AVTCORE
Internet-Draft
Intended status: Standards Track
Expires: 26 December 2022

J. Ott
M. Engelbart
Technical University Munich

24 June 2022

RTP over QUIC
draft-engelbart-rtp-over-quic-04

Abstract

This document specifies a minimal mapping for encapsulating RTP and RTCP packets within QUIC. It also discusses how to leverage state from the QUIC implementation in the endpoints to reduce the exchange of RTCP packets and how to implement congestion control.

Discussion Venues

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

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

Status of This Memo

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

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

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

This Internet-Draft will expire on 26 December 2022.

Table of Contents

1. Introduction

The Real-time Transport Protocol (RTP) [RFC3550] is generally used to carry real-time media for conversational media sessions, such as video conferences, across the Internet. Since RTP requires real-time delivery and is tolerant to packet losses, the default underlying transport protocol has been UDP, recently with DTLS on top to secure the media exchange and occasionally TCP (and possibly TLS) as a fallback. With the advent of QUIC [RFC9000] and, most notably, its unreliable DATAGRAM extension [RFC9221], another secure transport protocol becomes available. QUIC and its DATAGRAMs combine desirable properties for real-time traffic (e.g., no unnecessary retransmissions, avoiding head-of-line blocking) with a secure end-to-end transport that is also expected to work well through NATs and firewalls.

Moreover, with QUIC's multiplexing capabilities, reliable and unreliable transport connections as, e.g., needed for WebRTC, can be established with only a single port used at either end of the connection. This document defines a mapping of how to carry RTP over QUIC. The focus is on RTP and RTCP packet mapping and on reducing the amount of RTCP traffic by leveraging state information readily available within a QUIC endpoint. This document also describes different options for implementing congestion control for RTP over QUIC.

The scope of this document is limited to unicast RTP/RTCP.

This document does not cover signaling for session setup. Signaling for RTP over QUIC can be defined in separate documents such as [I-D.draft-dawkins-avtcore-sdp-rtp-quic] does for SDP.

Note that this draft is similar in spirit to but differs in numerous ways from [I-D.draft-hurst-quic-rtp-tunnelling].

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.

The following terms are used:

Datagram: Datagrams exist in UDP as well as in QUICs unreliable

datagram extension. If not explicitly noted differently, the term datagram in this document refers to a QUIC Datagram as defined in [RFC9221].

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

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

Media Encoder: An entity that is used by an application to produce a stream of encoded media, which can be packetized in RTP packets to be transmitted over QUIC.

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

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

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

3. Scope

RTP over QUIC mostly defines an application usage of QUIC [I-D.draft-ietf-quic-applicability]. As a baseline, the specification does not expect more than a standard QUIC implementation as defined in [RFC8999], [RFC9000], [RFC9001], and [RFC9002]. Nevertheless, the specification can benefit from QUIC extesions such as QUIC datagrams [RFC9221] as described below. Moreover, this document describes how a QUIC implementation and its API can be extended to improve efficiency of the protocol operation.

On top of QUIC, this document defines an encapsulation of RTP and RTCP packets.

The scope of this document is limited to carrying RTP over QUIC. It does not attempt to enhance QUIC for real-time media or define a replacement or evolution of RTP. Such new media transport protocols may be covered elsewhere, e.g., in the MOQ WG.

Protocols for negotiating connection setup and the associated parameters are defined separately, e.g., in [I-D.draft-dawkins-avtcore-sdp-rtp-quic].

4. Protocol Overview

This document introduces a mapping of the Real-time Transport Protocol (RTP) to the QUIC transport protocol. RTP over QUIC allows the use of QUIC streams and unreliable QUIC datagrams to transport real-time data, and thus, the QUIC implementation MUST support QUICs unreliable datagram extension, if RTP packets should be sent over QUIC datagrams. Since datagram frames cannot be fragmented, the QUIC implementation MUST also provide a way to query the maximum datagram size so that an application can create RTP packets that always fit into a QUIC datagram frame.

[RFC3550] specifies that RTP sessions need to be transmitted on different transport addresses to allow multiplexing between them. RTP over QUIC uses a different approach to leverage the advantages of QUIC connections without managing a separate QUIC connection per RTP session. QUIC 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 are flow identifiers prepended to each datagram frame as described in Section 2.1 of [I-D.draft-ietf-masque-h3-datagram]. RTP over QUIC uses a flow identifier to replace the network address and port number to multiplex many RTP sessions over the same QUIC connection.

