efficient transmission and rendering of haptic signals, to enable immersive experiences. It supports multiple types of perceptions, including the most common vibrotactile (sense of touch that perceives vibrations) and kinaesthetic perceptions (tactile resistance or force), but also other, less common perceptions, including for example the sense of temperature or texture. It also supports two approaches for encoding haptic signals: a "quantized" approach based on samples of measured data, and a "descriptive" approach where the signal is synthesized using a combination of functions. Both quantized and descriptive data can be encoded in a human-readable exchange format based on JSON (.hjif), or in a binary packetized format for distribution and streaming (.hmpg). This last format is referred to as the MPEG-I Haptic Stream (MIHS) format and is a base for the RTP payload format described in this document.

4.2. MPEG-I Haptic Stream (MIHS) format

MIHS is a stream format used to transport haptic data. Haptic data including haptic effects is packetized according to the MIHS format, and delivered to actuators, which operate according to the provided effects. The MIHS format has two level packetization, MIHS units and MIHS packets.

MIHS units are composed of a MIHS unit header and zero or more MIHS packets. Four types of MIHS units are defined. An initialization MIHS unit contains MIHS packets carrying metadata necessary to reset and initialize a haptic decoder, including a timestamp. A temporal MIHS unit contains one or more MIHS packets defining time-dependent effects and providing modalities such as pressure, velocity, and acceleration. The duration of a temporal unit is a positive number. A spatial MIHS unit contains one or more MIHS packets providing time-independent effects, such as vibrotactile texture, stiffness, and friction. The duration of a spatial unit is always zero. A silent MIHS unit indicates that there is no effect during a time interval and its duration is a positive number.

A MIHS unit can be marked as independent or dependent. When a decoder processes an independent unit, it resets the previous effects and therefore provides a haptic experience independent from any previous MIHS unit. A dependent unit is the continuation of previous MIHS units and cannot be independently decoded and rendered without having decoded previous MIHS unit(s). Initialization and spatial MIHS units are always independent units. Temporal and silent MIHS units can be dependent or independent units.

Figure 1 illustrates a succession of MIHS units in a MIHS stream.

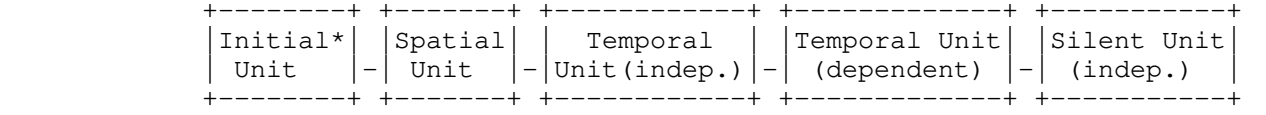


Figure 1: Example of MIHS stream

5. Payload format for haptics

5.1. RTP header Usage

The RTP header is defined in [RFC3550] and represented in Figure 2. Some of the header field values are interpreted as follows.

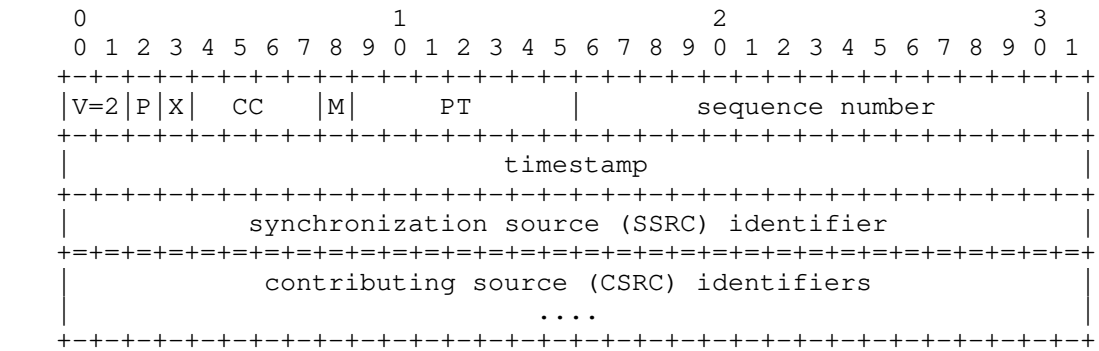


Figure 2: RTP header for Haptic.

Payload type (PT): 7 bits. The assignment of a payload type MUST be performed either through the profile used or in a dynamic way.

Time Stamp (TS): 32 bits. A timestamp representing the sampling time of the first sample of the MIHS unit in the RTP payload. The clock frequency MUST be set to the sample rate of the encoded haptic data and is conveyed out-of-band (e.g., as an SDP parameter).

Marker bit (M): 1 bit. The marker bit SHOULD be set to one in the first non-silent RTP packet after a period of haptic silence. This enables jitter buffer adaptation and haptics device washout (i.e., reset to a neutral position) prior to the beginning of the burst with minimal impact on the quality of experience for the end user. The marker bit in all other packets MUST be set to zero.

5.2. Payload Header

The RTP Payload Header follows the RTP header. Figure 3 describes RTP Payload Header.

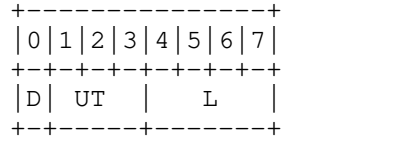


Figure 3: RTP payload header for Haptic.

D (Dependency, 1 bit): this field is used to indicate whether the MIHS unit included in the RTP payload is, when its value is one, dependent or, when its value is zero, independent.

UT (Unit Type, 3 bits): this field indicates the type of the MIHS unit included in the RTP payload. UT field values are listed in Figure 4.

L (MIHS Layer, 4 bits): this field is an integer value which indicates the priority order of the MIHS unit included in the RTP payload, as determined by the haptic sender (e.g., by the haptic codec), based on application-specific needs. For example, the sender may use the MIHS layer to prioritize perceptions with the largest impact on the end-user experience. Zero corresponds to the highest priority. The semantic of individual MIHS layers is not specified and left for the application to assign.

5.3. Payload Structures

Two different types of RTP packet payload structures are specified. The single unit payload structure contains a single MIHS unit. The fragmented unit payload structure contains a subset of a MIHS unit. The unit type (UT) field of the RTP payload header Figure 4 identifies both the payload structure and, in the case of a single unit structure, also identifies the type of MIHS unit present in the payload.

Editor's Note: consider if it would be useful to add the ability to aggregate multiple MIHS units in a single RTP payload - for instance, to aggregate multiple MIHS units with different layer values into a single RTP payload .

Unit Type	Payload Structure	Name
0	N/A	Reserved
1	Single	Initialization MIHS Unit
2	Single	Temporal MIHS Unit
3	Single	Spatial MIHS Unit
4	Single	Silent MIHS Unit
7	Frag	Fragmented Packet

Figure 4: Payload structure type for haptic

The payload structures are represented in Figure 4. The single unit payload structure is specified in Section 5.3.1. The fragmented unit payload structure is specified in Section 5.3.2.

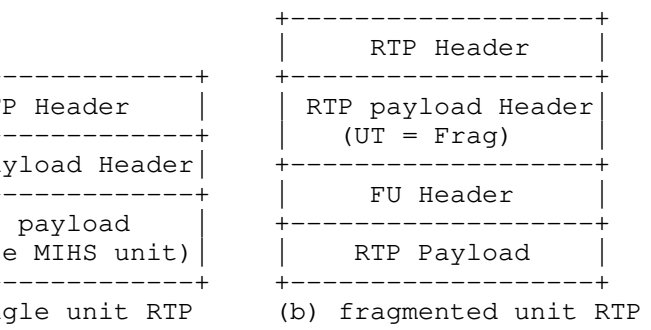


Figure 5: RTP Transmission mode

5.3.1. Single Unit Payload Structure

In a single unit payload structure, as described in Figure 5, the RTP packet contains the RTP header, followed by the payload header and one single MIHS unit. The payload header follows the structure described in Section 5.2. The payload contains a MIHS unit as defined in [ISO.IEC.23090-31].

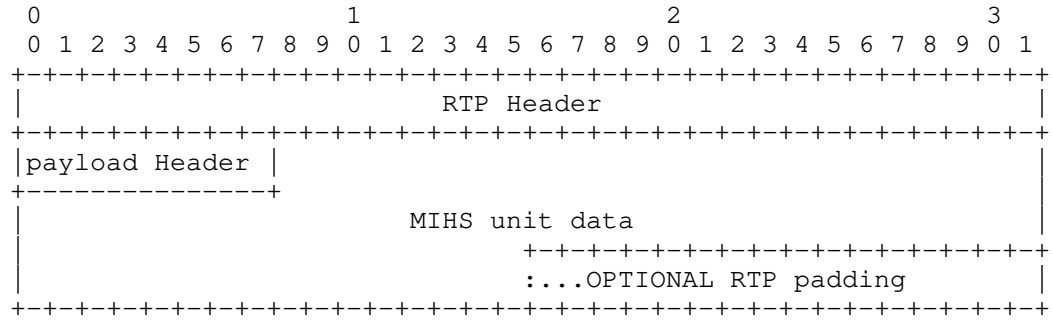


Figure 6: Single unit payload structure

5.3.2. Fragmented Unit Payload Structure

In a fragmented unit payload structure, as described in Figure 7, the RTP packet contains the RTP header, followed by the payload header, a Fragmented Unit (FU) header, and a MIHS unit fragment. The payload header follows the structure described in Section 5.2. The value of the UT field of the payload header is 7. The FU header follows the structure described in Figure 8.

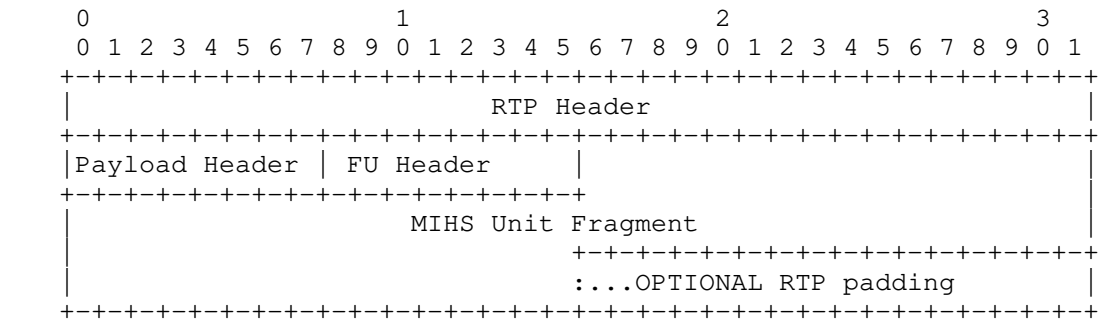


Figure 7: Fragmentation unit header

FU headers are used to enable fragmenting a single MIHS unit into multiple RTP packets. Fragments of the same MIHS 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). FUs MUST NOT be nested, i.e., an FU MUST NOT contain a subset of another FU.

Figure 8 describes a FU header, including the following fields:

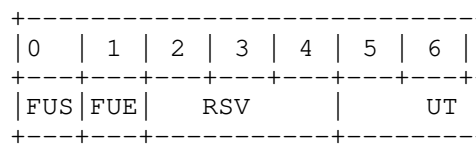


Figure 8: Fragmentation unit header

FUS (Fragmented Unit Start, 1 bit): this field MUST be set to 1 for the first fragment, and 0 for the other fragments.

FUE (Fragmented Unit End, 1 bit): this field MUST be set to 1 for the last fragment, and 0 for the other fragments.

RSV (Reserved, 3 bits): these bits MUST be set to 0 by the sender and ignored by the receiver.

UT (Unit Type, 3 bits): this field indicates the type of the MIHS unit this fragment belongs to, using values defined in Figure 4.

The use of MIHS unit fragmentation in RTP means that a media receiver can receive some fragments, but not other fragments. The missing fragments will typically not be retransmitted by RTP. This results in partially received MIHS units, which can be either dropped or used by the decoding application, based on implementation.

5.4. MIHS Units Transmission Considerations

The following considerations apply for the streaming of MIHS units over RTP:

The MIHS format enables variable duration units and uses initialization MIHS units to declare the duration of subsequent non-zero duration MIHS units, as well as the variation of this duration. A sender SHOULD set constant or low-variability (e.g., lower than the playout buffer) durations in initialization MIHS units, for RTP streaming. This enables the receiver to determine early (e.g., using a timer) when a unit has been lost and make the decoder more robust to RTP packet loss. If a sender sends MIHS units with high duration variations, the receiver may need to wait for a long period of time (e.g., the upper bound of the duration variation), to determine if a MIHS unit was lost in transmission. Whether this behavior is acceptable or not is application dependent.

The MIHS format uses silent MIHS units to signal haptic silence. A sender MAY decide not to send silent units, to save network resources. Since, from a receiver standpoint, a missed MIHS unit may originate from a not-sent silent unit, or a lost packet, a sender MAY send one, or a few, MIHS silent units at the beginning of a haptic silence. If a media receiver receives a MIHS silent unit, the receiver SHOULD assume that silence is intended until the reception of a non-silent MIHS unit. This can reduce the number of false detection of lost RTP packets by the decoder.

6. Payload Format Parameters

This memo updates the 'hmpg' haptic subtype defined in section 4.3.3 of [I-D.ietf-mediaman-haptics] for use with the MPEG-I haptics streamable binary coding format described in ISO/IEC DIS 23090-31: Haptics coding [ISO.IEC.23090-31]. This memo especially defines optional parameters for this type in Section 6.2. A mapping of the parameters into the Session Description Protocol (SDP) [RFC8866] is also provided for applications that use SDP. Equivalent parameters could be defined elsewhere for use with control protocols that do not use SDP. The receiver MUST ignore any parameter unspecified in this memo.

6.1. Media Type Registration Update

The following entries identify the media type being updated:

Type name: haptics

Subtype name: hmpg

The following entries are replaced by this memo:

Optional parameters: see section 6.2 of RFC XXX (note to RFC editor: replace with this RFC's number).

Person & email address to contact for further information: Yeshwant Muthusamy (yeshwant@yeshvik.com) and Hyunsik Yang (hyunsik.yang@interdigital.com)

6.2. Optional Parameters Definition

hmpg-ver provides the year of the edition and amendment of ISO/IEC 23090-31 that this file conforms to, as defined in [ISO.IEC.23090-31]. MPEG_haptics object.version is a string which may hold values such as XXXX or XXXX-Y where XXXX is the year of publication and Y is the amendment number, if any. For the initial release of the specifications, the value is "2023".

hmpg-profile indicates the profile used to generate the encoded stream as defined in [ISO.IEC.23090-31]: MPEG_haptics object.profile is a string which may in the initial release of the specifications hold the values "simple-parametric" or "main".

hmpg-lvl indicates the level used to generate the encoded stream as defined in [ISO.IEC.23090-31]: MPEG_haptics object.level is an integer which may in the initial release of the specifications hold the value 1 or 2.

hmpg-maxlod indicates the maximum level of details to use for the avatar(s). The avatar level of detail (LOD) is defined in [ISO.IEC.23090-31]: MPEG_haptics.avatar object.lod is an integer which may in the initial release of the specifications hold 0 or a positive integer.

hmpg-avtypes indicates, using a coma-separated list, types of haptic perception represented by the avatar(s). The avatar type is defined in [ISO.IEC.23090-31]: MPEG_haptics.avatar object.type is an integer which may in the initial release of the specifications hold values among "Vibration", "Pressure", "Temperature", "Custom".

hmpg-modalities indicates, using a coma-separated list, haptic perception modalities (e.g., pressure, acceleration, velocity, position, temperature, etc.). The perception modality is defined in [ISO.IEC.23090-31]: MPEG_haptics.perception object.perception_modality is a string which may in the initial release of the specifications hold values among "Pressure", "Acceleration", "Velocity", "Position", "Temperature", "Vibrotactile", "Water", "Wind", "Force", "Electrotactile", "Vibrotactile Texture", "Stiffness", "Friction", "Humidity", "User-defined Temporal", "User-defined Spatial", "Other".

hmpg-bodypartmask indicates, using a bitmask, the location of the devices or actuators on the body. The body part mask is defined in [ISO.IEC.23090-31]: MPEG_haptics.reference_device object.body_part_mask is a 32-bit integer which may in the initial release of the specifications hold a bit mask using bit positions defined in table 7 of [ISO.IEC.23090-31].

hmpg-maxfreq indicates the maximum frequency of haptic data for vibrotactile perceptions (Hz). Maximum frequency is defined in [ISO.IEC.23090-31]: MPEG_haptics.reference_device object.maximum_frequency is defined as an integer or floating-point number in the initial release of the specifications.

hmpg-minfreq indicates the minimum frequency of haptic data for vibrotactile perceptions (Hz). Minimum frequency is defined in [ISO.IEC.23090-31]: MPEG_haptics.reference_device object.minimum_frequency is defined as an integer or floating-point number in the initial release of the specifications.

hmpg-dvctypes indicates, using a coma-separated list, the types of actuators. The device type is defined in [ISO.IEC.23090-31]: MPEG_haptics.reference_device object.type is a string which may in the initial release of the specifications hold values among "LRA", "VCA", "ERM", "Piezo" or "Unknown".

hmpg-silencesupp indicates whether silence suppression should be used (1) or not (0). The default value shall be 1.

7. SDP Considerations

The mapping of above defined payload format media type to the corresponding fields in the Session Description Protocol (SDP) is done according to [RFC8866].

The media name in the "m=" line of SDP MUST be haptics.

The encoding name in the "a=rtpmap" line of SDP MUST be hmpg

The clock rate in the "a=rtpmap" line may be any sampling rate, typically 8000.

The OPTIONAL parameters (defined in Section 6.2), when present, MUST be included in the "a=fmtp" line of SDP. This is expressed as a media type string, in the form of a semicolon-separated list of parameter=value pairs.

An example of media representation corresponding to the hmpg RTP payload in SDP is as follows:

```
m=haptics 43291 UDP/TLS/RTP/SAVPF 115
a=rtpmap:115 hmpg/8000
a=fmtp:115 hmpg-profile=1;hmpg-lvl=1;hmpg-ver=2023
```

7.1. SDP Offer/Answer Considerations

When using the offer/answer procedure described in [RFC3264] to negotiate the use of haptic, the following considerations apply:

The haptic signal can be sampled at different rates. The MPEG Haptics Coding standard does not mandate a specific frequency. A typical sample rate is 8000Hz.

The parameter 'hmpg-ver' indicates the version of the haptic standard specification. If it is not specified, the initial version of the MPEG Haptic Coding specification SHOULD be assumed, although the sender and receiver MAY use a specific value based on an out-of-band agreement. The parameter 'hmpg-profile' is used to restrict the number of tools used (e.g., the simple-parametric profile fits enable simpler implementations than the main profile). If it is not specified, the most general profile "main" SHOULD be assumed, although the sender and receiver MAY use a specific value based on an out-of-band agreement. The parameter 'hmpg-lvl' is used to further characterize implementations within a given profile, e.g., according to the maximum supported number of channels, bands, and perceptions. If it is not specified, the most general level "2" SHOULD be assumed, although the sender and receiver MAY use a specific version based on an out-of-band agreement.

Other parameters can be used to indicate bitstream properties as well as receiver capabilities. The parameters 'hmpg-maxlod', 'hmpg-avtypes', 'hmpg-bodypartmask', 'hmpg-maxfreq', 'hmpg-minfreq', 'hmpg-dvctypes', and 'hmpg-modalities' can be sent by a sender to reflect the characteristics of bitstreams and can be set by a receiver to reflect the nature and capabilities of local actuator devices, or a preferred set of bitstream properties. For example, different receivers may have different sets of local actuators, in which case

these parameters can be used to select a stream adapted to the receiver. In some other cases, some receivers may indicate a preference for a set of bitstream properties such as perceptions, min/max frequency, or body-part-mask, which contribute the most to the user experience for a given application, in which case these parameters can be used to select a stream which include and possibly prioritizes those properties.

The parameter 'hmpg-silencesupp' can be used to indicate sender and receiver capabilities or preferences. This parameter indicates whether silence suppression should be used, as described in Section 5.4.

7.2. Declarative SDP considerations

When haptic content over RTP is offered with SDP in a declarative style, the parameters capable of indicating both bitstream properties as well as receiver capabilities are used to indicate only bitstream properties. For example, in this case, the parameters hmpg-maxlod, hmpg-bodypartmask, hmpg-maxfreq, hmpg-minfreq, hmpg-dvctypes, and hmpg-modalities declare the values used by the bitstream, not the capabilities for receiving bitstreams. A receiver of the SDP is required to support all parameters and values of the parameters provided; otherwise, the receiver MUST reject or not participate in the session. It falls on the creator of the session to use values that are expected to be supported by the receiving application.

8. Congestion control consideration

The general congestion control considerations for transporting RTP data apply to HMPG haptics over RTP as well [RFC3550].

It is possible to adapt network bandwidth by adjusting either the encoder bit rate or by adjusting the stream content (e.g., level of detail, body parts, actuator frequency range, target device types, modalities).

In case of congestion, a receiver or intermediate node MAY prioritize independent packets over dependent ones, since the non reception of an independent MIHS unit can prevent the decoding of multiple subsequent dependent MIHS units. In case of congestion, a receiver or intermediate node MAY prioritize initialization MIHS units over other units, since initialization MIHS units contain metadata used to re-initialize the decoder, and MAY drop silent MIHS units before other types of MIHS units, since a receiver may interpret a missing MIHS unit as a silence. It is also possible, using the layer field of the RTP payload header, to allocate MIHS units to different layers based on their content, to prioritize haptic data contributing the

most to the user experience. In case of congestion, intermediate nodes and receivers SHOULD use the MIHS layer value to determine the relative importance of haptic RTP packets.

9. Security Considerations

This RTP payload format is subject to security threats commonly associated with RTP payload formats, as well as threats specific to the interaction of haptic devices with the physical world, and threats associated with the use of compression by the codec. Security consideration for threats commonly associated with RTP payload formats are outlined in [RFC3550], as well as in RTP profiles such as RTP/AVP [RFC3551]), RTP/AVPF [RFC4585], RTP/SAVP [RFC3711], or RTP/SAVPF [RFC5124].

Haptic sensors and actuators operate within the physical environment. This introduces the potential for information leakage through sensors, or damage to actuators due to data tampering. Additionally, misusing the functionalities of actuators (such as force, position, temperature, vibration, electro-tactile, etc.) may pose a risk of harm to the user, for example by setting keyframe parameters (e.g., amplitude, position, frequency) or channel gain to a value that surpasses a permissible range. While individual devices can implement security measures to reduce or eliminate those risks on a per-device basis, in some cases harm can be inflicted by setting values which are permissible for the individual device. For example, causing contact with the physical environment or triggering unexpected force feedback can potentially harm the user. Each haptic system should therefore implement system-dependent security measures, which is more error prone. To limit the risk that attackers exploit weaknesses in haptic systems, it is important that haptic transmission should be protected against malicious traffic injection or tempering.

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]. Applications SHOULD use one or more appropriate strong security mechanisms.

The haptic codec used with this payload format uses a compression algorithm (see sections 8.2.8.5 and 8.3.3.2 in [ISO.IEC.23090-31]). An attacker may inject pathological datagrams into the stream which are complex to decode and cause the receiver to be overloaded, similarly to [RFC3551].

End-to-end security with authentication, integrity, or confidentiality protection will prevent a Media-Aware Network Element (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.

10. IANA Considerations

This memo updates the media type registration of haptics/hmpg with IANA, in Section 6.

11. References

11.1. Normative References

[ISO.IEC.23090-31]
ISO/IEC, "Text of ISO/IEC FDIS 23090-31 MPEG Haptics Coding", ISO/IEC 23090-31, 2024, <https://isotc.iso.org/livelink/livelink/open/jtc1sc29wg7>.

11.2. Informative References

[I-D.ietf-mediaman-haptics]
Muthusamy, Y. K. and C. Ullrich, "The 'haptics' Top-level Media Type", Work in Progress, Internet-Draft, draft-ietf-mediaman-haptics-05, 27 July 2023, <https://datatracker.ietf.org/doc/html/draft-ietf-mediaman-haptics-05>.

[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/rfc/rfc2119>.

[RFC2736] Handley, M. and C. Perkins, "Guidelines for Writers of RTP Payload Format Specifications", BCP 36, RFC 2736, DOI 10.17487/RFC2736, December 1999, <https://www.rfc-editor.org/rfc/rfc2736>.

[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/rfc/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/rfc/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/rfc/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/rfc/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/rfc/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/rfc/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/rfc/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/rfc/rfc7202>.

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

[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/rfc/rfc8174>.

[RFC8866] Begen, A., Kyzivat, P., Perkins, C., and M. Handley, "SDP: Session Description Protocol", RFC 8866, DOI 10.17487/RFC8866, January 2021, <https://www.rfc-editor.org/rfc/rfc8866>.

Authors' Addresses

Hyunsik Yang
InterDigital
United States of America
Email: hyunsik.yang@interdigital.com

Xavier de Foy
InterDigital
Canada
Email: xavier.defoy@interdigital.com

AVTCORE Working Group
INTERNET-DRAFT
Category: Standards Track
Expires: July 24, 2024

B. Aboba
P. Hancke
Microsoft Corporation

24 January 2024

H.265 Profile for WebRTC
draft-ietf-avtcore-hevc-webrtc-02.txt

Abstract

RFC 7742 defines WebRTC video processing and codec requirements, including guidance for endpoints supporting the VP8 and H.264 codecs, which are mandatory to implement. With support for H.265 under development in WebRTC browsers, similar guidance is needed for browsers considering support for the H.265 codec, whose RTP payload format is defined in RFC 7798.

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 July 24, 2024.

Table of Contents

1. Introduction

"RTP Payload Format for High Efficiency Video Coding (HEVC)" [RFC7798] defines the encapsulation of H.265 [H.265] within the Real-time Transport Protocol (RTP) [RFC3550]. While "WebRTC Video Processing and Codec Requirements" [RFC7742] provides guidance for endpoints supporting the mandatory to implement VP8 and H.264 codecs, it does not cover H.265. With H.265 support under development within browsers [HEVC-WebKit][HEVC-Chrome] there is a need to for an interoperability profile of [RFC7798] for WebRTC implementations choosing to support H.265.

1.1. 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.

2. H.265 Support

Support for the H.265 video codec is OPTIONAL for WebRTC browsers and non-browsers. Implementations supporting H.265 that conform to this specification MUST support receiving H.265 and MAY support sending H.265.

For the H.265 [H.265] codec, endpoints MUST support the payload formats defined in [RFC7798]. In addition, they MUST support Main Profile Level 3.1 (level-id=93) and SHOULD support Main Profile Level 4 (level-id=120).

[RFC7798] Section 4.5 defines how Temporal Scalability Control Information (TSCI) is communicated using PACI Extensions defined in [RFC7798] Section 4.4.4.2. A WebRTC implementation that has negotiated use of RTP header extensions containing TSCI information (such as the Dependency Descriptor [DD]) SHOULD NOT send TSCI information within the PACI. If TSCI information is being received in an RTP header extension, implementations MUST ignore TSCI information contained in the PACI.

Implementations of the H.265 codec have utilized a wide variety of optional parameters. To improve interoperability, the following parameter settings are specified:

level-id: Implementations SHOULD include this parameter within SDP and MUST interpret it when receiving it. If no level-id is present, a value of 93 (i.e., level 3.1) MUST be inferred.

tx-mode: Implementations SHOULD NOT include this parameter within SDP. If no tx-mode parameter is present, a value of "SRST" MUST be inferred. Implementations MUST support "SRST"; support for "MRST" and "MRMT" is OPTIONAL. Implementations that do not support "MRST" or "MRMT" MUST NOT include these tx-mode values in SDP.

sprop-sps, sprop-pps, sprop-vps, sprop-sei: H.265 allows sequence and picture information to be sent both in-band and out-of-band. WebRTC implementations MUST signal this information in-band. This means that WebRTC implementations MUST NOT include these parameters in the SDP they generate, and SHOULD silently ignore these parameters if they are received. An IDR/CRA/BLA sent MUST always be preceded by the relevant parameter sets sent in a packet (not necessarily a separate packet) with the same RTP timestamp as the IDR/CRA/BLA.

When the use of the video orientation (CVO) RTP header extension is not signaled as part of the SDP, H.265 implementations MAY send and SHOULD support proper interpretation of Display Orientation SEI messages.

