Audio/Video Transport Core Maintenance
Internet-Draft
Intended status: Experimental
Expires: 21 September 2025

M. Engelbart
J. Ott
Technical University of Munich
S. Dawkins
Tencent America LLC
20 March 2025

RTP over QUIC (RoQ)
draft-ietf-avtcore-rtp-over-quic-14

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

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].

Packet: The term "packet" is ambiguous. Without a qualifier, "packet" could refer to a UDP packet, or a QUIC packet, or an RTP/RTCP packet encapsulated in QUIC, or a media packet encapsulated in RTP. If not explicitly qualified, the term "packet" in this document refers to a QUIC packet.

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 RTP 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 an RTP packet is no longer needed and knows that the RTP 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 RTP 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.

5.4. Encapsulation Considerations for RTCP

The same encapsulation as described above for RTP packets can also be used to carry RTCP packets back from the receiver to the sender. Both RTP and RTCP can be transported in either QUIC DATAGRAM frames or QUIC STREAM frames. If a receiver sends aggregated RTCP reports for multiple RTP streams, the flow identifier no longer matches the flow identifier for a single RTP stream. Thus the sender always needs to inspect the received RTCP packet independent of the flow identifier used to the RTCP flow to determine to which of the RTP flows the received packets apply. This is also the reason why bidirectional streams are not allowed, as the received RTCP packets would not necessarily apply to the same RTP stream being sent on the same flow. In addition, allowing a bidirectional stream could result in a situation where the sender has closed its side of the QUIC stream, but the receiver continues to send RTCP in the opposite direction. Thus it makes more sense if the sender and receiver agree on one or multiple unidirectional streams to transport any RTCP messages.

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 RTP 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.

9.1. RTP Considerations

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.

9.2. RTCP Considerations

RTCP was originally defined to be used with UDP, which implies (1) the only buffering present would be at the IP interface level, so that transmission timing is largely under the control of the application, and (2) that the overhead, _avg_rtcp_size_, used to compute the RTCP transmission interval could be deterministically computed by adding the IP and UDP headers. Both change when carrying RTCP over QUIC and they change in different ways when using QUIC

streams vs. QUIC datagrams.

9.2.1. RTCP over QUIC datagrams

When sending RTCP packets in QUIC datagrams this implies that an RTCP packet may not be immediately transmitted as it is subject to queuing and multiplexing with RTP packets and subject to QUIC congestion control. This means that a sending timestamp added to an RTCP packet, e.g., in an SR packet, may differ in unforeseeable ways from the actual time when the RTCP packet gets sent into the network while these are usually fairly close to each other for RTP-over-UDP. Effectively, we have a application sending timestamp _t_a_ and the network transmission timestamp _t_n_. Applications just have to be aware that RTCP does not measure the network level RTT but rather the application layer RTT.

Moreover, the true overhead per RTCP packet cannot easily be determined: this is because, in addition to adding the IP and UDP headers, the QUIC (short) header and the QUIC datagram frame header are to be considered but their sizes vary and it is unknown which other frames may be sent along in the same UDP packet. Any lower bound that can be determined could be affected by the version of QUIC being used. An example estimation of the overhead including IP, UDP and QUIC headers is given in Appendix C.

It is thus suggested that application developers recognize that per-RTCP packet overhead will always be an estimate, and include IP, UDP, QUIC, and DATAGRAM header sizes as a conservative heuristic. While this value may not be precisely accurate, it follows the example of RTP over UDP in [RFC3550], which includes the RTP and UDP header sizes, and adding the additional QUIC and DATAGRAM header sizes avoids the immediate problem of significantly understating avg_rtcp_size, resulting in an underestimate of the cost of sending additional RTCP reports.

9.2.2. RTCP over QUIC streams

The above considerations from Section 9.2.1 get even more complex when transmitting RTCP reliably over QUIC streams: it is unknown if (and how many) retransmissions occurred.

For RTT computations, again, this means that the application must consider that it observes the application layer RTT including retransmissions, where retransmissions also contribute to the observed jitter.

For overhead computation, retransmissions are not explicitly considered nor is the multiplexing with other streams.