A congestion controller can be plugged in to adapt the media bitrate to the available bandwidth. This document does not mandate any congestion control algorithm. Some examples include Network-Assisted Dynamic Adaptation (NADA) [RFC8698] and Self-Clocked Rate Adaptation for Multimedia (SCReAM) [RFC8298]. These congestion control algorithms require some feedback about the network's performance to calculate target bitrates. Traditionally this feedback is generated at the receiver and sent back to the sender via RTCP. Since QUIC also collects some metrics about the network's performance, these metrics can be used to generate the required feedback at the sender-side and provide it to the congestion controller to avoid the additional overhead of the RTCP stream.

5. Connection Establishment and ALPN

QUIC requires the use of ALPN [RFC7301] tokens during connection setup. RTP over QUIC uses "rtp-mux-quic" as ALPN token in the TLS handshake (see also Section 12.

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

Editor's note: "rtp-mux-quic" indicates that RTP and other protocols may be multiplexed on the same QUIC connection using a flow identifier as described in Section 6. Applications are responsible for negotiation of protocols in use by appropriate use of a signaling protocol such as SDP.

Editor's note: This implies that applications cannot use RTP over QUIC as specified in this document over WebTransport.

5.1. Draft version identification

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

RTP over QUIC uses the token "rtp-mux-quic" to identify itself in ALPN.

Only implementations of the final, published RFC can identify themselves as "rtp-mux-quic". 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-engelbart-rtp-over-quic-04 is identified using the string "rtp-mux-quic-04".

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

6. Encapsulation

QUIC supports two transport methods: reliable streams [RFC9000] and unreliable datagrams [RFC9221]. This document specifies a mapping of RTP to both of the transport modes. The encapsulation format for RTP over QUIC is described in Figure 1.

Section 6.1 and Section 6.2 explain the specifics of mapping of RTP to QUIC streams and QUIC datagrams respectively.

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

Figure 1: RTP over QUIC Payload Format

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

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

For multiplexing different RTP and other data streams on the same QUIC connection, each RTP/RTCP packet is prefixed with a flow identifier. A flow identifier is a QUIC variable-length integer which must be unique per stream.

RTP and RTCP packets of a single RTP session MAY be sent using the same flow identifier (following the procedures defined in [RFC5761], or they MAY be sent using different flow identifiers. The respective mode of operation MUST be indicated using the appropriate signaling.

RTP and RTCP packets of different RTP sessions MUST be sent using different flow identifiers.

Differentiating RTP/RTCP packets of different RTP sessions from non-RTP/RTCP datagrams is the responsibility of the application by means of appropriate use of flow identifiers and the corresponding signaling.

This specification defines two ways of carrying RTP packets in QUIC: 1) using reliable QUIC streams and 2) using unreliable QUIC DATAGRAMs. Every RTP session MUST choose exactly one way of carrying RTP and RTCP packets, different RTP sessions MAY choose different ways.

6.1. QUIC Streams

An application MUST open a new QUIC stream for each Application Data Unit (ADU). Each ADU MUST be encapsulated in a single RTP packet and the application MUST not send more than one RTP packet per stream. Opening a new stream for each packet adds implicit framing to RTP packets, allows to receive packets without strict ordering and gives an application the possibility to cancel certain packets.

Large RTP packets sent on a stream will be fragmented in smaller QUIC frames, that are transmitted reliably and in order, such that a receiving application can read a complete packet from the stream. No retransmission has to be implemented by the application, since QUIC frames that are lost in transit are retransmitted by the QUIC connection. If it is known to either the sender or the receiver, that a packet, which was not yet successfully and completely transmitted, is no longer needed, either side can close the stream.

Editor's Note: We considered adding a framing like the one described in [RFC4571] to send multiple RTP packets on one stream, but we don't think it is worth the additional overhead only to reduce the number of streams. Moreover, putting multiple ADUs into a single stream would also require defining policies when to use the same (and which) stream and when to open a new one.

Editor's Note: Note, however, that using a single frame per stream in a single RTP packet may cause interworking issues when a translator wants to forward packets received via RTP-over-QUIC to an endpoint as UDP packets because the received ADUs may exceed the MTU size or even maximum UDP packet size.