[RFC7798] Section 8.3 specifies the use of the Reference Picture Selection Indication (RPSI) in H.265. Implementations MUST use the RPSI feedback message only as a reference picture selection request, and MUST NOT use it as positive acknowledgement. Receivers that detect that H.265 encoder-decoder synchronization has been lost SHOULD generate an RPSI feedback message if support for RPSI has been negotiated, unless the receiver has knowledge that the sender does not support RPSI. Such knowledge can be established during capability exchange or through previously sent RPSI requests that were not replied to by the sender through the use of a non-IRAP picture. An RTP packet-stream sender that receives an RPSI message MUST act on that message, and SHOULD change the reference picture.

Unless otherwise signaled, WebRTC implementations that support H.265 MUST encode and decode pixels with an implied 1:1 (square) aspect ratio.

3. Security Considerations

This document is subject to the security considerations described in Section 7 of [RFC7742].

In addition to those security considerations, H.265 implementers are advised to take note of the "Security Considerations" Section 9 of [RFC7798], including requirements pertaining to SEI messages.

4. IANA Considerations

This document does not require actions by IANA.

5. References

5.1. Normative References

[DD] Alliance for Open Media (AoMedia), "Dependency Descriptor RTP Header Extension", https://aomediacodec.github.io/av1-rtp-spec/#dependency-descriptor-rtp-header-extension, retrieved September 19, 2023.

[H.265] ITU-T, "High efficiency video coding", ITU-T Recommendation H.265, April 2013.

[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>.

[RFC7742] Roach, A. B., "WebRTC Video Processing and Codec Requirements", RFC 7742, DOI 10.17487/RFC7742, March 2016, <https://www.rfc-editor.org/info/rfc7742>.

[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>.

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

5.2. Informative References

[HEVC-WebKit] Shin, S. Qiu, J. and J. Zhu, "WebRTC HEVC RFC 7798 RTP Payload Format Implementation", https://github.com/WebKit/WebKit/pull/15494 (work in progress), retrieved July 9, 2023.

[HEVC-Chrome] "Issue 13485: Need the support of H.265", https://bugs.chromium.org/p/webrtc/issues/detail?id=13485 (work in progress), submitted December 8, 2021.

Acknowledgments

We would like to thank Stephan Wenger, Jonathan Lennox, Harald Alvestrand, Jianlin Qiu and Philip Eliasson for their discussions of this problem space.

Authors' Addresses

Bernard Aboba
Microsoft Corporation
One Microsoft Way
Redmond, WA 98052
United States of America

Email: bernard.aboba@gmail.com

Philipp Hancke
Microsoft Corporation
Talinn, Estonia

Email: philipp.hancke@gmail.com

AVTCORE Working Group
Internet-Draft
Intended status: Standards Track
Expires: 14 April 2024

Y. He
Qualcomm
C. Herglotz
FAU
E. Francois
InterDigital

12 October 2023

RTP Control Protocol (RTCP) Messages for Temporal-Spatial Resolution draft-ietf-avtcore-rtcp-green-metadata-02

Abstract

This specification describes an RTCP feedback message format for the ISO/IEC International Standard 23001-11, known as Energy Efficient Media Consumption (Green metadata), developed by the ISO/IEC JTC 1/SC 29/ WG 3 MPEG System. The RTCP payload format specified in this specification enables receivers to provide feedback to the senders and thus allows for short-term adaptation and feedback-based energy efficient mechanisms to be implemented. The payload format has broad applicability in real-time video communication services.

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 14 April 2024.

Table of Contents

1. Introduction

ISO/IEC 23001-11 specification, Energy Efficient Media Consumption (Green metadata) [GreenMetadata], specifies metadata that facilitates reduction of energy usage during media consumption. Two main types of metadata are defined in the specification. The first type consists of metadata generated by a video encoder which provides information about the decoding complexity of the delivered bitstream and about the quality of the decoded content. This first type of metadata is conveyed via the supplemental enhancement information (SEI) message mechanism specified in the video coding standard ITU-T Recommendation H.264 and ISO/IEC 14496-10 [AVC], H.265 and ISO/IEC 23008-5 [HEVC], H.266 and ISO/IEC 23090-3 [VVC].

The second type consists of metadata generated by a decoder as feedback conveyed to the encoder to adapt the decoder energy consumption. This specification focuses on this second type of metadata which is conveyed as extension of RTCP feedback messages [RFC4585]. The feedback in the second type of metadata specified in ISO/IEC 23001-11 [GreenMetadata] includes decoder operations reduction request, coding tools configuration request and temporal and spatial scaling request. This specification defines new RTCP payload format for the temporal and spatial resolution request and notification feedback message.

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 here.

3. Abbreviations

AVPF: The extended RTP profile for RTCP-based feedback

FCI: Feedback Control Information [RFC4585]

FMT: Feedback Message Type [RFC4585]

PSFB: Payload-specific FB message [RFC4585]

TSRR: Temporal-Spatial Resolution Request

TSRN: Temporal-Spatial Resolution Notification

CCM: Codec Control Messages [RFC5104]

4. Format of RTCP Feedback Messages

This document extends the RTCP feedback messages defined in the RTP/AVPF [RFC4585] and [RFC5104] by defining a Green Metadata feedback message. The message can be used by the receiver to inform the sender of the desirable coding temporal resolution (frame rate) and spatial resolution of the bitstream delivered, and by the sender to indicate the coding temporal and spatial resolution it will use henceforth.

RTCP Green Metadata feedback message follows a similar message format as RTCP Temporal-Spatial Trade-off Request and Notification [RFC5104]. The message may be sent in a regular full compound RTCP packet or in an early RTCP packet, as per the RTP/AVPF rules.

This specification specifies two additional payload-specific feedback messages: Temporal-Spatial Resolution Request (TSRR) and Temporal-Spatial Resolution Notification (TSRN)

4.1. Temporal-Spatial Resolution Request

The TSRR feedback message is identified by RTCP packet type value PT=PSFB and FMT=11.

The FCI field MUST contain one or more TSRR FCI entries.

4.1.1. Message format

The content of the FCI entry for the Temporal-Spatial Resolution Request is depicted in Figure 1.

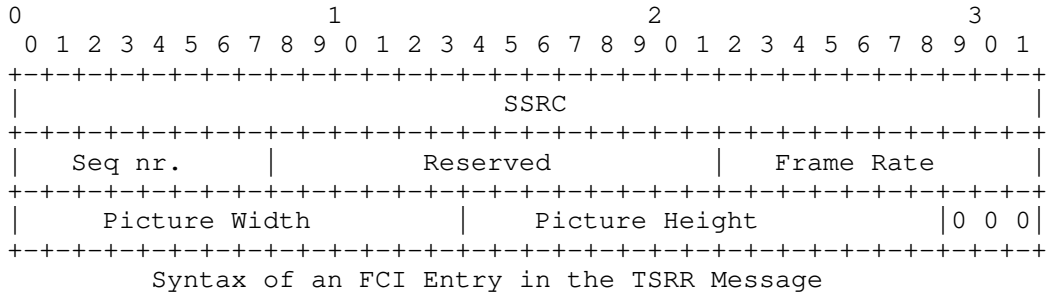


Figure 1

SSRC (32 bits): The Synchronization Source (SSRC) of the media sender that is requested to apply the frame rate and picture resolution.

Seq nr. (8 bits): Request sequence number. The sequence number space is unique for pairing of the SSRC of request source and the SSRC of the request target. The sequence number SHALL be increased by 1 modulo 256 for each new command. A repetition SHALL NOT increase the sequence number. The initial value is arbitrary.

Reserved (14 bits): All bits SHALL be set to 0 by the sender and SHALL be ignored on reception.

Frame Rate (10 bits): frames_per_second. This field specifies the frame rate as defined in clause 5.3 of [GreenMetadata]. An integer value between 1 and 1023 that indicates the coding frame rate that is requested. The value of Frame Rate equal to 0 is illegal.

Picture Width (14 bits): pic_width_in_luma_samples. This field specifies the picture width as defined in clause 5.3 of [GreenMetadata]. An integer value between 1 and 16383 that indicates the coding picture width in the units of luma samples that is requested. The value of Picture Width equal to 0 is illegal.

Picture Height (14 bits): pic_height_in_luma_samples. This specifies the picture height as defined in clause 5.3 of [GreenMetadata]. An integer value between 1 and 16383 that indicates the coding picture height in the units of luma samples that is requested. The value of Picture Height equal to 0 is illegal.

4.1.2. Semantics

A decoder can suggest a temporal-spatial resolution by sending a TSRR message to an encoder. If the encoder is capable of adjusting its temporal-spatial resolution, it SHOULD take into account the received TSRR message for future coding of pictures. The temporal and spatial resolutuions in a TSRR message SHALL be less than or equal to the temporal and spatial resolutions negotiated via SDP.

The reaction to the reception of more than one TSRR message by a media sender from different media receivers is left open to the implementation. The selected Frame Rate, Picture Width and Picture Height SHALL be communicated to the media receivers by means of the TSRN message (see section Section 4.2).

Within the common packet header for feedback messages (as defined in section 6.1 of [RFC4585]), the "SSRC of packet sender" field indicates the source of the request, and the "SSRC of media source" is not used and SHALL be set to 0. The SSRCs of the media senders to which the TSRR applies are in the corresponding FCI entries.

A TSRR message MAY contain requests to multiple media senders, using one FCI entry per target media sender.

4.1.3. Timing Rules

The timing follows the rules outlined in section 3 of [RFC4585]. This request message is not time critical and SHOULD be sent using regular RTCP timing. Only if it is known that the user interface requires quick feedback, the message MAY be sent with early or immediate feedback timing.

4.1.4. Handling of Message in Mixers and Translators

A mixer or media translator that encodes content sent to the session participant issuing the TSRR SHALL consider the request to determine if it can fulfill it by changing its own encoding parameters. A media translator unable to fulfill the request MAY forward the request unaltered towards the media sender. A mixer encoding for multiple session participants will need to consider the joint needs of these participants before generating a TSRR on its own behalf towards the media sender.

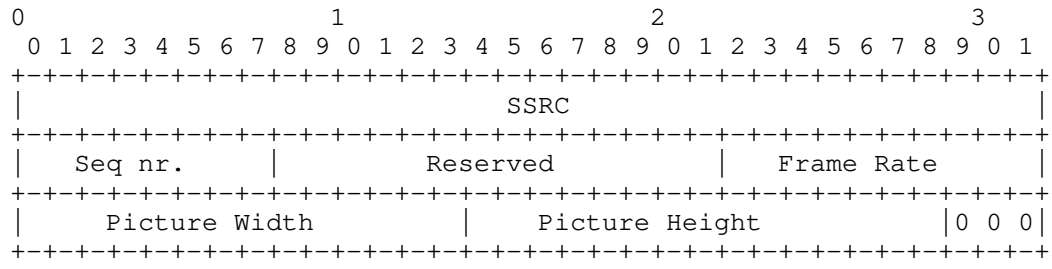
4.2. Temporal-Spatial Resolution Notification (TSRN)

The TSRN message is identified by RTCP packet type value PT=PSFB and FMT=12.

The FCI field SHALL contain one or more TSRN FCI entries.

4.2.1. Message format

The content of the FCI entry for the Temporal-Spatial Resolution Notification is depicted in Figure 2.



Syntax of an FCI Entry in the TSRN Message

Figure 2

SSRC (32 bits): The Synchronization Source (SSRC) of the source of the TSRR that resulted in this notification.

Seq nr. (8 bits): The sequence number value from the TSRR that is being acknowledged.

Reserved (14 bits): All bits SHALL be set to 0 by the sender and SHALL be ignored on reception.

Frame Rate (10 bits): The frame rate the media sender is using henceforth.

Picture Width (14 bits): The coding picture width the media sender is using henceforth.

Picture Height (14 bits): The coding picture height the media sender is using henceforth.

It is to note that the returned value (Frame Rate, Picture Width, Picture Height) may differ from the requested one, for example, in cases where a media encoder cannot change its frame rate or picture resolution, or when the requested temporal and spatial resolutions are larger than the temporal and spatial resolutions negotiated via SDP, or when pre-recorded content is used.

4.2.2. Semantics

This feedback message is used to acknowledge the reception of a TSRR. For each TSRR received targeted at the session participant, a TSRN FCI entry SHALL be sent in a TSRN feedback message. A single TSRN message MAY acknowledge multiple requests using multiple FCI entries. The Frame Rate, Picture Width and Picture Height value included SHALL be the same in all FCI entries of the TSRN message. Including an FCI for each requestor allows each requesting entity to determine that the media sender received the request. The notification SHALL also be sent in response to TSRR repetitions received. If the request receiver has received TSRR with several different sequence numbers from a single requestor, it SHALL only respond to the request with the highest (modulo 256) sequence number. Note that the highest sequence number may be a smaller integer value due to the wrapping of the field. Appendix A.1 of [RFC3550] has an algorithm for keeping track of the highest received sequence number for RTP packets; it could be adapted for this usage.

The TSRN SHALL include the Temporal-Spatial Resolution Frame Rate, Picture Width and Picture Height that will be used as a result of the request. This is not necessarily the same Frame Rate, Picture Width and Picture Height as requested, as the media sender may need to aggregate requests from several requesting session participants. It may also have some other policies or rules that limit the selection.

Within the common packet header for feedback messages (as defined in section 6.1 of [RFC4585]), the "SSRC of packet sender" field indicates the source of the Notification, and the "SSRC of media source" is not used and SHALL be set to 0. The SSRCs of the requesting entities to which the Notification applies are in the corresponding FCI entries.

4.2.3. Timing Rules

The timing follows the rules outlined in section 3 of [RFC4585]. This acknowledgement message is not extremely time critical and SHOULD be sent using regular RTCP timing.

4.2.4. Handling of TSRN in Mixers and Translators

A mixer or translator that acts upon a TSRR SHALL also send the corresponding TSRN. In cases where it needs to forward a TSRR itself, the notification message MAY need to be delayed until the TSRR has been responded to.

5. Security Considerations

The defined messages have certain properties that have security implications. These must be addressed and taken into account by users of this protocol.

Spoofed or maliciously created feedback messages of the type defined in this specification can have the following implications:

* severely reduced picture resolution due to false TSRR messages that sets the picture width and height to a very low value;

* severely reduced frame rate due to false TSRR messages that sets the frame rate to a very low value.

* severely increasedd picture resolution due to false TSRR messages that sets the picture width and height to a value that is larger than the value negotiated via SDP;

* severely increased frame rate due to false TSRR messages that sets the frame rate to a value that is larger than the value negotiated via SDP.

To prevent these attacks, there is a need to apply authentication and integrity protection of the feedback messages. This can be accomplished against threats external to the current RTP session using the RTP profile that combines Secure RTP [SRTP] and AVPF into SAVPF [SAVPF]. In the mixer cases, separate security contexts and filtering can be applied between the mixer and the participants, thus protecting other users on the mixer from a misbehaving participant.

6. SDP Definitions

The capability of handling messages defined in this specification MAY be exchanged at a higher layer such as SDP. This specification follows all the rules defined in AVPF [RFC4585] and CCM [RFC5104] for an "rtcp-fb" attribute relating to the payload type in a session description.

6.1. Extension of the rtcp-fb Attribute

This specification defines a new parameter "tsrr" to the "ccm" feedback value defined in CCM [RFC5104] to indicate support of the Temporal-Spatial Resolution Request/Notification (TSRR/TSRN). All the rules described in [RFC4585] for rtcp-fb attribute relating to payload type and to multiple rtcp-fb attributes in a session description also apply to the new feedback messages defined in this specification.

```
rtcp-fb-ccm-param =/ SP "tsrr" ; Temporal-Spatial Resolution
```

6.2. Examples

Example 1: The following SDP describes a point-to-point video call with H.266, with the originator of the call declaring its capability to support the FIR and TSRR/TSRN codec control messages. The SDP is carried in a high-level signaling protocol like SIP.

```
v=0
o=alice 3203093520 3203093520 IN IP4 host.example.com
s=Point-to-Point call
c=IN IP4 192.0.2.124
m=audio 49170 RTP/AVP 0
a=rtpmap:0 PCMU/8000
m=video 51372 RTP/AVPF 98
a=rtpmap:98 H266/90000
a=rtcp-fb:98 ccm tsrr
a=rtcp-fb:98 ccm fir
```

In the above example, when the sender receives a TSRR message from the remote party it is capable of adjusting the trade-off as indicated in the RTCP TSRN feedback message.

Example 2: The following example describes the Offer/Answer implications for the codec control messages. The offerer wishes to support "tsrr", "fir" and "tmmbr". The offered SDP is

```
-------------> Offer
```

```
v=0
o=alice 3203093520 3203093520 IN IP4 host.example.com
s=Offer/Answer
c=IN IP4 192.0.2.124
m=audio 49170 RTP/AVP 0
a=rtpmap:0 PCMU/8000
m=video 51372 RTP/AVPF 98
a=rtpmap:98 H266/90000
a=rtcp-fb:98 ccm tsrr
a=rtcp-fb:98 ccm fir
a=rtcp-fb:* ccm tmmbr smaxpr=120
```

The answerer wishes to support only the FIR and TSRR/TSRN messages and the answerer SDP is

```
<---------------- Answer

v=0
o=alice 3203093520 3203093524 IN IP4 otherhost.example.com
s=Offer/Answer
c=IN IP4 192.0.2.37
m=audio 47190 RTP/AVP 0
a=rtpmap:0 PCMU/8000
m=video 53273 RTP/AVPF 98
a=rtpmap:98 H266/90000
a=rtcp-fb:98 ccm tsrr
a=rtcp-fb:98 ccm fir
```

7. IANA Considerations

Placeholder

8. References

8.1. Normative References

[GreenMetadata] "ISO/IEC DIS 23001-11, Information technology - MPEG Systems Technologies - Part 11: Energy-Efficient Media Consumption (Green Metadata)", 2022, <https://www.iso.org/standard/83674.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>.

[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>.

[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>.

[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>.

8.2. Informative References

[AVC] "Advanced video coding, ITU-T Recommendation H.264", 2021, <https://www.itu.int/rec/T-REC-H.264>.

[HEVC] "High efficiency video coding, ITU-T Recommendation H.265", 2021, <https://www.itu.int/rec/T-REC-H.265>.

[SAVPF] Ott, J. and E. Carrara, ""Extended Secure RTP Profile for RTCP-based Feedback (RTP/SAVPF)"", 2008, <https://datatracker.ietf.org/doc/pdf/rfc5124>.

[SRTP] Baugher, M., McGrew, D., Naslund, M., Carrara, E., and K. Norrman, "The Secure Real-time Transport Protocol(SRTP)", 2004, <https://datatracker.ietf.org/doc/pdf/rfc3711>.

[VVC] "Versatile Video Coding, ITU-T Recommendation H.266", 2022, <http://www.itu.int/rec/T-REC-H.266>.

Appendix A. Change History

To RFC Editor: PLEASE REMOVE ThIS SECTION BEFORE PUBLICATION

draft-ietf-avtcore-rtcp-green-metadata-00 ....initial version

draft-ietf-avtcore-rtcp-green-metadata-01 ....title and editorial changes

draft-ietf-avtcore-rtcp-green-metadata-02 ....editorial changes

Authors' Addresses

Yong He
Qualcomm
5775 Morehouse Drive
San Diego, 92121
United States of America
Email: yonghe@qti.qualcomm.com

Christian Herglotz
FAU
Schlossplatz 4
91054 Erlangen
Germany
Email: christian.herglotz@fau.de

Edouard Francois
InterDigital
975 Avenue des Champs Blancs
35576 Cesson-Sevigne
France
Email: edouard.francois@interdigital.com

Audio/Video Transport Core Maintenance
Internet-Draft
Intended status: Standards Track
Expires: 20 May 2024

P.-A. Lemieux, Ed.
Sandflow Consulting LLC
D. S. Taubman
University of New South Wales

17 November 2023

RTP Payload Format for sub-codestream latency JPEG 2000 streaming
draft-ietf-avtcore-rtp-j2k-scl-00

Abstract

This RTP payload format defines the streaming of a video signal encoded as a sequence of JPEG 2000 codestreams. The format allows sub-codestream latency, such that the first RTP packet for a given codestream can be emitted before the entire codestream is available.

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 20 May 2024.

Table of Contents

1. Introduction

This RTP payload format specifies the streaming of a video signal encoded as a sequence of JPEG 2000 codestreams.

In addition to supporting a variety of frame scanning techniques (progressive, interlaced and progressive segmented frame) and image characteristics, the payload format includes the following features specifically designed for streaming applications:

* the payload format allows sub-codestream latency such that the first RTP packet of a given codestream to be emitted before the entire codestream is available. Specifically, the payload format does not rely on the JPEG 2000 PLM and PLT marker segments for recovery after RTP Packet loss since these markers can only be written after the codestream is complete and are thus incompatible with sub-codestream latency. Instead, the payload format includes payload header fields (ORDH, ORDB, POS and PID) that indicates whether the RTP packet contains a resynchronization (resync) point and how a recipient can restart codestream processing from that resync point. This contrasts with [RFC5371], which also specifies an RTP payload format for JPEG 2000, but relies on codestream structures that cannot be emitted until the entire codestream is available.

* as in [RFC4175], the payload header contains an extension (ESEQ) to the standard 16-bit RTP sequence number, enabling the payload format to accommodate high data rates without ambiguity. This is necessary as the standard sequence number will roll over very quickly for high data rates likely to be encountered in this application. For example, the standard sequence number will roll over in 0.5 seconds with a 1-Gbps video stream with RTP Packet sizes of at least 1000 octets, which can be a problem for detecting loss and out-of-order packets particularly in instances where the round-trip time is greater than the roll over period (0.5 seconds in this example).

* the payload header optionally contains a temporal offset (PTSTAMP) relative to the first RTP Packet with the same value of RTP timestamp field (Section 5.2). The higher resolution of PTSTAMP compared to the timestamp allows receivers to recover the sender's clock more rapidly.

Finally, the payload format also makes use of the unique scalability features of JPEG 2000 to allow a network agent or recipient to discard resolutions and/or quality layers merely by inspecting payload headers (QUAL and RES fields), without having to parse the underlying codestream.

2. Requirements Language

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.

3. Media format description

The following summarizes the structure of the JPEG 2000 codestream, which is specified in detail at [jpeg2000-1].

NOTE: as described at Section 6, a JPEG 2000 codestream allows capabilities defined in any part of the JPEG 2000 family of standards, including those specified in [jpeg2000-2] and [jpeg2000-15].

JPEG 2000 represents an image as one or more components, e.g., R, G and B, each uniformly sampled on a common rectangular reference grid. An image can be further divided into contiguous rectangular tiles that are each independently coded and decoded.

NOTE: This payload format allows the transmission of multi-tile images as multiple single-tile images per stream, as specified at Section 6.2.

JPEG 2000 codes each image as a standalone codestream. Each codestream consists of (i) marker segments, which contain coding parameters and metadata, and (ii) coded data.

The codestream starts with an SOC marker segment and ends with an EOC marker segment. The main header of the codestream consists of marker segments between the SOC and first SOT marker segment and contains information that applies to the codestream in its entirety. It is generally impossible to decode a codestream without its main header.

The rest of the codestream consists of additional marker segments (tile-part headers) interleaved with coded image data.

The coded image data ultimately consists of code-blocks, each containing coded samples belonging to a rectangular (spatial) region within one resolution level of one component. Code-blocks are further collected into precincts, which, accordingly, represents code-blocks belonging to a spatial region within one resolution level of one component.

The image coded data can be arranged into several progression orders, which dictates which aspect of the image appears first in the codestream (in terms of byte offset). The progression orders are parameterized according to:

Position (P) The first lines of the image come before the last lines of the image.

Component (C) The first component of the image come before the last component of the image.

Resolution Layer (R) The information needed to reconstruct the lower spatial resolutions of the image come before the information needed to reconstruct the higher spatial resolutions of the image.

Quality Layer (L) The information needed to reconstruct the most-significant bits of each sample come before the information needed to reconstruct the least-significant bit of each sample.

For example, in the PRCL progression order, the information needed to reconstruct the first lines of the image come before that needed to reconstruct the last lines of the image and, within a collection of lines, the information needed to reconstruct the lower spatial resolutions of the image come before the information needed to reconstruct the higher spatial resolutions. This progression order is particular useful for sub-frame latency operations.

4. Video signal description

This RTP payload format supports three distinct video frame scanning techniques:

* Progressive frame

* Interlaced frame, where each frame consists of two fields. Field 1 occurs temporarily before Field 2. The height in lines of each field is half the height of the image.

* Progressive segmented frame (PsF), where each frame consists of two segments. Segment 1 contains the odd lines (1, 3, 5, 7,...) of a frame and Segment 2 contains the even lines (2, 4, 6, 8,...) of the same frame, where lines from the top of the frame to the bottom of the frame are numbered sequentially starting at 1.

All frames are scanned left to right, top to bottom.

5. Payload Format

5.1. General

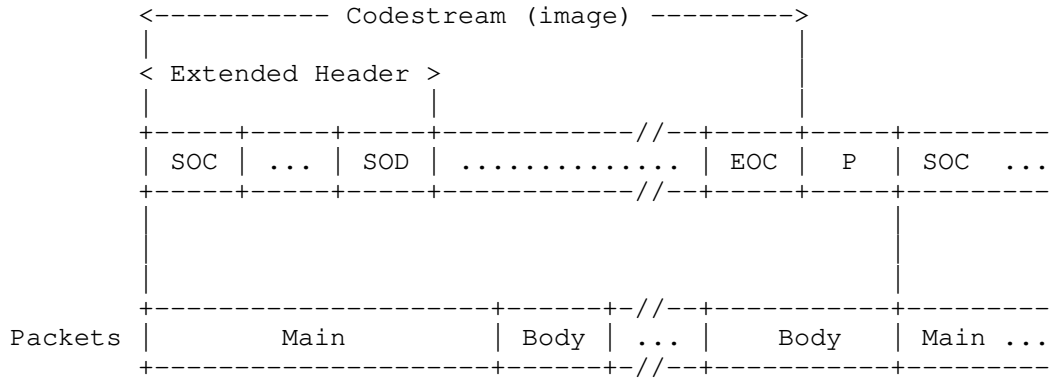


Figure 1: Packetization of a sequence of JPEG 2000 codestreams (not to scale). P are arbitrary padding bytes.

Each RTP packet, as specified at [RFC3550], is either a Main Packet or a Body Packet.

A Main Packet consists of the following ordered sequence of structures concatenated without gaps:

* the RTP Fixed Header;

* a Main Packet Payload Header, as specified at Section 5.3; and

* the payload, which consists of a JPEG 2000 codestream fragment.

A Body Packet consists of the following ordered sequence of structures concatenated without gaps:

* the RTP Fixed Header;

* a Body Packet Payload Header, as specified at Section 5.4; and

* the payload, which consists of a JPEG 2000 codestream fragment.

When concatenated, the sequence of JPEG 2000 codestream fragments emitted by the sender MUST be a sequence of JPEG 2000 codestreams where two successive JPEG 2000 codestreams MAY be separated by one or more arbitrary padding bytes.

The JPEG 2000 codestreams MUST conform to Section 6.

The padding bytes MUST be ignored by the recipient.

NOTE: Padding bytes can be used to achieve constant bit rate transmission.

A JPEG 2000 codestream consists of the bytes between, and including, the SOC and EOC markers, as defined in [jpeg2000-1].

A JPEG 2000 codestream fragment does not necessarily contain complete JPEG 2000 packets, as defined in [jpeg2000-1].

A JPEG 2000 codestream Extended Header consists of the bytes between, and including, the SOC marker and the first SOD marker.

The payload of a Body Packet MUST NOT contain any bytes of the JPEG 2000 codestream Extended Header.

The payload of a Main Packet MUST contain at least one byte of the JPEG 2000 codestream Extended Header and MAY contain bytes other than those of the JPEG 2000 codestream Extended Header.

A payload MUST NOT contain bytes from more than one JPEG 2000 codestream.

5.2. RTP Fixed Header Usage

The following RTP header fields have a specific meaning in the context of this payload format:

marker
: 1 The payload contains an EOC marker.

: 0 Otherwise

timestamp
: The timestamp is the presentation time of the image to which the payload belongs.

: The timestamp clock rate is 90 kHz.

The timestamp of successive progressive frames MUST advance at regular increments based on the instantaneous video frame rate.

The timestamp of Field 1 of successive interlaced frames MUST advance at regular increments based on the instantaneous video frame rate, and the Timestamp of Field 2 MUST be offset from the timestamp of Field 1 by one half of the instantaneous frame period.

