Audio/Video Transport Core Maintenance
Internet-Draft
Intended status: Informational
Expires: 6 April 2025

H. Alvestrand
Google

3 October 2024

Absolute Capture Timestamp RTP header extension
draft-alvestrand-avtcore-abs-capture-time-00

Abstract

This document describes an RTP header extension that can be used to carry information about the capture time of a video frame / audio sample, and include information that allows the capture time to be estimated by the receiver when the frame may have passed over multiple hops before reaching the receiver.

About This Document

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

The latest revision of this draft can be found at https://alvestrand.github.io/id-abs-capture-timestamp/draft-alvestrand-avtcore-abs-capture-timestamp.html. Status information for this document may be found at https://datatracker.ietf.org/doc/draft-alvestrand-avtcore-abs-capture-time/.

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

Source for this draft and an issue tracker can be found at https://github.com/alvestrand/id-abs-capture-timestamp.

Status of This Memo

Table of Contents

1. Introduction

When dealing with separate media streams originating from a single source, it is often desirable to present these in such a fashion that media generated at the same time is presented at the same time; the most well known form of this (between an audio stream and a video stream) is called "lip-sync".

In a simple setup with one source system, a single network hop and one destination system, this is usually done by lining up RTP timestamps. However, when multiple hops and multiple systems are involved, this task becomes more difficult; in particular, when one

desires to synchronize media from multiple sources with independent clocks, where the media may have traveled over multiple network hops between the source and destination.

This memo describes one mechanism for providing more information to make such synchronization possible.

2. Conventions and Definitions

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

3. Absolute Capture Time

The Absolute Capture Time extension is used to stamp RTP packets with a NTP timestamp showing when the first audio or video frame in a packet was originally captured. The intent of this extension is to provide a way to accomplish audio-to-video synchronization when RTCP-terminating intermediate systems (e.g. mixers) are involved.

Name: "Absolute Capture Time"; "RTP Header Extension for Absolute Capture Time"

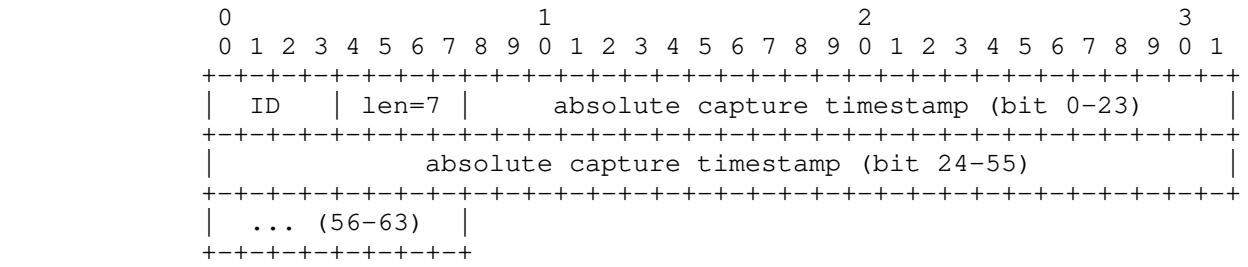
Formal name: http://www.webrtc.org/experiments/rtp-hdrext/abs-capture-time (http://www.webrtc.org/experiments/rtp-hdrext/abs-capture-time)

Status: This extension is defined here to allow for experimentation. Experience with the experiment has shown that it is useful; this draft therefore presents it to the IETF for consideration of whether to standardize it or leave it as a proprietary extension.

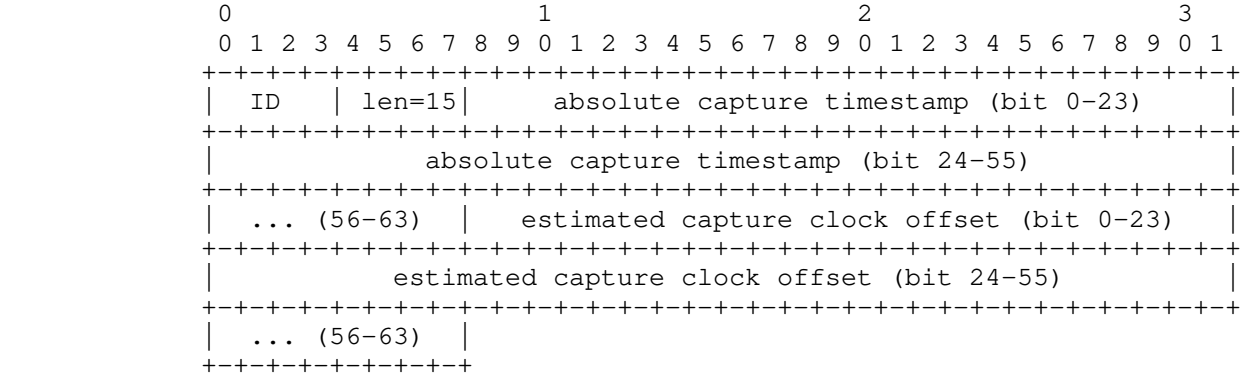
3.1. RTP header extension format

3.1.1. Data layout overview

Data layout of the shortened version of abs-capture-time with a 1-byte header + 8 bytes of data:



Data layout of the extended version of abs-capture-time with a 1-byte header + 16 bytes of data:



3.1.2. Data layout details

3.1.2.1. Absolute capture timestamp

Absolute capture timestamp is the NTP timestamp of when the first frame in a packet was originally captured. This timestamp MUST be based on the same clock as the clock used to generate NTP timestamps for RTCP sender reports on the capture system.

It’s not always possible to do an NTP clock readout at the exact moment of when a media frame is captured. A capture system MAY postpone the readout until a more convenient time. A capture system SHOULD have known delays (e.g. from hardware buffers) subtracted from the readout to make the final timestamp as close to the actual capture time as possible.

This field is encoded as a 64-bit unsigned fixed-point number with the high 32 bits for the timestamp in seconds and low 32 bits for the fractional part. This is also known as the UQ32.32 format and is what the RTP specification defines as the canonical format to represent NTP timestamps.

3.1.2.2. Estimated capture clock offset

Estimated capture clock offset is the sender's estimate of the offset between its own NTP clock and the capture system's NTP clock. The sender is here defined as the system that owns the NTP clock used to generate the NTP timestamps for the RTCP sender reports on this stream. The sender system is typically either the capture system or a mixer.

This field is encoded as a 64-bit two's complement *signed* fixed-point number with the high 32 bits for the seconds and low 32 bits for the fractional part. It's intended to make it easy for a receiver, that knows how to estimate the sender system's NTP clock, to also estimate the capture system's NTP clock:

Capture NTP Clock = Sender NTP Clock + Capture Clock Offset

If the estimated capture clock offset is not present (short format), it means that the sending system does not have enough data to compute a clock offset.

3.1.3. Further details

3.1.3.1. Capture system

The capture system generates the timestamp as close as possible to the true capture time. This may involve subtracting known delays in the capture pipeline from the time at which the system clock is read.

The capture time SHOULD be from the same clock as used to generate the NTP timestamp in RTP Sender Reports (SR) ([RFC3550] section 6.4.1), and indicate this by setting the "estimated capture clock offset" to zero; if this is not possible, the "estimated capture clock offset" MUST indicate the offset between the clock used for the capture timestamp and the clock used for RTP Sender Reports.

3.1.3.2. Intermediate systems

An intermediate system MAY compute the outgoing capture clock offset as follows:

* Start with the "estimated capture clock offset" from the incoming packet

* Add the estimated offset between the sender's NTP clock and the intermediate's NTP clock (see Section 3.1.3.4)

This should give a reasonable estimate of the offset between the capture system's clock and the NTP timestamps sent in SR blocks by the intermediate system.

An intermediate system (e.g. mixer) MAY adjust these timestamps as needed. It MAY also choose to rewrite the timestamps completely, using its own NTP clock as reference clock, if it wants to present itself as a capture system for A/V-sync purposes.

3.1.3.3. End systems

A receiver can use the same algorithm as intermediate systems in order to compute the approximate time in the receiver's NTP clock at which the packet was generated. This should be more comparable between source systems with different clocks than just using the raw timestamp.

A receiver MUST treat the first CSRC in the CSRC list of a received packet as if it belongs to the capture system. If the CSRC list is empty, then the receiver MUST treat the SSRC as if it belongs to the capture system. Mixers SHOULD put the most prominent CSRC as the first CSRC in a packet's CSRC list.

3.1.3.4. Estimating the NTP clock offset

The NTP clock offset can be calculated from an SR packet in the following way:

* Take the NTP timestamp from the SR packet
* Subtract the arrival time of the SR packet, in the receiver's NTP clock
* Add half the estimated RTT between the sender and the receiver

The resulting number should be a reasonable approximation of the offset between the two clocks, with positive numbers indicating that the sender's clock is running ahead, and negative numbers indicate that the sender's clock is running behind.

Note that this method is sensitive to a number of issues:

* Clock drift means that you have to continuously monitor and update the offset
* RTT variance will cause variation in offset; a smoothed value should be used

This document is not normative about how the NTP clock offset is estimated.

3.1.3.5. Timestamp interpolation

A sender SHOULD save bandwidth by not sending abs-capture-time with every RTP packet. It SHOULD still send them at regular intervals (e.g. every second) to help mitigate the impact of clock drift and packet loss. Mixers SHOULD always send abs-capture-time with the first RTP packet after changing capture system.

A receiver SHOULD memorize the capture system (i.e. CSRC/SSRC), capture timestamp, and RTP timestamp of the most recently received abs-capture-time packet on each received stream. It can then use that information, in combination with RTP timestamps of packets without abs-capture-time, to extrapolate missing capture timestamps.

Timestamp interpolation works fine as long as there's reasonably low NTP/RTP clock drift. This is not always true. Senders that detect "jumps" between its NTP and RTP clock mappings SHOULD send abs-capture-time with the first RTP packet after such a thing happening.

4. Security Considerations

This extension carries information that may allow an attacker to identify different media streams on a connection. However, this information is already carried in the RTP SSRC, which is not encrypted, so it is unlikely that much additional information is exposed.

5. IANA Considerations

If the WG decides that this extension should be registered as a standardized extension, IANA is requested to perform the appropriate registration.

If the WG decides that this is a private extension, the URL xxxxx is used to identify the extension.

6. 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>.

[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>.

Acknowledgments

Chen Xing, for writing the original version of this specification.

Author's Address

Harald Alvestrand
Google
Email: hta@google.com

Audio/Video Transport Core Maintenance
Internet-Draft
Intended status: Standards Track
Expires: 29 March 2025

M. Engelbart
J. Ott
Technical University Munich
L. Kondrad
Nokia Technologies
25 September 2024

RTCP Messages for Point Cloud Prioritization
draft-engelbart-avtcore-rtcp-point-cloud-roi-00

Abstract

This document specifies RTCP messages and RTP header extensions for exchanging parameters of real-time streamed point clouds. A sender can notify receivers of the currently applied parameters, such as selected regions, and their parameters, such as the respective resolutions and included point attributes. A receiver can request updates to the same parameters using RTCP feedback messages.

About This Document

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

The latest revision of this draft can be found at https://mengelbart.github.io/draft-engelbart-avtcore-rtcp-point-cloud-roi/draft-engelbart-avtcore-rtcp-point-cloud-roi.html. Status information for this document may be found at https://datatracker.ietf.org/doc/draft-engelbart-avtcore-rtcp-point-cloud-roi/.

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

Source for this draft and an issue tracker can be found at https://github.com/mengelbart/draft-engelbart-avtcore-rtcp-point-cloud-roi.

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 29 March 2025.

Table of Contents

1. Introduction

A point cloud is a set of data points in a three-dimensional coordinate system where each point is defined by its three coordinates. Point clouds can represent three-dimensional environments, such as a vehicle's surroundings. Each point in a point cloud may optionally be associated with additional attributes. Attributes can, for example, be colors or reflectance of objects in the scene. Sequences of point clouds can be generated by sensors such as Lidar ("light detection and ranging"), Radar ("radio detection and ranging"), or a multi-camera setup. Due to the high number of points in a scene, the bandwidth requirements of transmitting point clouds over a network are often higher than those of streaming video. In video streaming, efficient codecs are used to reduce the bandwidth requirement. Similar codecs are being developed for point clouds. However, when streaming point cloud data, consumers of the data usually aren't equally interested in each point. For further processing, focusing on some regions of a point cloud stream is often sufficient, allowing the producer of a point cloud sequence to filter out points of lower interest to further reduce the bandwidth requirements and prioritize bandwidth usage for points of higher importance. When selecting the regions to prioritize and deciding which points can be excluded from transmission, producers and consumers need a mechanism to signal a) which regions of a scene are currently being prioritized and b) request updates to prioritize different regions in the future. This document describes such a mechanism for real-time transmission of point clouds building on the RTP Control protocol RTCP, which is part of the Real-time Transport Protocol RTP. Additionally, this document provides RTP header extensions for senders to inform receivers about the currently applied parameters. The information might be useful for receivers in determining if an RTCP message requesting an update was received and acted upon by the sender.

In Section 4, this document first defines a set of shared encoding schemes to represent point cloud parameters. Section 5 and Section 6 define RTCP messages and RTP header extensions, respectively. The RTCP messages and RTP header extensions reference the encoding scheme in Section 4. Section 7 defines the SDP parameters required to negotiate the usage of the presented RTCP messages and header extensions.

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. Definitions and Abbreviations

TODO: Add definitions or remove section

4. Region and Parameter Encodings

This section introduces encodings for point cloud regions and their parameters, such as the included attributes and the level of detail. It also provides an encoding for the definition of a viewport, including a dynamically adaptable precision.

The encodings described in the following subsections are reused in RTCP feedback messages and RTP header extension as described in Section 5 and Section 6.

4.1. Octree Encoding of Point Cloud Regions

This section defines an encoding that is reused in RTCP message types to index regions in a point cloud. The encoding uses a recursive octree encoding to signal the presence of certain regions of a point cloud. The encoding can be used to set parameters for the present regions, e.g., receivers can signal individual priority of every region or sets of attributes to include in every region.

The root node in every tree is a single byte where every bit indicates whether a child node of an octant is present. Child nodes are appended in pre-order traversal order. A zero-byte encodes a leaf node. The order of octants by the signs of the points of the X-, Y-, and Z-coordinates in each octant is given in Table 1.

Regions in the octree are considered present when a leaf node encoding that region is present. For example, a single zero-byte encodes a single leaf node, and the root region covering the complete space is present. An encoded value of the two bytes 0x4000 would indicate that only the octant with X<0, Y>=0, and Z>=0 is present.

Bit	X	Y	Z
0	+	+	+
1	-	+	+
2	-	-	+
3	+	-	+
4	+	+	-
5	-	+	-
6	-	-	-
7	+	-	-

Table 1: Octant to bit mapping order with bit 0 being the most significant bit and bit 7 being the least significant bit.

The octree encoding can be used in absolute and relative forms. In the absolute form, the bounding box of the octree is implicitly defined by the space covered by the point cloud-generating source. In the relative form, the bounding box can be explicitly defined by setting the min and max values of the X-, Y-, and Z-coordinates. The coordinates are prefixed before the octree encoding as four-byte integers each, as shown in Figure 1.

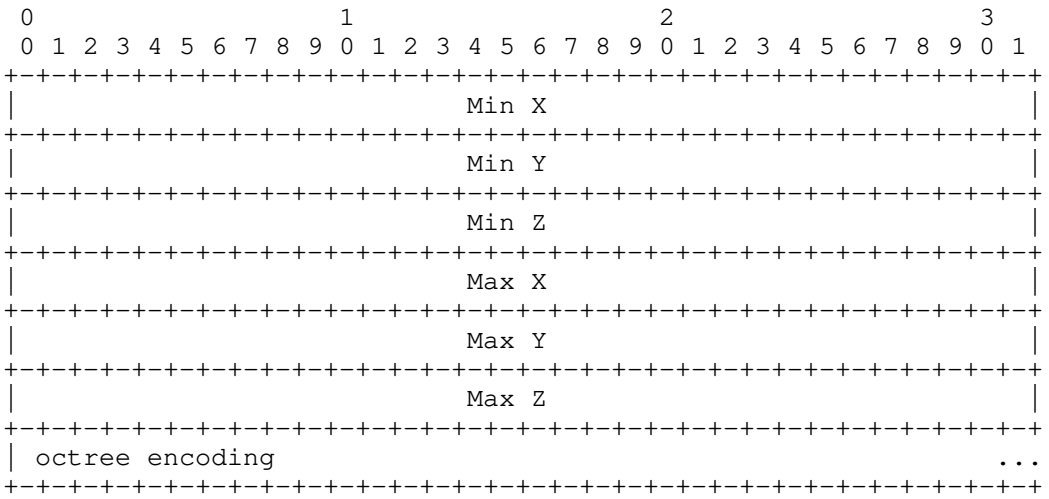


Figure 1: Relative octree encoding using an explicitly defined bounding box

4.2. Region-Dependent Attributes Encoding

Attributes are additional values optionally associated with every point in a point cloud. The available attributes depend on the context of an application and thus need to be configured out-of-band. The attribute encoding described in this section requires mapping attributes to a bitmask value. Section 7 describes one option to negotiate the mapping when using SDP.

To signal the presence of attributes per region, senders and receivers can use the octree encoding presented in Section 4.1. For every region presented in the octree encoding, N bytes will be used to indicate the presence of attributes by setting the bits of the bitmask of the respective attributes to true. The size of N depends on the number of attributes and is implicitly given by the smallest number of bytes that can represent the largest bitmask negotiated during signaling.

Bitmask values can be shared among attributes to allow signaling of attribute sets that always occur in combination.

5. Format of RTCP Feedback Messages

This document describes RTCP message types that can be used to signal interest in the prioritization of regions and their parameters. Receivers can signal an interest in receiving a region with a higher priority. Different regions can be requested in different resolutions or with different sets of attributes included. Additionally, receivers can request a viewport precision update.

A sender can acknowledge a parameter update using the RTP header extensions described in Section 6. Receivers should not retransmit the same request multiple times to avoid unnecessary overhead, but if the receiver can assume that a request was lost, it may retransmit the request. The request should not be retransmitted earlier than at least one RTT after the first request was transmitted.