6.2. QUIC Datagrams

RTP packets can be sent in QUIC datagrams. QUIC datagrams are an extension to QUIC described in [RFC9221]. QUIC datagrams preserve frame boundaries, thus a single RTP packet can be mapped to a single QUIC datagram, without the need for an additional framing. Senders SHOULD consider the header overhead associated with QUIC datagrams and ensure that the RTP/RTCP packets, including their payloads, QUIC, and IP headers, will fit into path MTU.

If an application wishes to retransmit lost RTP packets, the retransmission has to be implemented by the application by sending a new datagram for the RTP packet, because QUIC datagrams are not retransmitted on loss (see also Section 7.1 for loss signaling).

7. RTCP

The RTP Control Protocol (RTCP) allows RTP senders and receivers to exchange control information to monitor connection statistics and to identify and synchronize streams. Many of the statistics contained in RTCP packets overlap with the connection statistics collected by a QUIC connection. To avoid using up bandwidth for duplicated control information, the information SHOULD only be sent at one protocol layer. QUIC relies on certain control frames to be sent.

In general, applications MAY send RTCP without any restrictions. This document specifies a baseline for replacing some of the RTCP packet types by mapping the contents to QUIC connection statistics. Future documents can extend this mapping for other RTCP format types. It is RECOMMENDED to expose relevant information from the QUIC layer to the application instead of exchanging addtional RTCP packets, where applicable.

This section discusses what information can be exposed from the QUIC connection layer to reduce the RTCP overhead and which type of RTCP messages cannot be replaced by similar feedback from the transport layer. The list of RTCP packets in this section is not exhaustive and similar considerations SHOULD be taken into account before exchanging any other type of RTCP control packets.

7.1. Transport Layer Feedback

This section explains how some of the RTCP packet types which 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.

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. For RTP packets that are sent over QUIC streams, an RTP packet can be considered acknowledged, when all frames which carried fragments of the RTP packet were acknowledged.

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

Some of the transport layer feedback that can be implemented in RTCP contains information that is not included in QUIC by default, but can be added via QUIC extensions. One important example are arrival timestamps, which are not part of QUIC's default acknowledgment frames, but can be added using [I-D.draft-smith-quic-receive-ts] or [I-D.draft-huitema-quic-ts]. Another extension, that can improve the precision of the feedback from QUIC is [I-D.draft-ietf-quic-ack-frequency], which allows a sender to control the delay of acknowledgments sent by the receiver.

* _Receiver Reports_ (PT=201, Name=RR, [RFC3550])
  - _Fraction lost_: The fraction of lost packets can be directly infered from QUIC's acknowledgments. The calculation SHOULD include all packets up to the acknowledged RTP packet with the highest RTP sequence number. Later packets SHOULD be ignored,

since they may still be in flight, unless other QUIC packets that were sent after the datagram frame, were already acknowledged.

- _Cumulative lost_: Similar to the fraction of lost packets, the cumulative loss can be infered from QUIC's acknowledgments including all packets up to the latest acknowledged packet.

- _Highest Sequence Number received_: The highest sequence number received is the sequence number of all RTP packets that were acknowledged.

- Interarrival jitter: If QUIC acknowledgments carry timestamps as described in one of the extensions referenced above, senders can infer from QUIC acks the interarrival jitter from the arrival timestamps.

- Last SR: Similar to RTP arrival times, the arrival time of RTCP Sender Reports can be inferred from QUIC acknowledgments, if they include timestamps.

- Delay since last SR: This field is not required when the receiver reports are entirely replaced by QUIC feedback.

* _Negative Acknowledgments_ (PT=205, FMT=1, Name=Generic NACK, [RFC4585])

  - The generic negative acknowledgment packet contains information about packets which the receiver considered lost. Section 6.2.1. of [RFC4585] recommends to use this feature only, if the underlying protocol cannot provide similar feedback. QUIC does not provide negative acknowledgments, but can detect lost packets through acknowledgments.

* _ECN Feedback_ (PT=205, FMT=8, Name=RTCP-ECN-FB, [RFC6679])

  - ECN feedback 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 which are listed below. QUIC supports ECN reporting through acknowledgments. If the connection supports ECN, the reporting of ECN counts SHOULD be done using QUIC acknowledgments.

* _Congestion Control Feedback_ (PT=205, FMT=11, Name=CCFB, [RFC8888])

  - RTP Congestion Control Feedback contains acknowledgments, arrival timestamps and ECN notifications for each received packet. Acknowledgments and ECNs can be infered 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].