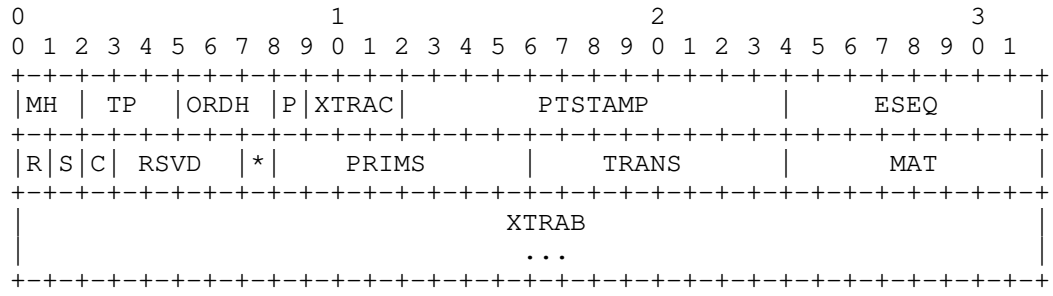
The timestamp of both segments of a progressive segmented frame MUST be equal.

timestamp of all RTP packets of a given image MUST be equal.

sequence number
The low-order bits of the RTP sequence number. The higher order bits of the RTP sequence number are contained in the ESEQ field.

5.3. Main Packet Payload Header

Figure 1 specifies the structure of the payload header. Fields are interpreted as unsigned binary integers in network order.



* RANGE

Figure 2: Structure of the Main Packet Payload Header

MH

0
The RTP Packet is a Body Packet.

1
The RTP Packet is a Main Packet and the codestream has more than one Main Packet. The next RTP Packet is a Main Packet.

2
The RTP Packet is a Main Packet and the codestream has more than one Main Packet. The next RTP Packet is a Body Packet.

3
The RTP Packet is a Main Packet and the codestream has exactly one Main Packet.

TP
Indicates the scanning structure of the image to which the payload belongs.

0
Progressive frame.

1
Field 1 of an interlaced frame, where the first line of the field is the first line of the frame.

2
Field 2 of an interlaced frame, where the first line of the field is the second line of the frame.

3
Field 1 of an interlaced frame, where the first line of the field is the second line of the frame.

4
Field 2 of an interlaced frame, where the first line of the field is the first line of the frame.

5
Segment 1 of a progressive segmented frame, where the first line of the image is the first line of the frame.

6
Segment 2 of a progressive segmented frame, where the first line of the image is the second line of the frame.

7
Extension value. See Section 8.6 and Section 7.8.

ORDH
Specifies the progression order used by the codestream and whether resync points are signaled.

0
Resync points are not necessarily signaled. The progression order can vary over the codestream.

1
The progression order is LRCP for the entire codestream. The first resync point is specified in every Body Packet that contains one or more resync points.

2
The progression order is RLCP for the entire codestream. The first resync point is specified in every Body Packet that contains one or more resync points.

3
The progression order is RPCL for the entire codestream. The first resync point is specified in every Body Packet that contains one or more resync points.

4
The progression order is PCRL for the entire codestream. The first resync point is specified in every Body Packet that contains one or more resync points.

5
The progression order is CPRL for the entire codestream. The first resync point is specified in every Body Packet that contains one or more resync points.

6
The progression order is PRCL for the entire codestream. The first resync point is specified in every Body Packet that contains one or more resync points.

7
The progression order can vary over the codestream. The first resync point is specified in every Body Packet that contains one or more resync points.

ORDH MUST be 0 is the codestream consists of more than one tile.

NOTE: Only ORDH = 4 and ORDH = 6 allow sub-codestream latency streaming.

NOTE: Progression order PRCL is defined in [jpeg2000-2]. The other progression orders are specified in [jpeg2000-1].

P

0
PTSTAMP is not used.

1
PTSTAMP is used.

XTRAC
Length, in multiples of 4 bytes, of the XTRAB field.

PTSTAMP
PTSTAMP = (timestamp + TOFF) mod 4096, if P = 1 in the Main Packet of this codestream.

TOFF is the transmission time of this RTP Packet, in the timebase of the timestamp clock and relative to the first packet with the same timestamp value.

TOFF = 0 in the first RTP Packet with the same timestamp value.

PTSTAMP = 0, if P = 0 in the Main Packet of this codestream.

NOTE: As described at Section 7.4 and Section 8.1, PTSTAMP is intended to improve clock recovery at the receiver and only applies when the transmission time of two consecutive RTP packets with identical timestamp fields differ by no more than 45 ms = 4095/90,000. [RFC5450] provides addresses the general case when a RTP packet is transmitted at a time other than its nominal transmission time.

ESEQ
The high order bits of the extended sequence number.

R
Determines whether Main Packet and codestream header information can be reused across codestreams.

1
All Main Packets in this stream, as identified by its SSRC value:

* MUST have identical Main Packet Payload Headers, with the exception of their TP, MH, ESEQ and PTSTAMP fields;

* MUST contain the same codestream main header information, with the exception of the SOT and COM marker segments, and any pointer marker segments; and

* MUST NOT contain bytes other than Extended Header bytes.

0
Otherwise

S
0
Component colorimetry is not specified, and left to the session or the application.

PRIMS, TRANS and MAT and RANGE MUST be zero.

1
Component colorimetry is specified by the PRIMS, TRANS and MAT and RANGE fields.

The codestream components MUST conform to one of the combinations at Table 1.

Combination name	Component index			
	0	1	2	3
Y	Y			
YA	Y	A		
RGB	R	G	B	
RGBA	R	G	B	A
YCbCr	Y	C_B	C_R	
YCbCrA	Y	C_B	C_R	A
The channel A is an opacity channel. The minimum sample value (0) indicates a completely transparent sample, and the maximum sample value (as determined by the bit depth of the codestream component) indicates a completely opaque sample. The opacity channel MUST map to a component with unsigned samples.				

Table 1: Mapping of codestream components to color channels

C
0
Code-block caching is not in use.

1
Code-block caching is in use.

R MUST be equal to 1.

RSVD
Reserved value. See Section 8.5 and Section 7.7.

RANGE
Value of the VideoFullRangeFlag specified in [rec-itu-t-h273]

PRIMS
One of the ColourPrimaries values specified in [rec-itu-t-h273]

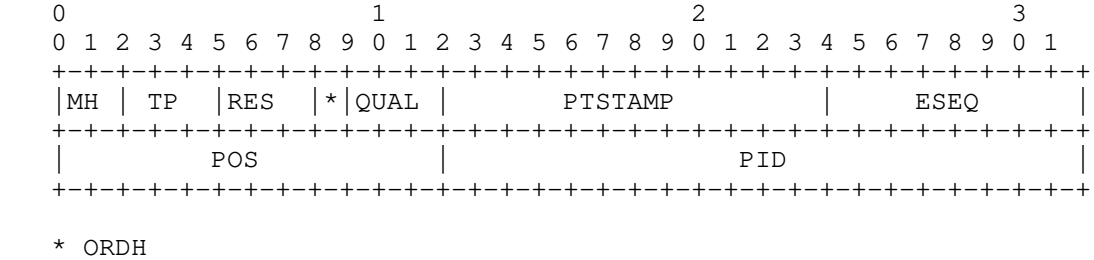
TRANS
One of the TransferCharacteristics values specified in [rec-itu-t-h273]

MAT
One of the MatrixCoefficients values specified in [rec-itu-t-h273]

XTRAB
Allows the contents of the Main Packet Payload Header to be extended in the future. See Section 8.4 and Section 7.6.

5.4. Body Packet Payload Header

Figure 3 specifies the structure of the Body Packet Payload Header. Fields are interpreted as unsigned binary integers in network order.



* ORDH

Figure 3: Structure of the Body Packet Payload Header

RES
0
The payload can contribute to all resolution layers.

Otherwise
The payload contains at least one byte of one JPEG 2000 packet belonging to resolution level (N_L + RES - 7) but does not contain any byte of any JPEG 2000 packet belonging to lower resolution levels. N_L is the number of decomposition levels of the codestream.

ORDB
0
No resync point is specified for the payload.

1
The payload contains a resync point.

ORDB MUST be 0 is the codestream consists of more than one tile.

QUAL
0
The payload can contribute to all quality layers.

Otherwise
The payload contributes only to quality layer index QUAL or above.

POS
Byte offset from the start of the payload to the first byte of the resync point belonging to the precinct identified by PID.

POS MUST be 0 if ORDB = 0.

PID
Unique identifier of the precinct of the resync point.

PID = c + s * num_components

where:

* _c_ is the index (starting from 0) of the image component to which the precinct belongs;

* _s_ is a sequence number which identifies the precinct within its tile-component; and

* _num_components_ is the number of components of the codestream.

If PID is present, the payload MUST NOT contain codestream bytes from more than one precinct.

PID MUST be 0 if ORDB = 0.

NOTE: PID is identical to precinct identifier I specified in [jpeg2000-9].

6. JPEG 2000 codestream requirements

6.1. General

The JPEG 2000 codestream MAY include capabilities beyond those specified at [jpeg2000-1], including those specified in [jpeg2000-2] and [jpeg2000-15].

NOTE: The Rsiz parameter and CAP marker segments of each JPEG 2000 codestream contain detailed information on the capabilities necessary to decode the codestream.

NOTE: The caps media type parameter defined in Section 9.2 allows applications to signal required device capabilities.

NOTE: The block coder specified at [jpeg2000-15] improves throughput and reduces latency compared to the original arithmetic block coder defined in [jpeg2000-1].

For interlaced or progressive segmented frames, the height specified in the JPEG 2000 main header MUST be the height in lines of the field or the segment, respectively.

If any decomposition level involves only horizontal decomposition then no decomposition level MUST involve only vertical decomposition; and conversely, if any decomposition level involves only vertical decomposition then no decomposition level MUST involve only horizontal decomposition.

6.2. Transmitting multi-tile images as multiple single-tile images

A sequence of multi-tile images can be transmitted by splitting it into multiple sequences of single-tile images, where:

* each sequence of single-tile images corresponds to a unique tile of the multi-tile image;

* each sequence of single-tile images is transmitted in a separate RTP stream;

* the coordinates of each single-tile image are expressed using the coordinate system of the multi-tile image; and

* the bitstreams of each of each single-tile image are identical to the corresponding bitstreams in the multi-tile image.

Such sequences of single-tile images are identified using the tile media type parameter specified at Section 9.2.

7. Sender requirements

7.1. Main Packet

Only Main Packets MAY contain bytes of the JPEG 2000 codestream Extended Header.

The sender MUST either emit a single Main Packet with MH = 3, or one or more Main Packets with MH = 1 followed by a single Main Packet with MH = 2.

The Main Packet Payload Headers fields MUST be identical in all Main Packet of a given codestream, with the exception of:

* MH;

* ESEQ; and

* PTSTAMP.

7.2. RTP Packet filtering

A network agent MAY strip out RTP Packet from a codestream that are of no interest to a particular client, e.g., based on a resolution or a spatial region of interest. Such a network agent SHOULD include a CSRC identifier to identify the SSRC field of the original source from which content was stripped.

7.3. Resync point

A resync point is the first byte of JPEG 2000 packet header data for a precinct and for which PID < 2^24.

NOTE: Resync points cannot be specified if the codestream consists of more than one tile (ORDB and ORDH are both equal to zero). To transmit codestreams that consist of more than one tile and benefit from resync points, the technique specified at Section 6.2 can be used.

NOTE: A resync point can be used by a receiver to process a codestream even if earlier packets in the codestream have been corrupted, lost or deliberately discarded by a network agent. As a

corollary, resync points can be used by a network agent to discard packets that are not relevant to a given rendering resolution or region of interest. Resync points play a role similar to pointer marker segments, albeit tailored for high bandwidth low latency streaming applications.

7.4. PTSTAMP field

A sender SHOULD set P = 1, but only if it can generate PTSTAMP accurately.

PTSTAMP can be derived from the same clock that is used to produce the 32-bit timestamp field in the RTP fixed header. Specifically, a sender maintains, at least conceptually, a 32-bit counter that is incremented by a 90kHz clock. The counter is sampled at the point when each RTP Packet is or SHOULD be at least notionally transmitted and the 12 LSBs of the sample are stored in the PTSTAMP field.

If P = 1, then the transmission time TOFF (as defined at Section 5.3) for two consecutive RTP packets with identical timestamp fields MUST NOT differ by more than 4095.

7.5. RES field

A sender SHOULD set RES > 0 whenever possible.

NOTE: While a sender can always safely set RES = 0, this makes it more difficult to discard packets based on resolution, as described at Section 8.3.

7.6. Extra information

The sender MUST set the value of XTRAC to 0.

Future edition of this specification can permit other values.

7.7. Reserved values

The sender MUST set reserved values to 0.

Future edition of this specification can specify other values such that these values can be ignored by receivers that conform to this specification.

7.8. Extension values

A sender MUST NOT use an extension value.

7.9. Code-block caching

This section applies only if C = 1.

A sender can improve bandwidth efficiency by only occasionally transmitting code-blocks corresponding to static portions of the video and otherwise transmitting empty code-blocks. When C = 1, and as described at Section 8.7, a receiver maintains a simple cache of previously received code-blocks, which it uses to replace empty code-blocks.

A sender alone determines which and when code-blocks are replaced with empty code-blocks.

The sender cannot however determine with certainty the state of the receiver's cache: some code-blocks might have been lost in transit, the sender doesn't know exactly when the receiver started processing the stream, etc.

A code-block is _empty_ if:

* it does not contribute code-bytes as specified in the parent JPEG 2000 packet header; or

* if the code-block conforms to [jpeg2000-15], contains an HT cleanup segment and the first two bytes of the Magsgn byte-stream are between 0xFF80 and 0xFF8F.

NOTE: the last condition allows the encoder to insert padding bytes to achieve a constant bit rate even when code-block does not contribute code-bytes, as suggested at [jpeg2000-15], F.4.

8. Receiver

8.1. PTSTAMP

Receivers can use PTSTAMP values to accelerate sender clock recovery since PTSTAMP typically updates more regularly than timestamp.

8.2. QUAL

A receiver can discard packets where QUAL > N if it is interested in reconstructing an image that only incorporates quality layers N and below.

8.3. RES

The JPEG 2000 coding process decomposes an image using a sequence of discrete wavelet transforms (DWT) stages.

Decomposition level	Resolution level	Subbands	Keep all Body Packets with RES equal to or less than this value...	... to decode an image with at most these dimensions
1	5	HL1,LH1,HH1	7	W x H
2	4	HL2,LH2,HH2	6	(W/2) x (H/2)
3	3	HL3,LH3,HH3	5	(W/4) x (H/4)
4	2	HL4,LH4,HH4	4	(W/8) x (H/8)
5	1	HL5,LH5,HH5	3	(W/16) x (H/16)
5	0	LL5	2	(W/32) x (H/32)

Table 2: Optional discarding of Body Packets based on the value of the RES field when decoding a reduced resolution image, in the case where N_L = 5 and all DWT stages consist of both horizontal and vertical transforms. The image has nominal width and height of W x H.

Table 2 illustrates the case where each DWT stage consists of both horizontal and vertical transforms, which is the only mode supported in [jpeg2000-1]. The first stage transforms the image into (i) the image at half-resolution (LL1 sub-bands) and (ii) residual high-frequency data (HH1, LH1, HL1 sub-bands). The second stage transforms the image at half-resolution (LL1 sub-bands) into the image at quarter resolution (LL2 sub-bands) and residual high-frequency data (HH2, LH2, HL2 sub-bands). This process is repeated N_L times, where N_L is the number of decomposition levels as defined in the COD and COC marker segments of the codestream.

The decoding process reconstructs the image by reversing the coding process, starting with the lowest resolution image stored in the codestream (LL_(N_L)).

As a result, it is possible to reconstruct a lower resolution of the image by stopping the decoding process at a selected stage. For example, in order to reconstruct the image at quarter resolution (LL2), only sub-bands with index greater than 2, e.g., HL3, LH3, HH3, HL4, LH4, HH4, etc., are necessary. In other words, a receiver that wishes to reconstruct an image at quarter resolution could discard all packets where RES >= 6 since those packets can only contribute to HL1, LH1, HH1, HL2, LH2 and HH2 sub-bands.

In the case where all DWT stages consist of both horizontal and vertical transforms, the maximum decodable resolution is reduced by a factor of 2^(7 - N) if all Body Packets where RES > N are discarded.

Decomposition level	Resolution level	Subbands	Keep all Body Packets with RES equal to or less than this value...	... to decode an image with at most these dimensions
1	5	HL1,LH1,HH1	7	W x H
2	4	HL2,LH2,HH2	6	(W/2) x (H/2)
3	3	HX3	5	(W/4) x (H/2)
4	2	HX4	4	(W/8) x (H/2)
5	1	HX5	3	(W/16) x (H/2)
5	0	LX5	2	(W/32) x (H/2)

Table 3: Optional discarding of Body Packets based on the value of the RES field when decoding a reduced resolution image, in the case where N_L = 5 and some DWT stages consist of only horizontal transforms. The image has nominal width and height of W x H.

Table 3 illustrates the case where some of DWT stage consist of only horizontal transforms, as specified at Annex F of [jpeg2000-2].

A receiver can therefore discard all Body Packets where RES is greater than some threshold value if it is interested in decoding an image with its resolution reduced by a factor determined by the threshold value, as illustrated in Table 2 and Table 3.

8.4. Extra information

The receiver MUST accept values XTRAC other than 0 and MUST ignore the value of XTRAB, whose length is given by XTRAC.

Future edition of this specification can specify XTRAB contents such that this content can be ignored by receivers that conform to this specification.

8.5. Reserved values

The receiver MUST ignore the value of reserved values.

8.6. Extension values

The receiver MUST discard an RTP packet that contains any extension value.

8.7. Code-block caching

This section applies only if C = 1.

When C = 1, and as specified in Section 7.9, the sender can improve bandwidth efficiency by only occasionally transmitting code-blocks corresponding to static portions of the video and otherwise transmitting empty code-blocks, as defined at Section 7.9.

When decoding a codestream, and for each code-block in the codestream:

* if the code-block in the codestream is empty, the receiver MUST replace it with a matching code-block from the cache, if one exists; or

* if the code-block in the codestream is not empty, the receiver MUST replace any matching code-block from the cache with the code-block in the codestream.

Two code-blocks are _matching_ if the following characteristics are identical for both: spatial coordinates, resolution level, component, sub-band and value of the TP field of the parent RTP packet.

9. Media Type

9.1. General

This RTP payload format is identified using the media type defined at Section 9.2, which is registered in accordance with [RFC4855] and using the template of [RFC6838].

9.2. Definition

Type name
video

Subtype name
jpeg2000-scl

Required parameters
None

Optional parameters
pixel
Specifies the pixel format used by the video sequence.

The parameter MUST be a URI-reference as specified in [RFC3986].

If the parameter is a relative-ref as specified in [RFC3986], then it MUST be equal to one of the pixel formats specified in Table 5 and the RTP header and payload MUST conform with the characteristics of that pixel format.

If the parameter is not a relative-ref, the specification of the pixel format is left to the application that defined the URI.

If the parameter is not specified, the pixel format is unspecified.

sample
Specifies the format of the samples in each component of the codestream.

The parameter MUST be a URI-reference as specified in [RFC3986].

If the parameter is a relative-ref as specified in [RFC3986], then it MUST be equal to one of the formats specified in Appendix C and the stream MUST conform with the characteristics of that format.

If the parameter is not a relative-ref, the specification of the sample format is left to the application that defined the URI.

If the parameter is not specified, the sample format is unspecified.

width
: Maximum width in pixels of each image. Integer between 0 and 4,294,967,295.

  The parameter MUST be a sequence of 1 or more digits.

  If the parameter is not specified, the maximum width is unspecified.

height
: Maximum height in pixels of each image. Integer between 0 and 4,294,967,295.

  The parameter MUST be a sequence of 1 or more digits.

  If the parameter is not specified, the maximum height is unspecified.

signal
: Specifies the sequence of image types.

  The parameter MUST be a URI-reference as specified in [RFC3986].

  If the parameter is a relative-ref as specified in [RFC3986], then it MUST be equal to one of the signal formats specified in Appendix B and the image sequence MUST conform to that signal format.

  If the parameter is not a relative-ref, the specification of the pixel format is left to the application that defined the URI.

  If the parameter is not specified, the stream consists of an arbitrary sequence of image types.

caps
: The parameters contains a list of sets of constraints to which the stream conforms, with each set of constraints identified using an absolute-URI defined by an application.

  The parameter MUST conform to the uri-list syntax expressed using ABNF ([RFC5234]):

```
uri-list = absolute-URI *(";" absolute-URI)
```

  Each absolute-URI MUST NOT contain any ";" character.

The application that defines the absolute-URI MUST associate it with a set of constraints to which the stream conforms. Such constraints can, for example, include the maximum height and width of images.

If the parameter is not specified, constraints, beyond those specified in this document, are unspecified.

tile

The parameter MUST conform to the tile syntax expressed using ABNF ([RFC5234]):

```
tile        = tile-index "&" image-siz
tile-index  = %x31-39 *%x30-39
image-siz   = 1*HEXDIG
```

If the tile parameter is present, each image MUST correspond to one tile of a multi-tile image, as defined in Section 6.2.

tile-index is the index of the tile in the multi-tile image.

image-siz contains the SIZ marker segment parameters of the multi-tile image, encoded as a case insensitive hexadecimal string.

The SIZ parameters of each single-tile image MUST conform to the following:

* Xsiz MUST be equal to the smaller of (i) the coordinate of the right edge of tile index tile-index in the multi-tile image and (ii) Xsiz of the multi-tile image.

* Ysiz MUST be equal to the smaller of (i) the coordinate of the bottom edge of tile index tile-index in the multi-tile image and (ii) Ysiz of the multi-tile image.

* XOsiz MUST be equal to the larger of (i) the coordinate of the left edge of tile index tile-index in the multi-tile image and (ii) XOsiz of the multi-tile image.

* YOsiz MUST be equal to the larger of (i) the coordinate of the top edge of tile index tile-index in the multi-tile image and (ii) YOsiz of the multi-tile image.

* XTOsiz MUST be equal to the coordinate of the left edge of tile index tile-index in the multi-tile image.

* YTOsiz MUST be equal to the coordinate of the top edge of tile index tile-index in the multi-tile image.

* All other parameters MUST be equal to that in the multi-tile image.

Figure 4 illustrates an example where a multi-tile image that consists of two tiles is transmitted as two single-tile images (images 1 and 2). Figure 5 and Table 4 describe the tile and SIZ parameter values, respectively.

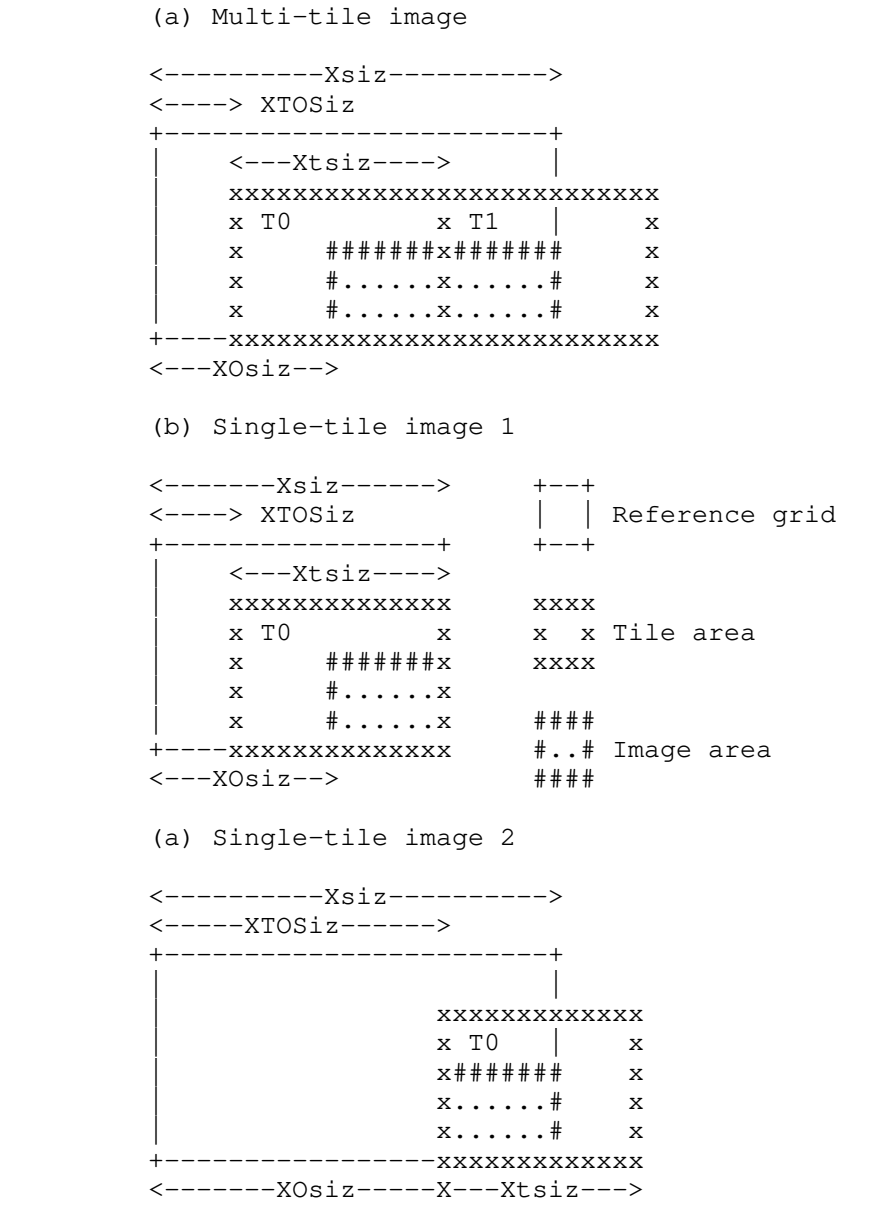


Figure 4: Ssiz parameters for a multi-tile image and two corresponding single-tile images 1 and 2.

NOTE: '\' line wrapping per RFC 8792

```
tile=0&00000190000000c8000000c800000000000000\
c8000000c8000000640000000000001080000
```

Figure 5: Example tile parameter for a multi-tile image that consists of two tiles.

SIZ parameter	Two-tile image	Image where tile-index = 1	Image where tile-index = 2
XSiz	400	300	400
YSiz	200	200	200
XTSiz	200	200	200
YTSiz	200	200	200
XOSiz	200	200	300
YOSiz	0	0	0
XTOSiz	100	100	300
YTOSiz	0	0	0

Table 4: Selected SIZ parameters for a two-tile image and two corresponding single-time images.

cache
The value of the parameter MUST be either false or true.

If the parameter is true, the field C MAY be 0 or 1; otherwise the field C MUST be 0.

If the parameter is not specified, then the parameter is equal to false.

Encoding considerations
This media type is framed and binary, see Section 4.8 of [RFC6838].

Security considerations
See Section 12.

Interoperability considerations
The RTP stream is a sequence of JPEG 2000 images. An implementation that conforms to the family of JPEG 2000 standards can decode and attempt to display each image.

Published specification
This document

Applications that use this media type
video streaming and communication

Person and email address to contact for further information
Pierre-Anthony Lemieux <pal@sandflow.com>

Intended usage
COMMON

Restrictions on Usage
This media type depends on RTP framing, and hence is only defined for use with RTP as specified at [RFC3550]. Transport within other framing protocols is not defined at the time.

Author
Pierre-Anthony Lemieux (mailto:pal@sandflow.com)

Change controller
IETF Audio/Video Transport Core Maintenance Working Group delegated from the IESG.

10. Mapping to the Session Description Protocol (SDP)

The mapping of the payload format media type and its parameters to SDP, as specified in [RFC8866] MUST be done according to Section 3 of [RFC4855].

11. IANA Considerations

This memo requests that IANA registers the content type specified at Section 9. The media type is also requested to be added to the IANA registry for RTP Payload Format MIME types (http://www.iana.org/assignments/rtp-parameters).

12. Security considerations

RTP packets using the payload format specified in this document are subject to the security considerations discussed in [RFC3550] , and in any applicable RTP profile such as [RFC3551], [RFC4585], [RFC3711], [RFC5124]. However, as [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 [RFC7201]. Applications SHOULD use one or more appropriate strong security mechanisms. The rest of this Security Considerations section discusses the security impacting properties of the payload format itself.

This RTP payload format and its media decoder do not exhibit any significant non-uniformity in the receiver-side computational complexity for RTP Packet processing, and thus are unlikely to pose a denial-of-service threat due to the receipt of pathological data. Nor does the RTP payload format contain any active content.