To keep the complexity under control, it is again suggested that application developers recognize that per-RTCP packet overhead will always be an estimate, and these estimates should include plausible values for IP, UDP, QUIC, and QUIC STREAM frame header sizes. While this value may not be precisely accurate, it follows the example of RoQ over DATAGRAMs in Section 9.2.1}, and again avoids the immediate problem of significantly understating avg_rtcp_size, resulting in an underestimate of the cost of sending additional RTCP reports.

9.2.3. Mixed operations

As noted in Section 9, applications may use QUIC streams, QUIC DATAGRAMs, or a mixture, and this extends to choices for RTP and RTCP. While applications may, in principle, mix sending RTP and RTCP via QUIC streams and via QUIC DATAGRAMs, doing so has unforeseeable implications on timing and reordering and overhead.

Using the same QUIC primitives for both RTP and RTCP when transporting a single media stream will be safer than mixing QUIC primitives - for example, using QUIC streams to carry RTP media payloads and QUIC DATAGRAMs to carry RTCP, or vice versa. If an application uses both streams and datagrams to selectively obtain reliable transmission for some RTP media payloads but not for others, it is strongly suggested that the application developer knowingly choose which RTT observations they are interested in, while remaining aware of the advice included in Section 9.2.

Even this awareness may not be "safe enough" - for example, [RFC9221] allows QUIC DATAGRAM frames to be coalesced with other QUIC frames, and recommends, but does not require, QUIC DATAGRAMs to be sent as soon as possible, or to be delivered to a receiving application immediately. [RFC9221] also recommends, but does not require, a QUIC implementation to provide an API for prioritization between QUIC streams and QUIC DATAGRAMs.

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.

Because RTP applications do use the application timestamps contained in RTCP packets in a variety of ways, a RoQ implementation that provides and event-driven API can allow RoQ applications to generate RTCP packets "at the last moment", when the RoQ application is able to send the RTCP packet, and allow RoQ applications to notice that QUIC congestion control is limiting the ability of the RoQ application to send RTCP packets without this delay.

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 RTP 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-12].

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: 07 January 2025

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

14.4. meetecho/imquic

Ogranization: Meetecho

Implementation: imquic [imquic]

Description: QUIC library with RTP Over QUIC (RoQ) and Media Over QUIC (MoQT) support

Level of Maturity: alpha

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. Basic client and server examples are available as a demo, and the library was also used to test interoperability with WebRTC via an open source gateway.

Version Compatibility: [I-D.draft-ietf-avtcore-rtp-over-quic-12]

Licensing: MIT

Implementation Experience: :

Contact Information: Lorenzo Miniero (lorenzo@meetecho.com)

Last Updated: 07 January 2025

14.5. gstreamer/gst-plugin-quinn

Organization: asymptotic

Implementation: gst-plugin-quinn [gst-plugin-quinn]

Description: GStreamer plugin with support for QUIC and RTP Over QUIC (RoQ)

Level of Maturity: alpha

Coverage: The library supports sending and receiving RTP packets using QUIC streams and QUIC DATAGRAMs, and supports multiplexing using flow identifiers. Using stream per packet is not supported at the moment. Applications using this GStreamer plugin are responsible for any required out-of-band signalling, and managing RTP sessions. quinnquicmux and quinnquicdemux provide RoQ

functionality with the QUIC transport handled by quinnquicsink and quinnquicsrc plugins. Applications can choose whether to send RTP packets over streams or DATAGRAMs. Basic examples are available in the repository.

Version Compatibility: [I-D.draft-ietf-avtcore-rtp-over-quic-12]

Licensing: MPL

Implementation Experience: :

Contact Information: Sanchayan Maity (sanchayan@asymptotic.io)

Arun Raghavan (arun@asymptotic.io)

Last Updated: 21 March 2025

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" registry group.

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-plugin-quinn]
"gst-plugin-quinn", n.d., <https://gitlab.freedesktop.org/gstreamer/gst-plugins-rs/-/tree/main/net/quinn>.

[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-05, 12 February 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-avtcore-rtcp-green-metadata-05>.

[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-12]
Engelbart, M., Ott, J., and S. Dawkins, "RTP over QUIC (RoQ)", Work in Progress, Internet-Draft, draft-ietf-avtcore-rtp-over-quic-12, 21 October 2024, <https://datatracker.ietf.org/doc/html/draft-ietf-avtcore-rtp-over-quic-12>.