* _Extended Reports_ (PT=207, Name=XR, [RFC3611])

  - Extended Reports offer an extensible framework for a variety of different control messages. Some of the standard report blocks which can be implemented in extended reports such as loss RLE or ECNs can be implemented in QUIC, too. For other report blocks, it SHOULD be evaluated individually, if the contained information can be transmitted using QUIC instead.

7.2. Application Layer Repair and other Control Messages

While the previous section presented some RTCP packet that can be replaced by QUIC features, QUIC cannot replace all of the available RTCP packet types. This mostly affects RTCP packet types which carry control information that is to be interpreted by the application layer instead of the transport itself.

Sender Reports (PT=200, Name=SR, [RFC3550]) are similar to _Receiver Reports_. They are sent by media senders and additionally contain a NTP and a RTP timestamp and the number of packets and octets transmitted by the sender. The timestamps can be used by a receiver to synchronize streams. QUIC cannot provide a similar control information, since it does not know about RTP timestamps. A QUIC receiver can also not calculate the packet or octet counts, since it does not know about lost datagrams. Thus, sender reports are required in RTP over QUIC to synchronize streams at the receiver. The sender reports SHOULD not contain any receiver report blocks, as the information can be infered from the QUIC transport as explained in the previous section.

Next to carrying transmission statistics, RTCP packets can contain application layer control information, that cannot directly be mapped to QUIC. This includes for example the _Source Description_ (PT=202, Name=SDES), _Bye_ (PT=203, Name=BYE) and _Application_ (PT=204, Name=APP) packet types from [RFC3550] or many of the payload specific feedback messages (PT=206) defined in [RFC4585], which can for example be used to control the codec behavior of the sender. Since QUIC does not provide any kind of application layer control messaging, these RTCP packet types SHOULD be used in the same way as they would be used over any other transport protocol.

8. Congestion Control

Like any other application on the internet, RTP over QUIC needs to perform congestion control to avoid overloading the network.

QUIC is a congestion controlled transport protocol. Senders are required to employ some form of congestion control. The default congestion control specified for QUIC is an alogrithm similar to TCP NewReno, but senders are free to choose any congestion control algorithm as long as they follow the guidelines specified in Section 3 of [RFC8085].

RTP does not specify a congestion controller, but provides feedback formats for congestion control (e.g. [RFC8888]) as well as different congestion control algorithms in separate RFCs (e.g. SCReAM [RFC8298] and NADA [RFC8698]). The congestion control algorithms for RTP are specifically tailored for real-time transmissions at low latencies. The available congestion control algorithms for RTP expose a target_bitrate that can be used to dynamically reconfigure media codecs to produce media at a rate that can be sent in real-time under the observed network conditions.

This section defines two architectures for congestion control and bandwidth estimation for RTP over QUIC, but it does not mandate any specific congestion control algorithm to use. The section also discusses congestion control implications of using shared or multiple separate QUIC connections to send and receive multiple independent data streams.

It is assumed that the congestion controller in use provides a pacing mechanism to determine when a packet can be sent to avoid bursts. The currently proposed congestion control algorithms for real-time communications provide such pacing mechanisms. The use of congestion controllers which don’t provide a pacing mechanism is out of scope of this document.

> *TODO:* Add considerations for bandwidth shares when a QUIC connection is shared between RTP and non-RTP streams?

8.1. Congestion Control at the QUIC layer

If congestion control is to be applied at the transport layer, it is RECOMMENDED to configure the QUIC Implementation to use a delay-based real-time congestion control algorithm instead of a loss-based algorithm. The currently available delay-based congestion control algorithms depend on detailed arrival time feedback to estimate the current one-way delay between sender and receiver. Since QUIC does not provide arrival timestamps in its acknowledgments, the QUIC

implementations of the sender and receiver MUST use an extension to add this information to QUICs acknowledgment frames, e.g. [I-D.draft-smith-quic-receive-ts].

If congestion control is done by the QUIC implementation, the application needs a mechanism to query the currently available bandwidth to adapt media codec configurations. The employed congestion controller of the QUIC connection SHOULD expose such an API to the application. If a current bandwidth estimation is not available from the QUIC congestion controller, the sender can either implement an alternative bandwidth estimation at the application layer as described in Section 8.2 or a receiver can feedback the observed bandwidth through RTCP, e.g., using [I-D.draft-alvestrand-rmcat-remb].