Security considerations related to the JPEG 2000 codestream contained in the payload are discussed at Section 3 of [RFC3745].

13. References

13.1. Normative References

[jpeg2000-1]
ITU-T, "Recommendation ITU-T T.800, JPEG 2000 image coding system: Core coding system", June 2019.

[jpeg2000-2]
ITU-T, "Recommendation ITU-T T.801, JPEG 2000 image coding system: Extensions", June 2021.

[jpeg2000-15]
ITU-T, "Recommendation ITU-T T.814, JPEG 2000 image coding system: High-throughput JPEG 2000", June 2019.

[rec-itu-t-h273]
ITU-T, "Recommendation ITU-T H.273, Coding-independent code points for video signal type identification", July 2021.

[jpeg2000-9]
ITU-T, "JPEG 2000 image coding system: Interactivity tools, APIs and protocols", January 2005.

[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>.

[RFC8866] Begen, A., Kyzivat, P., Perkins, C., and M. Handley, "SDP: Session Description Protocol", RFC 8866, DOI 10.17487/RFC8866, January 2021, <https://www.rfc-editor.org/info/rfc8866>.

[RFC4855] Casner, S., "Media Type Registration of RTP Payload Formats", RFC 4855, DOI 10.17487/RFC4855, February 2007, <https://www.rfc-editor.org/info/rfc4855>.

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

[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>.

[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>.

[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>.

13.2. Informative References

[RFC5371] Futemma, S., Itakura, E., and A. Leung, "RTP Payload Format for JPEG 2000 Video Streams", RFC 5371, DOI 10.17487/RFC5371, October 2008, <https://www.rfc-editor.org/info/rfc5371>.

[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>.

[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>.

[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>.

[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>.

[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>.

[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>.

[RFC3745] Singer, D., Clark, R., and D. Lee, "MIME Type Registrations for JPEG 2000 (ISO/IEC 15444)", RFC 3745, DOI 10.17487/RFC3745, April 2004, <https://www.rfc-editor.org/info/rfc3745>.

[RFC5450] Singer, D. and H. Desineni, "Transmission Time Offsets in RTP Streams", RFC 5450, DOI 10.17487/RFC5450, March 2009, <https://www.rfc-editor.org/info/rfc5450>.

Appendix A. Pixel formats

Table 5 defines pixel formats.

NAME	SAMP	COMPS	TRANS	PRIMS	MAT	VFR	Mapping in Table 1
rgb444sdr	4:4:4	RGB	1	1	0	0, 1	RGB
rgb444wcg	4:4:4	RGB	1	9	0	0, 1	RGB
rgb444pq	4:4:4	RGB	16	9	0	0, 1	RGB
rgb444hlg	4:4:4	RGB	18	9	0	0, 1	RGB
ycbcr420sdr	4:2:0	YCbCr	1	1	1	0	YCbCr
ycbcr422sdr	4:2:2	YCbCr	1	1	1	0	YCbCr
ycbcr422wcg	4:2:2	YCbCr	1	9	9	0	YCbCr
ycbcr422pq	4:2:2	YCbCr	16	9	9	0	YCbCr
ycbcr422hlg	4:2:2	YCbCr	18	9	9	0	YCbCr

Table 5: Defined pixel formats

Each pixel format is characterized by the following:

NAME
: Identifies the pixel format

COMPS
: RGB Each codestream contains exactly three components, associated with the R, G and B color channels, in order.

: YCbCr Each codestream contains exactly three components, associated with the Y, C_b and C_r color channels, in order.

SAMP
: 4:2:0 The C_b and C_r color channels are subsampled horizontally and vertically by 1/2.

: 4:2:2 The C_b and C_r color channels are subsampled horizontally by 1/2.

4:4:4 No color channels are sub-sampled.

TRANS
: Identifies the transfer characteristics allowed by the pixel format, as defined at [rec-itu-t-h273]

PRIMS
: Identifies the color primaries allowed by the pixel format, as defined at [rec-itu-t-h273]

MAT
: Identifies the matrix coefficients allowed by the pixel format, as defined at [rec-itu-t-h273]

VFR
: Allows values of the VideoFullRangeFlag defined at [rec-itu-t-h273]

Appendix B. Signal formats

prog
: The stream MUST only consist of a sequence of progressive frames.

psf
: Progressive segmented frame (PsF) stream. The stream MUST only consist of an alternating sequence of first segment and second segment.

tff
: Interlaced stream. The stream MUST only consist of an alternating sequence of first field and second field, where the first line of the first field is the first line of the frame.

bff
: Interlaced stream. The stream MUST only consist of an alternating sequence of first field and second field, where the first line of the first field is the second line of the frame.

Appendix C. Sample formats

8
: All components consist of unsigned 8-bit integer samples.

10
: All components consist of unsigned 10-bit integer samples.

12
: All components consist of unsigned 12-bit integer samples.

16
All components consist of unsigned 16-bit integer samples.

Appendix D. Summary of Changes (Informative)

D.1. Introduction

This Appendix summarizes substantive changes across revisions of this specification. This summary is informative and not intended to be exhaustive.

D.2. Changes from draft-ietf-avtcore-rtp-j2k-scl-00

* Allow multi-tile images in a single stream, in addition to allowing multi-tile images to be transmitted as multiple single-tile streams, as specified at Section 6.2.

* Fix incorrect TRANS values at Table 5.

Authors' Addresses

Pierre-Anthony Lemieux (editor)
Sandflow Consulting LLC
San Mateo, CA
United States of America
Email: pal@sandflow.com

David Scott Taubman
University of New South Wales
Sydney
Australia
Email: d.taubman@unsw.edu.au

Audio/Video Transport Core Maintenance
Internet-Draft
Intended status: Experimental
Expires: 5 September 2024

J. Ott
M. Engelbart
Technical University Munich
S. Dawkins
Tencent America LLC

4 March 2024

RTP over QUIC (RoQ)
draft-ietf-avtcore-rtp-over-quic-09

Abstract

This document specifies a minimal mapping for encapsulating Real-time Transport Protocol (RTP) and RTP Control Protocol (RTCP) packets within the QUIC protocol. This mapping is called RTP over QUIC (RoQ). It also discusses how to leverage state from the QUIC implementation in the endpoints, in order to reduce the need to exchange RTCP packets and how to implement congestion control and rate adaptation without relying on RTCP feedback.

Discussion Venues

This note is to be removed before publishing as an RFC.

Discussion of this document takes place on the Audio/Video Transport Core Maintenance Working Group mailing list (avt@ietf.org), which is archived at https://mailarchive.ietf.org/arch/browse/avt/.

Source for this draft and an issue tracker can be found at https://github.com/mengelbart/rtp-over-quic-draft.

Status of This Memo

Table of Contents

1. Introduction

This document specifies a minimal mapping for encapsulating Real-time Transport Protocol (RTP) [RFC3550] and RTP Control Protocol (RTCP) [RFC3550] packets within the QUIC protocol ([RFC9000]). This mapping is called RTP over QUIC (RoQ). It also discusses how to leverage state from the QUIC implementation in the endpoints, in order to reduce the need to exchange RTCP packets, and how to implement congestion control and rate adaptation without relying on RTCP feedback.

1.1. Background

The Real-time Transport Protocol (RTP) [RFC3550] is generally used to carry real-time media for conversational media sessions, such as video conferences, across the Internet. Since RTP requires real-time delivery and is tolerant to packet losses, the default underlying transport protocol has been UDP, recently with DTLS on top to secure the media exchange and occasionally TCP (and possibly TLS) as a fallback.

This specification describes an application usage of QUIC ([RFC9308]). As a baseline, the specification does not expect more than a standard QUIC implementation as defined in [RFC8999], [RFC9000], [RFC9001], and [RFC9002], providing a secure end-to-end transport that is also expected to work well through NATs and firewalls. Beyond this baseline, real-time applications can benefit from QUIC extensions such as unreliable DATAGRAMs [RFC9221], which provides additional desirable properties for real-time traffic (e.g., no unnecessary retransmissions, avoiding head-of-line blocking).

1.2. What's in Scope for this Specification

This document defines a mapping for RTP and RTCP over QUIC, called RoQ, and describes ways to reduce the amount of RTCP traffic by leveraging state information readily available within a QUIC endpoint. This document also describes different options for implementing congestion control and rate adaptation for RoQ.

This specification focuses on providing a secure encapsulation of RTP and RTCP packets for transmission over QUIC. The expected usage is wherever RTP is used to carry media packets, allowing QUIC in place of other transport protocols such as TCP, UDP, SCTP, DTLS, etc. That is, we expect RoQ to be used in contexts in which a signaling protocol is used to announce or negotiate a media encapsulation and the associated transport parameters (such as IP address, port number). RoQ is not intended as a stand-alone media transport, although media transport parameters could be statically configured.

The above implies that RoQ is targeted at peer-to-peer operation; but it may also be used in client-server-style settings, e.g., when talking to a conference server as described in RFC 7667 ([RFC7667]), or, if RoQ is used to replace RTSP ([RFC7826]), to a media server.

An appropriate rate adaptation algorithm can be plugged in to adapt the media bitrate to the available bandwidth. This document does not mandate any specific rate adaptation mechanism, so the application can use a rate adaption mechanism of its choice.

Moreover, this document describes how a QUIC implementation and its API can be extended to improve efficiency of the RoQ protocol operation.

RoQ does not impact the usage of RTP Audio Video Profiles (AVP) ([RFC3551]), or any RTP-based mechanisms, even though it may render some of them unnecessary, e.g., Secure Real-Time Transport Prococol (SRTP) ([RFC3711]) might not be needed, because end-to-end security is already provided by QUIC, and double encryption by QUIC and by SRTP might have more costs than benefits. Nor does RoQ limit the use of RTCP-based mechanisms, even though some information or functions obtained by using RTCP mechanisms may also be available from the underlying QUIC implementation by other means.

Between two (or more) endpoints, RoQ supports multiplexing multiple RTP-based media streams within a single QUIC connection and thus using a single (destination IP address, destination port number, source IP address, source port number, protocol) 5-tuple. We note that multiple independent QUIC connections may be established in parallel using the same destination IP address, destination port number, source IP address, source port number, protocol) 5-tuple., e.g. to carry different media channels. These connections would be logically independent of one another.

1.3. What's Out of Scope for this Specification

This document does not attempt to enhance QUIC for real-time media or define a replacement for, or evolution of, RTP. Work to map other media transport protocols to QUIC is under way elsewhere in the IETF.

RoQ is designed for use with point-to-point connections, because QUIC itself is not defined for multicast operation. The scope of this document is limited to unicast RTP/RTCP, even though nothing would or should prevent its use in multicast setups once QUIC supports multicast.

RoQ does not define congestion control and rate adaptation algorithms for use with media. However, Section 7 discusses options for how congestion control and rate adaptation could be performed at the QUIC and/or at the RTP layer, and Section 11 describes the information available at the QUIC layer that could be exposed via an API for the benefit of RTP layer implementation.

RoQ does not define prioritization mechanisms when handling different media as those would be dependent on the media themselves and their relationships. Prioritization is left to the application using RoQ.

This document does not cover signaling for session setup. SDP for RoQ is defined in separate documents such as [I-D.draft-dawkins-avtcore-sdp-rtp-quic], and can be carried in any signaling protocol that can carry SDP, including the Session Initiation Protocol (SIP) ([RFC3261]), Real-Time Protocols for Browser-Based Applications (RTCWeb) ([RFC8825]), or WebRTC-HTTP Ingestion Protocol (WHIP) ([I-D.draft-ietf-wish-whip]).

2. Terminology and Notation

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

> *Note to the Reader:* the meaning of the terms "congestion control" and "rate adaptation" in the IETF community have evolved over the decades since "slow start" and "congestion avoidance" were added as mandatory to implement in TCP, in Section 4.2.2.15 of [RFC1122]. Historically, "congestion control" usually referred to "achieving network stability" ([VJMK88]), by protecting the network from senders who continue to transmit packets that exceed the ability of the network to carry them, even after packet loss occurs (called "congestion collapse").
>
> Modern general-purpose "congestion control" algorithms have moved beyond avoiding congestion collapse, and work to avoid "bufferbloat", which causes increasing round-trip delays, as described in Section 7.2.
>
> "Rate adaptation" more commonly refers to strategies intended to guide senders on when to send "the next packet", so that one-way delays along the network path remain minimal.

When RTP runs over QUIC, as described in this specification, QUIC is performing congestion control, and the RTP application is responsible for performing rate adaptation.

In this document, these terms are used with the meanings listed below, with the recognition that not all the references in this document use these terms in the same way.

The following terms are used:

Bandwidth Estimation: An algorithm to estimate the available bandwidth of a link in a network. Such an estimation can be used for rate adaptation, i.e., adapt the rate at which an application transmits data.

Congestion Control: A mechanism to limit the aggregate amount of data that has been sent over a path to a receiver, but has not been acknowledged by the receiver. This prevents a sender from overwhelming the capacity of a path between a sender and a receiver, causing some outstanding data to be discarded before the receiver can receive the data and acknowledge it to the sender.

Datagram: The term "datagram" is ambiguous. Without a qualifier, "datagram" could refer to a UDP packet, or a QUIC DATAGRAM frame, as defined in QUIC's unreliable DATAGRAM extension [RFC9221], or an RTP packet encapsulated in UDP, or an RTP packet capsulated in QUIC DATAGRAM frame. If not explicitly qualified, the term "datagram" in this document refers to an RTP packet, and the uppercase "DATAGRAM" refers to a QUIC DATAGRAM frame. This document also uses the term "RoQ datagram" as a short form of "RTP packet encapsulated in a QUIC DATAGRAM frame".

Endpoint: A QUIC server or client that participates in an RoQ session.

Frame: A QUIC frame as defined in [RFC9000].

Rate Adaptation: An application-level mechanism that adjusts the sending rate of an application in order to respond to changing path conditions. For example, an application sending video might respond to indications of congestion by adjusting the resolution of the video it is sending.

Receiver: An endpoint that receives media in RTP packets and may send or receive RTCP packets.

Sender: An endpoint that sends media in RTP packets and may send or receive RTCP packets.

Stream: The term "stream" is ambiguous. Without a qualifier, "stream" could refer to a QUIC STREAM frame, as defined in [RFC9000], or a series of QUIC STREAM frames in a single stream, or a series of RTP packets encapsulated in QUIC STREAM frames. If not explicitly qualified, the term "STREAM" in this document refers to a QUIC STREAM frame, and "stream" in this document refers to one or more RTP packets encapsulated in QUIC STREAM frames. This document also uses the term "RoQ stream" as a short form of "one or more RTP packets encapsulated in QUIC STREAM frames".

Packet diagrams in this document use the format defined in Section 1.3 of [RFC9000] to illustrate the order and size of fields.

3. Protocol Overview

This document introduces a mapping of the Real-time Transport Protocol (RTP) to the QUIC transport protocol. RoQ allows the use of QUIC streams and QUIC DATAGRAMs to transport real-time data, and thus, the QUIC implementation MUST support QUIC's DATAGRAM extension, if RTP packets are to be sent over QUIC DATAGRAMs.

[RFC3550] specifies that RTP sessions need to be transmitted on different transport addresses to allow multiplexing between them. RoQ uses a different approach to leverage the advantages of QUIC connections without managing a separate QUIC connection per RTP session. [RFC9221] does not provide demultiplexing between different flows on DATAGRAMs but suggests that an application implement a demultiplexing mechanism if required. An example of such a mechanism would be flow identifiers prepended to each DATAGRAM frame as described in Section 2.1 of [I-D.draft-ietf-masque-h3-datagram]. RoQ uses a flow identifier to replace the network address and port number to multiplex many RTP sessions over the same QUIC connection.

An RTP application is responsible for determining what to send in an encoded media stream, and how to send that encoded media stream within a targeted bitrate.

This document does not mandate how an application determines what to send in an encoded media stream, because decisions about what to send within a targeted bitrate, and how to adapt to changes in the targeted bitrate, can be application and codec-specific. For example, adjusting quantization in response to changing network conditions may work well in many cases, but if what's being shared is video that includes text, maintaining readability is important.

As of this writing, the IETF has produced two Experimental-track congestion control specifications, Network-Assisted Dynamic Adaptation (NADA) [RFC8698] and Self-Clocked Rate Adaptation for Multimedia (SCReAM) [RFC8298]. These congestion control algorithms require some feedback about the network's performance to calculate target bitrates. Traditionally this feedback is generated at the receiver and sent back to the sender via RTCP.

Since QUIC also collects some metrics about the network's performance, these metrics can be used to generate the required feedback at the sender-side and provide it to the congestion control algorithm to avoid the additional overhead of the RTCP stream. This is discussed in more detail in Section 9.

3.1. Motivations

From time to time, someone asks the reasonable question, "why should anyone implement and deploy RoQ"? This reasonable question deserves a better answer than "because we can". Upon reflection, the following motivations seem useful to state.

The motivations in this section are in no particular order, and this reflects the reality that not all implementers and deployers would agree on "the most important motivations".

3.1.1. "Always-On" Transport-level Authentication and Encryption

Although application-level mechanisms to encrypt RTP and RTCP payloads have existed since the introduction of Secure Real-time Transport Protocol (SRTP) [RFC3711], encryption of RTP and RTCP header fields and contributing sources has only been defined recently (in Cryptex [RFC9335], and both SRTP and Cryptex are optional capabilities for RTP.

This is in sharp contrast to "always-on" transport-level encryption in the QUIC protocol, using Transport Layer Security (TLS 1.3) as described in [RFC9001]. QUIC implementations always authenticate the entirety of each packet, and encrypt as much of each packet as is practical, even switching from "long headers", which expose more QUIC header fields needed to establish a connection, to "short headers", which only expose the absolute minimum QUIC header fields needed to identify the connection to the receiver, so that the QUIC payload is presented to the right QUIC application [RFC8999].

3.1.2. "Always-On" Internet-Safe Congestion Avoidance

When RTP is carried directly over UDP, as is commonly done, the underlying UDP protocol provides no transport services beyond path multiplexing using UDP ports. All congestion avoidance behavior is up to the RTP application itself, and if anything goes wrong with the application resulting in an RTP sender failing to recognize that it is contributing to path congestion, the "worst case" response is to invoke RTP "circuit breaker" procedures [RFC8083], resulting in "ceasing transmission", as described in Section 4.5 of [RFC8083]. Because RTCP-based circuit breakers only detect long-lived congestion, a response based on these mechanisms will not happen quickly.

In contrast, when RTP is carried over QUIC, QUIC implementations maintain their own estimates of key transport parameters needed to detect and respond to possible congestion, and these are independent of any measurements RTP senders and receivers are maintaining. The result is that even if an RTP sender continues to "send", QUIC congestion avoidance procedures (for example, the procedures defined in [RFC9002]) will cause the RTP packets to be buffered while QUIC responds to detected packet loss. This happens without RTP senders taking any action, but the RTP sender has no control over this QUIC mechanism.

Moreover, when a single QUIC connection is used to multiplex both RTP-RTCP and non-RTP packets as described in Section 3.1.5, the shared QUIC connection will still be Internet-safe, with no coordination required.

While QUIC's response to congestion ensures that RoQ will be "Internet-safe", from the network's perspective, it is helpful to remember that a QUIC sender responds to detected congestion by delaying packets that are already available to send, to give the path to the QUIC receiver time to recover from congestion.

* If the QUIC connection encapsulates RTP, this means that some RTP packets will be delayed, and will arrive at the receiver later than a user of the RTP flow might prefer.

* If the QUIC connection also encapsulates RTCP, this means that these RTCP messages will also be delayed, and will not be sent in a timely manner. This delay can interfere with a sender's ability to stabilize rate control and achieve audio/video synchronization.

Taken as a whole,

* Timely RTP stream-level rate adaptation will give a better user experience by minimizing endpoint queuing delays and packet loss,

* but in the presence of packet loss, QUIC connection-level congestion control will respond more quickly to the end of congestion than RTP "circuit breakers".

3.1.3. RTP Rate Adaptation Based on QUIC Feedback

RTP makes use of a large number of RTP-specific feedback mechanisms because when RTP is carried directly over UDP, there is no other way to receive feedback. Some of these mechanisms are specific to the type of media RTP is sending, but others can be mapped from underlying QUIC implementations that are using this feedback to perform congestion control for any QUIC connection, regardless of the application reflected in the QUIC STREAM [RFC9000] and DATAGRAM [RFC9221] frames. This is described in (much) more detail in Section 7 on rate adaptation, and in Section 9 on replacing RTCP and RTP header extensions with QUIC feedback.

One word of caution is in order - RTP implementations may rely on at least some minimal periodic RTCP feedback, in order to determine that an RTP flow is still active, and is not causing sustained congestion (as described in [RFC8083], but since this "periodicity" is measured in seconds, the impact of this "duplicate" feedback on path bandwidth utilization is likely close to zero.

3.1.4. Path MTU Discovery and RTP Media Coalescence

The minimum Path MTU supported by conformant QUIC implementations is 1200 bytes [RFC9000], and in addition, QUIC implementations allow senders to use either DPLPMTUD ([RFC8899]) or PMTUD ([RFC1191], [RFC8201]) to determine the actual MTU size that the receiver and path between sender and receiver support, which can be even larger.

This is especially useful in certain conferencing topologies, where otherwise senders have no choice but to use the lowest path MTU for all conference participants, but even in point-to-point RTP sessions, this also allows senders to piggyback audio media in the same UDP packet as video media, for example, and also allows QUIC receivers to piggyback QUIC ACK frames on any QUIC packets being transmitted in the other direction.

3.1.5. Multiplexing RTP, RTCP, and Non-RTP Flows on a Single QUIC Connection

This specification defines a flow identifier for multiplexing multiple RTP and RTCP ports on the same QUIC connection to conserve ports, especially at NATs and Firewalls. Section 5.1 describes the multiplexing in more detail. Future extensions could further build on the flow identifier to multiplex RTP/RTCP with other protocols on the same connection, as long as these protocols can co-exist with RTP/RTCP without interfering with the ability of this connection to carry real-time media.

3.1.6. Exploiting Multiple Paths

Although there is much interest in multiplexing flows on a single QUIC connection as described in Section 3.1.5, QUIC also provides the capability of establishing and validating multiple paths for a single QUIC connection as described in Section 9 of [RFC9000]. Once multiple paths have been validated, a sender can migrate from one path to another with no additional signaling, allowing an endpoint to move from one endpoint address to another without interruption, as long as only a single path is in active use at any point in time.

Connection migration may be desireable for a number of reasons, but to give one example, this allows a QUIC connection to survive address changes due to a middlebox allocating a new outgoing port, or even a new outgoing IP address.

The Multipath Extension for QUIC [I-D.draft-ietf-quic-multipath] would allow the application to actively use two or more paths simultaneously, but in all other respects, this functionality is the same as QUIC connection migration.

A sender can use these capabilities to more effectively exploit multiple paths between sender and receiver with no action required from the application, even if these paths have different path characteristics. Examples of these different path characteristics include handling paths differently if one path has higher available bandwidth and the other has lower one-way latency, or if one is a more costly cellular path and the other is a less costly WiFi path.

Some of these differences can be detected by QUIC itself, while other differences must be described to QUIC based on policy, etc. Possible RTP implementation strategies for path selection and utilization are not discussed in this specification.

3.1.7. Exploiting New QUIC Capabilities

The first version of the QUIC protocol described in [RFC9000] has been completed, but extensions to QUIC are still under active development in the IETF. Because of this, using QUIC as a transport for a mature protocol like RTP allows developers to exploit new transport capabilities as they become available.

3.2. RTP with QUIC Streams, QUIC Datagrams, and a Mixture of Both

This document describes the use of QUIC streams and DATAGRAMs as RTP encapsulations, but does not take a position on which encapsulation an application should use. Indeed, an application can use both QUIC streams and DATAGRAM encapsulations. The choice of encapsulation is left to the application developer, but it is worth noting the differences.

QUIC [RFC9000] was initially designed to carry HTTP [RFC9114] in QUIC STREAM frames, and QUIC STREAM frames provide what HTTP application developers require - for example, QUIC STREAM frames provide a stateful, connection-oriented, flow-controlled, reliable, ordered stream of bytes to an application. QUIC STREAM frames can be multiplexed over a single QUIC connection, using stream IDs to demultiplex incoming messages.

QUIC Datagrams [RFC9221] were developed as a QUIC extension, intended to support applications that do not require reliable delivery of application data. This extension defines two DATAGRAM frame types (one including a length field, the other not including a length field), and these DATAGRAM frames can co-exist with QUIC STREAM frames within a single QUIC connection, sharing the connection's cryptographic and authentication context, and congestion controller context.

There is no default relative priority between DATAGRAM frames with respect to each other, and there is no default priority between DATAGRAM frames and QUIC STREAM frames. The implementation likely presents an API to allow appplications to assign relative priorities, but this is not mandated by the standard and may not be present in all implementations.

Because DATAGRAMs are an extension to QUIC, they inherit a great deal of functionality from QUIC (much of which is described in Section 3.1); so much so that it is easier to explain what DATAGRAMs do NOT inherit.

* DATAGRAM frames do not provide any explicit flow control signaling. This means that a QUIC receiver may not be able to commit the necessary resources to process incoming frames, but the purpose for DATAGRAM frames is to carry application-level information that can be lost and will not be retransmitted,

* DATAGRAM frames do inherit the QUIC connection's congestion controller. This means that although there is no frame-level flow control, DATAGRAM frames may be delayed until the controller allows them to be sent, or dropped (with an optional notification to the sending application). Implementations can also delay sending DATAGRAM frames to maintain consistent packet pacing (as described in Section 7.7 of [RFC9002]), and can allow an application to specify a sending expiration time, but these capabilities are not mandated by the standard and may not be present in all implementations.

* DATAGRAM frames cannot be fragmented. They are limited in size by the max_datagram_frame_size transport parameter, and further limited by the max_udp_payload_size transport parameter and the Maximum Transmission Unit (MTU) of the path between endpoints.

* DATAGRAM frames belong to a QUIC connection as a whole. There is no QUIC-level way to multiplex/demultiplex DATAGRAM frames within a single QUIC connection. Any multiplexing identifiers must be added, interpreted, and removed by an application, and they will be sent as part of the payload of the DATAGRAM frame itself.

Because DATAGRAMs are an extension to QUIC, a RoQ endpoint cannot count on a RoQ peer supporting that extension. The RoQ endpoint may discover that its peer does not support DATAGRAMs while using signaling to set up QUIC connections, but may also discover that its peer has not negotiated the use of this extension during the QUIC handshake. When this happens, the RoQ endpoint needs to make a decision about what to do next.

* If the use of DATAGRAMs was critical for the application, the endpoint can simply close the QUIC connection, allowing someone or something to correct this mismatch, so that DATAGRAMs can be used.

* If the use of DATAGRAMs was not critical for the application, the endpoint can negotiate the use of QUIC STREAM frames instead.

3.3. Supported RTP Topologies

RoQ only supports some of the RTP topologies described in [RFC7667]. Most notably, due to QUIC [RFC9000] being a purely IP unicast protocol at the time of writing, RoQ cannot be used as a transport protocol for any of the paths that rely on IP multicast in several multicast topologies (e.g., _Topo-ASM_, _Topo-SSM_, _Topo-SSM-RAMS_).

Some "multicast topologies" can include IP unicast paths (e.g., _Topo-SSM_, _Topo-SSM-RAMS_). In these cases, the unicast paths can use RoQ.

RTP supports different types of translators and mixers. Whenever a middlebox such as a translator or a mixer needs to access the content of RTP/RTCP-packets, the QUIC connection has to be terminated at that middlebox.

RoQ streams (see Section 5.2) can support much larger RTP packet sizes than other transport protocols such as UDP can, which can lead to problems with transport translators which translate from RoQ to RTP over a different transport protocol. A similar problem can occur if a translator needs to translate from RTP over UDP to RoQ over DATAGRAMs, where the max_datagram_frame_size of a QUIC DATAGRAM may be smaller than the MTU of a UDP datagram. In both cases, the translator may need to rewrite the RTP packets to fit into the smaller MTU of the other protocol. Such a translator may need codec-specific knowledge to packetize the payload of the incoming RTP packets in smaller RTP packets.

Additional details are provided in the following table.