The following sections define the available message types in detail. All RTCP feedback messages use the common header format shown in Figure 2. The first eight bytes of the header follow the format of RTCP message headers defined in [RFC3550]. The common header is followed by an additional byte consisting of flags describing the payload.

The payload of the RTCP messages following the header contains one or more of the parameter encodings in the following order:

1. Absolute/Relative Octree Encoding

2. (optional) Priority

3. (optional) Attribute Encoding

4. (optional) Level-of-Detail Encoding

   Note: alternative form: <Header><{abs,rel}-octree>[<priority>][<attributes>][<level-of-detail>]

   TODO: In addition to the encodings already described here, one could come up with additional parameter requests such as viewport precision, and sender position and possibly others.

The semantics of the different payload formats are explained in the following subsections.

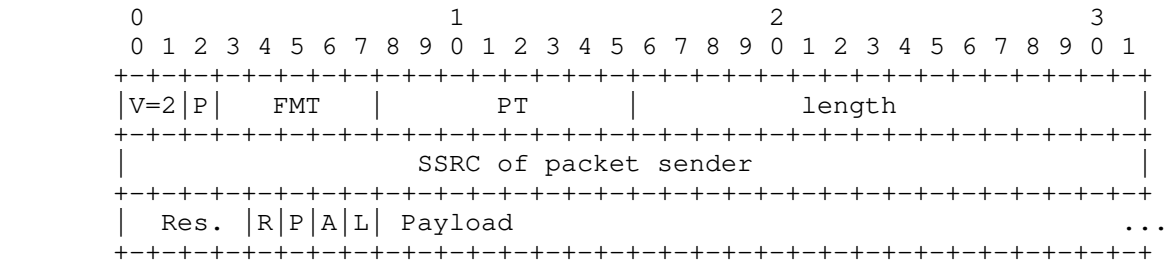


Figure 2: Common RTCP feedback message header

The fields V, P, SSRC and length are used as defined in the RTP specification [RFC3550]. The respective meaning is summarized below:

version (V): 2 bits This field identifies the RTP version. The current version is 2.

padding (P): 1 bit If set, the padding bit indicates that the packet contains additional padding octets at the end that are not part of the control information but are included in the length field.

Feedback message type (FMT): 5 bits The feedback message type is XX (*TODO*: Use correct value).

Payload type (PT): 8 bits The RTCP packet type is PSFB (206).

Relative (R): 1 bit This field distinguishes between absolute and relative region requests (see Section 4.1. If the bit is set, the message contains a relative octree encoding and the minimum and maximum fields described in Figure 1 are present.

Priority: (P): 1 bit If set, the RTCP message contains a priority request and the octree encoding is followed by one byte indicating the requested priority for each region encoded as leaf node in the octree encoding.

Attributes (A): 1 bit If this bit is set, the RTCP message contains an attribute request and the octree encoding is followed by an attribute encoding for every region encoded as leaf node in the octree encdoing.

Level-of-Detail (L): 1 bit If this bit is set, the RTCP message contains a level-of-detail request and the octree encoding (or the attribute encoding, if it is present) is followed by a level-of-detail encoding for every region encoded as leaf node in the octree encoding.

5.1. Region Requests

A receiver can use region of Interest signaling to indicate interest in some areas of a point cloud, and a sender can react by prioritizing the regions of interest when allocating bandwidth.

The region interest request uses the standard header defined in Figure 2.

5.2. Priority Requests

Receivers can append a priority encoding to the octree to signal fine-grained priorities per region. A priority is a value encoded as a single byte where a higher value indicates a higher priority. A priority request contains a priority byte for every region encoded as a leaf node in the preceding octree encoding.

5.3. Attribute Requests

By setting the header Flag A to 1, the receiver can include an attribute encoding as described in Section 4.2 to the feedback message to request attribute sets for every region encoded as a leaf node of the octree. The attribute encoding length depends on the number of negotiated attributes and their associated bitmask values. A single byte can indicate up to eight attributes or attribute sets. For example, if the two attributes, color, and reflectance, are negotiated with bitmask values of 0x01 and 0x02, one byte will be appended to the octree for every leaf node where the bits 0x01 and 0x02 are set to one for every region, which should include these attributes.

5.4. TODO: Region-Dependen Resolution or Level-of-Detail Request

TODO: Using the same mechanism as for attrbiutes, we can signal resolution per region, but we need to define how to express _resolution_.

5.5. Examples

An example encoding of a relative octree encoding followed by attribute and level-of-detail encodings:

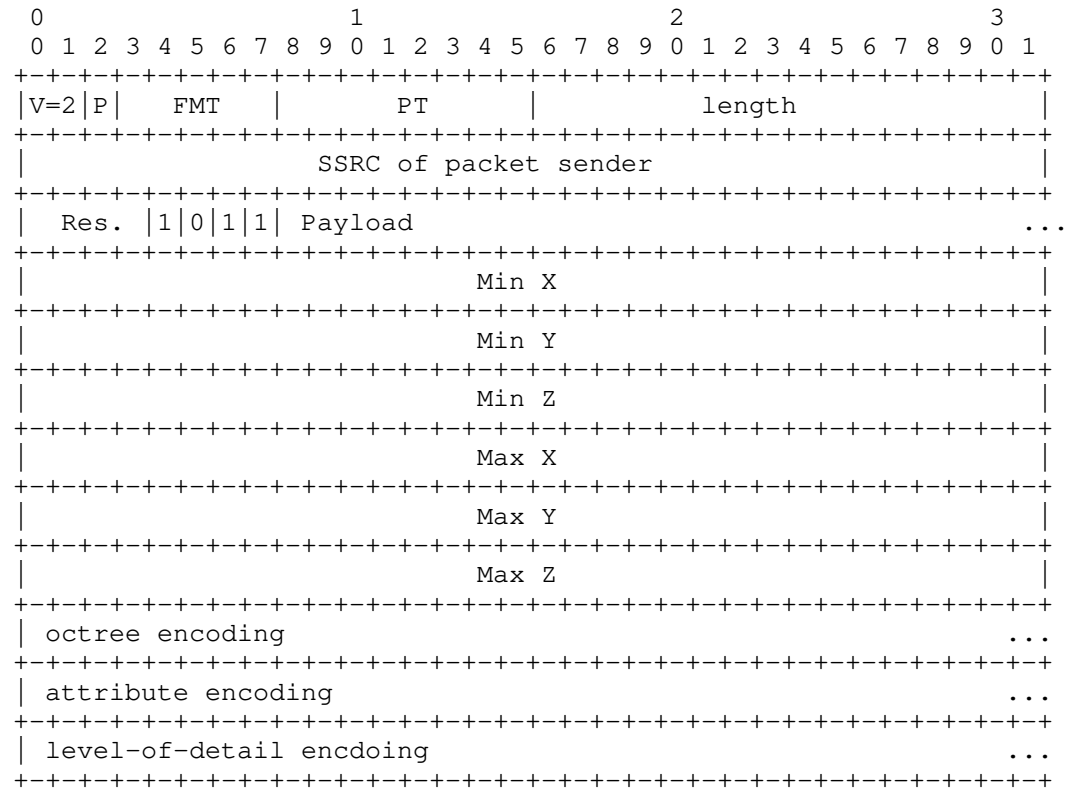


Figure 3: A RTCP feedback message containing a relative octree encoding, attribute encoding and level-of-detail encoding.

6. Format of RTP Header Extensions

RTP senders use RTP header extensions to acknowledge the applied parameters to the receiver. The parameters chosen by the sender may differ from the ones requested by the receiver. The header extension uses the two-byte header form defined in [RFC8285]. The payload of the header extension element uses the same format as the RTCP messages defined in Section 5. Each extension element begins with the same one-byte header (Figure 4) followed by an octree encoding and optional priority, attribute, or level-of-detail encodings.

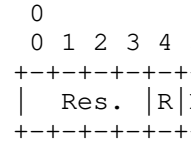


Figure 4: Common RTP header extension payload header

7. SDP Parameters

This section defines SDP parameters for negotiating usage of the RTCP messages and RTP header extension described in this document.

7.1. RTCP Feedback Messages

The rtcp-fb attribute is extended to indicate the capability to send or receive the RTCP feedback defined in this document. This document adds a new parameter "oerr" to the "ccm" feedback value defined in [RFC5104]. The "oerr" (Octree Encoded Region Request) parameter indicates support for the RTCP feedback message defined in Section 5.

rtcp-fb-ccm-param =/ SP "oerr" ; Octree Encoded Region Request

7.2. RTP Header Extensions

The URI for indicating support for the RTP Header extension described in Section 6 is "urn:ietf:params:rtp-hdrext:octree-region".

8. IANA Considerations

8.1. RTCP Feedback Message

The following value are requested to be registered as FMT value in the "FMT Values for PSFB Payload Types" Registry:

Name: OERR

Long Name: Octree Encoded Region Request

Value: *TODO*

Reference: *TODO:* This document

8.2. RTP Header Extension

The following URI is requested to be added to the RTP Compact Header Extensions registry:

Extension URI: urn:ietf:params:rtp-hdrext:octree-region

Description: Octree Encoded Region Information

Contact: mathis.engelbart@gmail.com

Reference: *TODO*: This document

9. Security Considerations

TODO

10. RFC Editor Considerations

Note to RFC Editor: This section may be removed after carrying out all the instructions of this section.

TODO: Consider

11. 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>.

[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>.

[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>.

[RFC8285] Singer, D., Desineni, H., and R. Even, Ed., "A General Mechanism for RTP Header Extensions", RFC 8285, DOI 10.17487/RFC8285, October 2017, <https://www.rfc-editor.org/rfc/rfc8285>.

Acknowledgments

TODO acknowledge.

Authors' Addresses

Mathis Engelbart
Technical University Munich
Email: mathis.engelbart@tum.de

JÃ¶rg Ott
Technical University Munich
Email: ott@in.tum.de

Lukasz Kondrad
Nokia Technologies
Email: lukasz.kondrad@nokia.com

AVTCORE Working Group
INTERNET-DRAFT
Category: Standards Track
Expires: April 6, 2025

B. Aboba
Microsoft Corporation
P. Hancke
Facebook Inc.
6 October 2024

H.265 Profile for WebRTC
draft-ietf-avtcore-hevc-webrtc-04.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 April 6, 2025.

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.

1.2. Abbreviations

AP	Aggregation Packet
BLA	Broken Link Access
CRA	Clean Random Access
FU	Fragmentation Unit
IDR	Instantaneous Decoding Refresh
IRAP	Intra Random Access Point
MANE	Media-Aware Network Element
MRMT	Multiple RTP streams on Multiple media Transports
MRST	Multiple RTP streams on a Single media Transport
NAL	Network Abstraction Layer
NALU	Network Abstraction Layer Unit
PACI	PAyload Content Information
PPS	Picture Parameter Set
SEI	Supplemental Enhancement Information
SFM	Selectively Forwarding Middlebox
SPS	Sequence Parameter Set
SRST	Single RTP stream on a Single media Transport
TID	Temporal Identifier
TSCI	Temporal Scalability Control Information
VCL	Video Coding Layer
VPS	Video Parameter Set

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).

All NAL units included within an RTP payload (including within APs) MUST have the same TID value. This includes both VCL and non-VCL NAL units. This ensures that an SFM will only forward RTP packets to a participant corresponding to the operating point chosen by the SFM. For example, if the SFM choses to only forward base layer frames to a participant, the participant will not also receive NAL units with a TID corresponding to an extension layer.

[RFC7798] Section 4.5 defines how 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 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" are 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
Facebook Inc.

Email: philipp.hancke@gmail.com

Audio/Video Transport Core Maintenance
Internet-Draft
Intended status: Standards Track
Expires: 11 April 2025

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

8 October 2024

RTP Payload Format for sub-codestream latency JPEG 2000 streaming
draft-ietf-avtcore-rtp-j2k-scl-04

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 11 April 2025.

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.

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

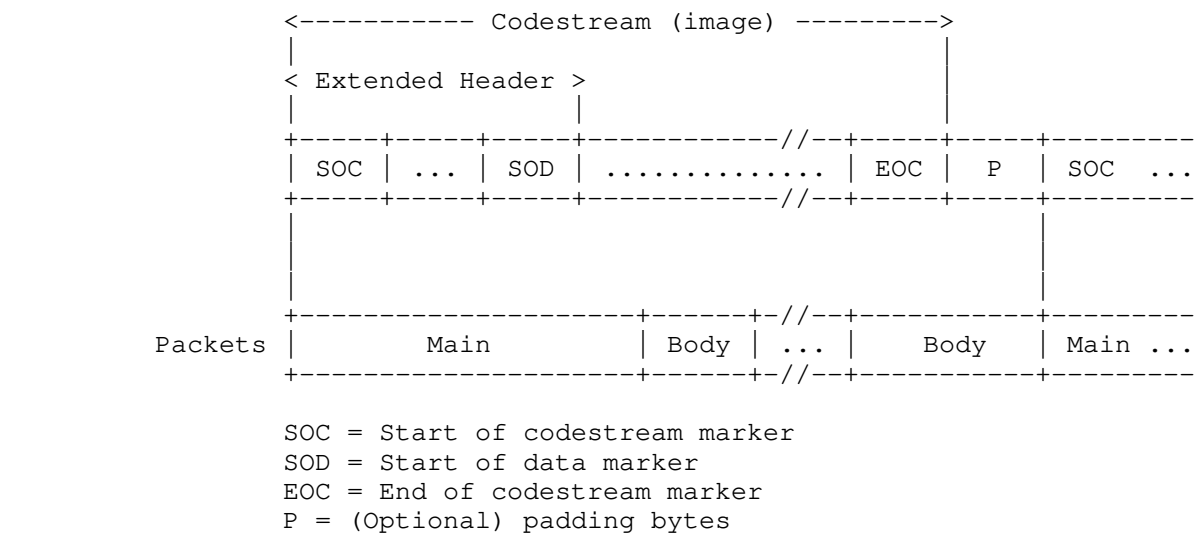


Figure 1: Packetization of a sequence of JPEG 2000 codestreams (not to scale).

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 (see Figure 1).

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.

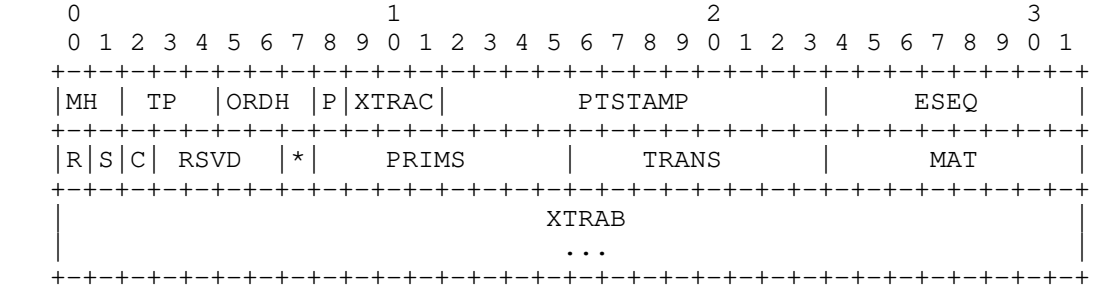
The higher order bits of the RTP sequence number are contained in the ESEQ field, which is specified at Section 5.3.

The RTP sequence number is calculated as follows:

ESEQ * 65536 + sequence number

5.3. Main Packet Payload Header

Figure 2 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 (Codestream Main Header Presence)
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 (Image Type)
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 (Progression Order [Main Packet])

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 (Precision Timestamp Presence)

0

PTSTAMP is not used.

1

PTSTAMP is used.

XTRAC (Extension Payload Length)

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

PTSTAMP (Precision Timestamp)

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 (Extended Sequence Number)

The high order bits of the RTP sequence number.

Section 5.2 specifies the low-order bits of the RTP sequence number and the formula to compute the RTP sequence number

R (Codestream Main Header Reuse)

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 (Parameterized Colorspace Presence)
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 (Code-block Caching Usage)

0
: Code-block caching is not in use.

1
: Code-block caching is in use.

R MUST be equal to 1.

RSVD (Reserved)
: Reserved value. See Section 8.5 and Section 7.7.

RANGE (Video Full Range Usage)
: Value of the VideoFullRangeFlag specified in [rec-itu-t-h273]

PRIMS (Color Primaries)
: One of the ColourPrimaries values specified in [rec-itu-t-h273]

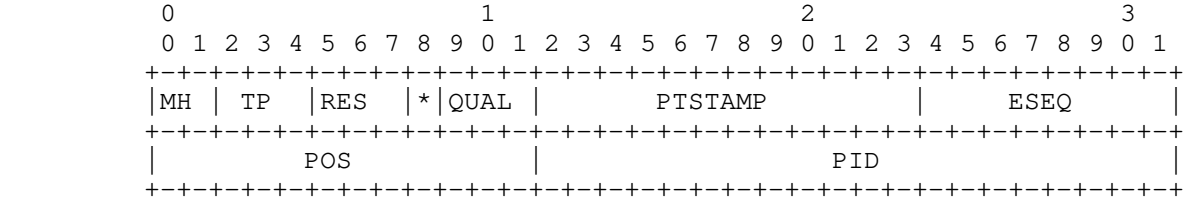
TRANS (Transfer Characteristics)
One of the TransferCharacteristics values specified in [rec-itu-t-h273]

MAT (Color Matrix Coefficients)
One of the MatrixCoefficients values specified in [rec-itu-t-h273]

XTRAB (Extension Payload)
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.



* ORDB

Figure 3: Structure of the Body Packet Payload Header

MH
See Section 5.3.

TP
See Section 5.3.

RES (Resolution Layers)
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 (Progression Order [Body Packet]
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 (Quality Layers)
0
The payload can contribute to all quality layers.

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

PTSTAMP
See Section 5.3.

ESEQ
See Section 5.3.

POS (Resync Point Offset)
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 (Precinct Identifier)
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.

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).

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 a 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	Sub-bands	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	Sub-bands	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 4 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.

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.

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 4 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 4: 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.

* Fix incorrect TRANS values.

D.3. Changes from draft-ietf-avtcore-rtp-j2k-scl-01

* Removed signaling for the transmission of multi-tile images as multiple single-tile image streams (the tile media type parameter).

D.4. Changes from draft-ietf-avtcore-rtp-j2k-scl-02

* Removed request for registration in the deprecated IANA registry for RTP Payload Format MIME types.