Editor's note: An alternative to the hard requirement to use a timestamp extension could be to use RTCP, but that would mean, that an application has to negotiate RTCP congestion control feedback which would then have to be passed to the QUIC congestion controller.

Editor's note: How can a QUIC connection be shared with non-RTP streams, when SCReAM/NADA/GCC is used as congestion controller? Can these algorithms be adapted to allow different streams including non-real-time streams? Do they even have to be adapted or _should_ this just work?

8.2. Congestion Control at the Application Layer

If an application cannot access a bandwidth estimation from the QUIC layer, or the QUIC implementation does not support a delay-based, low-latency congestion control algorithm, it can alternatively implement a bandwidth estimation algorithm at the application layer. Calculating a bandwidth estimation at the application layer can be done using the same bandwidth estimation algorithms as described in Section 8 (NADA, SCReAM). The bandwidth estimation algorithm typically needs some feedback on the transmission performance. This feedback can be collected following the guidelines in Section 7.

If the application implements full congestion control rather than just a bandwidth estimation at the application layer using a congestion controller that satisfies the requirements of Section 7 of [RFC9002], and the connection is only used to send real-time media which is subject to the application layer congestion control, it is RECOMMENDED to disable any other congestion control that is possibly running at the QUIC layer. Disabling the additional congestion controllers helps to avoid any interference between the different congestion controllers.

8.3. Shared QUIC connections

Two endpoints may want to 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. A straightforward solution is to establish multiple QUIC connections, one for each channel. Or all real-time channels are mapped to one QUIC connection, while a separate QUIC connection is created for the non-real-time channels. In both 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.

Alternatively, (all or a subset of) real-time and non-real-time channels are multiplexed onto a single, shared QUIC connection, which can be done by using the flow identifier described in Section 6. Applications multiplexing multiple streams in one connection SHOULD implement some form of stream prioritization or bandwidth allocation.

9. API Considerations

The mapping described in the previous sections poses some interface requirements on the QUIC implementation. Although a basic mapping should work without any of these requirements most of the optimizations regarding congestion control and RTCP mapping require certain functionalities to be exposed to the application. The following to sections contain a list of information that is required by an application to implement different optimizations (Section 9.1) and functions that a QUIC implementation SHOULD expose to an application (Section 9.2).

Each item in the following list can be considered individually. Any exposed information or function can be used by RTP over QUIC regardless of whether the other items are available. Thus, RTP over QUIC 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.

9.1. Information to be exported from QUIC

This section provides a list of items that an application might want to export from an underlying QUIC implementation. It is thus RECOMMENDED that a QUIC implementation exports the listed items to the application.

* _Maximum Datagram Size_: The maximum datagram size that the QUIC connection can transmit.

* _Datagram Acknowledgment and Loss_: Section 5.2 of [RFC9221] allows QUIC implementations to notify the application that a QUIC Datagram was acknowledged or that it believes a datagram was lost. The exposed information SHOULD include enough information to allow the application to maintain a mapping between the datagram that was acknowledged/lost and the RTP packet that was sent in that datagram.

* _Stream States_: The QUIC implementation SHOULD expose to a sender, how much of the data that was sent on a stream was successfully delivered and how much data is still outstanding to be sent or retransmitted.

* _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 SHOULD be exposed to the application.

* _ECN_: If ECN marks are available, they SHOULD be exposed to the application.

9.2. Functions to be exposed by QUIC

This sections lists functions that a QUIC implementation SHOULD expose to an application to implement different features of the mapping described in the previous sections of this document.

* _Cancel Streams_: To allow an application to cancel (re)transmission of packets that are no longer needed, the QUIC implementation MUST expose a way to cancel the corresponding QUIC streams.

* _Select Congestion Controller_: If congestion control is to be implemented at the QUIC connection layer as described in Section 8.1, the application must be able to choose an appropriate congestion control algorithm.

* _Disable Congestion Controller_: If congestion control is to be implemented at the application layer as described in Section 8.2, and the application layer is trusted to apply adequate congestion control, it is RECOMMENDEDto allow the application to disable QUIC layer congestion control entirely.