RFC 7667 Section	Shortcut Name	RTP over QUIC?	Comments
3.1 (https://datatracker.ietf.org/doc/html/rfc7667#section-3.1)	Topo-Point-to-Point	yes	
3.2.1.1 (https://datatracker.ietf.org/doc/html/rfc7667#section-3.2.1.1)	Topo-PtP-Relay	yes	Note-NAT
3.2.1.2 (https://datatracker.ietf.org/doc/html/rfc7667#section-3.2.1.2)	Topo-Trn-Translator	yes	Note-MTU Note-SEC
3.2.1.3	Topo-Media-	yes	Note-MTU

(https://datatracker.ietf.org/doc/html/rfc7667#section-3.2.1.3)	Translator		
3.2.2 (https://datatracker.ietf.org/doc/html/rfc7667#section-3.2.2)	Topo-Back-To-Back	yes	Note-SEC Note-MTU Note-MCast
3.3.1 (https://datatracker.ietf.org/doc/html/rfc7667#section-3.3.1)	Topo-ASM	no	Note-MCast
3.3.2 (https://datatracker.ietf.org/doc/html/rfc7667#section-3.3.2)	Topo-SSM	partly	Note-MCast Note-UCast-MCast
3.3.3 (https://datatracker.ietf.org/doc/html/rfc7667#section-3.3.3)	Topo-SSM-RAMS	partly	Note-MCast Note-MCast-UCast
3.4 (https://datatracker.ietf.org/doc/html/rfc7667#section-3.4)	Topo-Mesh	yes	Note-MCast
3.5.1 (https://datatracker.ietf.org/doc/html/rfc7667#section-3.5.1)	Topo-PtM-Trn-Translator	possibly	Note-MCast Note-MTU Note-Topo-PtM-Trn-Translator
3.6 (https://datatracker.ietf.org/doc/html/rfc7667#section-3.6)	Topo-Mixer	possibly	Note-MCast Note-Topo-Mixer
3.6.1 (https://datatracker.ietf.org/doc/html/rfc7667#section-3.6.1)	Media-Mixing-Mixer	partly	Note-Topo-Mixer
3.6.2 (https://datatracker.ietf.org/doc/html/rfc7667#section-3.6.2)	Media-Switching-Mixer	partly	Note-Topo-Mixer
3.7 (https://datatracker.ietf.org/doc/html/rfc7667#section-3.7)	Selective Forwarding Middlebox	yes	Note-MCast Note-Topo-Mixer
3.8	Topo-Video-	yes	Note-MTU

(https://datatracker.ietf.org/doc/html/rfc7667#section-3.8)	switch-MCU		Note-MCast Note-Topo-Mixer
3.9 (https://datatracker.ietf.org/doc/html/rfc7667#section-3.9)	Topo-RTCP-terminating-MCU	yes	Note-MTU Note-MCast Note-Topo-Mixer
3.10 (https://datatracker.ietf.org/doc/html/rfc7667#section-3.10)	Topo-Split-Terminal	yes	Note-MCast
3.11 (https://datatracker.ietf.org/doc/html/rfc7667#section-3.11)	Topo-Asymmetric	Possibly	Note-Warn, Note-MCast, Note-MTU

Table 1

Note-NAT: QUIC [RFC9000] doesn't support NAT traversal.

Note-MTU: Supported, but may require MTU adaptation.

Note-Sec: Note that RoQ provides mandatory security, and other RTP transports do not. Section 14 describes strategies to prevent the inadvertent disclosure of RTP sessions to unintended third parties.

Note-MCast: QUIC [RFC9000] cannot be used for multicast paths.

Note-UCast-MCast: The topology refers to a _Distribution Source_, which receives and relays RTP from a number of different media senders via unicast before relaying it to the receivers via multicast. QUIC can be used between the senders and the _Distribution Source_.

Note-MCast-UCast: The topology refers to a _Burst Source_ or _Retransmission Source_, which retransmits RTP to receivers via unicast. QUIC can be used between the _Retransmission Source_ and the receivers.

Note-Topo-PtM-Trn-Translator: Supports unicast paths between RTP sources and translators.

Note-Topo-Mixer: Supports unicast paths between RTP senders and mixers.

Note-Warn: Quote from [RFC7667]: _This topology is so problematic and it is so easy to get the RTCP processing wrong, that it is NOT RECOMMENDED to implement this topology._

4. Connection Establishment and ALPN

QUIC requires the use of ALPN [RFC7301] tokens during connection setup. RoQ uses "roq" as ALPN token in the TLS handshake (see also Section 15).

Note that the use of a given RTP profile is not reflected in the ALPN token even though it could be considered part of the application usage. This is simply because different RTP sessions, which may use different RTP profiles, may be carried within the same QUIC connection.

4.1. Draft version identification

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

RoQ uses the token "roq" to identify itself in ALPN.

Only implementations of the final, published RFC can identify themselves as "roq". Until such an RFC exists, implementations MUST NOT identify themselves using this string.

Implementations of draft versions of the protocol MUST add the string "-" and the corresponding draft number to the identifier. For example, draft-ietf-avtcore-rtp-over-quic-09 is identified using the string "roq-09".

Non-compatible experiments that are based on these draft versions MUST append the string "-" and an experiment name to the identifier.

5. Encapsulation

This section describes the encapsulation of RTP/RTCP packets in QUIC.

QUIC supports two transport methods: STREAM frames [RFC9000] and DATAGRAMs [RFC9221]. This document specifies mappings of RTP to both transport modes. Senders MAY combine both modes by sending some RTP/RTCP packets over the same or different QUIC streams and others in DATAGRAMs.

Section 5.1 introduces a multiplexing mechanism that supports multiplexing multiple RTP sessions and RTCP. Section 5.2 and Section 5.3 explain the specifics of mapping RTP to QUIC STREAM frames and DATAGRAMs, respectively.

5.1. Multiplexing

RoQ uses flow identifiers to multiplex different RTP and RTCP streams on a single QUIC connection. A flow identifier is a QUIC variable-length integer as described in Section 16 of [RFC9000]. Each flow identifier is associated with a stream of RTP or RTCP packets.

In a QUIC connection using the ALPN token defined in Section 4, every DATAGRAM and every QUIC stream MUST start with a flow identifier. A peer MUST NOT send any data in a DATAGRAM or STREAM frame that is not associated with the flow identifier which started the DATAGRAM or stream.

RTP and RTCP packets of different RTP sessions MUST use distinct flow identifiers. If peers wish to send multiple types of media in a single RTP session, they can do so by following [RFC8860].

A single RTP session can be associated with one or two flow identifiers. Thus, it is possible to send RTP and RTCP packets belonging to the same session using different flow identifiers. RTP and RTCP packets of a single RTP session can use the same flow identifier (following the procedures defined in [RFC5761]), or they can use different flow identifiers.

The association between flow identifiers and RTP/RTCP streams MUST be negotiated using appropriate signaling. If a receiver cannot associate a flow identifier with any RTP/RTCP stream, it MUST close the connection with the application error code ROQ_UNKNOWN_FLOW_ID.

Flow identifiers introduce some overhead in addition to the header overhead of RTP/RTCP and QUIC. QUIC variable-length integers require between one and eight bytes depending on the number expressed. Thus, it is advisable to use low numbers first and only use higher ones if necessary due to an increased number of flows.

5.2. QUIC Streams

To send RTP/RTCP packets over QUIC streams, a sender MUST open at least one new unidirectional QUIC stream. RoQ uses unidirectional streams, because there is no synchronous relationship between sent and received RTP/RTCP packets. A peer that receives a bidirectional stream with a flow identifier that is associated with an RTP or RTCP stream, MUST stop reading from the stream and send a CONNECTION_CLOSE

frame with the frame type set to APPLICATION_ERROR and the error code set to ROQ_STREAM_CREATION_ERROR.

A RoQ sender can open new QUIC streams for different packets using the same flow identifier, for example, to avoid head-of-line blocking.

Because a sender can continue sending on a lower stream number after starting packet transmission on a higher stream number, a RoQ receiver MUST be prepared to receive RoQ packets on any number of QUIC streams (subject to its limit on parallel open streams) and MUST not make assumptions about which RTP sequence numbers are carried in which streams.

5.2.1. Stream Encapsulation

Figure 1 shows the encapsulation format for RoQ Streams.

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

Figure 1: RoQ Streams Payload Format

Flow Identifier: Flow identifier to demultiplex different data flows on the same QUIC connection.

RTP/RTCP Payload: Contains the RTP/RTCP payload; see Figure 2

The payload in a QUIC STREAM frame starts with the flow identifier followed by one or more RTP/RTCP payloads. All RTP/RTCP payloads sent on a STREAM frame MUST belong to the RTP session with the same flow identifier.

Each payload begins with a length field indicating the length of the RTP/RTCP packet, followed by the packet itself, see Figure 2.

```
RTP/RTCP Payload {
  Length(i),
  RTP/RTCP Packet(..),
}
```

Figure 2: RTP/RTCP payload for QUIC STREAM frames

Length: A QUIC variable length integer (see Section 16 of [RFC9000]) describing the length of the following RTP/RTCP packets in bytes.

RTP/RTCP Packet: The RTP/RTCP packet to transmit.

5.2.2. Media Frame Cancellation

QUIC uses RESET_STREAM and STOP_SENDING frames to terminate the sending part of a stream and to request termination of an incoming stream by the sending peer respectively.

A RoQ sender can use RESET_STREAM if it knows that a packet, which was not yet successfully and completely transmitted, is no longer needed.

A RoQ receiver that is no longer interested in reading a certain partition of the media stream can signal this to the sending peer using a STOP_SENDING frame.

In both cases, the error code of the RESET_STREAM frame or the STOP_SENDING frame MUST be set to ROQ_FRAME_CANCELLED.

STOP_SENDING is not a request to the sender to stop sending RTP media, only an indication that a RoQ receiver stopped reading the QUIC stream being used. This can mean that the RoQ receiver is unable to make use of the media frames being received because they are "too old" to be used. A sender with additional media frames to send can continue sending them on another QUIC stream. Alternatively, new media frames can be sent as QUIC datagrams (see Section 5.3). In either case, a RoQ sender resuming operation after receiving STOP_SENDING can continue starting with the newest media frames available for sending. This allows a RoQ receiver to "fast forward" to media frames that are "new enough" to be used.

Any media frame that has already been sent on the QUIC stream that received the STOP_SENDING frame, MUST NOT be sent again on the new QUIC stream(s) or DATAGRAMs.

Note that an RTP receiver cannot request a reset of only a particular media frame because the sending QUIC implementation might already have sent data for the following frame on the same stream. In that case, STOP_SENDING and the following RESET_STREAM would also drop the following media frame and thus lead to unintentionally skipping one or more frames.

A translator that translates between two endpoints, both connected via QUIC, MUST forward RESET_STREAM frames received from one end to the other unless it forwards the RTP packets on encapsulated in DATAGRAMs.

Large RTP packets sent on a stream will be fragmented into smaller QUIC STREAM frames. The QUIC frames are transmitted reliably and in order such that a receiving application can read a complete RTP packet from the stream as long as the stream is not closed with a RESET_STREAM frame. No retransmission has to be implemented by the application since QUIC frames lost in transit are retransmitted by QUIC.

5.2.3. Flow control and MAX_STREAMS

In order to permit QUIC streams to open, a RoQ sender MUST configure non-zero minimum values for the number of permitted streams and the initial stream flow-control window. These minimum values control the number of parallel, or simultaneously active, RTP/RTCP flows. Endpoints that excessively restrict the number of streams or the flow-control window of these streams will increase the chance that the remote peer reaches the limit early and becomes blocked.

Opening new streams for new packets can implicitly limit the number of packets concurrently in transit because the QUIC receiver provides an upper bound of parallel streams, which it can update using QUIC MAX_STREAMS frames. The number of packets that have to be transmitted concurrently depends on several factors, such as the number of RTP streams within a QUIC connection, the bitrate of the media streams, and the maximum acceptable transmission delay of a given packet. Receivers are responsible for providing senders enough credit to open new streams for new packets anytime.

As an example, consider a conference scenario with 20 participants. Each participant receives audio and video streams of every other participant from a central server. If the sender opens a new QUIC stream for every frame at 30 frames per second video and 50 frames per second audio, it will open 1520 new QUIC streams per second. A receiver must provide at least that many credits for opening new unidirectional streams to the server every second.

In addition, the receiver should also consider the requirements of RTCP streams. These considerations may also be relevant when implementing signaling since it may be necessary to inform the receiver about how fast and how many stream credits it will have to provide to the media-sending peer.

5.3. QUIC DATAGRAMs

Senders can also transmit RTP packets in QUIC DATAGRAMs. DATAGRAMs are an extension to QUIC described in [RFC9221]. DATAGRAMs can only be used if the use of the DATAGRAM extension was successfully negotiated during the QUIC handshake. If the QUIC extension was signaled using a signaling protocol, but that extension was not negotiated during the QUIC handshake, a peer can close the connection with the ROQ_EXPECTATION_UNMET error code.

QUIC datagrams preserve frame boundaries. Thus, a single RTP packet can be mapped to a single QUIC datagram without additional framing. Because QUIC DATAGRAMs cannot be IP-fragmented (Section 5 of [RFC9221]), senders need to consider the header overhead associated with QUIC datagrams, and ensure that the RTP/RTCP packets, including their payloads, flow identifier, QUIC, and IP headers, will fit into the path MTU.

Figure 3 shows the encapsulation format for RoQ Datagrams.

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

Figure 3: RoQ Datagram Payload Format

Flow Identifier: Flow identifier to demultiplex different data flows on the same QUIC connection.

RTP/RTCP Packet: The RTP/RTCP packet to transmit.

RoQ senders need to be aware that QUIC uses the concept of QUIC frames. Different kinds of QUIC frames are used for different application and control data types. A single QUIC packet can contain more than one QUIC frame, including, for example, QUIC STREAM frames or DATAGRAM frames carrying application data and ACK frames carrying QUIC acknowledgements, as long as the overall size fits into the MTU. One implication is that the number of packets a QUIC stack transmits depends on whether it can fit ACK and DATAGRAM frames in the same QUIC packet. Suppose the application creates many DATAGRAM frames that fill up the QUIC packet. In that case, the QUIC stack might have to create additional packets for ACK- (and possibly other control-) frames. The additional overhead could, in some cases, be reduced if the application creates smaller RTP packets, such that the resulting DATAGRAM frame can fit into a QUIC packet that can also carry ACK frames.

Since DATAGRAMs are not retransmitted on loss (see also Section 9.4 for loss signaling), if an application wishes to retransmit lost RTP packets, the retransmission has to be implemented by the application. RTP retransmissions can be done in the same RTP session or a separate RTP session [RFC4588] and the flow identifier MUST be set to the flow identifier of the RTP session in which the retransmission happens.

6. Connection Shutdown

Either peer can close the connection for variety of reasons. If one of the peers wants to close the RoQ connection, the peer can use a QUIC CONNECTION_CLOSE frame with one of the error codes defined in Section 10.

7. Congestion Control and Rate Adaptation

Like any other application on the internet, RoQ applications need a mechanism to perform congestion control to avoid overloading the network. QUIC is a congestion-controlled transport protocol. RTP does not mandate a single congestion control mechanism. RTP suggests that the RTP profile defines congestion control according to the expected properties of the application's environment.

This document discusses aspects of transport level congestion control in Section 7.1 and application layer rate control in Section 7.2. It does not mandate any specific congestion control algorithm for QUIC or rate adaptation algorithm for RTP.

This document also gives guidance about avoiding problems with _nested_ congestion controllers in Section 7.2.

This document also discusses congestion control implications of using shared or multiple separate QUIC connections to send and receive multiple independent RTP/RTCP streams in Section 7.3.

7.1. Congestion Control at the Transport Layer

QUIC is a congestion-controlled transport protocol. Senders are required to employ some form of congestion control. The default congestion control specified for QUIC in [RFC9002] is similar to TCP NewReno [RFC6582], but senders are free to choose any congestion control algorithm as long as they follow the guidelines specified in Section 3 of [RFC8085], and QUIC implementors make use of this freedom.

Congestion control mechanisms are often implemented at the transport layer of the protocol stack, but can also be implemented at the application layer.

A congestion control mechanism could respond to actual packet loss (detected by timeouts), or to impending packet loss (signaled by mechanisms such as Explicit Congestion Notification [RFC3168]).

For real-time traffic, it is best that the QUIC implementation use a congestion controller that aims at keeping queues, and thus the latency, at intermediary network elements as short as possible. Delay-based congestion control algorithms might use, for example, an increasing one-way delay as a signal of impending congestion, and adjust the sending rate to prevent continued increases in one-way delay.

A wide variety of congestion control algorithms for real-time media have been developed (for example, "Google Congestion Controller" [I-D.draft-ietf-rmcat-gcc]). The IETF has defined two such algorithms in Experimental RFCs (SCReAM [RFC8298] and NADA [RFC8698]). These algorithms for RTP are specifically tailored for real-time transmissions at low latencies, but this section would apply to any congestion control algorithm that meets the requirements described in "Congestion Control Requirements for Interactive Real-Time Media" [RFC8836].

Some low latency congestion control algorithms depend on detailed arrival time feedback to estimate the current one-way delay between sender and receiver, which is unavailable in QUIC [RFC9000] without extensions. The QUIC implementations of the sender and receiver can use an extension to add this information to QUIC as described in Appendix A. An alternative to these dedicated real-time media congestion-control algorithms that QUIC implementations could support is the Low Latency, Low Loss, and Scalable Throughput (L4S) Internet Service [RFC9330], which can be used to limit growth in round-trip delays, due to increasing queuing delays. While L4S does not rely on a QUIC protocol extension, L4S does rely on support from network devices along the path from sender to receiver.

The application needs a mechanism to query the available bandwidth to adapt media codec configurations. If the employed congestion controller of the QUIC connection keeps an estimate of the available bandwidth, it could expose an API to the application to query the current estimate. If the congestion controller cannot provide a current bandwidth estimate to the application, the sender can implement an alternative bandwidth estimation at the application layer as described in Section 7.2.

It is assumed that the congestion controller in use provides a pacing mechanism to determine when a packet can be sent to avoid bursts and minimize variation in inter-packet arrival times. The currently proposed congestion control algorithms for real-time communications

(e.g., SCReAM and NADA) provide such pacing mechanisms, and QUIC recommends pacing for senders based on the congestion control algorithm.

7.2. Rate Adaptation at the Application Layer

RTP itself does not specify a congestion control algorithm, but [RFC8888] defines an RTCP feedback message intended to enable rate adaptation for interactive real-time traffic using RTP, and successful rate adaptation will accomplish congestion control as well.

If an application cannot access a bandwidth estimation from the QUIC layer, the application can alternatively implement a bandwidth estimation algorithm at the application layer. Congestion control algorithms for real-time media such as GCC [I-D.draft-ietf-rmcat-gcc], NADA [RFC8698], and SCReAM [RFC8298] expose a target_bitrate to dynamically reconfigure media codecs to produce media at the rate of the observed available bandwidth. Applications can use the same bandwidth estimation to adapt their rate when using QUIC. However, running an additional congestion control algorithm at the application layer can have unintended effects due to the interaction of two _nested_ congestion controllers.

If the application transmits media that does not saturate path bandwidth and paces its transmission, more heavy-handed congestion control mechanisms (drastic reductions in the sending rate when loss is detected, with much slower increases when losses are no longer being detected) should rarely come into play. If the application chooses RoQ as its transport, sends enough media to saturate the path bandwidth, and does not adapt its sending rate, drastic measures will be required to avoid sustained or oscillating congestion along the path.

Thus, applications are advised to only use the bandwidth estimation without running the complete congestion control algorithm at the application layer before passing data to the QUIC layer.

The bandwidth estimation algorithm typically needs some feedback on the transmission performance. This feedback can be collected via RTCP or following the guidelines in Section 9 and Section 11.

7.3. Sharing QUIC connections

Two endpoints may establish channels in order to exchange more than one type of data simultaneously. The channels can be intended to carry real-time RTP data or other non-real-time data. This can be realized in different ways.

* One straightforward solution is to establish multiple QUIC connections, one for each channel, whether the channel is used for real-time media or non-real-time data. This is a straightforward solution, but has the disadvantage that transport ports are more quickly exhausted and these are limited by the 16-bit UDP source and destination port number sizes [RFC768].

* Alternatively, all real-time channels are mapped to one QUIC connection, while a separate QUIC connection is created for the non-real-time channels.

* A third option is to multiplex all channels in a single QUIC connection via an extension to RoQ.

In the first two cases, the congestion controllers can be chosen to match the demands of the respective channels and the different QUIC connections will compete for the same resources in the network. No local prioritization of data across the different (types of) channels would be necessary.

Although it is possible to multiplex (all or a subset of) real-time and non-real-time channels onto a single, shared QUIC connection, by extending RoQ, the underlying QUIC implementation will likely use the same congestion controller for all channels in the shared QUIC connection. For this reason, applications multiplexing multiple streams in one connection will need to implement some form of stream prioritization or bandwidth allocation.

8. Guidance on Choosing QUIC Streams, QUIC DATAGRAMs, or a Mixture

As noted in Section 3.2, this specification does not take a position on using QUIC streams, QUIC DATAGRAMs, or some mixture of both, for any particular RoQ use case or application. It does seem useful to include observations that might guide implementers who will need to make choices about that.

One implementation goal might be to minimize processing overhead, for applications that are migrating from RTP over UDP to RoQ. These applications don’t rely on any transport protocol behaviors beyond UDP, which can be described as "nothing beyond IP, except multiplexing". They might be motivated by one or more of the

advantages of encapsulating RTP in QUIC that are described in Section 3.1, but they do not need any of the advantages that would apply when encapsulating RTP in QUIC streams. For these applications, simply placing each RTP packet in a QUIC DATAGRAM frame when it becomes available would be sufficient, with no QUIC streams at all.

Another implementation goal might be to prioritize specific types of video frames over other types. For these applications, placing each type of video frame in a separate QUIC stream would allow the RoQ receiver to focus on the most important video frames more easily. This also allows the implementer to rely on QUIC's "byte stream" abstraction, freeing the application from problems with MTU size restrictions that are present with QUIC DATAGRAMs. The application might use QUIC streams for all of the RTP packets carried over this specific QUIC connection, with no QUIC DATAGRAMs at all.

Some applications might have implementation goals that don't fit easily into "QUIC streams only" or "QUIC DATAGRAMs only" categories. For example, another implementation goal might be to use QUIC streams to carry RTP video frames, but to use QUIC DATAGRAMs to carry RTP audio frames, which are typically much smaller. Because humans tend to tolerate inconsistent behavior in video better than inconsistent behavior in audio, the application might add Forward Error Correction [RFC6363] to audio samples and encapsulate the result in QUIC DATAGRAMs while encapsulating RTP video packets in QUIC streams.

As noted in Section 5.1, all RoQ streams and datagrams begin with a flow identifier. This allows a RoQ sender to begin by encapsulating related RTP packets in a stream and then switch to carrying them in QUIC DATAGRAMs, or vice versa. RoQ receivers need to be prepared to accept any valid RTP packet with a given flow identifier, whether it started by being encapsulated in QUIC streams or in QUIC DATAGRAMs, and RoQ receivers need to be prepared to accept RTP flows that switch from QUIC stream encapsulation to QUIC DATAGRAMs, or vice versa.

Because QUIC provides a capability to migrate connections for various reasons, including recovering from a path failure (Section 9 of [RFC9000]), a RoQ sender has the opportunity to revisit decisions about which RTP packets are encapsulated in QUIC streams, and which RTP packets are encapsulated in QUIC DATAGRAMs, when a QUIC connection migrates. Again, RoQ receivers need to be prepated for this eventuality.

9. Replacing RTCP and RTP Header Extensions with QUIC Feedback

Because RTP has so often used UDP as its underlying transport protocol, and receiving little or no feedback from UDP, RTP implementations rely on feedback from the RTP Control Protocol (RTCP) so that RTP senders and receivers can exchange control information to monitor connection statistics and to identify and synchronize streams.

Because QUIC provides transport-level feedback, it can replace at least some RTP transport-level feedback with current QUIC feedback [RFC9000]. In addition, RTP-level feedback that is not available in QUIC by default can potentially be replaced with generally useful QUIC extensions in the future as described in Appendix B.6.

When statistics contained in RTCP packets are also available from QUIC or can be derived from statistics available from QUIC, it is desirable to provide these statistics at only one protocol layer. This avoids consumption of bandwidth to deliver equivalent control information at more than one level of the protocol stack. QUIC and RTCP both have rules describing when certain signals have to be sent. This document does not change any of the rules described by either protocol, but specifies a baseline for replacing some of the RTCP packet types by mapping the contents to QUIC connection statistics, and reducing the transmission frequency and bandwidth requirements for some RTCP packet types that must be transmitted periodically. Future documents can extend this mapping for other RTCP format types, and can make use of new QUIC extensions that become available over time. The mechanisms described in this section can enhance the statistics provided by RTCP and reduce the bandwidth overhead required by certain RTCP packets. Applications using RoQ need to adhere to the rules for RTCP feedback given by [RFC3550] and the RTP profiles in use.

Most statements about "QUIC" in Section 9 are applicable to both RTP encapsulated in QUIC STREAM frames and RTP encapsulated in DATAGRAMs. The differences are described in Section 9.1 and Section 9.2.

While RoQ places no restrictions on applications sending RTCP, this document assumes that the reason an implementer chooses to support RoQ is to obtain benefits beyond what’s available when RTP uses UDP as its underlying transport layer. Exposing relevant information from the QUIC layer to the application instead of exchanging additional RTCP packets, where applicable, will reduce the processing and bandwidth requirements for RoQ senders and receivers.

Section 9.4 discusses what information can be exposed from the QUIC connection layer to reduce the RTCP overhead.

9.1. RoQ Datagrams

QUIC DATAGRAMs are ack-eliciting packets, which means that an acknowledgment is triggered when a DATAGRAM frame is received. Thus, a sender can assume that an RTP packet arrived at the receiver or was lost in transit, using the QUIC acknowledgments of QUIC Datagram frames. In the following, an RTP packet is regarded as acknowledged when the QUIC Datagram frame that carried the RTP packet was acknowledged.

9.2. RoQ Streams

For RTP packets that are sent over QUIC streams, an RTP packet is considered acknowledged after all frames that carried fragments of the RTP packet were acknowledged.

When QUIC Streams are used, the application should be aware that the direct mapping proposed below may not reflect the real characteristics of the network. RTP packet loss can seem lower than actual packet loss due to QUIC's automatic retransmissions. Similarly, timing information might be incorrect due to retransmissions.

9.3. Multihop Topologies

In some topologies, RoQ may be used on only some of the links between multiple session participants. Other links may be using RTP over UDP, or over some other supported RTP encapsulation protocol, and some participants might be using implementations that don't support RoQ at all. These participants will not be able to infer feedback from QUIC, and they may receive less RTCP feedback than expected. On the other hand, participants using RoQ might not be aware that other participants are not using RoQ and send as little RTCP as possible since they assume their RoQ peer will be able to infer statistics from QUIC. There are two ways to solve this problem: if the middlebox translating between RoQ and RTP over other RTP transport protocols such as UDP or TCP provides Back-to-Back RTP sessions as described in Section 3.2.2 of [RFC7667], this middlebox can add RTCP packets for the participants not using RoQ by using the statistics the middlebox gets from QUIC and the mappings described in the following sections. If the middlebox does not provide Back-to-Back RTP sessions, participants may use additional signalling to let the RoQ participants know what RTCP is required.

9.4. Feedback Mappings

This section explains how some of the RTCP packet types that are used to signal reception statistics can be replaced by equivalent statistics that are already collected by QUIC. The following list explains how this mapping can be achieved for the individual fields of different RTCP packet types.

The list of RTCP packets in this section is not exhaustive, and similar considerations would apply when exchanging any other type of RTCP control packets using RoQ.

A more thorough analysis of RTCP Control Packet Types (in Appendix B.1), Generic RTP Feedback (RTPFB) (in Appendix B.3), Payload-specific RTP Feedback (PSFB) (in Appendix B.4), Extended Reports (in Appendix B.2), and RTP Header Extensions (in Appendix B.5), including the information that cannot be mapped from QUIC, can be found in Appendix B.

9.4.1. Negative Acknowledgments ("NACK")

Generic _Negative Acknowledgments_ (PT=205, FMT=1, Name=Generic NACK, [RFC4585]) contain information about RTP packets which the receiver considered lost. Section 6.2.1. of [RFC4585] recommends using this feature only if the underlying protocol cannot provide similar feedback. QUIC does not provide negative acknowledgments but can detect lost packets based on the Gap numbers contained in QUIC ACK frames (Section 6 of [RFC9002]).

