Network Working Group
Internet-Draft
Intended status: Informational
Expires: 28 April 2022

J. Ott
M. Engelbart
Technical University Munich

25 October 2021

RTP over QUIC
draft-engelbart-rtp-over-quic-01

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.

Discussion Venues

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

Discussion of this document takes place on the mailing list (), which is archived at .

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 28 April 2022.

Copyright Notice

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 fallback. With the advent of QUIC and, most notably, its unreliable DATAGRAM extension, 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 briefly touches upon how to signal media over QUIC using the Session Description Protocol (SDP) [RFC8866].

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

Note that this draft is similar in spirit to but differs in numerous ways from [QRT].

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:

Congestion Controller: QUIC specifies a congestion controller in Section 7 of [RFC9002], but the specific requirements for interactive real-time media, lead to the development of dedicated congestion control algorithms. The term congestion controller in this document refers to these algorithms which are dedicated to real-time applications and may be used next to or instead of the congestion controller specified by [RFC9002].

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

[QUIC-DATAGRAM].

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 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. Protocol Overview

This document introduces a mapping of the Real-time Transport Protocol (RTP) to the QUIC transport protocol. QUIC supports two transport methods: reliable streams and unreliable datagrams [RFC9000], [QUIC-DATAGRAM]. RTP over QUIC uses unreliable QUIC datagrams to transport real-time data.

[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, in order 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 implements a demultiplexing mechanism if it is required. An example of such a mechanism are flow identifiers prepended to each datagram frame as described in [H3-DATAGRAM]. RTP over QUIC uses a flow identifier as a replacement for 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 performance of the network in order 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 networks 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.

> *Editor's note:* Should the congestion controller work independently from the congestion controller used in QUIC, because the QUIC connection can simultaneously be used for other data streams, that need to be congestion controlled, too?

4. Local Interfaces

RTP over QUIC requires different components like QUIC implementations, congestion controllers and media encoders to work together. The interfaces of these components have to fulfill certain requirements which are described in this section.

4.1. QUIC Interface

If the used QUIC implementation is not directly incorporated into the RTP over QUIC mapping implementation, it has to fulfill the following interface requirements. The QUIC implementation MUST support QUICs unreliable datagram extension and it MUST provide a way to signal acknowledgments or losses of QUIC datagrams to the application. Since datagram frames cannot be fragmented, the QUIC implementation MUST 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.

Additionally, a QUIC implementation MUST expose the recorded RTT statistics as described in Section 5 of [RFC9002] to the application. These statistics include the latest generated RTT sample

(latest_rtt), the minimum observed RTT over a period of time (min_rtt), exponentially-weighted moving average (smoothed_rtt) and the mean deviation (rtt_var). These values are necessary to perform congestion control as explained in Section 4.2.

Section 7.1 of [RFC9002] also specifies how QUIC treats Explicit Congestion Notifications (ECN) if it is supported by the network path. If ECN counts can be exported from a QUIC implementation, these may be used to improve congestion control, too.

4.2. Congestion Controller Interface

There are different congestion control algorithms proposed by RMCAT to implement application layer congestion control for real-time communications. To estimate the currently available bandwidth, these algorithms keep track of the sent packets and typically require a list of successfully delivered packets together with the timestamps at which they were received by a receiver. The bandwidth estimation can then be used to decide, whether the media encoder can be configured to produce output at a higher or lower rate.

A congestion controller used for RTP over QUIC should be able to compute an adequate bandwidth estimation using the following inputs:

* t_current: A current timestamp

* pkt_status_list: A list of RTP packets that were acknowledged by the receiver

* pkt_delay_list: For each acknowledged RTP packet, a delay between the send- and receive-timestamps of the packet

* The RTT estimations calculated by QUIC as described in Section 5 of [RFC9002]:

  - latest_rtt: The latest RTT sample generated by QUIC.

  - min_rtt: The miminum RTT observed by QUIC over a period of time

  - smoothed_rtt: An exponentially-weighted moving average of the observed RTT values

- rtt_var: The mean deviation in the observed RTT values

* ecn: Optionally ECN marks may be used, if supported by the network and exposed by the QUIC implementation.

A congestion controller MUST expose a target_bitrate to which the encoder should be configured to fully utilize the available bandwidth.

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

4.3. Codec Interface

An application is expected to adapt the media bitrate to the observed available bandwidth by setting the media encoder to the target_bitrate that is computed by the congestion controller. Thus, the media encoder needs to offer a way to update its bitrate accordingly. An RTP over QUIC implementation can either expose the most recent target_bitrate produced by the congestion controller to the application, or accept a callback from the application, which updates the encoder bitrate whenever the congestion controller updates the target_bitrate.