10. Discussion

10.1. Flow Identifier

[RFC9221] suggests to use flow identifiers to multiplex different streams on QUIC Datagrams, which is implemented in Section 6, but it is unclear how applications can combine RTP over QUIC with other data streams using the same QUIC connections. If the non-RTP data streams use the same flow identifies, too and the application can make sure, that flow identifiers are unique, there should be no problem. Flow identifiers could be problematic, if different specifications for RTP and non-RTP data streams over QUIC mandate different incompatible flow identifiers.

10.2. Impact of Connection Migration

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

If a QUIC connection is established via a signaling channel, this signaling may have involved Interactive Connectivity Establishment (ICE) exchanges to determine and choose suitable (IP address, port number) pairs for the QUIC connection. Subsequent address change events may be noticed by QUIC via its connection migration handling and/or at the ICE or other application layer, e.g., by noticing changing IP addresses at the network interface. This may imply that the two signaling and data "layers" get (temporarily) out of sync.

> *Editor's Note:* It may be desirable that the API provides an indication of connection migration event for either case.

10.3. 0-RTT considerations

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

This specification defines carrying RTP on top of QUIC and thus does not finally define what the actual application data are going to be. RTP typically carries ephemeral media contents that is rendered and possibly recorded but otherwise causes no side effects. Moreover,

the amount of data that can be carried as 0-RTT data is rather limited. But it is the responsibility of the respective application to determine if 0-RTT data is permissible.

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

11. Security Considerations

RTP over QUIC 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 RTP over QUIC.

12. IANA Considerations

12.1. Registration of a RTP over QUIC Identification String

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

The "rtp-mux-quic" string identifies RTP over QUIC:

Protocol: RTP over QUIC

Identification Sequence: 0x72 0x74 0x70 0x2D 0x6F 0x76 0x65 0x72 0x2D 0x71 0x75 0x69 0x63 ("rtp-mux-quic")

Specification: This document

13. References

13.1. Normative References

[I-D.draft-huitema-quic-ts]
Huitema, C., "Quic Timestamps For Measuring One-Way Delays", Work in Progress, Internet-Draft, draft-huitema-quic-ts-07, 6 March 2022, <https://datatracker.ietf.org/doc/html/draft-huitema-quic-ts-07>.

[I-D.draft-ietf-quic-ack-frequency]
Iyengar, J. and I. Swett, "QUIC Acknowledgement Frequency", Work in Progress, Internet-Draft, draft-ietf-quic-ack-frequency-01, 25 October 2021, <https://datatracker.ietf.org/doc/html/draft-ietf-quic-ack-frequency-01>.

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

[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>.

[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>.

[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>.

[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>.

[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>.

[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>.

13.2. Informative References

[I-D.draft-alvestrand-rmcat-remb]
Alvestrand, H., "RTCP message for Receiver Estimated Maximum Bitrate", Work in Progress, Internet-Draft, draft-alvestrand-rmcat-remb-03, 21 October 2013, <https://datatracker.ietf.org/doc/html/draft-alvestrand-rmcat-remb-03>.

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

[I-D.draft-ietf-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-applicability]
Kuehlewind, M. and B. Trammell, "Applicability of the QUIC Transport Protocol", Work in Progress, Internet-Draft, draft-ietf-quic-applicability-16, 6 April 2022, <https://datatracker.ietf.org/doc/html/draft-ietf-quic-applicability-16>.

[RFC4571] Lazzaro, J., "Framing Real-time Transport Protocol (RTP) and RTP Control Protocol (RTCP) Packets over Connection-Oriented Transport", RFC 4571, DOI 10.17487/RFC4571, July 2006, <https://www.rfc-editor.org/rfc/rfc4571>.

[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>.

Appendix A. Experimental Results

An experimental implementation of the mapping described in this document can be found on Github (https://github.com/mengelbart/rtp-over-quic). The application implements the RTP over QUIC Datagrams mapping and implements SCReAM congestion control at the application layer. It can optionally disable the builtin QUIC congestion control (NewReno). The endpoints only use RTCP for congestion control feedback, which can optionally be disabled and replaced by the QUIC connection statistics as described in Section 7.1.

Experimental results of the implementation can be found on Github (https://github.com/mengelbart/rtp-over-quic-mininet), too.

Acknowledgments

The authors would like to thank Spencer Dawkins, Lucas Pardue and David Schinazi for their valuable comments and suggestions contributing to this document.

Authors' Addresses

Jörg Ott
Technical University Munich
Email: ott@in.tum.de

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