[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. K端hlewind, "QUIC Acknowledgment Frequency", Work in Progress, Internet-Draft, draft-ietf-quic-ack-frequency-11, 28 February 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-quic-ack-frequency-11>.

[I-D.draft-ietf-quic-multipath]
Liu, Y., Ma, Y., De Coninck, Q., Bonaventure, O., Huitema, C., and M. K端hlewind, "Multipath Extension for QUIC", Work in Progress, Internet-Draft, draft-ietf-quic-multipath-13, 3 March 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-quic-multipath-13>.

[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-16, 21 August 2024, <https://datatracker.ietf.org/doc/html/draft-ietf-wish-whip-16>.

[I-D.draft-smith-quic-receive-ts]
Smith, C., Swett, I., Beshay, J., Jaiswal, S., Purushothaman, I., and B. Schlinker, "QUIC Extended Acknowledgement for Reporting Packet Receive Timestamps", Work in Progress, Internet-Draft, draft-smith-quic-receive-ts-01, 3 March 2025, <https://datatracker.ietf.org/doc/html/draft-smith-quic-receive-ts-01>.

[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/>.

[imquic] "imquic", n.d., <https://github.com/meetecho/imquic>.

[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., T端xen, M., R端ngeler, I., and T. V旦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] K端hlewind, 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 RTCP 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 RTCP 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 /	QUIC could provide packet receive timestamps when using a timestamp extension that reports timestamp

		partly	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	RTP 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 RTCP 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 RTCP packets can be directly inferred from QUIC's acknowledgments. The calculation includes all RTP packets up to the acknowledged RTP packet with the highest RTP sequence number.

* _Cumulative lost_: Similar to the fraction of lost RTP 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 RTP 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 RTP 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 RTP 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.

Appendix C. Header overhead considerations

As discussed in Section 9.2, the header overhead of an RTP packet sent over RoQ cannot easily be determined. This section gives an estimation of the minimum and maximum header overhead of different combinations of STREAM and DATAGRAM frames using either IPv4 or IPv6. However, even this estimation is not exactly correct, since it does not take into account additional complications such as that RTP packets may be fragmented over multiple STREAM frames and that QUIC packets may contain more than a single FRAME, so that the RTCP overhead could thus be the shared overhead of multiple RTP packets being sent in different QUIC frames in the same QUIC packet.

* At least 20 Bytes (v4) or 40 Bytes (v6) IP header

* 8 Bytes UDP header

* 2-25 Bytes QUIC Short header packets

  - 1 Byte fixed header

- 0-20 Bytes Connection ID
- 1-4 Bytes Packet Number

* 2-25 Bytes STREAM frame header
- 1 Byte type
- 1-8 Bytes stream ID
- Optional 1-8 Bytes Offset
- Optional 1-8 Bytes Length

* 1-9 Bytes DATAGRAM frame header
- 1 Byte type
- Optional 1-8 Bytes length

* 1-8 Bytes RoQ Flow ID

* IPv4 with STREAM frames
- Minimum: 20+8+2+2+1=33 Bytes
- Maximum: 20+8+25+25+8=86 Bytes

* IPv6 with STREAM frames
- Minimum: 40+8+2+2+1=53 Bytes
- Maximum: 40+8+25+25+8=106 Bytes

* IPv4 with DATAGRAM frames
- Minimum: 20+8+2+1+1=32 Bytes
- Maximum: 20+8+25+9+8=70 Bytes

* IPv6 with DATAGRAM frames
- Minimum: 40+8+2+1+1=52 Bytes
- Maximum: 40+8+25+9+8=90 Bytes

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, Gurtej Singh Chandok, Lucas Pardue, Nils Ohlmeier, Sam Hurst, Sergio Garcia Murillo, and Vidhi Goel for their valuable comments and suggestions contributing to this document.

The authors would also like to thank Sam Hurst and Lorenzo Miniero for implementing RTP over QUIC.

Authors' Addresses

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

J旦rg Ott
Technical University of Munich
Email: ott@in.tum.de

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