5. Packet Format

All RTP and RTCP packets MUST be sent in QUIC datagram frames with the following format:


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

Figure 1: Datagram Payload Format

For multiplexing RTP sessions on the same QUIC connection, each RTP/

RTCP packet is prefixed with a flow identifier. This flow identifier serves as a replacement for using different transport addresses per session. 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, e.g., when using SDP as discussed in Section 7.

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

Differentiating RTP/RTCP datagrams 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.

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.

6. Protocol Operation

This section describes how senders and receivers can exchange RTP and RTCP packets using QUIC. While the receiver side is very simple, the sender side has to keep track of sent packets and corresponding acknowledgments to implement congestion control.

RTP/RTCP packets that are submitted to an RTP over QUIC implementation are buffered in a queue. The congestion controller defines the rate at which the next packet is dequeued and sent over the QUIC connection. Before a packet is sent, it is prefixed with the flow identifier described in Section 5 and encapsulated in a QUIC datagram.

The implementation has to keep track of sent RTP packets in order to build the feedback for a congestion controller described in Section 4.2. Each sent RTP packet is mapped to the datagram in which

it was sent over QUIC. When the QUIC implementation signals an

acknowledgment for a specific datagram, the packet that was sent in this datagram is marked as received. Together with the received mark, an estimation of the delay at which the packet was received by the peer can be stored. Assuming the RTT is divided equally between the link from the sender to the receiver and the link back to the sender, this estimation can be calculated by adding the latest_rtt divided by two to the send time of the datagram in which the RTP packet was sent. This mapping can later be used to create the pkt_status_list and the pkt_delay_list as described in Section 4.2.

In a regular interval, the pkt_status_list and the pkt_delay_list MUST be passed to the congestion controller together with the current timestamp t_current and the RTT statistics min_rtt, smoothed_rtt and rtt_var. If available, the feedback MAY also contain the ECN marks.

The feedback report can be passed to the congestion controller at a frequency specified by the used algorithm.

The congestion controller regularly outputs the target_bitrate, which is forwarded to the encoder using the interface described in Section 4.3.