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: 9 January 2025

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

8 July 2024

RTP over QUIC (RoQ)
draft-ietf-avtcore-rtp-over-quic-11

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).

This document also discusses how to leverage state that is already available from the QUIC implementation in the endpoints, in order to reduce the need to exchange RTCP packets, and describes different options for implementing congestion control and rate adaptation for RTP 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

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/.

Copyright Notice

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).

This document also discusses how to leverage state that is already available from the QUIC implementation in the endpoints, in order to reduce the need to exchange RTCP packets, and describes different options for implementing congestion control and rate adaptation for RTP 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 historically been UDP [RFC0768], but a large variety of other underlying transport protocols have been defined for various reasons (e.g., securing media exchange, or providing a fallback when UDP is blocked along a network path). This document describes RTP over QUIC, providing one more underlying transport protocol. The reasons for using QUIC as an underlying transport protocol are given in Section 3.1.

This document describes an application usage of QUIC ([RFC9308]). As a baseline, the document does not expect more than a standard QUIC implementation as defined in [RFC8999], [RFC9000], [RFC9001], and

[RFC9002], providing a secure end-to-end transport. 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 Document

This document 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 underlying transport protocols. We expect RoQ to be used in contexts where a signaling protocol is used to announce or negotiate a media encapsulation for RTP and the associated transport parameters (such as IP address, port number). RoQ does not provide a stand-alone media transport capability, because at a minimum, media transport parameters would need to be statically configured.

RoQ can be used in many of the point-to-point and multi-endpoint RTP topologies described in [RFC7667], and can be used with both decentralized and centralized control topologies. When RoQ is used in a decentralized topology, RTP packets are exchanged directly between ultimate RTP endpoints. When RoQ is used in a centralized topology, RTP packets transit one or more middleboxes which might function as mixers or translators between ultimate RTP endpoints. RoQ can also be used in RTP 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.

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 limit the usage of RTP Audio Video Profiles (AVP) ([RFC3551]), or any RTP-based mechanisms, although it might render some of them unnecessary, e.g., Secure Real-Time Transport Protocol (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, although some information or functions provided by using RTCP mechanisms might also be available from the underlying QUIC implementation.

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 can be established in parallel using the same 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 Document

This document does not 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.

This document does not specify RoQ for point-to-multipoint applications, because QUIC itself is not defined for multicast operation. The scope of this document is limited to unicast RTP, even though nothing would prevent its use in multicast setups if future QUIC extensions support multicast.

RoQ does not define new congestion control and rate adaptation algorithms for use with RTP media, and does not specify the use of particular congestion control and rate adaptation algorithms for use with RTP media. However, Section 8 discusses multiple ways that congestion control and rate adaptation could be performed at the QUIC and/or at the RTP layer, and Section 11 describes 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: [RFC3550] actually describes two closely-related protocols - the RTP Data Transfer Protocol Section 5 of [RFC3550], and the RTP Control Protocol Section 6 of [RFC3550]. In this document, the term "RTP" refers to the combination of RTP Data Transfer Protocol and RTP Control Protocol, because the distinction isn't relevant for encapsulation, and the term "RTCP" always refers to the RTP Control Protocol.

Note to the Reader: the meaning of the terms "congestion avoidance", "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 8.2.

"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 document, 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 in this document:

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, which might cause intermediaries on the path to drop packets before they arrive at the receiver.

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. This document uses the uppercase "DATAGRAM" to refer to a QUIC DATAGRAM frame and the term RoQ datagram as a short form of "RTP packet encapsulated in a 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 client or QUIC server that participates in an RoQ session. "A RoQ endpoint" is used in this document where that seems clearer than "an endpoint" without qualification.

Early data: Application data carried in a QUIC 0-RTT packet payload, as defined in [RFC9000]. In this document, the early data would be an RTP packet.

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

Peer: The term "peer" is ambiguous, and without a qualifier could be understood to refer to an RTP endpoint, a RoQ endpoint, or a QUIC endpoint. In this document, a "peer" is "the other RoQ endpoint that a RoQ endpoint is communicating with", and does not have anything to do with "peer-to-peer" operation versus "client-server" operation.

Rate Adaptation: An application-level mechanism that adjusts the sending rate of an application in response 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 might send or receive RTCP packets.

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

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

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 both QUIC streams and QUIC DATAGRAMs to transport real-time data, and thus, if RTP packets are to be sent over QUIC DATAGRAMs, the QUIC implementation MUST support QUIC's DATAGRAM extension.

[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 depend on the application and on the codec in use. For example, adjusting quantization in response to changing network conditions might 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 documents for real-time media, Network-Assisted Dynamic Adaptation (NADA) [RFC8698] and Self-Clocked Rate Adaptation for Multimedia (SCReAM) [RFC8298]. These congestion control algorithms use feedback about the network's performance to calculate target bitrates. When these algorithms are used with RTP, the necessary feedback is generated at the receiver and sent back to the sender via RTCP.

Since QUIC itself collects some metrics about the network's performance, these QUIC 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 10.

3.1. Motivation

From time to time, someone asks the reasonable question, "why would 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 payloads have existed since the introduction of the Secure Real-time Transport Protocol (SRTP) [RFC3711], the additional encryption of RTP 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 the QUIC header fields needed to establish a connection, to "short headers", which only expose the absolute minimum QUIC header fields needed to identify an existing connection to the receiver, so that the QUIC payload is presented to the correct QUIC application [RFC8999].

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

When RTP is carried directly over UDP, as is commonly done, the underlying UDP protocol provides multiplexing using UDP ports, but no transport services beyond multiplexing are provided to the application. All congestion control behavior is up to the RTP application itself, and if anything goes wrong with the application and this condition results in an RTP sender failing to recognize that it is contributing to path congestion, the "worst case" response is to invoke the RTP "circuit breaker" procedures [RFC8083]. These procedures result 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 estimates are independent of any measurements RTP senders and receivers are maintaining. The result is that even if an RTP sender attempts to "send" in the presence of persistent path congestion, QUIC congestion control 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 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, arriving at the receiver later than a consumer 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 will impact RTT measurements using RTCP and can interfere with a sender's ability to stabilize rate control and achieve audio/video synchronization.

In summary,

* 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 and possibly more smoothly to the end of congestion than RTP "circuit breakers".

3.1.3. RTP Rate Adaptation Based on QUIC Feedback

When RTP is carried directly over UDP, RTP makes use of a large number of RTP-specific feedback mechanisms because 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 payload. This is described in (much) more detail in Section 8 on rate adaptation, and in Section 10 on replacing RTCP and RTP header extensions with QUIC feedback.

One word of caution is in order - RTP implementations might 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]. Because the necessary "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 (Maximum Transmission Unit) supported by conformant QUIC implementations is 1200 bytes [RFC9000]. In addition, QUIC implementations allow senders to use either DPLPMTUD ([RFC8899]) or PMTUD ([RFC1191], [RFC8201]) to determine the actual Path MTU size that the receiver can accept, and that the path between sender and receiver can support. The actual Path MTU can be larger than the Minimum Path MTU.

This is especially useful in certain conferencing topologies, where otherwise senders would have no choice but to use the lowest Path MTU for all conference participants. 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 document 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 with other protocols on the same connection, as long as these protocols can co-exist with RTP 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 could be desirable 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 senders 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 document. Path scheduling APIs to let applications control these mechanisms are a topic for future research and might need further specification in future documents.

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 ought to use. Indeed, an application can use both QUIC streams and DATAGRAM encapsulations on the same QUIC connection. The choice of encapsulation is left to the application developer, but it is worth noting differences that are relevant when making this choice.

QUIC [RFC9000] was initially designed to carry HTTP [RFC9114] in QUIC streams, and QUIC streams provide what HTTP application developers need – for example, a stateful, connection-oriented, flow-controlled, reliable, ordered stream of bytes to an application. QUIC streams 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 need 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 streams 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. QUIC implementations can present an API to allow applications to assign relative priorities within a QUIC connection, but this is not mandated by the standard and might 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 might 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 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 Path MTU 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.

DATAGRAM frames do inherit the QUIC connection's congestion controller. This means that although there is no frame-level flow control, DATAGRAM frames can 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 might not be present in all implementations.

Because DATAGRAMs are an extension to QUIC, a RoQ endpoint cannot assume that its peer supports this extension. The RoQ endpoint might discover that its peer does not support DATAGRAMs in one of two ways:

* as part of the signaling process to set up QUIC connections, or

* during negotiation of the DATAGRAM extension during the QUIC handshake.

When either of these situations happen, 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 streams instead.

3.3. Supported RTP Topologies

RoQ supports only 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 needs to access the content of QUIC frames (e.g., _Topo-PtP-Translator_, _Topo-PtP-Relay_, _Topo-Trn-Translator_, _Topo-Media-Translator_), the QUIC connection will 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 when 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 can be smaller than the MTU of a UDP datagram. In both cases, the translator might need to rewrite the RTP packets to fit into the smaller MTU of the other protocol. Such a translator might 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 (https://datatracker.ietf.org/doc/html/rfc7667#section-3.2.1.3)	Topo-Media-Translator	yes	Note-MTU
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 (https://datatracker.ietf.org/doc/html/rfc7667#section-3.8)	Topo-Video-switch-MCU	yes	Note-MTU 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: Not supported, because QUIC [RFC9000] does not support NAT traversal.

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

Note-Sec: Note that because RoQ uses QUIC as its underlying transport, and QUIC authenticates the entirety of each packet and encrypts as much of each packet as is practical, RoQ secures both RTP headers and RTP payloads, while other RTP transports do not. Section 15 describes strategies to prevent the inadvertent disclosure of RTP sessions to unintended third parties.

Note-MCast: Not supported, because QUIC [RFC9000] does not support IP multicast.

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: Supported for IP unicast paths between RTP sources and translators.

Note-Topo-Mixer: Supported for IP 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 Application-Layer Protocol Negotiation

QUIC requires the use of Application-Layer Protocol Negotiation (ALPN) [RFC7301] tokens during connection setup. RoQ uses "roq" as the ALPN token, included as part of the TLS handshake (see also Section 16).

Note that the "roq" ALPN token is not tied to a specific RTP profile, even though the RTP profile could be considered part of the application usage. This allows different RTP sessions, which might use different RTP profiles, to 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 ALPN token "roq" to identify itself during QUIC connection setup.

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 packets in QUIC.

QUIC supports two transport methods: QUIC streams [RFC9000] and DATAGRAMs [RFC9221]. This document specifies mappings of RTP to both transport modes. Senders MAY combine both modes by sending some RTP 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 streams and DATAGRAMs, respectively.

5.1. Multiplexing

RoQ uses flow identifiers to multiplex different RTP 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 an RTP stream.

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

RTP packets of different RTP sessions MUST use distinct flow identifiers. If endpoints wish to send multiple types of media in a single RTP session, they can do so by following the guidance specified in [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.

Endpoints need to associate flow identifiers with RTP streams. Depending on the context of the application, the association can be statically configured, signaled using an out-of-band signaling mechanism (e.g., SDP), or applications might be able to identify the stream based on the RTP packets sent on the stream (e.g., by inspecting the payload type).

If an endpoint receives a flow identifier that it cannot associate with an RTP stream, the endpoint MAY close the connection using the ROQ_UNKNOWN_FLOW_ID error code. Closing the connection can be a valid response if it is not expected that out of band signaling is still ongoing and the application cannot handle unknown flow identifiers.

If the association of flow identifiers with RTP streams depends on out-of-band signaling, the signaling mechanism SHOULD be completed before the exchange of RTP packets using the new flow identifiers starts.

In cases where it cannot be guaranteed that signaling is completed before RTP packets are transmitted, streams or DATAGRAMs with a given flow identifer can arrive before the signaling finished. In that case, an endpoint cannot associate the stream or DATAGRAM with the corresponding RTP stream. The endpoint can buffer streams and DATAGRAMs using an unknown flow identifier until they can be associated with the corresponding RTP stream. To avoid resource exhaustion, the buffering endpoint MUST limit the number of streams and DATAGRAMs to buffer. If the number of buffered streams exceeds the limit on buffered streams, the endpoint MUST send a STOP_SENDING with the error code ROQ_UNKNOWN_FLOW_ID. It is an implementation's choice on which stream to send STOP_SENDING. If the number of buffered DATAGRAMs exceeds the limit on buffered DATAGRAMs, the endpoint MUST drop a DATAGRAM. It is an implementation's choice which DATAGRAMs to drop.

Flow identifiers introduce some overhead in addition to the header overhead of RTP and QUIC. QUIC variable-length integers require between one and eight bytes depending on the number expressed. Thus, using low numbers as session identifiers first will minimize this additional overhead.

5.2. QUIC Streams

To send RTP 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 packets. An endpoint that receives a bidirectional stream with a flow identifier that is associated with an RTP 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.

The underlying QUIC implementation might be acting as either a QUIC client or QUIC server, so the unidirectional QUIC stream can be either client-initiated or server-initiated, as described in

Section 2.1 of [RFC9000], depending on the role. The QUIC implementation's role is not controlled by RoQ, and can be negotiated using a separate signaling protocol.

A RoQ sender can open new QUIC streams for different RTP packets using the same flow identifier. This allows RoQ senders to use QUIC streams while avoiding head-of-line blocking.

Because a sender can continue sending on a stream with a lower stream identifier after starting packet transmission on a stream with a higher stream identifier, 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 any particular stream.

5.2.1. Stream Encapsulation

Figure 1 shows the encapsulation format for RoQ Streams.

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

Figure 1: RoQ Streams Payload Format

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

RTP Payload: Contains the RTP payload; see Figure 2

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

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

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

Figure 2: RTP payload for QUIC streams

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

RTP Packet: The RTP 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 receiver that is no longer interested in reading a certain portion of the media stream can signal this to the sending peer using a STOP_SENDING frame.

If a RoQ sender discovers that a packet is no longer needed and knows that the packet has not yet been successfully and completely transmitted, it can use RESET_STREAM to tell the RoQ receiver that the RoQ sender is discarding the packet.

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 to carry that RTP media. This can mean that the RoQ receiver is no longer able to use the media frames being received because they are "too old". A sender with additional media frames to send can continue sending them on another QUIC stream. Alternatively, new media frames can be sent as 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 a particular media frame because the sending QUIC implementation might already have sent data for one or more following media frames on the same stream. In that case, STOP_SENDING and the resulting RESET_STREAM would also discard the following media frames and thus lead to unintentionally skipping one or more media 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 encapsulated in DATAGRAMs.

QUIC implementations will fragment large RTP packets into smaller QUIC STREAM frames. The data carried in these QUIC STREAM frames is transmitted reliably and is delivered to the receiving application in order, so 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 data that was carried in QUIC frames that were lost in transit is 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 flows. Endpoints that excessively restrict the number of streams or the flow-control window of these streams will increase the chance that the sending 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 can 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 at any time.

As an example, consider a conference scenario with 20 participants. Each participant receives audio and video streams of every other participant from a central RTP middlebox. 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 RTP middlebox every second.

In addition, the receiver ought to also consider the requirements of RTCP streams. These considerations can also be relevant when implementing signaling since it can be necessary to inform the receiver about how many stream credits it will have to provide to the sending peer, and how rapidly it must provide these stream credits.