9.4.2. ECN Feedback ("ECN")

ECN Feedback (PT=205, FMT=8, Name=RTCP-ECN-FB, [RFC6679]) packets report the count of observed ECN-CE marks. [RFC6679] defines two RTCP reports, one packet type (with PT=205 and FMT=8), and a new report block for the extended reports. QUIC supports ECN reporting through acknowledgments. If the QUIC connection supports ECN, using QUIC acknowledgments to report ECN counts, rather than RTCP ECN feedback reports, reduces bandwidth and processing demands on the RTCP implementation.

9.4.3. Goodbye Packets ("BYE")

RTP session participants can use _Goodbye_ RTCP packets (PT=203, Name=BYE, [RFC3550]), to indicate that a source is no longer active. If the participant is also going to close the QUIC connection, the _BYE_ packet can be replaced by a QUIC CONNECTION_CLOSE frame. In this case, the reason for leaving can be transmitted in QUIC's CONNECTION_CLOSE _Reason Phrase_. However, if the participant wishes

to use this QUIC connection for any other multiplexed traffic, the participant has to use the BYE packet because the QUIC CONNECTION_CLOSE would close the entire QUIC connection for all other QUIC STREAM frames and DATAGRAMs.

10. Error Handling

The following error codes are defined for use when abruptly terminating RoQ streams, aborting reading of RoQ streams, or immediately closing RoQ connections.

ROQ_NO_ERROR (0x00): No error. This is used when the connection or stream needs to be closed, but there is no error to signal.

ROQ_GENERAL_ERROR (0x01): An error that does not match a more specific error code occured.

ROQ_INTERNAL_ERROR (0x02): An internal error has occured in the RoQ stack.

ROQ_PACKET_ERROR (0x03): Invalid payload format, e.g., length does not match packet, invalid flow id encoding, non-RTP on RTP-flow ID, etc.

ROQ_STREAM_CREATION_ERROR (0x04): The endpoint detected that its peer created a stream that violates the ROQ protocol.

ROQ_FRAME_CANCELLED (0x05): A receiving endpoint is using STOP_SENDING on the current stream to request new frames be sent on new streams. Similarly, a sender notifies a receiver that retransmissions of a frame were stopped using RESET_STREAM and new frames will be sent on new streams.

ROQ_UNKNOWN_FLOW_ID (0x06): An endpoint was unable to handle a flow identifier, e.g., because it was not signalled or because the endpoint does not support multiplexing using arbitrary flow identifiers.

ROQ_EXPECTATION_UNMET (0x07): Expectiations of the QUIC transport set by RoQ out-of-band signalling were not met by the QUIC connection.

11. RoQ-QUIC and RoQ-RTP API Considerations

The mapping described in the previous sections relies on the QUIC implementation passing some information to the RoQ implementation. Although RoQ will function without this information, some optimizations regarding rate adaptation and RTCP mapping require certain functionalities to be exposed to the application.

Each item in the following list can be considered individually. Any exposed information or function can be used by RoQ regardless of whether the other items are available. Thus, RoQ does not depend on the availability of all of the listed features but can apply different optimizations depending on the functionality exposed by the QUIC implementation.

* _Maximum Datagram Size_: The maximum DATAGRAM size that the QUIC connection can transmit on the network path to the QUIC receiver. If a RoQ sender using DATAGRAMs does not know the maximum DATAGRAM size for the path to the RoQ receiver, there are only two choices - either use heuristics to limit the size of RoQ messages, or be prepared to lose RoQ messages that were too large to be carried through the network path and delivered to the RoQ receiver.

* _Datagram Acknowledgment and Loss_: Section 5.2 of [RFC9221] allows QUIC implementations to notify the application that a DATAGRAM was acknowledged or that it believes a DATAGRAM was lost. Given the DATAGRAM acknowledgments and losses, the application can deduce which RTP packets arrived at the receiver and which were lost (see also Section 9.1).

* _Stream States_: The stream states include which parts of the data sent on a stream were successfully delivered and which are still outstanding to be sent or retransmitted. If an application keeps track of the RTP packets sent on a stream, their respective sizes, and in which order they were transmitted, it can infer which RTP packets were acknowledged according to the definition in Section 9.2.

* _Arrival timestamps_: If the QUIC connection uses a timestamp extension like [I-D.draft-smith-quic-receive-ts] or [I-D.draft-huitema-quic-ts], the arrival timestamps or one-way delays can support the application as described in Section 9 and Section 7.

* _Bandwidth Estimation_: If a bandwidth estimation is available in the QUIC implementation, exposing it avoids the implementation of an additional bandwidth estimation algorithm in the application.

* _ECN_: If ECN marks are available, they can support the bandwidth estimation of the application if necessary.

One goal for the RoQ protocol is to shield RTP applications from the details of QUIC encapsulation, so the RTP application doesn't need much information about QUIC from RoQ. One exception is that it may be desirable that the RoQ implementation provides an indication of connection migration to the RTP application.

12. Discussion

12.1. Impact of Connection Migration

RTP sessions are characterized by a continuous flow of packets into either or both directions. A connection migration may lead to pausing media transmission until reachability of the peer under the new address is validated. This may lead to short breaks in media delivery in the order of RTT and, if RTCP is used for RTT measurements, may cause spikes in observed delays. Application layer congestion control mechanisms (and also packet repair schemes such as retransmissions) need to be prepared to cope with such spikes. As noted in Section 11, it may be desirable that the RoQ implementation provides an indication of connection migration to the RTP application, to assist in coping.

12.2. 0-RTT considerations

For repeated connections between peers, the initiator of a QUIC connection can use 0-RTT data for both QUIC STREAM frames and DATAGRAMs. As such packets are subject to replay attacks, applications shall carefully specify which data types and operations are allowed. 0-RTT data may be beneficial for use with RoQ to reduce the risk of media clipping, e.g., at the beginning of a conversation.

This specification defines carrying RTP on top of QUIC and thus does not finally define what the actual application data are going to be. RTP typically carries ephemeral media contents that is rendered and possibly recorded but otherwise causes no side effects. Moreover, the amount of data that can be carried as 0-RTT data is rather limited. But it is the responsibility of the respective application to determine if 0-RTT data is permissible.

> *Editor's Note:* Since the QUIC connection will often be created in the context of an existing signaling relationship (e.g., using WebRTC or SIP), specific 0-RTT keying material could be exchanged to prevent replays across sessions. Within the same connection, replayed media packets would be discarded as duplicates by the receiver.

12.3. Coalescing RTP packets in single QUIC packet

Applications have some control over how the QUIC stack maps application data to QUIC frames, but applications cannot control how the QUIC stack maps STREAM and DATAGRAM frames to QUIC packets Section 13 of [RFC9000] and Section 5 of [RFC9308].

* When RTP payloads are carried over QUIC streams, the RTP payload is treated as an ordered byte stream that will be carried in QUIC STREAM frames, with no effort to match application data boundaries.

* When RTP payloads are carried over DATAGRAMs, each RTP payload data unit is mapped into a DATAGRAM frame, but

* QUIC implementations can include multiple STREAM frames from different streams and one or more DATAGRAM frames into a single QUIC packet, and may include other QUIC frames as well.

QUIC stacks are allowed to wait for a short period of time if the queued QUIC packet is shorter than the path MTU, in order to optimize for bandwidth utilization instead of latency, while real-time applications usually prefer to optimize for latency rather than bandwidth utilization. This waiting interval is under the QUIC implementation's control, and might be based on knowledge about application sending behavior or heuristics to determine whether and for how long to wait.

When there are a lot of small DATAGRAM frames (e.g., an audio stream) and a lot of large DATAGRAM frames (e.g., a video stream), it may be a good idea to make sure the audio frames can be included in a QUIC packet that also carries video frames (i.e., the video frames don't fill the whole QUIC packet). Otherwise, the QUIC stack may have to send additional small packets only carrying single audio frames, which would waste some bandwidth.

Application designers are advised to take these considerations into account when selecting and configuring a QUIC stack for use with RoQ.

13. Directions for Future work

This specification represents considerable work and discussion within the IETF, and describes RoQ in sufficient detail that an implementer can build a RoQ application, but we recognize that additional work is likely, after we have sufficient experience with RoQ to guide that work. Possible directions would include

* Better guidance on transport for RTCP (for example, when to use QUIC streams vs. QUIC datagrams).

* Better guidance on the use of realtime-friendly congestion control algorithms (for example, Copa [Copa], L4S [RFC9330], etc.).

* Better guidance for congestion control and rate adaptation for multiple RoQ flows (whether streams or datagrams).

* Possible guidance for connection sharing between RoQ and non-RoQ flows, including considerations for congestion control and rate adaptation, scheduling, prioritization, and which ALPNs to use.

For these reasons, publication of this specification as a stable reference for implementers to test with, and report results, seems useful.

In addition, as noted in Section 3.1.7, one of the motivations for using QUIC as a transport for RTP is to exploit new QUIC extensions as they become available. We noted several proposed QUIC extensions in Appendix A, but these proposals are all solving relevant problems, and those problems are worthy of attention, no matter how they are solved for the QUIC protocol.

* Guidance for using RoQ with QUIC connection migration and over multiple paths. We note that the Multipath Extension for QUIC [I-D.draft-ietf-quic-multipath] has been adopted and is relatively mature.

* Guidance for using RoQ with QUIC NAT traversal solutions. This could use Interactive Connectivity Establishment (ICE) [RFC8445] or other NAT traversal solutions.

* Guidance for improved jitter calculations to use with congestion control and rate adaptation.

* Guidance for other aspects of QUIC performance optimization relying on extensions.

Other QUIC extensions, not yet proposed, may also be useful with RoQ.

14. Security Considerations

RoQ is subject to the security considerations of RTP described in Section 9 of [RFC3550] and the security considerations of any RTP profile in use.

The security considerations for the QUIC protocol and DATAGRAM extension described in Section 21 of [RFC9000], Section 9 of [RFC9001], Section 8 of [RFC9002] and Section 6 of [RFC9221] also apply to RoQ.

Note that RoQ provides mandatory security, and other RTP transports do not. In order to prevent the inadvertent disclosure of RTP sessions to unintended third parties, RTP topologies described in Section 3.3 that include middleboxes supporting both RoQ and non-RoQ paths MUST forward RTP packets on non-RoQ paths using a secure AVP profile ([RFC3711], [RFC4585], or another AVP profile providing equivalent RTP-level security), whether or not RoQ senders are using a secure AVP profile for those RTP packets.

15. IANA Considerations

This document registers a new ALPN protocol ID (in Section 15.1) and creates a new registry that manages the assignment of error code points in RoQ (in Section 15.2).

15.1. Registration of a RoQ Identification String

This document creates a new registration for the identification of RoQ in the "TLS Application-Layer Protocol Negotiation (ALPN) Protocol IDs" registry [RFC7301].

The "roq" string identifies RoQ:

Protocol: RTP over QUIC (RoQ)

Identification Sequence: 0x72 0x6F 0x71 ("roq")

Specification: This document

15.2. RoQ Error Codes Registry

This document establishes a registry for RoQ error codes. The "RTP over QUIC (RoQ) Error Codes" registry manages a 62-bit space and is listed under the "Real-Time Transport Protocol (RTP) Parameters" heading.

The new error codes registry created in this document operates under the QUIC registration policy documented in Section 22.1 of [RFC9000]. This registry includes the common set of fields listed in Section 22.1.1 of [RFC9000].

Permanent registrations in this registry are assigned using the Specification Required policy ([RFC8126]), except for values between 0x00 and 0x3f (in hexadecimal; inclusive), which are assigned using Standards Action or IESG Approval as defined in Sections 4.9 and 4.10 of [RFC8126].

Registrations for error codes are required to include a description of the error code. An expert reviewer is advised to examine new registrations for possible duplication or interaction with existing error codes.

In addition to common fields as described in Section Section 22.1 of [RFC9000], this registry includes two additional fields. Permanent registrations in this registry MUST include the following fields:

Name: A name for the error code.

Description: A brief description of the error code semantics, which can be a summary if a specification reference is provided.

The initial allocations in this registry are all assigned permanent status and list a change controller of the IETF and a contact of the AVTCORE working group (avt@ietf.org).

The entries in Table 2 are registered by this document.

Value	Name	Description	Specification
0x00	ROQ_NO_ERROR	No Error	Section 10
0x01	ROQ_GENERAL_ERROR	General error	Section 10
0x02	ROQ_INTERNAL_ERROR	Internal Error	Section 10
0x03	ROQ_PACKET_ERROR	Invalid payload format	Section 10
0x04	ROQ_STREAM_CREATION_ERROR	Invalid stream type	Section 10
0x05	ROQ_FRAME_CANCELLED	Frame cancelled	Section 10
0x06	ROQ_UNKNOWN_FLOW_ID	Unknown Flow ID	Section 10
0x07	ROQ_EXPECTATION_UNMET	Externally signalled requirement unmet	Section 10

Table 2: Initial RoQ Error Codes

16. References

16.1. 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/rfc/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/rfc/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/rfc/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/rfc/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/rfc/rfc4585>.

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

[RFC6363] Watson, M., Begen, A., and V. Roca, "Forward Error Correction (FEC) Framework", RFC 6363, DOI 10.17487/RFC6363, October 2011, <https://www.rfc-editor.org/rfc/rfc6363>.

[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/rfc/rfc6679>.

[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/rfc/rfc7301>.

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

[RFC768] Postel, J., "User Datagram Protocol", STD 6, RFC 768, DOI 10.17487/RFC0768, August 1980, <https://www.rfc-editor.org/rfc/rfc768>.

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

[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/rfc/rfc8174>.

[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/rfc/rfc8298>.

[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/rfc/rfc8698>.

[RFC8836] Jesup, R. and Z. Sarker, Ed., "Congestion Control Requirements for Interactive Real-Time Media", RFC 8836, DOI 10.17487/RFC8836, January 2021, <https://www.rfc-editor.org/rfc/rfc8836>.

[RFC8888] Sarker, Z., Perkins, C., Singh, V., and M. Ramalho, "RTP Control Protocol (RTCP) Feedback for Congestion Control", RFC 8888, DOI 10.17487/RFC8888, January 2021, <https://www.rfc-editor.org/rfc/rfc8888>.

[RFC8999] Thomson, M., "Version-Independent Properties of QUIC", RFC 8999, DOI 10.17487/RFC8999, May 2021, <https://www.rfc-editor.org/rfc/rfc8999>.

[RFC9000] Iyengar, J., Ed. and M. Thomson, Ed., "QUIC: A UDP-Based Multiplexed and Secure Transport", RFC 9000, DOI 10.17487/RFC9000, May 2021, <https://www.rfc-editor.org/rfc/rfc9000>.

[RFC9001] Thomson, M., Ed. and S. Turner, Ed., "Using TLS to Secure QUIC", RFC 9001, DOI 10.17487/RFC9001, May 2021, <https://www.rfc-editor.org/rfc/rfc9001>.

[RFC9002] Iyengar, J., Ed. and I. Swett, Ed., "QUIC Loss Detection and Congestion Control", RFC 9002, DOI 10.17487/RFC9002, May 2021, <https://www.rfc-editor.org/rfc/rfc9002>.

[RFC9221] Pauly, T., Kinnear, E., and D. Schinazi, "An Unreliable Datagram Extension to QUIC", RFC 9221, DOI 10.17487/RFC9221, March 2022, <https://www.rfc-editor.org/rfc/rfc9221>.

16.2. Informative References

[Copa] "Copa: Practical Delay-Based Congestion Control for the Internet", 2018, <https://web.mit.edu/copa/>.

[I-D.draft-dawkins-avtcore-sdp-rtp-quic]
Dawkins, S., "SDP Offer/Answer for RTP using QUIC as Transport", Work in Progress, Internet-Draft, draft-dawkins-avtcore-sdp-rtp-quic-00, 28 January 2022, <https://datatracker.ietf.org/doc/html/draft-dawkins-avtcore-sdp-rtp-quic-00>.

[I-D.draft-huitema-quic-ts]
Huitema, C., "Quic Timestamps For Measuring One-Way Delays", Work in Progress, Internet-Draft, draft-huitema-quic-ts-08, 28 August 2022, <https://datatracker.ietf.org/doc/html/draft-huitema-quic-ts-08>.

[I-D.draft-hurst-quic-rtp-tunnelling]
Hurst, S., "QRT: QUIC RTP Tunnelling", Work in Progress, Internet-Draft, draft-hurst-quic-rtp-tunnelling-01, 28 January 2021, <https://datatracker.ietf.org/doc/html/draft-hurst-quic-rtp-tunnelling-01>.

[I-D.draft-ietf-avtcore-rtcp-green-metadata]
He, Y., Herglotz, C., and E. Francois, "RTP Control Protocol (RTCP) Messages for Temporal-Spatial Resolution", Work in Progress, Internet-Draft, draft-ietf-avtcore-rtcp-green-metadata-02, 12 October 2023, <https://datatracker.ietf.org/doc/html/draft-ietf-avtcore-rtcp-green-metadata-02>.

[I-D.draft-ietf-avtext-lrr-07]
Lennox, J., Hong, D., Uberti, J., Holmer, S., and M. Flodman, "The Layer Refresh Request (LRR) RTCP Feedback Message", Work in Progress, Internet-Draft, draft-ietf-avtext-lrr-07, 2 July 2017, <https://datatracker.ietf.org/doc/html/draft-ietf-avtext-lrr-07>.

[I-D.draft-ietf-masque-h3-datagram]
Schinazi, D. and L. Pardue, "HTTP Datagrams and the Capsule Protocol", Work in Progress, Internet-Draft, draft-ietf-masque-h3-datagram-11, 17 June 2022, <https://datatracker.ietf.org/doc/html/draft-ietf-masque-h3-datagram-11>.

[I-D.draft-ietf-quic-ack-frequency]
Iyengar, J., Swett, I., and M. Kühlewind, "QUIC Acknowledgement Frequency", Work in Progress, Internet-Draft, draft-ietf-quic-ack-frequency-07, 27 October 2023, <https://datatracker.ietf.org/doc/html/draft-ietf-quic-ack-frequency-07>.

[I-D.draft-ietf-quic-multipath]
Liu, Y., Ma, Y., De Coninck, Q., Bonaventure, O., Huitema, C., and M. Kühlewind, "Multipath Extension for QUIC", Work in Progress, Internet-Draft, draft-ietf-quic-multipath-06, 23 October 2023, <https://datatracker.ietf.org/doc/html/draft-ietf-quic-multipath-06>.

[I-D.draft-ietf-quic-reliable-stream-reset]
Seemann, M. and K. Oku, "QUIC Stream Resets with Partial Delivery", Work in Progress, Internet-Draft, draft-ietf-quic-reliable-stream-reset-06, 28 February 2024, <https://datatracker.ietf.org/doc/html/draft-ietf-quic-reliable-stream-reset-06>.

[I-D.draft-ietf-rmcat-gcc]
Holmer, S., Lundin, H., Carlucci, G., De Cicco, L., and S. Mascolo, "A Google Congestion Control Algorithm for Real-Time Communication", Work in Progress, Internet-Draft, draft-ietf-rmcat-gcc-02, 8 July 2016, <https://datatracker.ietf.org/doc/html/draft-ietf-rmcat-gcc-02>.

[I-D.draft-ietf-wish-whip]
Murillo, S. G. and A. Gouaillard, "WebRTC-HTTP ingestion protocol (WHIP)", Work in Progress, Internet-Draft, draft-ietf-wish-whip-13, 7 February 2024, <https://datatracker.ietf.org/doc/html/draft-ietf-wish-whip-13>.

[I-D.draft-smith-quic-receive-ts]
Smith, C. and I. Swett, "QUIC Extension for Reporting Packet Receive Timestamps", Work in Progress, Internet-Draft, draft-smith-quic-receive-ts-00, 25 October 2021, <https://datatracker.ietf.org/doc/html/draft-smith-quic-receive-ts-00>.

[I-D.draft-thomson-quic-enough]
Thomson, M., "Signaling That a QUIC Receiver Has Enough Stream Data", Work in Progress, Internet-Draft, draft-thomson-quic-enough-00, 30 March 2023, <https://datatracker.ietf.org/doc/html/draft-thomson-quic-enough-00>.

[IANA-RTCP-FMT-PSFB-PT]
"FMT Values for PSFB Payload Types", n.d., <https://www.iana.org/assignments/rtp-parameters/rtp-parameters.xhtml#rtp-parameters-9>.

[IANA-RTCP-FMT-RTPFB-PT]
"FMT Values for RTPFB Payload Types", n.d., <https://www.iana.org/assignments/rtp-parameters/rtp-parameters.xhtml#rtp-parameters-8>.

[IANA-RTCP-PT]
"RTCP Control Packet Types (PT)", n.d., <https://www.iana.org/assignments/rtp-parameters/rtp-parameters.xhtml#rtp-parameters-4>.

[IANA-RTCP-XR-BT]
"RTCP XR Block Type", n.d., <https://www.iana.org/assignments/rtcp-xr-block-types/rtcp-xr-block-types.xhtml#rtcp-xr-block-types-1>.

[IANA-RTP-CHE]
"RTP Compact Header Extensions", n.d., <https://www.iana.org/assignments/rtp-parameters/rtp-parameters.xhtml#rtp-parameters-10>.

[IANA-RTP-SDES-CHE]
"RTP SDES Compact Header Extensions", n.d., <https://www.iana.org/assignments/rtp-parameters/rtp-parameters.xhtml#sdes-compact-header-extensions>.

[RFC1122] Braden, R., Ed., "Requirements for Internet Hosts - Communication Layers", STD 3, RFC 1122, DOI 10.17487/RFC1122, October 1989, <https://www.rfc-editor.org/rfc/rfc1122>.

[RFC1191] Mogul, J. and S. Deering, "Path MTU discovery", RFC 1191, DOI 10.17487/RFC1191, November 1990, <https://www.rfc-editor.org/rfc/rfc1191>.

[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/rfc/rfc3168>.

[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/rfc/rfc3261>.

[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/rfc/rfc3711>.

[RFC5093] Hunt, G., "BT's eXtended Network Quality RTP Control Protocol Extended Reports (RTCP XR XNQ)", RFC 5093, DOI 10.17487/RFC5093, December 2007, <https://www.rfc-editor.org/rfc/rfc5093>.

[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/rfc/rfc5104>.

[RFC5450] Singer, D. and H. Desineni, "Transmission Time Offsets in RTP Streams", RFC 5450, DOI 10.17487/RFC5450, March 2009, <https://www.rfc-editor.org/rfc/rfc5450>.

[RFC5484] Singer, D., "Associating Time-Codes with RTP Streams", RFC 5484, DOI 10.17487/RFC5484, March 2009, <https://www.rfc-editor.org/rfc/rfc5484>.

[RFC5725] Begen, A., Hsu, D., and M. Lague, "Post-Repair Loss RLE Report Block Type for RTP Control Protocol (RTCP) Extended Reports (XRs)", RFC 5725, DOI 10.17487/RFC5725, February 2010, <https://www.rfc-editor.org/rfc/rfc5725>.

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

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

[RFC6051] Perkins, C. and T. Schierl, "Rapid Synchronisation of RTP Flows", RFC 6051, DOI 10.17487/RFC6051, November 2010, <https://www.rfc-editor.org/rfc/rfc6051>.

[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/rfc/rfc6284>.

[RFC6285] Ver Steeg, B., Begen, A., Van Caenegem, T., and Z. Vax, "Unicast-Based Rapid Acquisition of Multicast RTP Sessions", RFC 6285, DOI 10.17487/RFC6285, June 2011, <https://www.rfc-editor.org/rfc/rfc6285>.

[RFC6332] Begen, A. and E. Friedrich, "Multicast Acquisition Report Block Type for RTP Control Protocol (RTCP) Extended Reports (XRs)", RFC 6332, DOI 10.17487/RFC6332, July 2011, <https://www.rfc-editor.org/rfc/rfc6332>.

[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/rfc/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/rfc/rfc6465>.

[RFC6582] Henderson, T., Floyd, S., Gurtov, A., and Y. Nishida, "The NewReno Modification to TCP's Fast Recovery Algorithm", RFC 6582, DOI 10.17487/RFC6582, April 2012, <https://www.rfc-editor.org/rfc/rfc6582>.

[RFC6642] Wu, Q., Ed., Xia, F., and R. Even, "RTP Control Protocol (RTCP) Extension for a Third-Party Loss Report", RFC 6642, DOI 10.17487/RFC6642, June 2012, <https://www.rfc-editor.org/rfc/rfc6642>.

[RFC6776] Clark, A. and Q. Wu, "Measurement Identity and Information Reporting Using a Source Description (SDES) Item and an RTCP Extended Report (XR) Block", RFC 6776, DOI 10.17487/RFC6776, October 2012, <https://www.rfc-editor.org/rfc/rfc6776>.

[RFC6798] Clark, A. and Q. Wu, "RTP Control Protocol (RTCP) Extended Report (XR) Block for Packet Delay Variation Metric Reporting", RFC 6798, DOI 10.17487/RFC6798, November 2012, <https://www.rfc-editor.org/rfc/rfc6798>.

[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/rfc/rfc6843>.

[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/rfc/rfc6904>.

[RFC6958] Clark, A., Zhang, S., Zhao, J., and Q. Wu, Ed., "RTP Control Protocol (RTCP) Extended Report (XR) Block for Burst/Gap Loss Metric Reporting", RFC 6958, DOI 10.17487/RFC6958, May 2013, <https://www.rfc-editor.org/rfc/rfc6958>.

[RFC6990] Huang, R., Wu, Q., Asaeda, H., and G. Zorn, "RTP Control Protocol (RTCP) Extended Report (XR) Block for MPEG-2 Transport Stream (TS) Program Specific Information (PSI) Independent Decodability Statistics Metrics Reporting", RFC 6990, DOI 10.17487/RFC6990, August 2013, <https://www.rfc-editor.org/rfc/rfc6990>.

[RFC7002] Clark, A., Zorn, G., and Q. Wu, "RTP Control Protocol (RTCP) Extended Report (XR) Block for Discard Count Metric Reporting", RFC 7002, DOI 10.17487/RFC7002, September 2013, <https://www.rfc-editor.org/rfc/rfc7002>.

[RFC7003] Clark, A., Huang, R., and Q. Wu, Ed., "RTP Control Protocol (RTCP) Extended Report (XR) Block for Burst/Gap Discard Metric Reporting", RFC 7003, DOI 10.17487/RFC7003, September 2013, <https://www.rfc-editor.org/rfc/rfc7003>.

[RFC7004] Zorn, G., Schott, R., Wu, Q., Ed., and R. Huang, "RTP Control Protocol (RTCP) Extended Report (XR) Blocks for Summary Statistics Metrics Reporting", RFC 7004, DOI 10.17487/RFC7004, September 2013, <https://www.rfc-editor.org/rfc/rfc7004>.

[RFC7005] Clark, A., Singh, V., and Q. Wu, "RTP Control Protocol (RTCP) Extended Report (XR) Block for De-Jitter Buffer Metric Reporting", RFC 7005, DOI 10.17487/RFC7005, September 2013, <https://www.rfc-editor.org/rfc/rfc7005>.

[RFC7097] Ott, J., Singh, V., Ed., and I. Curcio, "RTP Control Protocol (RTCP) Extended Report (XR) for RLE of Discarded Packets", RFC 7097, DOI 10.17487/RFC7097, January 2014, <https://www.rfc-editor.org/rfc/rfc7097>.

[RFC7243] Singh, V., Ed., Ott, J., and I. Curcio, "RTP Control Protocol (RTCP) Extended Report (XR) Block for the Bytes Discarded Metric", RFC 7243, DOI 10.17487/RFC7243, May 2014, <https://www.rfc-editor.org/rfc/rfc7243>.

[RFC7244] Asaeda, H., Wu, Q., and R. Huang, "RTP Control Protocol (RTCP) Extended Report (XR) Blocks for Synchronization Delay and Offset Metrics Reporting", RFC 7244, DOI 10.17487/RFC7244, May 2014, <https://www.rfc-editor.org/rfc/rfc7244>.

[RFC7266] Clark, A., Wu, Q., Schott, R., and G. Zorn, "RTP Control Protocol (RTCP) Extended Report (XR) Blocks for Mean Opinion Score (MOS) Metric Reporting", RFC 7266, DOI 10.17487/RFC7266, June 2014, <https://www.rfc-editor.org/rfc/rfc7266>.

[RFC7272] van Brandenburg, R., Stokking, H., van Deventer, O., Boronat, F., Montagud, M., and K. Gross, "Inter-Destination Media Synchronization (IDMS) Using the RTP Control Protocol (RTCP)", RFC 7272, DOI 10.17487/RFC7272, June 2014, <https://www.rfc-editor.org/rfc/rfc7272>.