5.3. QUIC DATAGRAMs

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

DATAGRAMs preserve application frame boundaries. Thus, a single RTP packet can be mapped to a single DATAGRAM without additional framing. Because QUIC DATAGRAMs cannot be IP-fragmented (Section 5 of [RFC9221]), senders need to consider the header overhead associated with DATAGRAMs, and ensure that the RTP 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 Packet (..),
}
```

Figure 3: RoQ Datagram Payload Format

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

RTP Packet: The RTP packet to transmit.

RoQ senders need to be aware that QUIC uses the concept of QUIC frames, and QUIC connections use different kinds of QUIC frames to carry 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 acknowledgments, 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 would need to create additional packets for ACK frames, 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. Another implication is that multiple RTP packets in either QUIC streams or QUIC DATAGRAMs might be encapsulated in a single QUIC packet, which is discussed in more detail in Section 12.3.

Since DATAGRAMs are not retransmitted on loss (see also Section 10.4 for loss signaling), if an application is using DATAGRAMs and wishes to retransmit lost RTP packets, the application has to carry out that retransmission. RTP retransmissions can be done in the same RTP session or in a different 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 endpoint can close the connection for any of a variety of reasons. If one of the endpoints wants to close the RoQ connection, the endpoint can use a QUIC CONNECTION_CLOSE frame with one of the error codes defined in Section 7.

7. 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 occurred.

ROQ_INTERNAL_ERROR (0x02): An internal error has occurred 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, e.g., a bidirectional stream, for sending RTP packets.

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 signaled or because the endpoint does not support multiplexing using arbitrary flow identifiers.

ROQ_EXPECTATION_UNMET (0x07): RoQ out-of-band signaling set expectations for QUIC transport, but the resulting QUIC connection would not meet those expectations.

8. 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 8.1 and application layer rate control in Section 8.2. 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 8.2.

This document also discusses congestion control implications of using shared or multiple separate QUIC connections to send and receive multiple independent RTP streams in Section 8.3.

8.1. 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 uses a congestion controller that aims at keeping queues at intermediary network elements, and thus latency, 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 transmission at low latencies, but the guidance in 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. QUIC implementations can use an extension to add this information to QUIC as described in Appendix A. In addition to these dedicated real-time media congestion-control algorithms, QUIC implementations could support the Low Latency, Low Loss, and Scalable Throughput (L4S) Internet Service [RFC9330], which limits growth in round-trip delays that result from 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 also 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 8.2.

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 the QUIC exemplary congestion control algorithm (Section 7.7 of [RFC9002]) recommends pacing for senders.

8.2. 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 an RTP application paces its media transmission at a rate that does not saturate path bandwidth, 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) ought to rarely come into play. If an RTP application chooses RoQ as its transport, sends enough media to saturate the available path bandwidth, and does not adapt its sending rate, these 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 10 and Section 11.

8.3. Sharing QUIC connections

Two endpoints can establish channels 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.

* 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, whether real-time or non-real-time, 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 real-time and non-real-time channels in one connection will need to implement some form of prioritization or bandwidth allocation for the different channels.

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

As noted in Section 3.2, this document does not take a position on using QUIC streams, QUIC DATAGRAMs, or a 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 "IP plus multiplexing". The implementers 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, using 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 dealing with MTU size restrictions, in contrast to the need to fit RTP packets into 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 neatly 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 RTP audio packets 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 RoQ datagrams begin with a flow identifier. This allows a RoQ sender to begin by encapsulating related RTP packets in QUIC streams 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]), when a QUIC connection migrates, 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. Again, RoQ receivers need to be prepared for this eventuality.

10. Replacing RTCP and RTP Header Extensions with QUIC Feedback

Because RTP has so often used UDP as its underlying transport protocol, receiving little or no transport feedback, existing 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 media streams.

Because QUIC can provide 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 feedback provided by 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 are to be sent. This document does not change any of the rules described by either protocol. Rather, it 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 still need to adhere to the rules for RTCP feedback given by [RFC3550] and the RTP profiles in use.

Most statements about "QUIC" in Section 10 are applicable to both RTP packets encapsulated in QUIC streams and RTP packets encapsulated in DATAGRAMs. The differences are described in Section 10.1 and Section 10.2.

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 processing and bandwidth overhead for RoQ senders and receivers.

Section 10.4 discusses what information can be exposed from the QUIC connection layer to reduce the RTCP overhead.

10.1. 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.

10.2. RoQ Streams

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

When QUIC streams are used, the implementer needs to be aware that the direct mapping proposed below might 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 can be incorrect due to retransmissions or transmission delays introduced by the QUIC stack.

10.3. Multihop Topologies

In some topologies, RoQ might only be used on some of the links between multiple session participants. Other links might be using RTP over UDP, or over some other supported RTP encapsulation protocol, and some participants might be using RTP implementations that don't support RoQ at all. These participants will not be able to infer feedback from QUIC, and they might receive less RTCP feedback than expected. This situation can arise when participants using RoQ are not aware that other participants are not using RoQ and minimize their use of RTCP, assuming 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 can use additional signaling to let the RoQ participants know what RTCP is required.

10.4. 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 including the information that cannot be mapped from QUIC can be found in Appendix B: 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).

10.4.1. 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]).

10.4.2. 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.

10.4.3. 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 streams and DATAGRAMs.

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.

* _initial_max_data transport_: If the QUIC receiver has indicated a willingness to accept 0-RTT packets with early data, this is the maximum size that the QUIC sender can use, as described in Section 12.2.

* _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 10.1).

* _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 10.2.

* _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 10 and Section 8.

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

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

* _RTT_: The RTT estimations as described in Section 5 of [RFC9002].

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, but some information will be valuable. For example, it could be desirable that the RoQ implementation provides an indication of connection migration to the RTP application.

12. Discussion

This section contains topics that are worth mentioning, but don't fit well into other sections of the document.

12.1. Impact of Connection Migration

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

12.2. 0-RTT and Early Data considerations

RoQ applications, like any other RTP applications, want to establish a media path quickly, reducing clipping at the beginning of a conversation. For repeated connections between endpoints that have previously carried out a full TLS handshake procedure, the initiator of a QUIC connection can use 0-RTT packets with "early data" to include application data in a packet that is used to establish a connection.

As 0-RTT packets are subject to replay attacks, RoQ applications MUST carefully specify which data types and operations are allowed.

Section 9.2 of [RFC9001] says

> Application protocols MUST either prohibit the use of extensions that carry application semantics in 0-RTT or provide replay mitigation strategies.

For the purposes of this discussion, RoQ is an application protocol that allows the use of 0-RTT.

RoQ application developers ought to take the considerations described in Section 12.2.1 and Section 12.2.2 into account when deciding whether to use 0-RTT with early data for an application.

12.2.1. Effect of 0-RTT Rejection for RoQ using Early Data

If the goal for using early data is to reduce clipping, a QUIC endpoint is relying on the other QUIC endpoint to accept the 0-RTT packet carrying early data containing media.

A QUIC endpoint indicates its willingness to accept a 0-RTT packet containing early data by sending the TLS early_data extension in the NewSessionTicket message with the max_early_data_size parameter set to the sentinel value 0xffffffff. The amount of data that a QUIC client can send in QUIC 0-RTT is controlled by the initial_max_data transport parameter supplied by the QUIC server. This is described in more detail in Section 4.6.1 of [RFC9001].

If a QUIC endpoint is not willing to accept a 0-RTT packet containing early data, but receives one anyway, the QUIC endpoint rejects the 0-RTT packet by sending EncryptedExtensions without an early_data extension. This is described in more detail, in Section 4.6.2 of [RFC9001]. This necessarily means that a QUIC endpoint attempting to convey RoQ media is now subject to at least one additional RTT delay, as it must send a QUIC Initial packet and perform a full QUIC handshake before it can send RoQ media.

12.2.2. Effect of 0-RTT Replay Attacks for RoQ using Early Data

Including "early data" in the packet payload in any QUIC 0-RTT packet exposes the application to an additional risk, of accepting "early data" from a 0-RTT packet that has been replayed.

While it is true that

* RTP typically carries ephemeral media contents that is rendered and possibly recorded but otherwise causes no side effects,

* the amount of data that can be carried as packet payload in a 0-RTT packet is rather limited, and

* RTP implementations are likely to discard any replayed media packets as duplicates,

it is still the responsibility of the RoQ application to determine whether the benefits of using 0-RTT with early data outweigh the risks.

Since the QUIC connection will often be created in the context of an existing signaling relationship (e.g., using WebRTC or SIP), a careful RoQ implementer can exchange specific 0-RTT keying material to prevent replays across sessions.

12.3. Coalescing RTP packets in a 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 can 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 could 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 could 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 might 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 document 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 (Section 13.1) and as new QUIC extensions become available (Section 13.2).

13.1. Future Work Resulting from Implementation and Deployment Experience

Possible directions would include

* More guidance on transport for RTCP (for example, when to use QUIC streams vs. DATAGRAMs) including guidance on prioritization between streams and DATAGRAMs for the performance of RTCP.

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

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

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

* Investigation of the effects of delaying or dropping DATAGRAMs due to congestion before they can be transmitted by the QUIC stack.

* Implementation of translating middleboxes for translating between RoQ and RTP over UDP. As described in Section 3.3, RoQ can be used to connect to some RTP middleboxes using some topologies, and these middleboxes might be connecting RoQ endpoints and non-RoQ endpoints, so will need to translate between RoQ and RTP over UDP.

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

13.2. Future Work Resulting from New QUIC Extensions

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 specific proposed QUIC extensions in Appendix A, but these proposals are all solving relevant problems, and those problems are worth solving for the QUIC protocol, whether the listed proposals are used in the solution or

not.

* Guidance for using RoQ with QUIC connection migration and over multiple paths. QUIC connection migration was already defined in [RFC9000], and the Multipath Extension for QUIC [I-D.draft-ietf-quic-multipath] has been adopted and is relatively mature, so this is likely to be the first new QUIC extension we address.

* 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, might also be useful with RoQ.

14. Implementation Status

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

This section records the status of known implementations of the protocol defined by this specification at the time of posting of this Internet-Draft, and is based on a proposal described in [RFC7942]. The description of implementations in this section is intended to assist the IETF in its decision processes in progressing drafts to RFCs. Please note that the listing of any individual implementation here does not imply endorsement by the IETF. Furthermore, no effort has been spent to verify the information presented here that was supplied by IETF contributors. This is not intended as, and must not be construed to be, a catalog of available implementations or their features. Readers are advised to note that other implementations may exist.

According to [RFC7942], "this will allow reviewers and working groups to assign due consideration to documents that have the benefit of running code, which may serve as evidence of valuable experimentation and feedback that have made the implemented protocols more mature. It is up to the individual working groups to use this information as they see fit".

14.1. mengelbart/roq

Ogranization: Technical University of Munich

Implementation: [roq]

Description: _roq_ is a library implementing the basic encapsulation described in Section 5. The library uses the Go programming language and supports the [quic-go] QUIC implementation.

Level of Maturity: prototype

Coverage: : The library supports sending and receiving RTP and RTCP packets using QUIC streams and QUIC DATAGRAMs, and supports multiplexing using flow identifiers. Applications using the library are responsible for appropriate signaling, setting up QUIC connections, and managing RTP sessions. Applications choose whether to send RTP and RTCP packets over streams or DATAGRAMs, and applications also have control over the QUIC and RTP congestion controllers in use since they control the QUIC connection setup and can thus configure the QUIC stack they use to their preferences.

Version Compatibility: The library implements [I-D.draft-ietf-avtcore-rtp-over-quic-10].

Licensing: MIT License

Implementation Experience: The implementer reports they have no experience with the topics discussed in Section 13. RoQ relies on out-of-band signaling for connection establishment, and since there is currently no specification for SDP for RoQ, applications using the library have to statically configure connection information to allow testing

Contact Information: Mathis Engelbart (mathis.engelbart@gmail.com)

Last Updated: 25 May 2024

14.2. bbc/gst-roq

Ogranization: BBC Research and Development

Implementation: RTP-over-QUIC elements for GStreamer [gst-roq]

Description: _gst-quic-transport_ provides a set of GStreamer plugins implementing QUIC transport. _gst-roq_ provides a set of GStreamer plugins implementing RoQ.

Level of Maturity: research

Coverage: The plugins support sending and receiving RTP and RTCP packets using QUIC streams and QUIC DATAGRAMs, and supports multiplexing using flow identifiers. GStreamer pipelines that use the RoQ plugins found in the _gst-roq_ repository can make use of the plugins found in the _gst-quic-transport_ repository to set up QUIC connections. RTP sessions can be managed by existing GStreamer plugins available in the standard GStreamer release. GStreamer pipeline applications choose whether to send RTP and RTCP packets over streams or DATAGRAMs.

Version Compatibility: The library implements [I-D.draft-ietf-avtcore-rtp-over-quic-05].

Licensing: GNU Lesser General Public License v2.1

Implementation Experience: The implementer reports they have no experience with the topics discussed in Section 13. Both in-band and out-of-band signalling for RoQ media sessions is in active development via an implementation of [I-D.draft-hurst-sip-quic-00], which re-uses the GStreamer plugins described above.

Contact Information: Sam Hurst (sam.hurst@bbc.co.uk)

Last Updated: 05 June 2024

14.3. mengelbart/rtp-over-quic

Ogranization: Technical University of Munich

Implementation: RTP over QUIC [RTP-over-QUIC]

Description: _RTP over QUIC_ is a experimental implementation of the mapping described in an earlier version of this document.

Level of Maturity: research

Coverage: The application implements the RoQ DATAGRAMs mapping and implements SCReAM congestion control at the application layer. It can optionally disable the built-in 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 10.4. Experimental results of the implementation can be found in [RoQ-Mininet].

Version Compatibility: [I-D.draft-ietf-avtcore-rtp-over-quic-00]

Licensing: MIT

Implementation Experience: See [RoQ-Mininet]

Contact Information: Mathis Engelbart (mathis.engelbart@gmail.com)

Last Updated: 25 May 2024

15. 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.

16. IANA Considerations

This document registers a new ALPN protocol ID (in Section 16.1) and creates a new registry that manages the assignment of error code points in RoQ (in Section 16.2).

16.1. 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

16.2. 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 7
0x01	ROQ_GENERAL_ERROR	General error	Section 7
0x02	ROQ_INTERNAL_ERROR	Internal Error	Section 7
0x03	ROQ_PACKET_ERROR	Invalid payload format	Section 7
0x04	ROQ_STREAM_CREATION_ERROR	Invalid stream type	Section 7
0x05	ROQ_FRAME_CANCELLED	Frame cancelled	Section 7
0x06	ROQ_UNKNOWN_FLOW_ID	Unknown Flow ID	Section 7
0x07	ROQ_EXPECTATION_UNMET	Externally signaled requirement unmet	Section 7

Table 2: Initial RoQ Error Codes

17. References

17.1. 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>.

[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>.

17.2. Informative References

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

[gst-roq] "RTP-over-QUIC elements for GStreamer", n.d., <https://github.com/bbc/gst-roq>.

[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-hurst-sip-quic-00]
Hurst, S., "SIP-over-QUIC: Session Initiation Protocol over QUIC Transport", Work in Progress, Internet-Draft, draft-hurst-sip-quic-00, 6 November 2022, <https://datatracker.ietf.org/doc/html/draft-hurst-sip-quic-00>.

[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-03, 8 April 2024, <https://datatracker.ietf.org/doc/html/draft-ietf-avtcore-rtcp-green-metadata-03>.

[I-D.draft-ietf-avtcore-rtp-over-quic-00]
Ott, J. and M. Engelbart, "RTP over QUIC", Work in Progress, Internet-Draft, draft-ietf-avtcore-rtp-over-quic-00, 26 July 2022, <https://datatracker.ietf.org/doc/html/draft-ietf-avtcore-rtp-over-quic-00>.

[I-D.draft-ietf-avtcore-rtp-over-quic-05]
Ott, J., Engelbart, M., and S. Dawkins, "RTP over QUIC (RoQ)", Work in Progress, Internet-Draft, draft-ietf-

avtcore-rtp-over-quic-05, 26 July 2023, <https://datatracker.ietf.org/doc/html/draft-ietf-avtcore-rtp-over-quic-05>.

[I-D.draft-ietf-avtcore-rtp-over-quic-10]
Ott, J., Engelbart, M., and S. Dawkins, "RTP over QUIC (RoQ)", Work in Progress, Internet-Draft, draft-ietf-avtcore-rtp-over-quic-10, 8 May 2024, <https://datatracker.ietf.org/doc/html/draft-ietf-avtcore-rtp-over-quic-10>.

[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. KÃ¼hlewind, "QUIC Acknowledgment Frequency", Work in Progress, Internet-Draft, draft-ietf-quic-ack-frequency-09, 30 April 2024, <https://datatracker.ietf.org/doc/html/draft-ietf-quic-ack-frequency-09>.

[I-D.draft-ietf-quic-multipath]
Liu, Y., Ma, Y., De Coninck, Q., Bonaventure, O., Huitema, C., and M. KÃ¼hlewind, "Multipath Extension for QUIC", Work in Progress, Internet-Draft, draft-ietf-quic-multipath-09, 21 June 2024, <https://datatracker.ietf.org/doc/html/draft-ietf-quic-multipath-09>.

[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-14, 3 May 2024, <https://datatracker.ietf.org/doc/html/draft-ietf-wish-whip-14>.

[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>.

[IEEE-1733-2011]
"IEEE 1733-2011 Standard for Layer 3 Transport Protocol for Time-Sensitive Applications in Local Area Networks", n.d., <https://standards.ieee.org/ieee/1733/4748/>.

[quic-go] "A QUIC implementation in pure Go", n.d., <https://github.com/quic-go/quic-go>.

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

[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>.

[RFC7942] Sheffer, Y. and A. Farrel, "Improving Awareness of Running Code: The Implementation Status Section", BCP 205, RFC 7942, DOI 10.17487/RFC7942, July 2016, <https://www.rfc-editor.org/rfc/rfc7942>.

[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., TÃ¼xen, M., RÃ¼ngeler, I., and T. VÃ¶lker, "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] KÃ¼hlewind, 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>.

[roq] "RTP over QUIC (RoQ)", n.d., <https://github.com/mengelbart/roq>.

[RoQ-Mininet] "Congestion Control for RTP over QUIC Simulations", n.d., <https://github.com/mengelbart/rtp-over-quic-mininet>.

[RTP-over-QUIC] "RTP over QUIC", n.d., <https://github.com/mengelbart/rtp-over-quic>.

[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 could 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 such as _Quic Timestamps For Measuring One-Way Delays_ [I-D.draft-huitema-quic-ts] or _QUIC Extension for Reporting Packet Receive Timestamps_ [I-D.draft-smith-quic-receive-ts], the arrival timestamps or one-way delays could be exposed to the application for improved bandwidth estimation or RTCP mappings as described in Section 10 and Appendix B.

* _QUIC Acknowledgment Frequency_ [I-D.draft-ietf-quic-ack-frequency] can be used by a sender to optimize the acknowledgment behavior 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 ought to 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 10.

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 10.4. _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 10.4.3
Application-defined	APP	204	[RFC3550]	no	
Generic RTP Feedback	RTPFB	205	[RFC4585]	partly	see Appendix B.3
Payload-specific	PSFB	206	[RFC4585]	partly	see Appendix B.4
extended report	XR	207	[RFC3611]	partly	see Appendix B.2

AVB RTCP packet	AVB	208	[IEEE-1733-2011]	no	
Receiver Summary Information	RSI	209	[RFC5760]	no	
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 10.1 and Section 10.2
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 10.4.2
Measurement Information Block	[RFC6776]	no	

Packet Delay Variation Metrics Block	[RFC6798]	no	QUIC timestamps can be used to achieve the same goal
Delay Metrics Block	[RFC6843]	no	QUIC has RTT and can provide timestamps for one-way delay, but QUIC timestamps cannot provide 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 Report)	[RFC7243]	no	
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 10.4.1
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 to tell a QUIC implementation "don’t ask for retransmission".
RTCP-ECN-FB	RTCP ECN Feedback	[RFC6679]	partly	see Section 10.4.2
PAUSE-RESUME	Media Pause/ Resume	[RFC7728]	no	
DBI	Delay Budget Information (DBI)	[_3GPP-TS-26.114]		
CCFB	RTP Congestion Control Feedback	[RFC8888]	QUIC extension needed	see 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 Loss Early Indication	[RFC6642]
ROI	Video region-of-interest	[_3GPP-TS-26.114]

	(ROI)	
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 orientation (CVO) feature, see clause 6.2.3	[_3GPP-TS-26.114]	probably not(?)
urn:3gpp:video-orientation	Coordination of video orientation (CVO) feature,	[_3GPP-TS-26.114]	probably not(?)

	see clause 6.2.3		
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 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 media 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 media 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 might 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.

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.

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

Mathis Engelbart
Technical University of Munich
Email: mathis.engelbart@gmail.com

JÃ¶rg Ott
Technical University of Munich

Email: ott@in.tum.de

Spencer Dawkins
Tencent America LLC
Email: spencerdawkins.ietf@gmail.com

avtcore
Internet-Draft
Intended status: Standards Track
Expires: 23 March 2025

L. Ilola
L. Kondrad
Nokia Technologies

19 September 2024

RTP Payload Format for Visual Volumetric Video-based Coding (V3C) draft-ietf-avtcore-rtp-v3c-07

Abstract

A visual volumetric video-based coding (V3C) [ISO.IEC.23090-5] bitstream is composed of V3C units that contain V3C atlas sub-bitstreams, V3C video sub-bitstreams, and a V3C parameter set. This memo describes an RTP payload format for V3C atlas sub-bitstreams. The RTP payload format for V3C video sub-bitstreams is defined by relevant Internet Standards for the applicable video codec. The V3C RTP payload format allows for packetization of one or more V3C atlas Network Abstraction Layer (NAL) units in an RTP packet payload as well as fragmentation of a V3C atlas NAL unit into multiple RTP packets. The memo also describes the mechanisms for grouping RTP streams of V3C component sub-bitstreams, providing a complete solution for streaming V3C encoded content.

Status of This Memo

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

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

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

This Internet-Draft will expire on 23 March 2025.

Table of Contents

1. Introduction

Volumetric video, similar to traditional 2D video, when uncompressed, is represented by a large amount of data. The Visual Volumetric Video-based Coding (V3C) specification [ISO.IEC.23090-5] leverages the compression efficiency of existing 2D video codecs to reduce the amount of data needed for storage and transmission of volumetric video. V3C is a generic mechanism for volumetric video coding, and it can be used by applications targeting volumetric content, such as point clouds, Video-based Point Cloud Compression (V-PCC) [ISO.IEC.23090-5], and immersive video with depth, MPEG Immersive Video (MIV) [ISO.IEC.23090-12].

V3C encoder converts volumetric frames, i.e., 3D volumetric information, into a collection of 2D frames and associated data, known as atlas data. The converted 2D frames are subsequently coded using any video or image codec, e.g., ISO/IEC International Standard 14496-10 (Advanced Video Coding, AVC/H.264) [ISO.IEC.14496-10], ISO/IEC International Standard 23008-2 (High Efficiency Video Coding, HEVC/H.265) [ISO.IEC.23008-2] or ISO/IEC International Standard 23090-3 (Versatile Video Coding, VVC/H.266) [ISO.IEC.23090-3]. The atlas data is coded with mechanisms specified in [ISO.IEC.23090-5].

V3C utilizes high level syntax (HLS) design, familiar from traditional 2D video codecs, to represent the associated coded data, i.e., atlas data. The coded atlas data is represented by Network Abstraction Layer (NAL) units. Consequently, RTP payload format for V3C atlas data described in this memo shares design philosophy, security, congestion control, and overall implementation complexity with the other NAL unit-based RTP payload formats such as the ones defined in [RFC6184], [RFC6190], and [RFC7798].

2. Conventions

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

All fields defined in this specification related to RTP payload structures SHALL be considered in network order.

3. Definitions, and abbreviations

3.1. Definitions

3.1.1. General

This document uses the definitions of [ISO.IEC.23090-5]. Section 3.1.2 below lists relevant definitions from [ISO.IEC.23090-5] for convenience.

3.1.2. Definitions from the V3C specification

atlas: collection of 2D bounding boxes and their associated information placed onto a rectangular frame and corresponding to a volume in 3D space on which volumetric data is rendered.

atlas bitstream: sequence of bits that forms the representation of atlas frames and associated data forming one or more coded atlas sequences.

atlas coding layer NAL unit: collective term for coded atlas tile layer NAL units and the subset of NAL units that have reserved values of nal_unit_type that are classified as being of type class equal to ACL in this document.

atlas frame: 2D rectangular array of atlas samples onto which patches are projected and additional information related to the patches, corresponding to a volumetric frame.

attribute: scalar or vector property optionally associated with each point in a volumetric frame such as colour, reflectance, surface normal, timestamps, material ID, etc.

coded atlas sequence: sequence of coded atlas access units, in decoding order, of an IRAP coded atlas access unit, followed by zero or more coded atlas access units that are not IRAP coded atlas access units, including all subsequent access units up to but not including any subsequent coded atlas access unit that is an IRAP coded atlas access unit.

coded atlas access unit: set of atlas NAL units that are associated with each other according to a specified classification rule, are consecutive in decoding order, and contain all atlas NAL units pertaining to one particular output time.

coded visual volumetric video-based coding (V3C) sequence: sequence of V3C atlas and video sub-bitstream(s) identified and separated by appropriate delimiters, required to start with a VPS, included in at least one V3C unit or provided through external means.

network abstraction layer unit: syntax structure containing an indication of the type of data to follow and bytes containing that data in the form of an RBSP.

patch: rectangular region within an atlas associated with volumetric information.

raw byte sequence payload: syntax structure containing an integer number of bytes that is encapsulated in a NAL unit and that is either empty or has the form of a string of data bits containing syntax elements followed by an RBSP stop bit and zero or more subsequent bits equal to 0.

tile: independently decodable rectangular region of an atlas frame.

visual volumetric video-based coding (V3C) atlas sub-bitstream: extracted sub-bitstream from the V3C bitstream containing whole or portion of an atlas bitstream.

visual volumetric video-based coding (V3C) video sub-bitstream: extracted sub-bitstream from the V3C bitstream containing whole or portion of a video bitstream.

visual volumetric video-based coding (V3C) component: atlas, occupancy, geometry, or attribute of a particular type that is associated with a V3C volumetric content representation.

visual volumetric video-based coding (V3C) parameter set: syntax structure containing syntax elements that apply to zero or more entire CVSs and may be referred to by syntax elements found in the V3C unit header.

volumetric frame: set of 3D points specified by their cartesian coordinates and zero or more corresponding sets of attributes at a particular time instance.

3.2. Abbreviations

ACL atlas coding layer

AP aggregation packet

AU aggregation unit

CVS coded V3C sequence

DON decoding order number

IRAP intra random access point

MTU maximum transmission unit

NAL network abstraction layer

NALU NAL unit

RBSP raw byte sequence payload

V3C visual volumetric video-based coding

VPS V3C parameter set

4. Media format description

4.1. Overview of the V3C codec (informative)

V3C encoding of a volumetric frame is achieved through a conversion of the volumetric frame from its 3D representation into multiple 2D representations and a generation of associated data documenting such conversions and transformations. The associated data, also known as the atlas data, provides information on how to reproject the 2D representations back into the 3D volumetric frame.

2D representations, known as V3C video components, of volumetric frame are encoded using traditional 2D video codecs. V3C video component may, for example, include occupancy, geometry, or attribute data. The occupancy data informs a V3C decoder which pixels in other V3C video components contribute to reconstructed 3D representation. The geometry data describes information on the position of the reconstructed voxels, while attribute data provides additional properties for the voxels, e.g., colour or material information.

Atlas data, known as V3C atlas component, provides information to interpret V3C video components and enables the reconstruction from a 2D representation back into a 3D representation of volumetric frame. Atlas data is composed of a collection of patches. Each patch identifies a region in the V3C video components and provides information necessary to perform the appropriate inverse projection of the indicated region back into 3D space. The shape of the patch region is determined by a 2D bounding box associated with each patch as well as their coding order. The shape of these patches is also further refined based on occupancy data.

To enable parallelization, random access, as well as a variety of other functionalities, an atlas frame can be divided into one or more rectangular partitions referred to as tiles. Tiles are not allowed to overlap and should be independently decodable. An atlas frame may contain regions that are not associated with any tile or patch.

The binary form of V3C video components, i.e., video bitstream, and V3C atlas components, i.e., atlas bitstream, can be grouped and represented by a single V3C bitstream. The V3C bitstream is composed of a set of V3C units. Each V3C unit has a V3C unit header and a V3C unit payload. The V3C unit header describes the V3C unit type for the payload. V3C unit payload contains V3C video components, V3C atlas components or a V3C parameter set. V3C video components, i.e., occupancy, geometry, or attribute components, correspond to video data units (e.g., NAL units defined in [ISO.IEC.23008-2]) that could be decoded by an appropriate video decoder. An example of V3C bitstream consisting of a V3C parameter set, atlas bitstream and three video component bitsreams (geometry, occupancy, attribute) is provided in Figure 1.

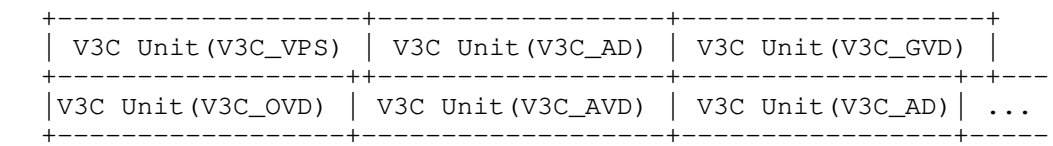


Figure 1: Example of V3C bitstream

4.2. V3C parameter set (informative)

While this memo intends to describe encapsulation of V3C atlas data, aspects related to signalling of V3C parameter set are also considered. V3C parameter set is encapsulated in its own V3C unit, which allows decoupling the transmission of V3C parameter set from the V3C video and atlas components. V3C parameter set may be transmitted by external means (e.g., as a result of the capability exchange) or through a (reliable or unreliable) control protocol. This memo provides information on how a V3C parameter set may be signalled as part of session description protocol, see Section 9.

Generally, it is useful to signal V3C parameter set out-of-band, because it describes what overall resources are needed to decode and reconstruct the associated V3C bitstream. Signalling it dynamically as part of an RTP stream might result in undefined behaviour when receiver does not have the required capabilities to decode the received V3C video component sub-bitstreams or when reconstruction process relies on information that the receiver does not support.

4.3. V3C atlas and video components (informative)

4.3.1. General

In V3C bitstream the atlas component is identified by vuh_unit_type equal to V3C_AD, or V3C_CAD in case of common atlas data, in the V3C unit header. The V3C atlas component consists of atlas NAL units that define header and payload pairs, see Section 4.3.2. V3C video components are identified by vuh_unit_type equal to V3C_OVD, V3C_GVD, V3C_AVD, and V3C_PVD. V3C video components can be further differentiated by other values in the V3C unit header such as vuh_attribute_index, vuh_attribute_partition_index, vuh_map_index and vuh_auxiliary_video_flag. By mapping V3C parameter set information to vuh_attribute_index, a V3C decoder identifies which attribute a given V3C video component contains, e.g., colour.

The information supplied by V3C unit header should be provided in one form or another to a V3C decoder, e.g., as part of SDP as described in this memo in Section 9. The four-byte V3C unit header syntax and semantics are copied below as defined in [ISO.IEC.23090-5], but the syntax is subject to change. Implementations should always refer to the latest specification of [ISO.IEC.23090-5]. The syntax of four-byte V3C unit header is provided here for informative purposes only.

```
v3c_unit_header( ) {
 unsigned int(5) vuh_unit_type;
 if( vuh_unit_type == V3C_AVD || vuh_unit_type == V3C_GVD ||
   vuh_unit_type == V3C_OVD || vuh_unit_type == V3C_AD ||
   vuh_unit_type == V3C_CAD || vuh_unit_type == V3C_PVD ) {
   unsigned int(4) vuh_v3c_parameter_set_id;
  }
  if( vuh_unit_type == V3C_AVD || vuh_unit_type == V3C_GVD ||
    vuh_unit_type == V3C_OVD || vuh_unit_type == V3C_AD ||
    vuh_unit_type == V3C_PVD ) {
    unsigned int(6) vuh_atlas_id;
  }
  if( vuh_unit_type == V3C_AVD ) {
    unsigned int(7) vuh_attribute_index;
    unsigned int(5) vuh_attribute_partition_index;
    unsigned int(4) vuh_map_index;
    unsigned int(1) vuh_auxiliary_video_flag;
  }
  else if( vuh_unit_type == V3C_GVD ) {
    unsigned int(4) vuh_map_index;
    unsigned int(1) vuh_auxiliary_video_flag;
    bit(12) vuh_reserved_zero_12bits;
  }
  else if( vuh_unit_type == V3C_OVD || vuh_unit_type == V3C_AD ||
      vuh_unit_type == V3C_PVD) {
    bit(17) vuh_reserved_zero_17bits;
  }
  else if( vuh_unit_type == V3C_CAD ) {
    bit(23) vuh_reserved_zero_23bits;
  }
  else {
    bit(27) vuh_reserved_zero_27bits;
  }
}
```

vuh_unit_type indicates the V3C unit type for the V3C component as specified in [ISO.IEC.23090-5]. As a convenience, the mapping table from vuh_unit_type values to semantics is copied below in Table 1.

vuh_unit_type	Identifier	V3C unit type	Description
0	V3C_VPS	V3C parameter set	V3C level parameters
1	V3C_AD	Atlas data	Atlas information
2	V3C_OVD	Occupancy video data	Occupancy information
3	V3C_GVD	Geometry video data	Geometry information
4	V3C_AVD	Attribute video data	Attribute information
5	V3C_PVD	Packed video data	Packing information
6	V3C_CAD	Common atlas data	Information that is common for atlases in a CVS. Specified in ISO/IEC 23090-12
7...31	V3C_RSVD	Reserved	-

Table 1: V3C unit type semantics

vuh_v3c_parameter_set_id specifies the value of vps_v3c_parameter_set_id for the active V3C VPS.

vuh_atlas_id specifies the ID of the atlas that corresponds to the current V3C unit.

vuh_attribute_index indicates the index of the attribute data carried in the Attribute Video Data unit.

vuh_attribute_partition_index indicates the index of the attribute dimension group carried in the attribute video data unit.

vuh_map_index when present indicates the map index of the current geometry or attribute stream. When not present, the map index of the current geometry or attribute sub-bitstream is derived based on the type of the sub-bitstream.

vuh_auxiliary_video_flag equal indicates if the associated geometry or attribute video data unit is a RAW and/or EOM coded points video only sub-bitstream.

4.3.2. Atlas NAL units

Atlas NAL unit (nal_unit(NumBytesInNalUnit)) is a byte-aligned syntax structure defined by [ISO.IEC.23090-5] to carry atlas data. Atlas NAL unit always contains a 16-bit NAL unit header (nal_unit_header()), which indicates among other things the type of the NAL unit (nal_unit_type). The payload of a NAL unit refers to the NAL unit excluding the NAL unit header. The Atlas NAL unit syntax and semantics are copied here as defined in [ISO.IEC.23090-5].