[RFC7294] Clark, A., Zorn, G., Bi, C., and Q. Wu, "RTP Control Protocol (RTCP) Extended Report (XR) Blocks for Concealment Metrics Reporting on Audio Applications", RFC 7294, DOI 10.17487/RFC7294, July 2014, <https://www.rfc-editor.org/rfc/rfc7294>.

[RFC7380] Tong, J., Bi, C., Ed., Even, R., Wu, Q., Ed., and R. Huang, "RTP Control Protocol (RTCP) Extended Report (XR) Block for MPEG2 Transport Stream (TS) Program Specific

Information (PSI) Decodability Statistics Metrics Reporting", RFC 7380, DOI 10.17487/RFC7380, November 2014, <https://www.rfc-editor.org/rfc/rfc7380>.

[RFC7509] Huang, R. and V. Singh, "RTP Control Protocol (RTCP) Extended Report (XR) for Post-Repair Loss Count Metrics", RFC 7509, DOI 10.17487/RFC7509, May 2015, <https://www.rfc-editor.org/rfc/rfc7509>.

[RFC7728] Burman, B., Akram, A., Even, R., and M. Westerlund, "RTP Stream Pause and Resume", RFC 7728, DOI 10.17487/RFC7728, February 2016, <https://www.rfc-editor.org/rfc/rfc7728>.

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

[RFC7867] Huang, R., "RTP Control Protocol (RTCP) Extended Report (XR) Block for Loss Concealment Metrics for Video Applications", RFC 7867, DOI 10.17487/RFC7867, July 2016, <https://www.rfc-editor.org/rfc/rfc7867>.

[RFC7941] Westerlund, M., Burman, B., Even, R., and M. Zanaty, "RTP Header Extension for the RTP Control Protocol (RTCP) Source Description Items", RFC 7941, DOI 10.17487/RFC7941, August 2016, <https://www.rfc-editor.org/rfc/rfc7941>.

[RFC8015] Singh, V., Perkins, C., Clark, A., and R. Huang, "RTP Control Protocol (RTCP) Extended Report (XR) Block for Independent Reporting of Burst/Gap Discard Metrics", RFC 8015, DOI 10.17487/RFC8015, November 2016, <https://www.rfc-editor.org/rfc/rfc8015>.

[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/rfc/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/rfc/rfc8085>.

[RFC8201] McCann, J., Deering, S., Mogul, J., and R. Hinden, Ed., "Path MTU Discovery for IP version 6", STD 87, RFC 8201, DOI 10.17487/RFC8201, July 2017, <https://www.rfc-editor.org/rfc/rfc8201>.

[RFC8286] Xia, J., Even, R., Huang, R., and L. Deng, "RTP/RTCP Extension for RTP Splicing Notification", RFC 8286, DOI 10.17487/RFC8286, October 2017, <https://www.rfc-editor.org/rfc/rfc8286>.

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

[RFC8825] Alvestrand, H., "Overview: Real-Time Protocols for Browser-Based Applications", RFC 8825, DOI 10.17487/RFC8825, January 2021, <https://www.rfc-editor.org/rfc/rfc8825>.

[RFC8849] Even, R. and J. Lennox, "Mapping RTP Streams to Controlling Multiple Streams for Telepresence (CLUE) Media Captures", RFC 8849, DOI 10.17487/RFC8849, January 2021, <https://www.rfc-editor.org/rfc/rfc8849>.

[RFC8852] Roach, A.B., Nandakumar, S., and P. Thatcher, "RTP Stream Identifier Source Description (SDES)", RFC 8852, DOI 10.17487/RFC8852, January 2021, <https://www.rfc-editor.org/rfc/rfc8852>.

[RFC8860] Westerlund, M., Perkins, C., and J. Lennox, "Sending Multiple Types of Media in a Single RTP Session", RFC 8860, DOI 10.17487/RFC8860, January 2021, <https://www.rfc-editor.org/rfc/rfc8860>.

[RFC8861] Lennox, J., Westerlund, M., Wu, Q., and C. Perkins, "Sending Multiple RTP Streams in a Single RTP Session: Grouping RTP Control Protocol (RTCP) Reception Statistics and Other Feedback", RFC 8861, DOI 10.17487/RFC8861, January 2021, <https://www.rfc-editor.org/rfc/rfc8861>.

[RFC8899] Fairhurst, G., Jones, T., Tüxen, M., Rüngeler, I., and T. Völker, "Packetization Layer Path MTU Discovery for Datagram Transports", RFC 8899, DOI 10.17487/RFC8899, September 2020, <https://www.rfc-editor.org/rfc/rfc8899>.

[RFC9114] Bishop, M., Ed., "HTTP/3", RFC 9114, DOI 10.17487/RFC9114, June 2022, <https://www.rfc-editor.org/rfc/rfc9114>.

[RFC9143] Holmberg, C., Alvestrand, H., and C. Jennings, "Negotiating Media Multiplexing Using the Session Description Protocol (SDP)", RFC 9143, DOI 10.17487/RFC9143, February 2022, <https://www.rfc-editor.org/rfc/rfc9143>.

[RFC9308] Kühlewind, M. and B. Trammell, "Applicability of the QUIC Transport Protocol", RFC 9308, DOI 10.17487/RFC9308, September 2022, <https://www.rfc-editor.org/rfc/rfc9308>.

[RFC9330] Briscoe, B., Ed., De Schepper, K., Bagnulo, M., and G. White, "Low Latency, Low Loss, and Scalable Throughput (L4S) Internet Service: Architecture", RFC 9330, DOI 10.17487/RFC9330, January 2023, <https://www.rfc-editor.org/rfc/rfc9330>.

[RFC9335] Uberti, J., Jennings, C., and S. Murillo, "Completely Encrypting RTP Header Extensions and Contributing Sources", RFC 9335, DOI 10.17487/RFC9335, January 2023, <https://www.rfc-editor.org/rfc/rfc9335>.

[VJMK88] "Congestion Avoidance and Control", November 1988, <https://ee.lbl.gov/papers/congavoid.pdf>.

[_3GPP-TS-26.114] "IP Multimedia Subsystem (IMS); Multimedia telephony; Media handling and interaction", 5 January 2023, <https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=1404>.

Appendix A. List of optional QUIC Extensions

The following is a list of QUIC protocol extensions that might be beneficial for RoQ, but are not required by RoQ.

* _An Unreliable Datagram Extension to QUIC_ [RFC9221]. Without support for unreliable DATAGRAMs, RoQ cannot use the encapsulation specified in Section 5.3, but can still use QUIC streams as specified in Section 5.2.

* A version of QUIC receive timestamps can be helpful for improved jitter calculations and congestion control. If the QUIC connection uses a timestamp extension like, the arrival timestamps or one-way delays could be exposed to the application for improved bandwidth estimation or RTCP mappings as described in Section 9 and Appendix B.

  - _Quic Timestamps For Measuring One-Way Delays_
    [I-D.draft-huitema-quic-ts]

  - _QUIC Extension for Reporting Packet Receive Timestamps_
    [I-D.draft-smith-quic-receive-ts]

* _QUIC Acknowledgement Frequency_
  [I-D.draft-ietf-quic-ack-frequency] can be used by a sender to optimize the acknowledgement behaviour of the receiver, e.g., to optimize congestion control.

* _Signaling That a QUIC Receiver Has Enough Stream Data_
  [I-D.draft-thomson-quic-enough] and _Reliable QUIC Stream Resets_
  [I-D.draft-ietf-quic-reliable-stream-reset] would allow RoQ senders and receivers to use versions of CLOSE_STREAM and STOP_SENDING that contain offsets. The offset could be used to reliably retransmit all frames up to a certain frame that should be cancelled before resuming transmission of further frames on new QUIC streams.

Appendix B. Considered RTCP Packet Types and RTP Header Extensions

This section lists all the RTCP packet types and RTP header extensions that were considered in the analysis described in Section 9.

Each subsection in Appendix B corresponds to an IANA registry, and includes a reference pointing to that registry.

Several but not all of these control packets and their attributes can be mapped from QUIC, as described in Section 9.4. _Mappable from QUIC_ has one of four values: _yes_, _partly_, _QUIC extension needed_, and _no_. _Partly_ is used for packet types for which some fields can be mapped from QUIC, but not all. _QUIC extension needed_ describes packet types which could be mapped with help from one or more QUIC extensions.

Examples of how certain packet types could be mapped with the help of QUIC extensions follow in Appendix B.6.

B.1. RTCP Control Packet Types

The IANA registry for this section is [IANA-RTCP-PT].

Name	Shortcut	PT	Defining Document	Mappable from QUIC	Comments
SMPTE time-code mapping	SMPTETC	194	[RFC5484]	no	
Extended inter-arrival jitter report	IJ	195	[RFC5450]	no	Would require send-timestamps, which are not provided by any QUIC extension today
Sender Reports	SR	200	[RFC3550]	QUIC extension needed / partly	see Appendix B.6.4 and Appendix B.6.1
Receiver Reports	RR	201	[RFC3550]	QUIC extension needed	see Appendix B.6.1
Source description	SDES	202	[RFC3550]	no	
Goodbye	BYE	203	[RFC3550]	partly	see Section 9.4.3
Application-defined	APP	204	[RFC3550]	no	
Generic RTP Feedback	RTPFB	205	[RFC4585]	partly	see Appendix B.3
Payload-specific	PSFB	205	[RFC4585]	partly	see Appendix B.4
extended report	XR	207	[RFC3611]	partly	see Appendix B.2
AVB RTCP packet	AVB				
Receiver	RSI	209	[RFC5760]	no	

Summary Information					
Port Mapping	TOKEN	210	[RFC6284]	no	
IDMS Settings	IDMS	211	[RFC7272]	no	
Reporting Group Reporting Sources	RGRS	212	[RFC8861]	no	
Splicing Notification Message	SNM	213	[RFC8286]	no	

Table 3

B.2. RTCP XR Block Type

The IANA registry for this section is [IANA-RTCP-XR-BT].

Name	Document	Mappable from QUIC	Comments
Loss RLE Report Block	[RFC3611]	yes	If only used for acknowledgment, could be replaced by QUIC acknowledgments, see Section 9.1 and Section 9.2
Duplicate RLE Report Block	[RFC3611]	no	
Packet Receipt Times Report Block	[RFC3611]	QUIC extension needed / partly	QUIC could provide packet receive timestamps when using a timestamp extension that reports timestamp for every received packet, such as [I-D.draft-smith-quic-receive-ts]. However, QUIC does not provide feedback in RTP timestamp format.
Receiver Reference Time Report Block	[RFC3611]	QUIC extension needed	Used together with DLRR Report Blocks to calculate RTTs of non-senders. RTT measurements can natively be provided by QUIC.

DLRR Report Block	[RFC3611]	QUIC extension needed	Used together with Receiver Reference Time Report Blocks to calculate RTTs of non-senders. RTT can natively be provided by QUIC.
Statistics Summary Report Block	[RFC3611]	QUIC extension needed / partly	Packet loss and jitter can be inferred from QUIC acknowledgments, if a timestamp extension is used (see [I-D.draft-smith-quic-receive-ts] or [I-D.draft-huitema-quic-ts]). The remaining fields cannot be mapped to QUIC.
VoIP Metrics Report Block	[RFC3611]	no	as in other reports above, only loss and RTT available
RTCP XR	[RFC5093]	no	
Texas Instruments Extended VoIP Quality Block			
Post-repair Loss RLE Report Block	[RFC5725]	no	
Multicast Acquisition Report Block	[RFC6332]	no	
IDMS Report Block	[RFC7272]	no	
ECN Summary Report	[RFC6679]	partly	see Section 9.4.2
Measurement Information Block	[RFC6776]	no	
Packet Delay Variation Metrics Block	[RFC6798]	no	QUIC timestamps may be used to achieve the same goal

Delay Metrics Block	[RFC6843]	no	QUIC has RTT and can provide timestamps for one-way delay, but no way of informing peers about end-to-end statistics when QUIC is only used on one segment of the path.
Burst/Gap Loss Summary Statistics Block	[RFC7004]	no	
Burst/Gap Discard Summary Statistics Block	[RFC7004]	no	
Frame Impairment Statistics Summary	[RFC7004]	no	
Burst/Gap Loss Metrics Block	[RFC6958]		no
Burst/Gap Discard Metrics Block	[RFC7003]	no	
MPEG2 Transport Stream PSI-Independent Decodability Statistics Metrics Block	[RFC6990]	no	
De-Jitter Buffer Metrics Block	[RFC7005]	no	
Discard Count Metrics Block	[RFC7002]	no	
DRLE (Discard RLE Report)	[RFC7097]	no	
BDR (Bytes Discarded	[RFC7243]	no	

Report)			
RFISD (RTP Flows Initial Synchronization Delay)	[RFC7244]	no	
RFSO (RTP Flows Synchronization Offset Metrics Block)	[RFC7244]	no	
MOS Metrics Block	[RFC7266]	no	
LCB (Loss Concealment Metrics Block)	[RFC7294], Section 4.1	no	
CSB (Concealed Seconds Metrics Block)	[RFC7294], Section 4.1	no	
MPEG2 Transport Stream PSI Decodability Statistics Metrics Block	[RFC7380]	no	
Post-Repair Loss Count Metrics Report Block	[RFC7509]	no	
Video Loss Concealment Metric Report Block	[RFC7867]	no	
Independent Burst/Gap Discard Metrics Block	[RFC8015]	no	

Table 4: Extended Report Blocks

B.3. FMT Values for RTP Feedback (RTPFB) Payload Types

The IANA registry for this section is [IANA-RTCP-FMT-RTPFB-PT].

Name	Long Name	Document	Mappable from QUIC	Comments
Generic NACK	Generic negative acknowledgement	[RFC4585]	partly	see Section 9.4.1
TMMBR	Temporary Maximum Media Stream Bit Rate Request	[RFC5104]	no	
TMMBN	Temporary Maximum Media Stream Bit Rate Notification	[RFC5104]	no	
RTCP-SR-REQ	RTCP Rapid Resynchronisation Request	[RFC6051]	no	
RAMS	Rapid Acquisition of Multicast Sessions	[RFC6285]	no	
TLLEI	Transport-Layer Third-Party Loss Early Indication	[RFC6642]	no	There is no way of telling QUIC peer "don't ask for retransmission", but QUIC would not ask that anyway, only RTCP NACK, if used.
RTCP-ECN-FB	RTCP ECN Feedback	[RFC6679]	partly	see Section 9.4.2
PAUSE-RESUME	Media Pause/ Resume	[RFC7728]	no	
DBI	Delay Budget Information (DBI)	[_3GPP-TS-26.114]		
CCFB	RTP Congestion	[RFC8888]	QUIC	see

	Control Feedback		extension needed	Appendix B.6.2

Table 5

B.4. FMT Values for Payload-Specific Feedback (PSFB) Payload Types

The IANA registry for this section is [IANA-RTCP-FMT-PSFB-PT].

Because QUIC is a generic transport protocol, QUIC feedback cannot replace the following Payload-specific RTP Feedback (PSFB) feedback.

Name	Long Name	Document
PLI	Picture Loss Indication	[RFC4585]
SLI	Slice Loss Indication	[RFC4585]
RPSI	Reference Picture Selection Indication	[RFC4585]
FIR	Full Intra Request Command	[RFC5104]
TSTR	Temporal-Spatial Trade-off Request	[RFC5104]
TSTN	Temporal-Spatial Trade-off Notification	[RFC5104]
VBCM	Video Back Channel Message	[RFC5104]
PSLEI	Payload-Specific Third-Party	[RFC6642]

	Loss Early Indication	
ROI	Video region-of-interest (ROI)	[_3GPP-TS-26.114]
LRR	Layer Refresh Request Command	[I-D.draft-ietf-avtext-lrr-07]
VP	Viewport (VP)	[_3GPP-TS-26.114]
AFB	Application Layer Feedback	[RFC4585]
TSRR	Temporal-Spatial Resolution Request	[I-D.draft-ietf-avtcore-rtcp-green-metadata]
TSRN	Temporal-Spatial Resolution Notification	[I-D.draft-ietf-avtcore-rtcp-green-metadata]

Table 6

B.5. RTP Header extensions

Like the payload-specific feedback packets, QUIC cannot directly replace the control information in the following header extensions. RoQ does not place restrictions on sending any RTP header extensions. However, some extensions, such as Transmission Time offsets [RFC5450] are used to improve network jitter calculation, which can be done in QUIC if a timestamp extension is used.

B.5.1. RTP Compact Header Extensions

The IANA registry for this section is [IANA-RTP-CHE].

Extension URI	Description	Reference	Mappable from QUIC
urn:ietf:params:rtp-hdrext:toffset	Transmission Time offsets	[RFC5450]	no
urn:ietf:params:rtp-hdrext:ssrc-audio-level	Audio Level	[RFC6464]	no
urn:ietf:params:rtp-hdrext:splicing-interval	Splicing Interval	[RFC8286]	no
urn:ietf:params:rtp-hdrext:smpte-tc	SMPTE time-code mapping	[RFC5484]	no
urn:ietf:params:rtp-hdrext:sdes	Reserved as base URN for RTCP SDES items that are also defined as RTP compact header extensions.	[RFC7941]	no
urn:ietf:params:rtp-hdrext:ntp-64	Synchronisation metadata: 64-bit timestamp format	[RFC6051]	no
urn:ietf:params:rtp-hdrext:ntp-56	Synchronisation metadata: 56-bit timestamp format	[RFC6051]	no
urn:ietf:params:rtp-hdrext:encrypt	Encrypted extension header element	[RFC6904]	no
urn:ietf:params:rtp-hdrext:csrc-audio-level	Mixer-to-client audio level indicators	[RFC6465]	no
urn:3gpp:video-orientation:6	Higher granularity (6-bit) coordination of video	[_3GPP-TS-26.114]	probably not(?)

	orientation (CVO) feature, see clause 6.2.3		
urn:3gpp:video-orientation	Coordination of video orientation (CVO) feature, see clause 6.2.3	[_3GPP-TS-26.114]	probably not(?)
urn:3gpp:roi-sent	Signalling of the arbitrary region-of-interest (ROI) information for the sent video, see clause 6.2.3.4	[_3GPP-TS-26.114]	probably not(?)
urn:3gpp:predefined-roi-sent	Signalling of the predefined region-of-interest (ROI) information for the sent video, see clause 6.2.3.4	[_3GPP-TS-26.114]	probably not(?)

Table 7

B.5.2. RTP SDES Compact Header Extensions

The IANA registry for this section is [IANA-RTP-SDES-CHE].

Extension URI	Description	Reference	Mappable from QUIC
urn:ietf:params:rtp-hdrext:sdes:cname	Source Description: Canonical End-Point Identifier (SDES CNAME)	[RFC7941]	no
urn:ietf:params:rtp-hdrext:sdes:rtp-stream-id	RTP Stream Identifier	[RFC8852]	no
urn:ietf:params:rtp-hdrext:sdes:repaired-rtp-stream-id	RTP Repaired Stream Identifier	[RFC8852]	no
urn:ietf:params:rtp-hdrext:sdes:CaptId	CLUE CaptId	[RFC8849]	no
urn:ietf:params:rtp-hdrext:sdes:mid	Media identification	[RFC9143]	no

Table 8

B.6. Examples

B.6.1. Mapping QUIC Feedback to RTCP Receiver Reports ("RR")

Considerations for mapping QUIC feedback into _Receiver Reports_ (PT=201, Name=RR, [RFC3550]) are:

* _Fraction lost_: When RTP packets are carried in QUIC datagrams, the fraction of lost packets can be directly inferred from QUIC's acknowledgments. The calculation includes all packets up to the acknowledged RTP packet with the highest RTP sequence number.

* _Cumulative lost_: Similar to the fraction of lost packets, the cumulative loss can be inferred from QUIC's acknowledgments, including all packets up to the latest acknowledged packet.

* _Highest Sequence Number received_: In RTCP, this field is a 32-bit field that contains the highest sequence number a receiver received in an RTP packet and the count of sequence number cycles the receiver has observed. A sender sends RTP packets in QUIC

packets and receives acknowledgments for the QUIC packets. By keeping a mapping from a QUIC packet to the RTP packets encapsulated in that QUIC packet, the sender can infer the highest sequence number and number of cycles seen by the receiver from QUIC acknowledgments.

* _Interarrival jitter_: If QUIC acknowledgments carry timestamps as described in [I-D.draft-smith-quic-receive-ts], senders can infer the interarrival jitter from the arrival timestamps in QUIC acknowledgments.

* _Last SR_: Similar to lost packets, the NTP timestamp of the last received sender report can be inferred from QUIC acknowledgments.

* _Delay since last SR_: This field is not required when the receiver reports are entirely replaced by QUIC feedback.

B.6.2. Congestion Control Feedback ("CCFB")

RTP _Congestion Control Feedback_ (PT=205, FMT=11, Name=CCFB, [RFC8888]) contains acknowledgments, arrival timestamps, and ECN notifications for each received packet. Acknowledgments and ECNs can be inferred from QUIC as described above. Arrival timestamps can be added through extended acknowledgment frames as described in [I-D.draft-smith-quic-receive-ts] or [I-D.draft-huitema-quic-ts].

B.6.3. Extended Report ("XR")

Extended Reports (PT=207, Name=XR, [RFC3611]) offer an extensible framework for a variety of different control messages. Some of the statistics that are defined as extended report blocks can be derived from QUIC, too. Other report blocks need to be evaluated individually to determine whether the contained information can be transmitted using QUIC instead. Table 4 in Appendix B.2 lists considerations for mapping QUIC feedback to some of the _Extended Reports_.

B.6.4. Application Layer Repair and other Control Messages

While Appendix B.6.1 presented some RTCP packets that can be replaced by QUIC features, QUIC cannot replace all of the defined RTCP packet types. This mostly affects RTCP packet types, which carry control information that is to be interpreted by the RTP application layer rather than the underlying transport protocol itself.

* _Sender Reports_ (PT=200, Name=SR, [RFC3550]) are similar to _Receiver Reports_, as described in Appendix B.6.1. They are sent by media senders and additionally contain an NTP and an RTP

timestamp and the number of packets and octets transmitted by the sender. The timestamps can be used by a receiver to synchronize streams. QUIC cannot provide similar control information since it does not know about RTP timestamps. A QUIC receiver cannot calculate the packet or octet counts since it does not know about lost datagrams. Thus, sender reports are necessary in RoQ to synchronize streams at the receiver.

In addition to carrying transmission statistics, RTCP packets can contain application layer control information that cannot directly be mapped to QUIC. Examples of this information may include:

* _Source Description_ (PT=202, Name=SDES) and _Application_ (PT=204, Name=APP) packet types from [RFC3550], or

* many of the payload-specific feedback messages (PT=206) defined in [RFC4585], used to control the codec behavior of the sender.

Since QUIC does not provide any kind of application layer control messaging, QUIC feedback cannot be mapped into these RTCP packet types. If the RTP application needs this information, the RTCP packet types are used in the same way as they would be used over any other transport protocol.

Appendix C. Experimental Results

An experimental implementation of the mapping described in this document can be found on Github (https://github.com/mengelbart/rtp-over-quic). The application implements the RoQ Datagrams mapping and implements SCReAM congestion control at the application layer. It can optionally disable the builtin QUIC congestion control (NewReno). The endpoints only use RTCP for congestion control feedback, which can optionally be disabled and replaced by the QUIC connection statistics as described in Section 9.4.

Experimental results of the implementation can be found on Github (https://github.com/mengelbart/rtp-over-quic-mininet), too.

Acknowledgments

Early versions of this document were similar in spirit to [I-D.draft-hurst-quic-rtp-tunnelling], although many details differ. The authors would like to thank Sam Hurst for providing his thoughts about how QUIC could be used to carry RTP.

The guidance in Section 5.2 about configuring the number of parallel unidirectional QUIC streams is based on Section 6.2 of [RFC9114], with obvious substitutions for RTP/RTCP.

The authors would like to thank Bernard Aboba, David Schinazi, Lucas Pardue, Sam Hurst, Sergio Garcia Murillo, and Vidhi Goel for their valuable comments and suggestions contributing to this document.

Authors' Addresses

Jörg Ott
Technical University Munich
Email: ott@in.tum.de

Mathis Engelbart
Technical University Munich
Email: mathis.engelbart@gmail.com

Spencer Dawkins
Tencent America LLC
Email: spencerdawkins.ietf@gmail.com

Payload Working Group
Internet-Draft
Intended status: Standards Track
Expires: 16 August 2024

D. Hanson
M. Faller
K. Maver
General Dynamics Mission Systems, Inc.

13 February 2024

RTP Payload Format for the Secure Communication Interoperability Protocol (SCIP) Codec
draft-ietf-avtcore-rtp-scip-09

Abstract

This document describes the RTP payload format of the Secure Communication Interoperability Protocol (SCIP). SCIP is an application layer protocol that provides end-to-end capability exchange, packetization/de-packetization of media, reliable transport, and payload encryption.

SCIP handles packetization/de-packetization of the encrypted media and acts as a tunneling protocol, treating SCIP payloads as opaque octets to be encapsulated within RTP payloads prior to transmission or decapsulated on reception. SCIP payloads are sized to fit within the maximum transmission unit (MTU) when transported over RTP thereby avoiding fragmentation.

SCIP transmits encrypted traffic and does not require the use of Secure RTP (SRTP) for payload protection. SCIP also provides for reliable transport at the application layer, so it is not necessary to negotiate RTCP retransmission capabilities.

To establish reliable communications using SCIP over RTP, it is important that middle boxes avoid parsing or modifying SCIP payloads. Because SCIP payloads are confidentiality and integrity protected and are only decipherable by the originating and receiving SCIP devices, modification of the payload by middle boxes would be detected as an integrity failure in SCIP devices, resulting in retransmission and/or communication failure. Middle boxes do not need to parse the SCIP payloads to correctly transport them. Not only is parsing unnecessary to tunnel/detunnel SCIP within RTP, but the parsing and filtering of SCIP payloads by middle boxes would likely lead to ossification of the evolving SCIP protocol.

Status of This Memo

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

This Internet-Draft will expire on 16 August 2024.

Table of Contents

1. Key Points

* SCIP is an application layer protocol that uses RTP as a transport. This document defines the SCIP media subtypes to be listed in the Session Description Protocol (SDP) and only requires a basic RTP transport channel for SCIP payloads. This basic transport channel is comparable to [RFC4040] Clearmode.

* SCIP is designed to be network agnostic. It can operate over any digital link, including non-IP modem-based PSTN and ISDN. The SCIP media subtypes listed in this document were developed for SCIP to operate over RTP.

* SCIP handles packetization/de-packetization of payloads by producing encrypted media packets that are not greater than the MTU size. The SCIP payload is opaque to the network, therefore, SCIP functions as a tunneling protocol for the encrypted media, without the need for middle boxes to parse SCIP payloads. Since SCIP payloads are integrity protected, modification of the SCIP payload is detected as an integrity violation by SCIP endpoints leading to communication failure.

* SCIP includes built-in mechanisms that negotiate protocol message versions and capabilities. To avoid SCIP protocol ossification (as described in [RFC9170]), it is important for middle boxes to not attempt parsing of the SCIP payload. As described in this document, such parsing serves no useful purpose.

2. Introduction

The purpose of this document is to provide enough information to enable SCIP payloads to be transported through the network without modification or filtering. The document provides a reference for network security policymakers; network equipment OEMs, administrators, and architects; procurement personnel; and government agency and commercial industry representatives.

The document details usage of the "audio/scip" and "video/scip" pseudo-codecs [AUDIOSCIP], [VIDEOSCIP] as a secure session establishment protocol and media transport protocol over RTP. It discusses (1) how encrypted audio and video codec payloads are transported over RTP; (2) the IP network layer not implementing SCIP as a protocol since SCIP operates at the application layer in endpoints; (3) the IP network layer enabling SCIP traffic to transparently pass through the network; (4) network devices not recognizing SCIP, and thus removing the scip codecs from the SDP

media payload declaration before forwarding to the next network node; and finally, (5) SCIP endpoint devices not operating on networks due to the scip media subtype removal from the SDP media payload declaration.