```
nal_unit_header(){
    bit(1) nal_forbidden_zero_bit;
    bit(6) nal_unit_type;
    bit(6) nal_layer_id;
    bit(3) nal_temporal_id_plus1;
}
nal_unit(NumBytesInNalUnit){
    nal_unit_header();
    NumBytesInRbsp = 0;
    for( i = 2; i < NumBytesInNalUnit; i++ )
      bit(8) rbsp_byte[ NumBytesInRbsp++ ];
}
```

nal_forbidden_zero_bit MUST be equal to 0. (F)

nal_unit_type indicates the type of the RBSP data structure contained in the NAL unit (NUT)

nal_layer_id indicates the identifier of the layer to which an ACL NAL unit belongs or the identifier of a layer to which a non-ACL NAL unit applies. (NLI)

nal_temporal_id_plus1 minus 1 indicates a temporal identifier for the NAL unit. The value of nal_temporal_id_plus1 MUST NOT be equal to 0. (TID)

4.4. Systems and transport interfaces (informative)

In addition to releasing specifications on V3C applications [ISO.IEC.23090-5] and [ISO.IEC.23090-12], MPEG conducted further systems level work on file formats to encapsulate compressed V3C content. The seventh edition of the ISOBMFF specification [ISO.IEC.14496-12] introduces a new media handler 'volv', intended to support volumetric visual media. It also specifies other structures to enable development of derived specifications detailing how various volumetric visual media may be stored in ISOBMFF.

One of such derived specifications is [ISO.IEC.23090-10], which defines how V3C content can be stored in a file and streamed over DASH. To a large extent ISO/IEC 23090-10 focuses on describing how ISOBMFF boxes and syntax elements may be used to store volumetric media, but in some cases new boxes and syntax elements are introduced to accommodate the fundamentally different type of new media. While the specification is not directly relevant for defining RTP payload format for V3C atlas data, it is a useful resource that may be considered especially when designing ingestion of encoded V3C content into RTP streaming pipelines.

5. V3C atlas RTP payload format

5.1. General

This section describes details related to V3C atlas RTP payload format definitions. Aspects related to RTP header, RTP payload header and general payload structure are considered. RTP payload format(s) for video components is defined in their respective RTP payload format specifications depending on the video codec used.

5.2. RTP header

The format of the RTP header is specified in [RFC3550] and replicated below in Figure 2 for convenience. This payload format uses the fields of the header in a manner consistent with that specification.

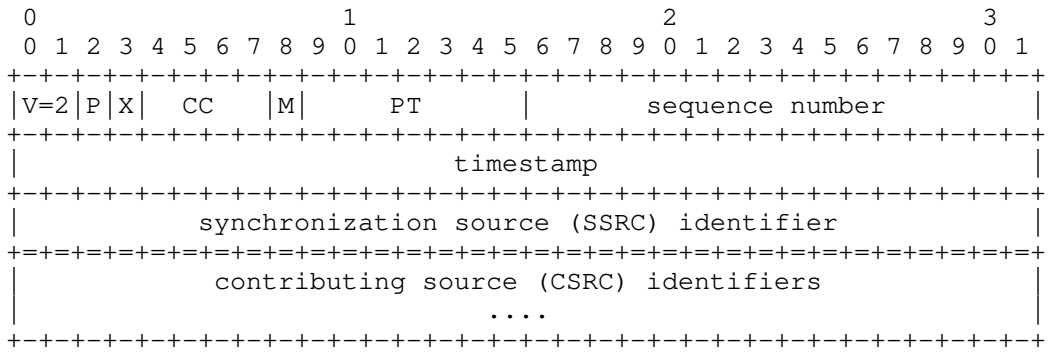


Figure 2: RTP Header

The RTP header information to be set according to this RTP payload format is set as follows:

Marker bit (M): 1 bit

Set for the last packet of the access unit, carried in the current RTP stream. This is in line with the normal use of the M bit in video formats to allow an efficient playout buffer handling.

Payload Type (PT): 7 bits

The assignment of an RTP payload type for this new packet format is outside the scope of this document and will not be specified here. The assignment of a payload type MUST be performed either through the profile used or in a dynamic way.

Sequence Number (SN): 16 bits

Set and used in accordance with [RFC3550]

Timestamp (32 bits):

The RTP timestamp is set to the sampling timestamp of the content. A 90 kHz clock rate MUST be used.

If the NAL unit has no timing properties of its own (e.g., parameter set and SEI NAL units), the RTP timestamp MUST be set to the RTP timestamp of the coded atlas of the access unit in which the NAL unit (according to Section 8.4.5.3 of [ISO.IEC.23090-5]) is included.

Receivers MUST use the RTP timestamp for the display process, even when the bitstream contains atlas frame timing SEI messages as specified in [ISO.IEC.23090-5].

Synchronization source (SSRC): 32 bits

Used to identify the source of the RTP packets. By definition a single SSRC is used for all parts of a single bitstream.

The remaining RTP header fields are used as specified in [RFC3550].

5.3. RTP payload header

The first two bytes of the payload of an RTP packet are referred to as the payload header. The payload header consists of the same fields (F, NUT, NLI, and TID) as the NAL unit header as shown in Section 4.3.2, irrespective of the type of the payload structure. For convenience the structure of RTP payload header is described below in Figure 3.

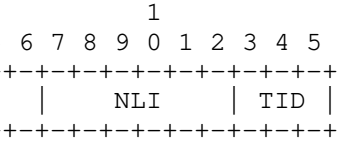


Figure 3: RTP Payload Header

F: nal_forbidden_zero_bit as specified in [ISO.IEC.23090-5] MUST be equal to 0.

NUT: nal_unit_type as specified in [ISO.IEC.23090-5] defines the type of the RBSP data structure contained in the NAL unit payload. NUT value could carry other meaning depending on the RTP packet type.

NLI: nal_layer_id as specified in [ISO.IEC.23090-5] defines the identifier of the layer to which an ACL NAL unit belongs or the identifier of a layer to which a non-ACL NAL unit applies.

TID: nal_temporal_id_plus1 minus 1 as specified in [ISO.IEC.23090-5] defines a temporal identifier for the NAL unit. The value of nal_temporal_id_plus1 MUST NOT be equal to 0.

5.4. Payload structures

5.4.1. General

Three different types of RTP packet payload structures are specified. A receiver can identify the payload structure by the first two bytes of the RTP packet payload, which co-serves as the RTP payload header. These two bytes are always structured as a NAL unit header. The NAL unit type field indicates which structure is present in the payload.

The three different payload structures are as follows:

* Single NAL Unit Packet: Contains a single NAL unit in the payload. This payload structure is specified in Section 5.4.2.

* Aggregation Packet: Contains multiple NAL units in a single RTP payload. This payload structure is specified in Section 5.4.3.

* Fragmentation Unit: Contains a subset of a single NAL unit. This payload structure is specified in Section 5.4.4.

NOTE: (informative) This memo does not limit the size of NAL units encapsulated in NAL unit packets and fragmentation units. [ISO.IEC.23090-5] does not restrict the maximum size of a NAL unit directly either. Instead, a NAL unit sample stream format may be used, which provides flexibility to signal NAL unit size up to UINT64_MAX bytes.

5.4.2. Single NAL unit packet

Single NAL unit packet contains exactly one NAL unit, and consists of an RTP payload header and following conditional fields: 16-bit DONL and 16-bit v3c-tile-id. The rest of the payload data contain the NAL unit payload data (excluding the NAL unit header). Single NAL unit packet MUST only contain atlas NAL units of the types defined in Table 4 of [ISO.IEC.23090-5]. The structure of the single NAL unit packet is shown below in Figure 4.

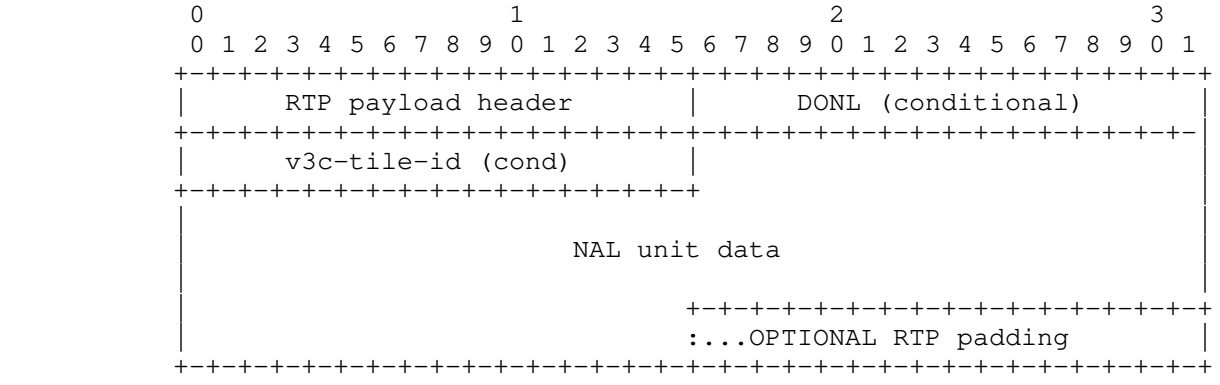


Figure 4: Single NAL unit packet

RTP payload header MUST be an exact copy of the NAL unit header of the contained NAL unit.

A NAL unit stream composed by de-packetizing single NAL unit packets in RTP sequence number order MUST conform to the NAL unit decoding order, when DONL is not present.

The DONL field, when present, specifies the value of the 16-bit decoding order number of the contained NAL unit. If sprop-max-don-diff is greater than 0 for any of the RTP streams, the DONL field MUST be present, and the variable DONL for the contained NAL unit is derived as equal to the value of the DONL field. Otherwise (sprop-max-don-diff is equal to 0 for all the RTP streams), the DONL field MUST NOT be present.

The v3c-tile-id field, when present, specifies the 16-bit tile identifier for the NAL unit, as signalled in V3C atlas tile header defined in [ISO.IEC.23090-5]. If sprop-v3c-tile-id-pres is equal to 1 and RTP payload header NUT is in range 0-35, inclusive, the v3c-tile-id field MUST be present. Otherwise, the v3c-tile-id field MUST NOT be present.

NOTE: (informative) Only values for NAL unit type (NUT) in range 0-35, inclusive, are allocated for atlas tile layer data in [ISO.IEC.23090-5].

5.4.3. Aggregation packet

Aggregation Packets (APs) enable the reduction of packetization overhead for small NAL units, such as most of the non-ACL NAL units, which are often only a few octets in size.

Aggregation packets MAY be used wrap multiple NAL units belonging to the same access unit in a single RTP payload. The first two bytes of the AP MUST contain RTP payload header. The NAL unit type (NUT) for the NAL unit header contained in the RTP payload header MUST be equal to 56, which falls in the unspecified range of the NAL unit types defined in [ISO.IEC.23090-5]. AP MAY contain a conditional v3c-tile-id field. AP MUST contain two or more aggregation units. The structure of AP is shown in Figure 5.

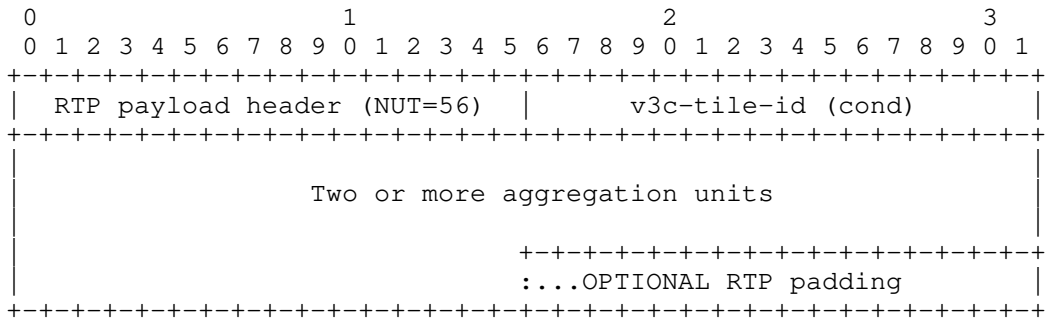


Figure 5: Aggregation Packet (AP)

The fields in the payload header are set as follows. The F bit MUST be equal to 0 if the F bit of each aggregated NAL unit is equal to zero; otherwise, it MUST be equal to 1. The NUT field MUST be equal to 56. The value of NLI MUST be equal to the lowest value of NLI of all the aggregated NAL units. The value of TID MUST be the lowest value of TID of all the aggregated NAL units.

All ACL NAL units in an aggregation packet have the same TID value since they belong to the same access unit. However, the packet MAY contain non-ACL NAL units for which the TID value in the NAL unit header MAY be different than the TID value of the ACL NAL units in the same AP.

The v3c-tile-id field, when present, specifies the 16-bit tile identifier for all ACL NAL units in the AP. If sprop-v3c-tile-id-pres is equal to 1, the v3c-tile-id field MUST be present. Otherwise, the v3c-tile-id field MUST NOT be present.

AP MUST carry at least two aggregation units (AU) and can carry as many aggregation units as necessary. However, the total amount of data in an AP MUST fit into an IP packet, and the size SHOULD be chosen so that the resulting IP packet is smaller than the MTU size so to avoid IP layer fragmentation. The structure of the AU depends both on the presence of the decoding order number, the sequence order of the AU in the AP and the presence of v3c-tile-id field. The structure of an AU is shown in Figure 6.

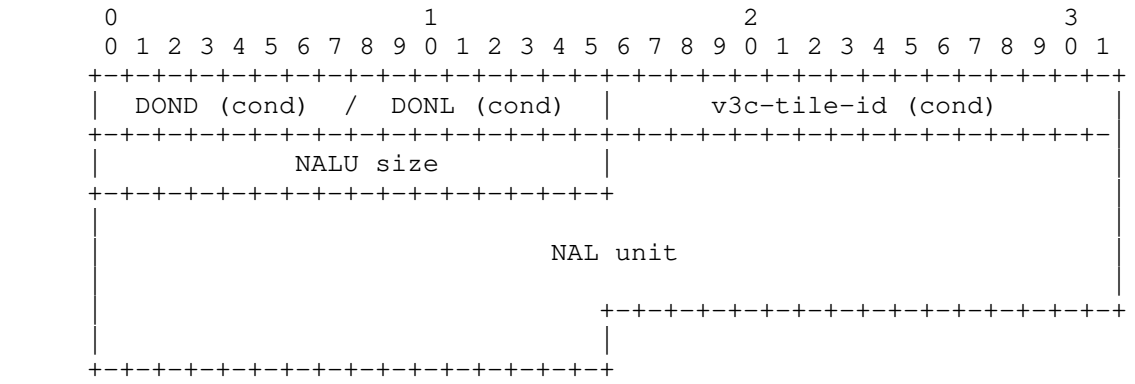


Figure 6: Aggregation Unit (AU)

If sprop-max-don-diff is greater than 0 for any of the RTP streams, an AU begins with the DOND / DONL field. The first AU in the AP contains DONL field, which specifies the 16-bit value of the decoding order number of the aggregated NAL unit. The variable DON for the aggregated NAL unit is derived as equal to the value of the DONL field. All subsequent AUs in the AP MUST contain an (8-bit) DOND field, which specifies the difference between the decoding order number values of the current aggregated NAL unit and the preceding aggregated NAL unit in the same AP. The variable DON for the aggregated NAL unit is derived as equal to the DON of the preceding aggregated NAL unit in the same AP plus the value of the DOND field plus 1 modulo 65536.

When sprop-max-don-diff is equal to 0 for all the RTP streams, DOND / DONL fields MUST NOT be present in an aggregation unit. The aggregation units MUST be stored in the aggregation packet so that the decoding order of the containing NAL units is preserved. This means that the first aggregation unit in the aggregation packet SHOULD contain the NAL unit that SHOULD be decoded first.

If sprop-v3c-tile-id-pres is equal to 2 and the AU NAL unit header type is in range 0-35, inclusive, the 16-bit v3c-tile-id field MUST be present in the aggregation unit after the conditional DOND/DONL field. Otherwise v3c-tile-id field MUST NOT be present in the aggregation unit.

The conditional fields of the aggregation unit are followed by a 16-bit NALU size field, which provides the size of the NAL unit (in bytes) in the aggregation unit. The remainder of the data in the aggregation unit SHOULD contain the NAL unit (including the unmodified NAL unit header).

5.4.4. Fragmentation unit

Fragmentation Units (FUs) are introduced to enable fragmenting a single NAL unit into multiple RTP packets, possibly without co-operation or knowledge of the encoder. A fragment of a NAL unit consists of an integer number of consecutive octets of that NAL unit. Fragments of the same NAL unit MUST be sent in consecutive order with ascending RTP sequence numbers (with no other RTP packets within the same RTP stream being sent between the first and last fragment.

When a NAL unit is fragmented and conveyed within FUs, it is referred to as a fragmented NAL unit. Aggregation packets MUST NOT be fragmented. FUs MUST NOT be nested; i.e., an FU MUST NOT contain a subset of another FU. The RTP header timestamp of an RTP packet carrying an FU is set to the NALU-time of the fragmented NAL unit.

A FU consists of an RTP payload header with NUT equal to 57, an 8-bit FU header, a conditional 16-bit DONL field, a conditional 16-bit v3c-tile-id field and an FU payload. The structure of an FU is illustrated below in Figure 7.

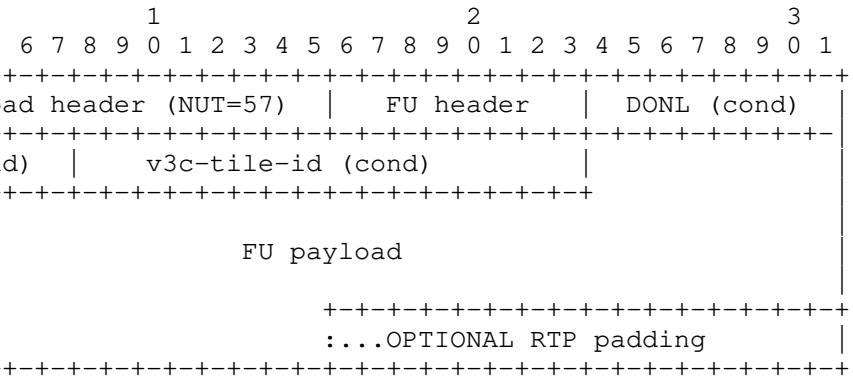


Figure 7: Fragmentation Unit

The fields in the RTP payload header are set as follows. The NUT field MUST be equal to 57. The rest of the fields MUST be equal to the fragmented NAL unit.

The FU header consists of an S bit, an E bit, and a 6-bit FUT field. The structure of FU header is illustrated below in Figure 8.

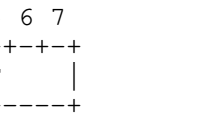


Figure 8: Fragmentation unit header

When set to 1, the S bit indicates the start of a fragmented NAL unit, i.e., the first byte of the FU payload is also the first byte of the payload of the fragmented NAL unit. When the FU payload is not the start of the fragmented NAL unit payload, the S bit MUST be set to 0.

When set to 1, the E bit indicates the end of a fragmented NAL unit, i.e., the last byte of the payload is also the last byte of the fragmented NAL unit. When the FU payload is not the last fragment of a fragmented NAL unit, the E bit MUST be set to 0.

The field FUT MUST be equal to the nal_unit_type field of the fragmented NAL unit.

A non-fragmented NAL unit MUST NOT be transmitted in one FU; i.e., the Start bit and End bit MUST NOT both be set to 1 in the same FU header.

The DONL field, when present, specifies the value of the 16-bit decoding order number of the fragmented NAL unit. If sprop-max-don-diff is greater than 0 for any of the RTP streams, and the S bit is equal to 1, the DONL field MUST be present in the FU, and the variable DON for the fragmented NAL unit is derived as equal to the value of the DONL field. Otherwise (sprop-max-don-diff is equal to 0 for all the RTP streams, or the S bit is equal to 0), the DONL field MUST NOT be present in the FU.

The v3c-tile-id field, when present, specifies the 16-bit tile identifier for the fragmented NAL unit. If sprop-v3c-tile-id-pres is equal to 1, FUT is in range 0-35, and the S bit is equal to 1, the v3c-tile-id field MUST be present after the conditional DONL field. Otherwise, the v3c-tile-id field MUST NOT be present.

The FU payload consists of fragments of the payload of the fragmented NAL unit so that if the FU payloads of consecutive FUs, starting with an FU with the S bit equal to 1 and ending with an FU with the E bit equal to 1, are sequentially concatenated, the payload of the fragmented NAL unit can be reconstructed.

The NAL unit header of the fragmented NAL unit is not included as such in the FU payload, but rather the information of the NAL unit header of the fragmented NAL unit is conveyed in F, NLI, and TID fields of the RTP payload headers of the FUs and the FUT field of the FU header. An FU payload MUST NOT be empty.

If an FU is lost, the receiver SHOULD discard all following fragmentation units in transmission order corresponding to the same fragmented NAL unit, unless the decoder in the receiver is known to be prepared to gracefully handle incomplete NAL units.

5.4.5. Example of fragmentation unit (informative)

This example illustrates how fragmentation unit may be used to divide one NAL unit into two RTP packets. The Figure 9 depicts the structure of the first packet with the first part of the fragmented NAL unit.

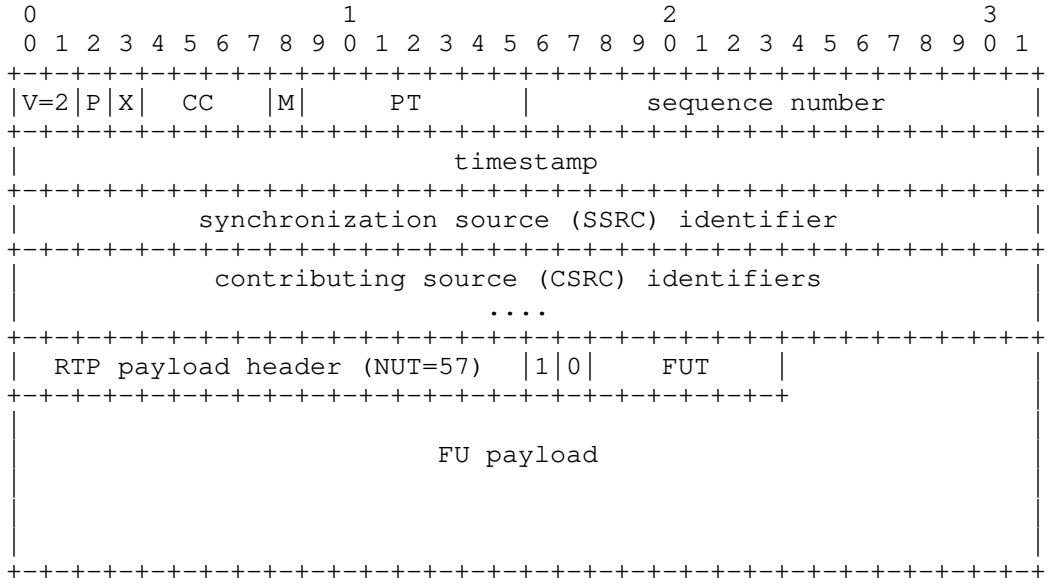


Figure 9: First packet of fragmented NAL unit

The Figure 10 visualizes the structure of the second packet with the rest of the fragmented NAL unit.

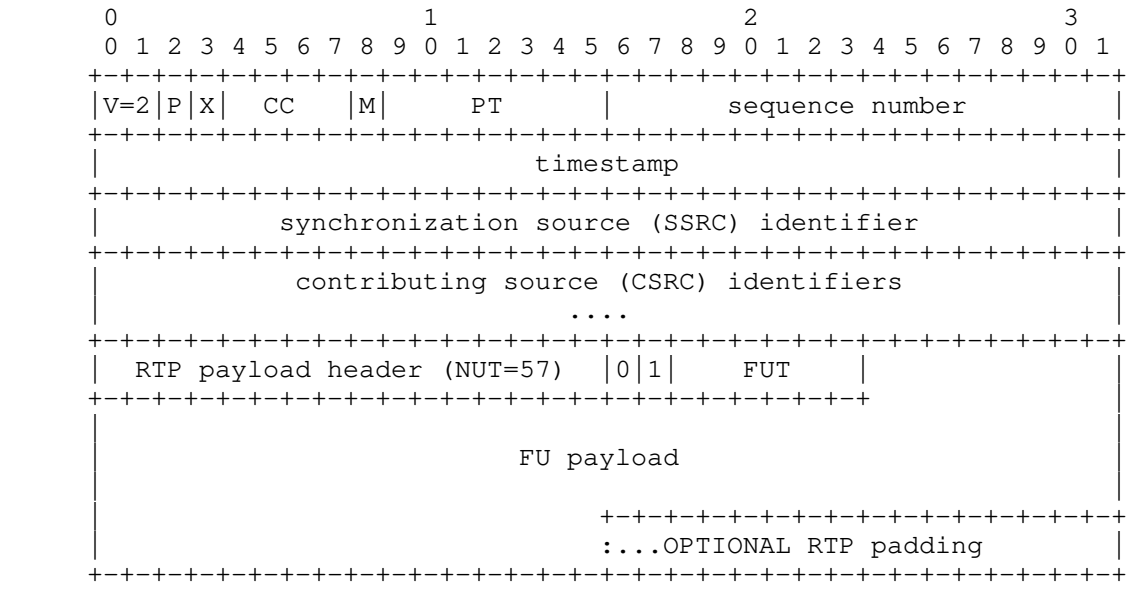


Figure 10: Second packet of fragmented NAL unit

5.5. Decoding order number

For each atlas NAL unit, the variable AbsDon is derived, representing the decoding order number that is indicative of the NAL unit decoding order. Let NAL unit n be the n-th NAL unit in transmission order within an RTP stream.

If sprop-max-don-diff is equal to 0 for all the RTP streams carrying the atlas bitstream, AbsDon[n], the value of AbsDon for NAL unit n, is derived as equal to n.

Otherwise (sprop-max-don-diff is greater than 0 for any of the RTP streams), AbsDon[n] is derived as follows, where DON[n] is the value of the variable DON for NAL unit n:

```
If (n == 0)
  AbsDon[n] = DON[0]
Else
  If (DON[n] == DON[n-1])
    AbsDon[n] = AbsDon[n-1]
  If (DON[n] > DON[n-1] and DON[n] - DON[n-1] < 32768)
    AbsDon[n] = AbsDon[n-1] + DON[n] - DON[n-1]
  If (DON[n] < DON[n-1] and DON[n-1] - DON[n] >= 32768)
    AbsDon[n] = AbsDon[n-1] + 65536 - DON[n-1] + DON[n]
  If (DON[n] > DON[n-1] and DON[n] - DON[n-1] >= 32768)
    AbsDon[n] = AbsDon[n-1] - (DON[n-1] + 65536 - DON[n])
  If (DON[n] < DON[n-1] and DON[n-1] - DON[n] < 32768)
    AbsDon[n] = AbsDon[n-1] - (DON[n-1] - DON[n])
```

For any two NAL units m and n, the following applies:

* AbsDon[n] greater than AbsDon[m] indicates that NAL unit n follows NAL unit m in NAL unit decoding order.

* When AbsDon[n] is equal to AbsDon[m], the NAL unit decoding order of the two NAL units can be in either order.

* AbsDon[n] less than AbsDon[m] indicates that NAL unit n precedes NAL unit m in decoding order.

6. Packetization and de-packetization rules

The following packetization rules apply for V3C atlas data:

* If sprop-max-don-diff is greater than 0 for any of the RTP streams, the transmission order of NAL units carried in the RTP stream MAY be different than the NAL unit decoding order and the NAL unit output order. Otherwise (sprop-max-don-diff is equal to 0 for all the RTP streams), the transmission order of NAL units carried in the RTP stream MUST be the same as the NAL unit decoding order.

* A NAL unit of a small size SHOULD be encapsulated in an aggregation packet together with one or more other NAL units in order to avoid the unnecessary packetization overhead for small NAL units. For example, non-ACL NAL units such as access unit delimiters, parameter sets, or SEI NAL units are typically small and can often be aggregated with ACL NAL units without violating MTU size constraints.

* Each non-ACL NAL unit SHOULD, when possible, from an MTU size perspective, be encapsulated in an aggregation packet together with its associated ACL NAL unit, as typically a non-ACL NAL unit would be meaningless without the associated ACL NAL unit being available.

* For carrying exactly one NAL unit in an RTP packet, a single NAL unit packet MUST be used

The general concept behind de-packetization is to get the NAL units out of the RTP packets in an RTP stream and all RTP streams the RTP stream depends on, if any, and pass them to the decoder in the NAL unit decoding order.

The de-packetization process is implementation dependent. Therefore, the following de-packetization rules SHOULD be taken as an example.

* All normal RTP mechanisms related to buffer management apply. In particular, duplicated or outdated RTP packets (as indicated by the RTP sequence number and the RTP timestamp) are removed. To determine the exact time for decoding, factors such as a possible intentional delay to allow for proper inter-stream synchronization must be factored in.

* NAL units with NAL unit type values in the range of 0 to 55, inclusive, MAY be passed to the decoder. NAL-unit-like structures with NAL unit type values in the range of 56 to 63, inclusive, MUST NOT be passed to the decoder.

* When sprop-max-don-diff is equal to 0 for the received RTP stream, the NAL units carried in the RTP stream MAY be directly passed to the decoder in their transmission order, which is identical to their decoding order.

* When sprop-max-don-diff is greater than 0 for any of the received RTP streams, the received NAL units need to be arranged into decoding order before handing them over to the decoder.

* For further de-packetization examples, the reader is referred to Section 6 of [RFC7798].

Regarding the packetization of V3C video component data, the respective RTP video payload specification(s) define how packetization and de-packetization SHOULD be handled.

7. Payload format parameters

This section specifies the optional parameters. 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.

7.1. Media type registration

The receiver MUST ignore any parameter unspecified in this memo.

Type name: application

Subtype name: v3c

Required parameters: N/A

Optional parameters: sprop-v3c-unit-header, sprop-v3c-unit-type, sprop-v3c-vps-id, sprop-v3c-atlas-id, sprop-v3c-attr-idx, sprop-v3c-attr-part-idx, sprop-v3c-map-idx, sprop-v3c-aux-video-flag, sprop-v3c-parameter-set, sprop-v3c-tile-id, sprop-v3c-tile-id-pres, sprop-v3c-atlas-data, sprop-v3c-common-atlas-data, sprop-v3c-sei, v3c-ptl-level-idc, v3c-ptl-tier-flag, v3c-ptl-codec-idc, v3c-ptl-toolset-idc, v3c-ptl-rec-idc and sprop-max-don-diff.

Encoding considerations: This type is only defined for transfer via RTP [RFC3550].

Security considerations: Please see Section 11.

Interoperability considerations: N/A

Published specification: Please refer to [ISO.IEC.23090-5]

Applications that use this media type: Any application that relies on V3C-based media services over RTP

Additional information: N/A

Person & email address to contact for further information:

Intended usage: COMMON

Restrictions on usage: N/A

Author: See Authors' Addresses section of this memo.

Change controller: IETF avtcore@ietf.org (mailto:avtcore@ietf.org)

Provisional registration? (standards tree only): No

7.2. Optional parameters definition

sprop-v3c-unit-header:

provides a V3C unit header bytes defined in [ISO.IEC.23090-5]. The value contains base64 encoded [RFC4648] representation of the 4 bytes of V3C unit header.

sprop-v3c-unit-type:

sprop-v3c-unit-type provides a V3C unit type value corresponding to vuh_unit_type defined in [ISO.IEC.23090-5], i.e., defines V3C sub-bitstream type.

sprop-v3c-vps-id:

sprop-v3c-vps-id provides a value corresponding to vuh_v3c_parameter_set_id defined in [ISO.IEC.23090-5].

sprop-v3c-atlas-id:

sprop-v3c-atlas-id provides a value corresponding to vuh_atlas_id defined in [ISO.IEC.23090-5].

sprop-v3c-attr-idx:

sprop-v3c-attr-idx provides a value corresponding to vuh_attribute_index defined in [ISO.IEC.23090-5].

sprop-v3c-attr-part-idx:

sprop-v3c-attr-part-idx provides a value corresponding to vuh_attribute_partition_index defined in [ISO.IEC.23090-5].

sprop-v3c-map-idx:

sprop-v3c-map-idx provides a value corresponding to vuh_map_index defined in [ISO.IEC.23090-5].

sprop-v3c-aux-video-flag:

sprop-v3c-aux-video-flag provides a value corresponding to vuh_auxiliary_video_flag defined in [ISO.IEC.23090-5].

sprop-v3c-parameter-set:

sprop-v3c-parameter-set provides V3C parameter set bytes as defined in [ISO.IEC.23090-5]. The value contains base64 encoded [RFC4648] representation of the V3C parameter set bytes.

sprop-v3c-tile-id:

sprop-v3c-tile-id indicates that the RTP stream contains only portion of the tiles in the atlas. sprop-v3c-tile-id is a comma-separated (',') list of integer values, which indicate the sprop-v3c-tile-ids that are present in the RTP stream.

sprop-v3c-tile-id-pres:

sprop-v3c-tile-id-pres indicates that the RTP packets contain v3c-tile-id field.

sprop-v3c-atlas-data:

sprop-v3c-atlas-data MAY be used to convey any atlas data NAL units of the V3C atlas sub bitstream for out-of-band transmission. The value is a comma-separated (',') list of encoded representations of the atlas NAL units as specified in [ISO.IEC.23090-5]. The NAL units SHOULD be encoded as base64 [RFC4648] representations.

sprop-v3c-common-atlas-data:

sprop-v3c-common-atlas-data MAY be used to convey common atlas data NAL units of the V3C common atlas sub bitstream for out-of-band transmission. The value is a comma-separated (',') list of encoded representations of the common atlas NAL units (i.e., NAL_CASPS and NAL_CAF_IDR) as specified in [ISO.IEC.23090-5]. The NAL units SHOULD be encoded as base64 [RFC4648] representations.

sprop-v3c-sei:

sprop-v3c-sei MAY be used to convey SEI NAL units of V3C atlas and common atlas sub bitstreams for out-of-band transmission. The value is a comma-separated (',') list of encoded representations of SEI NAL units (i.e., NAL_PREFIX_NSEI and NAL_SUFFIX_NSEI, NAL_PREFIX_ESEI, NAL_SUFFIX_ESEI) as specified in [ISO.IEC.23090-5]. The SEI NAL units SHOULD be encoded as base64 [RFC4648] representations.

v3c-ptl-level-idc:

v3c-ptl-level-idc provides a value corresponding to ptl_level_idc defined in [ISO.IEC.23090-5].

v3c-ptl-tier-flag:

v3c-ptl-tier-flag provides a value corresponding to ptl_tier_flag defined in [ISO.IEC.23090-5].

v3c-ptl-codec-idc:

v3c-ptl-codec-idc provides a value corresponding to ptl_profile_codec_group_idc defined in [ISO.IEC.23090-5].

v3c-ptl-toolset-idc:

v3c-ptl-toolset-idc provides a value corresponding to ptl_profile_toolset_idc defined in [ISO.IEC.23090-5].

v3c-ptl-rec-idc:

v3c-ptl-rec-idc provides a value corresponding to ptl_profile_reconstruction_idc defined in [ISO.IEC.23090-5].

sprop-max-don-diff:

If the transmission order of NAL units in the RTP stream(s) is the same as the decoding and NAL unit output order, this parameter must be equal to 0.

Otherwise, if the decoding order of the NAL units of the RTP stream(s) is the same as the NAL unit transmission order but not the same as NAL unit output order, the value of this parameter MUST be equal to 1.

Otherwise, this parameter specifies the maximum absolute difference between the decoding order number (i.e., AbsDon) values of any two NAL units naluA and naluB, where naluA follows naluB in decoding order and precedes naluB in transmission order.

The value of sprop-max-don-diff MUST be an integer in the range of 0 to 32767, inclusive.

When not present, the value of sprop-max-don-diff is inferred to be equal to 0.

8. Congestion control considerations

Congestion control for RTP SHALL be used in accordance with [RFC3550], and with any applicable RTP profile: e.g., [RFC3551]. An additional requirement if best-effort service is being used is users of this payload format MUST monitor packet loss to ensure that the packet loss rate is within acceptable parameters.

Simple bitrate adaptation for congestion control can be achieved when real-time coding is used for V3C video components, where quality parameter can be adaptively tuned. Video coding specifications MAY define further adaptation techniques.

Circuit Breakers [RFC8083] is an update to RTP [RFC3550] that defines criteria for when one is required to stop sending RTP Packet Streams. The circuit breakers is to be implemented and followed.

9. Session description protocol

A new attribute "v3cfmtp" is defined for carrying V3C format media type parameters in the corresponding fields of the Session Description Protocol (SDP). Grouping framework [RFC5888] is used to indicate which media lines (video and application) in the SDP constitute a V3C representation.

9.1. V3C format parameters "v3cfmtp" attribute

This memo defines a new attribute for SDP, intended to carry V3C specific media format parameters. Its functionality is similar to "a=fmtp", with the exception that it SHALL be used without the fmt-token and that it can be used also on a session level. The attribute allows to associate V3C specific media format parameters with any media line in SDP.

Contact name: See Authors' Addresses section of this memo.

Contact email address: See Authors' Addresses section of this memo.

Attribute name: v3cfmtp

Attribute value: v3cfmtp-value

Attribute syntax:

```
v3cfmtp-value = byte-string
; Notes:
; - The V3C format parameters are V3C media type parameters and
;   need to reflect their syntax.
```

Attribute semantics: "v3cfmtp-value" is a byte-string, which MUST contain at least one V3C specific media format parameter as defined in this memo. Multiple semicolon separated V3C media parameters MAY be stored in "v3c-format-specific-params" to be conveyed by SDP and given unchanged to the media tool that will use this format.

Usage level: session, media

Charset dependent: no

Purpose: This attribute allows parameters that are specific to a V3C format to be conveyed in a way that SDP does not have to understand them. It allows to associate V3C specific parameters with the session or with any media line. Parameters signalled as part of session level attribute take effect when conflicting parameters are signalled as media level attribute.

O/A procedures: v3cfmtp attribute MAY be present both in offers and answers.

Mux Category: NORMAL

Reference: "this memo"

NOTE: (informative) "this memo" to be replaced with the RFC number, once it becomes available.

Example: First line describes session level usage of the attribute, signaling a V3C parameter set. Second line describes media level attribute, signaling V3C unit header and profile tier level flag for the associated media line.