The United States, along with its NATO Partners, have implemented SCIP in secure voice, video, and data products operating on commercial, private, and tactical IP networks worldwide using the scip media subtype. The SCIP data traversing the network is encrypted, and network equipment in-line with the session cannot interpret the traffic stream in any way. SCIP-based RTP traffic is opaque and can vary significantly in structure and frequency making traffic profiling not possible. Also, as the SCIP protocol continues to evolve independently of this document, any network device that attempts to filter traffic (e.g., deep packet inspection) may cause unintended consequences in the future when changes to the SCIP traffic may not be recognized by the network device.

The SCIP protocol defined in SCIP-210 [SCIP210] includes built-in support for packetization/de-packetization, retransmission, capability exchange, version negotiation, and payload encryption. Since the traffic is encrypted, neither the RTP transport nor middle boxes can usefully parse or modify SCIP payloads; modifications are detected as integrity violations resulting in retransmission, and eventually, communication failure.

Because knowledge of the SCIP payload format is not needed to transport SCIP signaling or media through middle boxes, SCIP-210 represents an informative reference. While older versions of the SCIP-210 specification are publicly available, the authors strongly encourage network implementers to treat SCIP payloads as opaque octets. When handled correctly, such treatment does not require referring to SCIP-210, and any assumptions about the format of SCIP messages defined in SCIP-210 are likely to lead to protocol ossification and communication failures as the protocol evolves.

> Note: The IETF has not conducted a security review of SCIP and therefore has not verified the claims contained in this document.

2.1. 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 here.

Best current practices for writing an RTP payload format specification were followed [RFC2736] [RFC8088].

When referring to the Secure Communication Interoperability Protocol, the uppercase acronym "SCIP" is used. When referring to the media subtype scip, lowercase "scip" is used.

2.2. Abbreviations

The following abbreviations are used in this document.

AVP:	Audio/Video Profile
AVPF:	Audio/Video Profile Feedback
ICWG:	Interoperability Control Working Group
IICWG:	International Interoperability Control Working Group
NATO:	North Atlantic Treaty Organization
OEM:	Original Equipment Manufacturer
SAVP:	Secure Audio/Video Profile
SAVPF:	Secure Audio/Video Profile Feedback
SCIP:	Secure Communication Interoperability Protocol
SDP:	Session Description Protocol
SRTP:	Secure Real-Time Transport Protocol
STANAG:	Standardization Agreement

3. Background

The Secure Communication Interoperability Protocol (SCIP) allows the negotiation of several voice, data, and video applications using various cryptographic suites. SCIP also provides several important characteristics that have led to its broad acceptance as a secure communications protocol.

SCIP began in the United States as the Future Narrowband Digital Terminal (FNBDT) Protocol in the late 1990s. A combined U.S. Department of Defense and vendor consortium formed a governing organization named the Interoperability Control Working Group (ICWG) to manage the protocol. In time, the group expanded to include NATO, NATO partners and European vendors under the name International Interoperability Control Working Group (IICWG), which was later renamed the SCIP Working Group.

First generation SCIP devices operated on circuit-switched networks. SCIP was then expanded to radio and IP networks. The scip media subtype transports SCIP secure session establishment signaling and secure application traffic. The built-in negotiation and flexibility provided by the SCIP protocols make it a natural choice for many scenarios that require various secure applications and associated encryption suites. SCIP has been adopted by NATO in STANAG 5068. SCIP standards are currently available to participating government/ military communities and select OEMs of equipment that support SCIP.

However, SCIP must operate over global networks (including private and commercial networks). Without access to necessary information to support SCIP, some networks may not support the SCIP media subtypes. Issues may occur simply because information is not as readily available to OEMs, network administrators, and network architects.

This document provides essential information about audio/scip and video/scip media subtypes that enables network equipment manufacturers to include settings for "scip" as a known audio and video media subtype in their equipment. This enables network administrators to define and implement a compatible security policy which includes audio and video media subtypes "audio/scip" and "video/scip", respectively, as permitted codecs on the network.

All current IP-based SCIP endpoints implement "scip" as a media subtype. Registration of scip as a media subtype provides a common reference for network equipment manufacturers to recognize SCIP in an SDP payload declaration.

4. Payload Format

The "scip" media subtype indicates support for and identifies SCIP traffic that is being transported over RTP. Transcoding, lossy compression, or other data modifications MUST NOT be performed by the network on the SCIP RTP payload. The audio/scip and video/scip media subtype data streams within the network, including the VoIP network, MUST be a transparent relay and be treated as "clear-channel data", similar to the Clearmode media subtype defined by [RFC4040].

RFC 4040 is referenced because Clearmode does not define specific RTP payload content, packet size, or packet intervals, but rather enables Clearmode devices to signal that they support a compatible mode of operation and defines a transparent channel on which devices may communicate. This document takes a similar approach. Network devices that implement support for SCIP need to enable SCIP endpoints to signal that they support SCIP and provide a transparent channel on which SCIP endpoints may communicate.

SCIP is an application layer protocol that is defined in SCIP-210. The SCIP traffic consists of encrypted SCIP control messages and codec data. The payload size and interval will vary considerably depending on the state of the SCIP protocol within the SCIP device.

Figure 1 below illustrates the RTP payload format for SCIP.

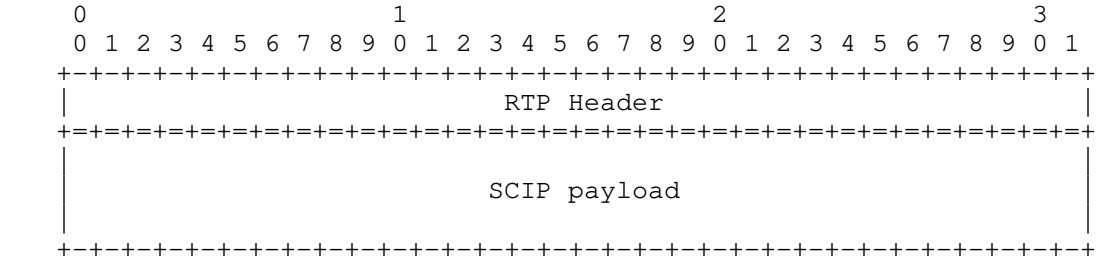


Figure 1: SCIP RTP Payload Format

The SCIP codec produces an encrypted bitstream that is transported over RTP. Unlike other codecs, SCIP does not have its own upper layer syntax (e.g., no Network Adaptation Layer (NAL) units), but rather encrypts the output of the audio/video codecs that it uses (e.g., G.729D, H.264 [RFC6184], etc.). SCIP achieves this by encapsulating the encrypted codec output that has been previously formatted according to the relevant RTP payload specification for that codec. SCIP endpoints MAY employ mechanisms, such as Inter-media RTP Synchronization as described in [RFC8088] Section 3.3.4, to synchronize audio/scip and video/scip streams.

Figure 2 below illustrates notionally how codec packets and SCIP control messages are packetized for transmission over RTP.

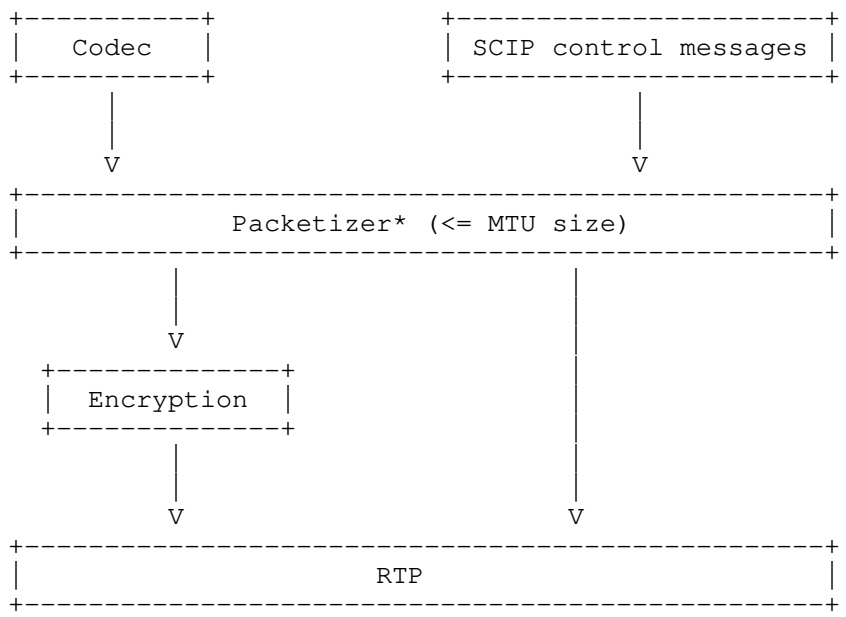


Figure 2: SCIP RTP Architecture

> * Packetizer: The SCIP application layer will ensure that all traffic sent to the RTP layer will not exceed the MTU size. The receiving SCIP RTP layer will handle packet identification, ordering, and reassembly. When required, the SCIP application layer handles error detection and retransmission.

As described above, the SCIP RTP payload format is variable and cannot be described in specificity in this document. Details can be found in SCIP-210. SCIP will continue to evolve and as such the SCIP RTP traffic MUST NOT be filtered by network devices based upon what currently is observed or documented. The focus of this document is for network devices to consider the SCIP RTP payload as opaque and allow it to traverse the network. Network devices MUST NOT modify SCIP RTP packets.

4.1. RTP Header Fields

The SCIP RTP header fields SHALL conform to RFC 3550.

SCIP traffic may be continuous or discontinuous. The Timestamp field MUST increment based on the sampling clock for discontinuous transmission as described in [RFC3550], Section 5.1. The Timestamp field for continuous transmission applications is dependent on the sampling rate of the media as specified in the media subtype's specification (e.g., MELPe). Note that during a SCIP session, both discontinuous and continuous traffic are highly probable.

The Marker bit SHALL be set to zero for discontinuous traffic. The Marker bit for continuous traffic is based on the underlying media subtype specification. The underlying media is opaque within SCIP RTP packets.

4.2. Congestion Control Considerations

The bitrate of SCIP may be adjusted depending on the capability of the underlying codec (such as MELPe [RFC8130], G.729D [RFC3551], etc.). The number of encoded audio frames per packet may also be adjusted to control congestion. Discontinuous transmission may also be used if supported by the underlying codec.

Since UDP does not provide congestion control, applications that use RTP over UDP SHOULD implement their own congestion control above the UDP layer [RFC8085] and MAY also implement a transport circuit breaker [RFC8083]. Work in the RTP Media Congestion Avoidance Techniques (RMCAT) working group [RMCAT] describes the interactions and conceptual interfaces necessary between the application components that relate to congestion control, including the RTP layer, the higher-level media codec control layer, and the lower-level transport interface, as well as components dedicated to congestion control functions.

Use of the packet loss feedback mechanisms in AVPF [RFC4585] and SAVPF [RFC5124] are OPTIONAL because SCIP itself manages retransmissions of some errored or lost packets. Specifically, the Payload-Specific Feedback Messages defined in RFC 4585 section 6.3 are OPTIONAL when transporting video data.

4.3. Use of Augmented RTP Transport Protocols with SCIP

The SCIP application layer protocol uses RTP as a basic transport for the audio/scip and video/scip payloads. Additional RTP transport protocols that do not modify the SCIP payload are considered OPTIONAL in this document and are discretionary for a SCIP device vendor to implement. Some examples include but are not limited to:

* RTP Payload Format for Generic Forward Error Correction [RFC5109]

* Multiplexing RTP Data and Control Packets on a Single Port [RFC5761]

* Symmetric RTP/RTP Control Protocol (RTCP) [RFC4961]

* Negotiating Media Multiplexing Using the Session Description Protocol (BUNDLE) [RFC9143]

5. Payload Format Parameters

The SCIP RTP payload format is identified using the scip media subtype, which is registered in accordance with [RFC4855] and per the media type registration template form [RFC6838]. A clock rate of 8000 Hz SHALL be used for "audio/scip". A clock rate of 90000 Hz SHALL be used for "video/scip".

5.1. Media Subtype "audio/scip"

Media type name: audio

Media subtype name: scip

Required parameters: N/A

Optional parameters: N/A

Encoding considerations: Binary. This media subtype is only defined for transfer via RTP. There SHALL be no encoding/decoding (transcoding) of the audio stream as it traverses the network.

Security considerations: See Section 7.

Interoperability considerations: N/A

Published specifications: [SCIP210]

Applications which use this media: N/A

Fragment Identifier considerations: none

Restrictions on usage: N/A

Additional information:

1. Deprecated alias names for this type: N/A

2. Magic number(s): N/A

3. File extension(s): N/A

4. Macintosh file type code: N/A

5. Object Identifiers: N/A

Person to contact for further information:

1. Name: Michael Faller and Daniel Hanson

2. Email: michael.faller@gd-ms.com and dan.hanson@gd-ms.com

Intended usage: Common

Authors:

Michael Faller - michael.faller@gd-ms.com

Daniel Hanson - dan.hanson@gd-ms.com

Change controller:

SCIP Working Group - ncia.cis3@ncia.nato.int

5.2. Media Subtype "video/scip"

Media type name: video

Media subtype name: scip

Required parameters: N/A

Optional parameters: N/A

Encoding considerations: Binary. This media subtype is only defined for transfer via RTP. There SHALL be no encoding/decoding (transcoding) of the video stream as it traverses the network.

Security considerations: See Section 7.

Interoperability considerations: N/A

Published specifications: [SCIP210]

Applications which use this media: N/A

Fragment Identifier considerations: none

Restrictions on usage: N/A

Additional information:

1. Deprecated alias names for this type: N/A
2. Magic number(s): N/A
3. File extension(s): N/A
4. Macintosh file type code: N/A
5. Object Identifiers: N/A

Person to contact for further information:

1. Name: Michael Faller and Daniel Hanson
2. Email: michael.faller@gd-ms.com and dan.hanson@gd-ms.com

Intended usage: Common

Authors:

Michael Faller - michael.faller@gd-ms.com

Daniel Hanson - dan.hanson@gd-ms.com

Change controller:

SCIP Working Group - ncia.cis3@ncia.nato.int

5.3. Mapping to SDP

The mapping of the above defined payload format media subtype and its parameters SHALL be implemented according to Section 3 of [RFC4855].

Since SCIP includes its own facilities for capabilities exchange, it is only necessary to negotiate the use of SCIP within SDP Offer/ Answer; the specific codecs to be encapsulated within SCIP are then negotiated via the exchange of SCIP control messages.

The information carried in the media type specification has a specific mapping to fields in the Session Description Protocol (SDP) [RFC8866], which is commonly used to describe RTP sessions. When SDP is used to specify sessions employing the SCIP codec, the mapping is as follows:

* The media type ("audio") goes in SDP "m=" as the media name for audio/scip, and the media type ("video") goes in SDP "m=" as the media name for video/scip.

* The media subtype ("scip") goes in SDP "a=rtpmap" as the encoding name. The required parameter "rate" also goes in "a=rtpmap" as the clock rate.

* The optional parameters "ptime" and "maxptime" go in the SDP "a=ptime" and "a=maxptime" attributes, respectively.

An example mapping for audio/scip is:

```
m=audio 50000 RTP/AVP 96
a=rtpmap:96 scip/8000
```

An example mapping for video/scip is:

```
m=video 50002 RTP/AVP 97
a=rtpmap:97 scip/90000
```

An example mapping for both audio/scip and video/scip is:

```
m=audio 50000 RTP/AVP 96
a=rtpmap:96 scip/8000
m=video 50002 RTP/AVP 97
a=rtpmap:97 scip/90000
```

5.4. SDP Offer/Answer Considerations

In accordance with the SDP Offer/Answer model [RFC3264], the SCIP device SHALL list the SCIP payload type number in order of preference in the "m" media line.

For example, an SDP Offer with scip as the preferred audio media subtype:

```
m=audio 50000 RTP/AVP 96 0 8
a=rtpmap:96 scip/8000
a=rtpmap:0 PCMU/8000
a=rtpmap:8 PCMA/8000
```

6. 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]. However, as "Securing the RTP Protocol 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. The rest of this Security Considerations section discusses the security impacting properties of the payload format itself.

This RTP payload format and its media decoder do not exhibit any significant non-uniformity in the receiver-side computational complexity for packet processing, and thus do not inherently pose a denial-of-service threat due to the receipt of pathological data. Nor does the RTP payload format contain any active content.

SCIP only encrypts the contents transported in the RTP payload; it does not protect the RTP header or RTCP packets. Applications requiring additional RTP header and/or RTCP security might consider mechanisms such as SRTP [RFC3711], however these additional mechanisms are considered OPTIONAL in this document.

7. IANA Considerations

The audio/scip and video/scip media subtypes have previously been registered with IANA [AUDIOSCIP] [VIDEOSCIP]. IANA should update [AUDIOSCIP] and [VIDEOSCIP] to reference this document upon publication.

8. SCIP Contact Information

The SCIP protocol is maintained by the SCIP Working Group. The current SCIP-210 specification may be requested from the email address below.

SCIP Working Group, CIS3 Partnership
NATO Communications and Information Agency
Oude Waalsdorperweg 61
2597 AK The Hague, Netherlands
Email: ncia.cis3@ncia.nato.int

An older public version of the SCIP-210 specification can be downloaded from https://www.iad.gov/SecurePhone/index.cfm.

9. References

9.1. 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>.

[RFC2736] Handley, M. and C. Perkins, "Guidelines for Writers of RTP Payload Format Specifications", BCP 36, RFC 2736, DOI 10.17487/RFC2736, December 1999, <https://www.rfc-editor.org/info/rfc2736>.

[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>.

[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>.

[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>.

[RFC8866] Begen, A., Kyzivat, P., Perkins, C., and M. Handley, "SDP: Session Description Protocol", RFC 8866, DOI 10.17487/RFC8866, January 2021, <https://www.rfc-editor.org/info/rfc8866>.

9.2. Informative References

[AUDIOSCIP] Faller, M. and D. Hanson, "audio/scip: Internet Assigned Numbers Authority (IANA)", 28 January 2021, <https://www.iana.org/assignments/media-types/audio/scip>.

[RFC4040] Kreuter, R., "RTP Payload Format for a 64 kbit/s Transparent Call", RFC 4040, DOI 10.17487/RFC4040, April 2005, <https://www.rfc-editor.org/info/rfc4040>.

[RFC4855] Casner, S., "Media Type Registration of RTP Payload Formats", RFC 4855, DOI 10.17487/RFC4855, February 2007, <https://www.rfc-editor.org/info/rfc4855>.

[RFC4961] Wing, D., "Symmetric RTP / RTP Control Protocol (RTCP)", BCP 131, RFC 4961, DOI 10.17487/RFC4961, July 2007, <https://www.rfc-editor.org/info/rfc4961>.

[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>.

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

[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>.

[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>.

[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>.

[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>.

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

[RFC8130] Demjanenko, V. and D. Satterlee, "RTP Payload Format for the Mixed Excitation Linear Prediction Enhanced (MELPe) Codec", RFC 8130, DOI 10.17487/RFC8130, March 2017, <https://www.rfc-editor.org/info/rfc8130>.

[RFC9143] Holmberg, C., Alvestrand, H., and C. Jennings, "Negotiating Media Multiplexing Using the Session Description Protocol (SDP)", RFC 9143, DOI 10.17487/RFC9143, February 2022, <https://www.rfc-editor.org/info/rfc9143>.

[RFC9170] Thomson, M. and T. Pauly, "Long-Term Viability of Protocol Extension Mechanisms", RFC 9170, DOI 10.17487/RFC9170, December 2021, <https://www.rfc-editor.org/info/rfc9170>.

[RMCAT] IETF, "RTP Media Congestion Avoidance Techniques (rmcat) Working Group", <https://datatracker.ietf.org/wg/rmcat/about/>.

[SCIP210] SCIP Working Group, "SCIP Signaling Plan", SCIP-210, r3.11, September 2023, <https://www.iad.gov/SecurePhone/index.cfm>.

[VIDEOSCIP] Faller, M. and D. Hanson, "video/scip: Internet Assigned Numbers Authority (IANA)", 28 January 2021, <https://www.iana.org/assignments/media-types/video/scip>.

Authors' Addresses

Daniel Hanson
General Dynamics Mission Systems, Inc.
150 Rustcraft Road
Dedham, MA 02026
United States of America
Email: dan.hanson@gd-ms.com

Michael Faller
General Dynamics Mission Systems, Inc.
150 Rustcraft Road
Dedham, MA 02026
United States of America
Email: michael.faller@gd-ms.com

Keith Maver
General Dynamics Mission Systems, Inc.
150 Rustcraft Road
Dedham, MA 02026
United States of America
Email: keith.maver@gd-ms.com

Audio/Video Transport Core Maintenance
Internet-Draft
Intended status: Standards Track
Expires: 10 May 2024

P. Thatcher
Microsoft

7 November 2023

RTP Payload Format for SFrame
draft-ietf-avtcore-rtp-sframe-00

Abstract

This document describes the RTP payload format of SFrame.

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 10 May 2024.

Copyright Notice

Table of Contents

1. Introduction

SFrame [I-D.draft-ietf-sframe-enc-01] describes an 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.

This document describes how to packetize a media frame encrypted using SFrame into RTP packets.

2. Terminology and Notation

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

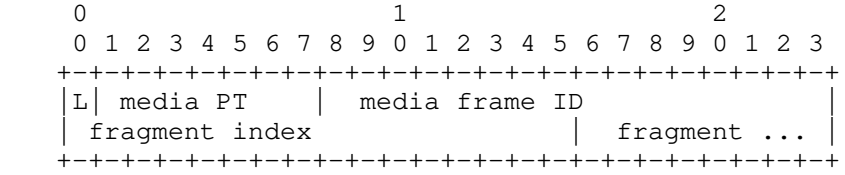
3. RTP Packetization of a media frame encrypted by SFrame

In order to packetize SFrame into RTP, packetization is done in 2 stages. In the first stage, before SFrame encryption, media is packetized into RTP packets in a way specific to the media format. In the second stage, each RTP packet from the first stage is packetized into RTP packets in a way specific to SFrame. SFrame encryption is applied to the payload of each RTP packet between the first and second stages.

For example, if a media frame to be encrypted by SFrame is encoded using VP8, the media frame is first packetized according to [RFC7741] into one RTP packets with VP8-specific payloads. Each of those VP8 RTP payloads are then encrypted using SFrame, resulting in an SFrame-encrypted RTP payload of VP8. SFrame-specific packetization is then applied to the SFrame-encrypted RTP payload of VP8, resulting in RTP packets with SFrame-specific RTP payloads.

SFrame-specific packetization is done by first breaking up the output of SFrame encryption into fragments, and then prepending some fragment metadata necessary for depacketization. Finally, fragments are combined with values from the RTP header of the output of the media-format-specific packetization.

The SFrame-specific RTP payloads (fragments with prepended metadata) have the following format:



The media PT must be the payload type of the output of the media-format-specific packetization. The frame index of the first fragment of each media frame MUST be 0. The frame index of each subsequent fragment MUST be one more than the previous fragment. The L bit MUST be 0 for all fragments except for the last one of the media frame. The media frame ID must be unique enough that a depacketizer may be able to differentiate the fragments of one media frame from another. The SSRC, timestamp, marker bit, CSRCs, and header extensions of the SFrame RTP packets MUST be the same as those of the output of the media-format-specific packetization. The payload type of the SFrame RTP packets must be a payload type that indicates the payload format defined in this document, and it must have a negotiated RTP clock rate that is the same as the media-format-specific RTP packet.

4. RTP depacketization of SFrame

Depacketization is done by doing the packetization process in reverse:

1. The fragments of a given media frame ID are grouped together in order of fragment index and concatenated together, resulting in a media frame encrypted by SFrame.

2. The media frame is decrypted using SFrame, resulting in a media-format-specific RTP payload.

3. The media-format-specific RTP payload is combined with the RTP headers of the RTP packet with fragment index 0, resulting in a media-format-specific RTP packet. The "media PT" from the SFrame RTP payload header is used as the payload type of the media-format-specific RTP packet.

4. The media-format-specific RTP packet is passed into a media-format-specific RTP depacketizer, resulting in a media frame.

5. SFrame payload type negotiation

Because the payload type of an RTP packet that results from SFrame-specific packetization must match the clock rate of the payload type of the RTP packet that results from media-format-specific packetization, it may be necessary to negotiate more than one SFrame payload type. For example, if one were to use SDP to negotiate payload types, the following payload types could be negotiated with different clock rates:

```
m=audio 50000 RTP/SAVPF 96
a=rtpmap:96 sframe/48000
m=video 50002 RTP/SAVPF 97
a=rtpmap:97 sframe/90000
```

6. Security Considerations

This document is subject to the security considerations of SFrame.

7. IANA Considerations

None

8. References

8.1. Normative References

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

[RFC7741] Westin, P., Lundin, H., Glover, M., Uberti, J., and F. Galligan, "RTP Payload Format for VP8 Video", RFC 7741, DOI 10.17487/RFC7741, March 2016, <https://www.rfc-editor.org/rfc/rfc7741>.

[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/rfc/rfc8174>.

8.2. Informative References

[I-D.draft-ietf-sframe-enc-01]
Omara, E., Uberti, J., Murillo, S. G., Barnes, R., and Y. Fablet, "Secure Frame (SFrame)", Work in Progress, Internet-Draft, draft-ietf-sframe-enc-01, 13 March 2023, <https://datatracker.ietf.org/doc/html/draft-ietf-sframe-enc-01>.

Author's Address

Peter Thatcher
Microsoft
Email: pthatcher@microsoft.com

AVTCORE Working Group
Internet-Draft
Intended status: Informational
Expires: 9 August 2024

S. Majali
Nvidia

6 February 2024

RTCP feedback Message Timing Configuration
draft-majali-avtcore-rtcp-fb-timing-cfg-00

Abstract

This specification describes configuring the Real-time Transport Control Protocol (RTCP) message feedback send time.

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 9 August 2024.

Copyright Notice

Table of Contents

1. Introduction

This document proposes controlling specific RTCP message feedback send time. This proposal help sender negotiate RTCP feedback send time, better flexibility in defining application behavior. This document defines a new Session Description Protocol (SDP) parameter to negotiate the timing configuration.

1.1. Requirements Language

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.

2. SDP Definitions

This section defines optional SDP parameters that are used to negotiate RTCP feedback message send time. Time defined is applicable to specific RTCP feedback message only.

An OPTIONAL RTCP feedback specific parameter, "fb-min-time", indicates the minimum period T_fb_min_time in milliseconds between two same RTCP feedback or wait time before sending feedback message.

The syntax is as follows:

a=rtcp-fb:<rtcp-fb-pt> <rtcp-fb-param>;fb-min-time=<fb-min-time-val>

where above parameters are explained in Section 4 of [RFC4585]

rtcp-fb-pt = /fmt ; as defined in SDP

rtcp-fb-param = SP "app" [SP byte-string]

```
                          / SP token [SP byte-string]

                          / ; empty

   fb-min-time-val        = feedback message minimum time value in
                          milliseconds
```

fb-min-time may have an OPTIONAL parameter sync-counter, indicates synchronization counter SYNC-CONTER helps synchronize RTCP feedback with RTP timestamp change.

If T0 is start of time, receiver keeps count of change in RTP timestamp as COUNT. Once COUNT is equal to parameter SYNC-CONTER or time elapsed is greater than or equal to T_fb_min_time, receiver sends the RTCP feedback. Receiver resets the counter and time, to determine when the next feedback is to be sent.

2.1. SDP description for RTCP feedback timing configuration

* Payload specific RTCP feedback PLI (Picture Loss Indication) with minimum interval of 50 milliseconds. Configuration can be used by the receiver to trigger PLI when no decodable unit is available to decode for 50ms.

  a=rtcp-fb:96 nack pli;fb-min-time=50

* RTCP feedback Generic NACK with minimum time of 1 milliseconds. Receiver to wait for 1 milliseconds before NACK RTCP feedback message is sent on packet loss.

  a=rtcp-fb:96 nack;fb-min-time=1

* RTCP feedback transport-cc with minimum time of 50 milliseconds and synchronization counter set to 3. Receiver to send transport-cc feedback on every 3rd change in RTP timestamp change or 50 milliseconds elapsed, whichever happens earliest.

  a=rtcp-fb:96 transport-cc ;fb-min-time=50;sync-counter=3

3. IANA Considerations

An OPTIONAL parameters, "fb-min-time", sync-counter are defined. See Section 3 for details.

4. Security Considerations

RTP packets using the payload format defined in this specification are subject to the general security considerations discussed in RTP Section 9 of [RFC3550]

5. References

5.1. 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>.

[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>.

[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>.

[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>.

Author's Address

Shridhar Majali
Nvidia
2788 San Tomas Expressway
Santa Clara, CA 95051
United States of America
Email: smajali@nvidia.com