```
a=v3cfmtp:sprop-v3c-parameter-set=AUH/AAAP/zwAAAAAACgIAtEAgQLAIAAUQ
BACWAM5QEDgQCAIAAAAABP8CzwAAAAAAAAQAAAtAE/wLPAAAAAAAg=;

a=v3cfmtp:sprop-v3c-unit-header=CAAAAA==;v3c-ptl-tier-flag=1;
```

9.2. Mapping of payload type parameters to SDP

9.2.1. For V3C atlas components

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

* The encoding name in the "a=rtpmap" line of SDP MUST be v3c

* The clock rate in the "a=rtpmap" line MUST be 90000.

* The OPTIONAL parameters sprop-v3c-atlas-data, sprop-v3c-common-atlas-data, sprop-v3c-sei, sprop-v3c-tile-id, sprop-v3c-tile-id-pres, when present, MUST be included in the "a=fmtp" line of SDP. This parameter is expressed as a media type string, in the form of a semicolon-separated list of parameter=value pairs.

* The OPTIONAL parameters sprop-v3c-unit-header, sprop-v3c-unit-type, sprop-v3c-vps-id, sprop-v3c-atlas-id, sprop-v3c-attr-idx, sprop-v3c-attr-part-idx, sprop-v3c-map-idx, sprop-v3c-aux-video-flag, sprop-max-don-diff, sprop-v3c-parameter-set, v3c-ptl-level-idc, v3c-ptl-tier-flag, v3c-ptl-codec-idc, v3c-ptl-toolset-idc, v3c-ptl-rec-idc, when present, MUST be included in the "a=v3cfmtp" line of SDP. This parameter is expressed as a media type string, in the form of a semicolon-separated list of parameter=value pairs.

The OPTIONAL parameters, when present in the V3C atlas component media line format parameters attribute, specify values that are valid for the coded V3C sequence until a new value is received in-band. Some OPTIONAL parameters, like sprop-v3c-parameter-set or sprop-v3c-unit-header, can't be carried in-band in the atlas stream and may thus be considered static for the session. The carriage of V3C payload format parameters in "a=fmtp" and "a=v3cfmtp" attributes is separated by logical context, where "a=fmtp" consists atlas level media format parameters and "a=v3cfmtp" contains V3C level media format parameters.

An example of media representation corresponding to atlas data component (V3C_AD), where static V3C parameter set and V3C unit header is carried out-of-band in SDP, is as follows:

```
m=application 49170 RTP/AVP 98
a=rtpmap:98 v3c/90000
a=fmtp:98 sprop-v3c-tile-id=0,1
a=v3cfmtp:sprop-v3c-unit-header=CAAAAA==;v3c-ptl-tier-flag=1;
  sprop-v3c-parameter-set=AQD/AAAP/zwAAAAAADwIAQ5BwAAOADjgQAADkA==
```

9.2.2. For V3C video components

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

* The encoding name in the "a=rtpmap" line of SDP can be any video subtype, e.g., H.264, H.265, H.266 etc.

* The clock rate in the "a=rtpmap" line MUST be 90000.

* The OPTIONAL parameters sprop-v3c-unit-header, sprop-v3c-unit-type, sprop-v3c-vps-id, sprop-v3c-atlas-id, sprop-v3c-attr-idx, sprop-v3c-attr-part-idx, sprop-v3c-map-idx, sprop-v3c-aux-video-flag, sprop-max-don-diff, sprop-v3c-parameter-set, sprop-v3c-atlas-data, sprop-v3c-common-atlas-data, sprop-v3c-sei, sprop-v3c-tile-id, sprop-v3c-tile-id-pres, v3c-ptl-level-idc, v3c-ptl-tier-flag, v3c-ptl-codec-idc, v3c-ptl-toolset-idc, v3c-ptl-rec-idc, when present, MUST be included in the "a=v3cfmtp" line of SDP. This parameter is expressed as a media type string, in the form of a semicolon-separated list of parameter=value pairs.

The OPTIONAL parameters, when present in the video media line V3C format parameters ("v3cfmtp") attribute, specify values that are considered static for the session.

An example of media representation corresponding to occupancy video component (V3C_OVD) in SDP is as follows:

```
m=video 49170 RTP/AVP 99
a=rtpmap:99 H265/90000
a=fmtp:99 sprop-max-don-diff=0;
a=v3cfmtp:sprop-v3c-unit-header=EAAAAA==
```

Below is an example of media representation corresponding to packed video component (V3C_PVD), where static V3C parameter set, atlas data and common atlas data are carried out-of-band in SDP. The values are considered static for the session, as they can’t be signaled in-band in the video stream.

```
m=video 49170 RTP/AVP 99
a=rtpmap:99 H265/90000
a=v3cfmtp:sprop-v3c-unit-header=KAAAAA==;
  sprop-v3c-parameter-set=AUH/AAAP/zwAAAAAACgIAtEAgQLAIAAUQBACWAM5Q
  EDgQCAIAAAAABP8CzwAAAAAAAAQAAAtAE/wLPAAAAAAAg=;
  sprop-v3c-atlas-data=SAGAFAQBaKjuXgABQEKA,SgHmIA==,LgFoDOAFAABaAA
  AAAAA+;
  sprop-v3c-common-atlas-data=YAEHgFA=,YgEAMAAAC/B0qcvv/Dbr/pTvb8oq
  fhC5JQVS9jn7kAQT/As9EFyrjRBcmxEQe+j5DuGbTT9mZmZAQAAAoA==
```

9.3. Grouping framework

Different V3C components MAY be represented by their own respective RTP streams, whose payload formats are defined in the respective specifications. V3C atlas data RTP payload format is defined in this memo, whereas the video component RTP payload formats are defined for example in [RFC6184] or [RFC7798]. A grouping tool, as defined in [RFC5888], is extended to idicate which media lines constitute a V3C representation.

Group attribute with V3C type is provided to allow application to identify "m" lines that belong to the same V3C bitstream. Grouping type V3C MUST be used with the group attribute. The tokens that follow are mapped to 'mid'-values of individual media lines in the SDP.

```
a=group:V3C <tokens>
```

The following example shows an SDP including four media lines, three describing V3C video components (PT:96=occupancy, PT:97=geometry, PT:98=attribute) and one V3C atlas component (PT:100). All the media lines are grouped under one V3C group. V3C parameter set is provided via session level V3C media format parameter attribute.

```
...
a=group:V3C 1 2 3 4
a=v3cfmtp:sprop-v3c-parameter-set=AQD/AAAP/zwAAAAAADwIAQ5BwAAOADjgQ
  AADkA==
m=video 40000 RTP/AVP 96
a=rtpmap:96 H264/90000
a=v3cfmtp:sprop-v3c-unit-header=EAAAAA==
a=mid:1
m=video 40002 RTP/AVP 97
a=rtpmap:97 H264/90000
a=v3cfmtp:sprop-v3c-unit-header=GAAAAA==
a=mid:2
m=video 40004 RTP/AVP 98
a=rtpmap:98 H264/90000
a=v3cfmtp:sprop-v3c-unit-header=IAAAAA==
a=mid:3
m=application 40008 RTP/AVP 100
a=rtpmap:100 v3c/90000
a=v3cfmtp:sprop-v3c-unit-header=CAAAAA==;
a=mid:4
```

The example below describes how content with two atlases can be signalled as separate streams. V3C parameter set is carried in a session level V3C media format parameter attribute and common atlas data are carried as part of the media level V3C media format parameter attribute corresponding to atlas zero. PT equal to 96, 97, 98 and 100 correspond to occupancy, geometry and attribute video component as well as atlas data component for atlas zero. PT equal to 101, 102, 103 and 104 correspond to respective components for atlas one.

```
...
a=group:V3C 1 2 3 4 5 6 7 8
a=v3cfmtp:sprop-v3c-parameter-set=AAUH/AAAP/zwAAABAADwIAWhBwAAOADjg
  QAADgAA8CAFoQcAADgA44EAAA6AkAgABRIA=;
m=video 40000 RTP/AVP 96
a=rtpmap:96 H264/90000
a=v3cfmtp:sprop-v3c-unit-header=EAAAAA==
a=mid:1
m=video 40002 RTP/AVP 97
a=rtpmap:97 H264/90000
a=v3cfmtp:sprop-v3c-unit-header=GAAAAA==
a=mid:2
m=video 40004 RTP/AVP 98
a=rtpmap:98 H264/90000
a=v3cfmtp:sprop-v3c-unit-header=IAAAAA==
a=mid:3
m=application 40008 RTP/AVP 100
a=rtpmap:100 v3c/90000
a=fmtp:100
  sprop-v3c-common-atlas-data=YAEHgFA=,YgEAMAAAa+96Z5v6VP1D+P7LzRsb
  WDJ/yz+ALzMZNfvCg2389Kjd+d6fZyM6QZBfhrDW3K0vaP2Rr8L+gLAq/ny3wAzs9
  veiXEjjS67MfH+H4xV/RgW4fkl/YkINe/OsWCOBwPAVLACCf4FnogwYZKIME6oiD9
  UCodqjLwCCf4FnogxqBiIMZNwiEBpJIduBUoCCf4FnogwOeSIMCaGiEA9VIdtGwwC
  Cf4FnogvB+aILvWIiEBB6IdqobKfmZmZoCmZmefmZmZoCmZmefmZmZoCmZmefmZmZ
  oCmZmdA=
a=v3cfmtp:sprop-v3c-unit-header=CAAAAA==;
a=mid:4
m=video 40010 RTP/AVP 101
a=rtpmap:101 H264/90000
a=v3cfmtp:sprop-v3c-unit-header=EAIAAA==
a=mid:5
m=video 40012 RTP/AVP 102
a=rtpmap:102 H264/90000
a=v3cfmtp:sprop-v3c-unit-header=GAIAAA==
a=mid:6
m=video 40014 RTP/AVP 103
a=rtpmap:103 H264/90000
a=v3cfmtp:sprop-v3c-unit-header=IAIAAA==
a=mid:7
m=application 40018 RTP/AVP 104
a=rtpmap:104 v3c/90000
a=v3cfmtp:sprop-v3c-unit-header=CAIAAA==
a=mid:8
```

9.4. Offer and answer considerations

V3C coded content consist of metadata, i.e. atlas data and the video coded bitstreams represented as separate media lines in the SDP (unless packed video is used Section 9.2.2). During the session negotiation offerer lists different V3C components and informs which media lines SHOULD be consumed together. The receiver CAN select media components as suggested by the offer or select a subset of the components and formulate the answer accordingly. This freedom allows the receiver to consume a subset of the V3C coded media in scenarios, where it is fully or partially ignorant of the V3C coding scheme.

An example of offer which only sends V3C content. The following example contains video components as three different versions (H.264, H.265, H.266). Further differences between the alternatives would be signaled as part of the media attribute parameters, as is the practice with regular video streams.

```
...
a=group:v3c 1 2 3 4
a=v3cfmtp:v3c-ptl-level-idc=60;
  sprop-v3c-parameter-set=AQD/AAAP/zwAAAAAADwIAQ5BwAAOADjgQAADkA==
m=video 40000 RTP/AVP 96 97 98
a=rtpmap:96 H264/90000
a=rtpmap:97 H265/90000
a=rtpmap:98 H266/90000
a=v3cfmtp:sprop-v3c-unit-type=2;sprop-v3c-vps-id=0;
  sprop-v3c-atlas-id=0
a=sendonly
a=mid:1
m=video 40002 RTP/AVP 99 100 101
a=rtpmap:99 H264/90000
a=rtpmap:100 H265/90000
a=rtpmap:101 H266/90000
a=v3cfmtp:sprop-v3c-unit-type=3;sprop-v3c-vps-id=0;
  sprop-v3c-atlas-id=0
a=mid:2
a=sendonly
m=video 40004 RTP/AVP 102 103 104
a=rtpmap:102 H264/90000
a=rtpmap:103 H265/90000
a=rtpmap:104 H266/90000
a=v3cfmtp:sprop-v3c-unit-type=4;sprop-v3c-vps-id=0;
  sprop-v3c-atlas-id=0
a=mid:3
a=sendonly
m=application 40006 RTP/AVP 105
a=rtpmap:105 v3c/90000
a=v3cfmtp:sprop-v3c-unit-type=1;sprop-v3c-vps-id=0;
  sprop-v3c-atlas-id=0;
a=mid:4
a=sendonly
```

An example of answer which only receives V3C data with the selected versions.

```
...
a=group:v3c 1 2 3 4
m=video 50000 RTP/AVP 96
a=rtpmap:96 H264/90000
a=recvonly
a=mid:1
m=video 50002 RTP/AVP 100
a=rtpmap:100 H265/90000
a=recvonly
a=mid:2
m=video 50004 RTP/AVP 104
a=rtpmap:104 H266/90000
a=recvonly
a=mid:3
m=application 50006 RTP/AVP 105
a=rtpmap:105 v3c/90000
a=recvonly
a=mid:4
```

An example offer, which allows bundling different V3C components into one stream, based on [RFC9143].

```
...
a=group:BUNDLE 1 2 3 4
a=group:v3c 1 2 3 4
m=video 40000 RTP/AVP 96
a=rtpmap:96 H264/90000
a=v3cfmtp:sprop-v3c-unit-type=2;sprop-v3c-vps-id=0;
  sprop-v3c-atlas-id=0
a=mid:1
a=extmap:1 urn:ietf:params:rtp-hdrext:sdes:mid
m=video 40002 RTP/AVP 97
a=rtpmap:97 H264/90000
a=v3cfmtp:sprop-v3c-unit-type=3;sprop-v3c-vps-id=0;
  sprop-v3c-atlas-id=0;
a=mid:2
a=extmap:1 urn:ietf:params:rtp-hdrext:sdes:mid
m=video 40004 RTP/AVP 98
a=rtpmap:98 H264/90000
a=v3cfmtp:sprop-v3c-unit-type=4;sprop-v3c-vps-id=0;
  sprop-v3c-atlas-id=0
a=mid:3
a=extmap:1 urn:ietf:params:rtp-hdrext:sdes:mid
m=application 40006 RTP/AVP 99
a=rtpmap:99 v3c/90000
a=v3cfmtp:sprop-v3c-unit-type=1;sprop-v3c-vps-id=0;
  sprop-v3c-atlas-id=0;
  sprop-v3c-parameter-set=AQD/AAAP/zwAAAAAADwIAQ5BwAAOADjgQAADkA==
a=mid:4
a=extmap:1 urn:ietf:params:rtp-hdrext:sdes:mid
```

An example answer, which accepts bundling of different V3C components.

```
a=group:BUNDLE 1 2 3 4
a=group:v3c 1 2 3 4
m=video 50000 RTP/AVP 96
a=rtpmap:96 H264/90000
a=mid:1
a=extmap:1 urn:ietf:params:rtp-hdrext:sdes:mid
m=video 0 RTP/AVP 97
a=rtpmap:97 H264/90000
a=bundle-only
a=mid:2
a=extmap:1 urn:ietf:params:rtp-hdrext:sdes:mid
m=video 0 RTP/AVP 98
a=rtpmap:98 H264/90000
a=bundle-only
a=mid:3
a=extmap:1 urn:ietf:params:rtp-hdrext:sdes:mid
m=application 0 RTP/AVP 99
a=rtpmap:99 v3c/90000
a=bundle-only
a=mid:4
a=extmap:1 urn:ietf:params:rtp-hdrext:sdes:mid
```

9.5. Declarative SDP considerations

When V3C 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 v3c-ptl-level-idc, v3c-ptl-tier-flag, v3c-ptl-codec-idc, v3c-ptl-toolset-idc and v3c-ptl-rec-idc 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.

10. IANA considerations

This memo contains three considerations to IANA: new media type, new SDP attribute and new grouping type.

10.1. V3C media type registration

A new media type will be registered with IANA; see Section Section 7.1.

10.2. V3C format parameters SDP attribute

This document defines a new session and media level SDP attribute: "v3cfmtp". This attribute will be registered by IANA under attribute-field names (<attribute-name>) registry in Session Description Protocol (SDP). The "v3cfmtp" attribute is used to convey V3C specific media format parameters for any media line. Its format is defined in Section Section 9.1. Further semantics are provided in Table 2.

Type	SDP Name	Usage Level	Mux Category	Reference
attribute	v3cfmtp	session, media	NORMAL	"this memo"

Table 2: Additional details for <attribute-name> Registry

NOTE: (informative) "this memo" to be replaced with the RFC number, once it becomes available.

10.3. V3C grouping type extension

Grouping is extended to establish relationships between substreams of a V3C representation. A new group type (V3C) for the group attribute will be registered as defined in Section 9.3. This document registers the semantics in Table 3 with IANA in the "Semantics for the ’group’ SDP Attribute" subregistry (under the "Session Description Protocol (SDP) Parameters" registry):

Semantics	Token	Mux Category	Reference
V3C grouping	V3C	NORMAL	"this memo"

Table 3: Additional semantics for V3C SDP group type

NOTE: (informative) "this memo" to be replaced with the RFC number, once it becomes available.

11. Security considerations

RTP packets using the payload format defined in this specification are subject to the security considerations discussed in the RTP specification [RFC3550], and in any 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 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 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 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.

12. References

12.1. Normative References

[ISO.IEC.23090-12]
ISO/IEC, "Information technology --- Coded representation of immersive media --- Part 12: MPEG Immersive video (MIV)", ISO/IEC 23090-12, 2022, <https://www.iso.org/standard/79113.html>.

[ISO.IEC.23090-5]
ISO/IEC, "Information technology --- Coded representation of immersive media --- Part 5: Visual volumetric video-based coding (V3C) and video-based point cloud compression (V-PCC)", ISO/IEC 23090-5, 2021, <https://www.iso.org/standard/73025.html>.

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

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

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

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

[RFC4585] Ott, J., Wenger, S., Sato, N., Burmeister, C., and J. Rey, "Extended RTP Profile for Real-time Transport Control Protocol (RTCP)-Based Feedback (RTP/AVPF)", RFC 4585, DOI 10.17487/RFC4585, July 2006, <https://www.rfc-editor.org/info/rfc4585>.

[RFC4648] Josefsson, S., "The Base16, Base32, and Base64 Data Encodings", RFC 4648, DOI 10.17487/RFC4648, October 2006, <https://www.rfc-editor.org/info/rfc4648>.

[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>.

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

[RFC6184] Wang, Y.-K., Even, R., Kristensen, T., and R. Jesup, "RTP Payload Format for H.264 Video", RFC 6184, DOI 10.17487/RFC6184, May 2011, <https://www.rfc-editor.org/info/rfc6184>.

[RFC6190] Wenger, S., Wang, Y.-K., Schierl, T., and A. Eleftheriadis, "RTP Payload Format for Scalable Video Coding", RFC 6190, DOI 10.17487/RFC6190, May 2011, <https://www.rfc-editor.org/info/rfc6190>.

[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>.

[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>.

[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>.

[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>.

12.2. Informative References

[ISO.IEC.14496-10]
ISO/IEC, "Information technology - Coding of audio-visual objects - Part 10: Advanced video coding", ISO/IEC 14496-10, 2020, <https://www.iso.org/standard/75400.html>.

[ISO.IEC.14496-12]
ISO/IEC, "Information technology --- Coding of audio-visual objects --- Part 12: ISO base media file format", ISO/IEC 14496-12, 2020, <https://www.iso.org/standard/74428.html>.

[ISO.IEC.23008-2]
ISO/IEC, "Information technology --- High efficiency coding and media delivery in heterogeneous environments --- Part 2: High efficiency video coding", ISO/IEC 23008-2, 2020, <https://www.iso.org/standard/75484.html>.

[ISO.IEC.23090-10]
ISO/IEC, "Information technology --- Coded representation of immersive media --- Part 10: Carriage of visual volumetric video-based coding data", ISO/IEC FDIS 23090-10, 2022, <https://www.iso.org/standard/78991.html>.

[ISO.IEC.23090-3]
ISO/IEC, "Information technology --- Coded representation of immersive media --- Part 3: Versatile video coding", ISO/IEC 23090-3, 2021, <https://www.iso.org/standard/73022.html>.

[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>.

Authors' Addresses

Lauri Ilola
Nokia Technologies
Hatanpaeaen valtatie 30
FI-33100 Tampere
Finland
Email: lauri.ilola@nokia.com

Lukasz Kondrad
Nokia Technologies
Werinherstrasse 91
D-81541 Munich
Germany
Email: lukasz.kondrad@nokia.com