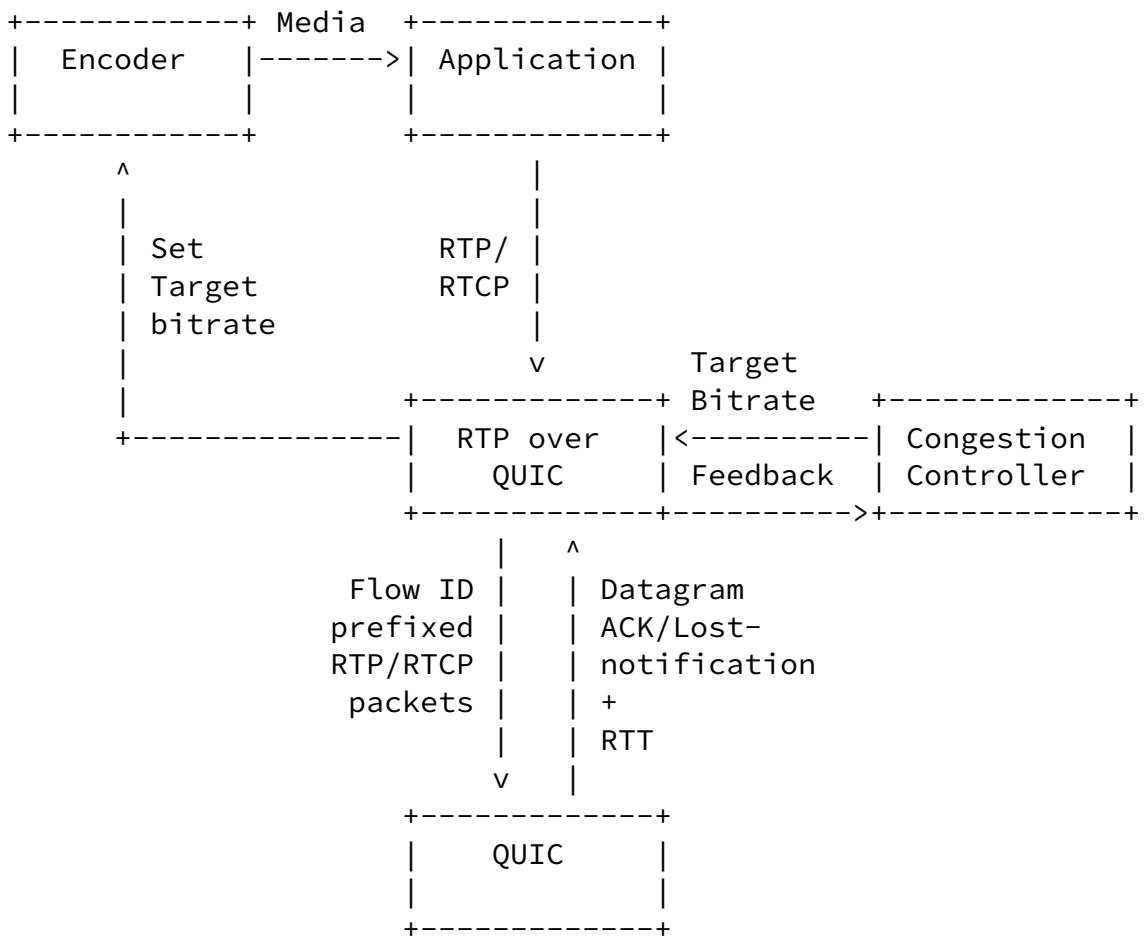

Figure 2: RTP over QUIC send flow

On receiving a datagram, an RTP over QUIC implementation strips off and parses the flow identifier to identify the stream to which the received RTP or RTCP packet belongs. The remaining content of the datagram is then passed to the RTP session which was assigned the given flow identifier.

7. SDP Signalling

QUIC is a connection-based protocol that supports connectionless transmissions of DATAGRAM frames within an established connection. As noted above, demultiplexing DATAGRAMS intended for different purposes is up to the application using QUIC.

There are several necessary steps to carry out jointly between the communicating peers to enable RTP over QUIC:

1. The protocol identifier for the m= lines MUST be "QUIC/RTP", combined as per [RFC8866] with the respective audiovisual profile: for example, "QUIC/RTP/AVP".

2. The peers need to decide whether to establish a new QUIC connection or whether to re-use an existing one. In case of establishing a new connection, the initiator and the responder (client and server) need to be determined. Signaling for this step MUST follow [RFC8122] on SDP attributes for connection-oriented media for the a=setup, a=connection, and a=fingerprint attributes. They MUST use the appropriate protocol identification as per 1.

3. The peers must provide a means for identifying RTP sessions carried in QUIC DATAGRAMS. To enable using a common transport connection for one, two, or more media sessions in the first place, the BUNDLE grouping framework MUST be used [RFC8843]. All media sections belonging to a bundle group, except the first one, MUST set the port in the m= line to zero and MUST include the a=bundle-only attribute.

   For disambiguating different RTP session, a reference needs to be provided for each m= line to allow associating this specific media session with a flow identifier. This could be achieved following different approaches:

   * Simply reusing the a=extmap attribute [RFC8285] and relying on RTP header extensions for demultiplexing different media

packets carried in QUIC DATAGRAM frames.

* Defining a variant or different flavor of the a=extmap attribute [RFC8285] that binds media sessions to flow identifiers used in QUIC DATAGRAMS.

  Editor's note: It is likely preferable to use multiplexing using QUIC DATAGRAM flow identifiers because this multiplexing mechanisms will also work across RTP and non-RTP media streams.

In either case, the corresponding identifiers MUST be treated independently for each direction of transmission, so that an endpoint MAY choose its own identifies and only uses SDP to inform its peer which RTP sessions use which identifiers.

To this end, SDP MUST be used to indicate the respective flow identifiers for RTP and RTCP of the different RTP sessions (for which we borrow inspiration from [RFC3605]).

4. The peers MUST agree, for each RTP session, whether or not to apply RTP/RTCP multiplexing. If multiplexing RTP and RTCP shall take place on the same flow identifier, this MUST be indicated using the attribute a=rtcp-mux.

A sample session setup offer (liberally borrowed and extended from [RFC8843] and [RFC8122] could look as follows:

```
v=0
o=alice 2890844526 2890844526 IN IP6 2001:db8::3
s=
c=IN IP6 2001:db8::3
t=0 0
a=group:BUNDLE abc xyz

m=audio 10000 QUIC/RTP/AVP 0 8 97
a=setup:actpass
a=connection:new
a=fingerprint:SHA-256 \
 12:DF:3E:5D:49:6B:19:E5:7C:AB:4A:AD:B9:B1:3F:82:18:3B:54:02:12:DF: \
 3E:5D:49:6B:19:E5:7C:AB:4A:AD
b=AS:200
a=mid:abc
a=rtcp-mux
a=rtpmap:0 PCMU/8000
a=rtpmap:8 PCMA/8000
a=rtpmap:97 iLBC/8000
a=extmap:1 urn:ietf:params:<tbd>

m=video 0 QUIC/RTP/AVP 31 32
b=AS:1000
a=bundle-only
a=mid:bar
a=rtcp-mux
a=rtpmap:31 H261/90000
a=rtpmap:32 MPV/90000
a=extmap:2 urn:ietf:params:<tbd>
```

Figure 3: SDP Offer

Signaling details to be worked out.

8. Used RTP/RTCP packet types

Any RTP packet can be sent over QUIC and no RTCP packets are used by default. Since QUIC already includes some features which are usually implemented by certain RTCP messages, RTP over QUIC implementations should not need to implement the following RTCP messages:

* PT=205, FMT=1, Name=Generic NACK: Provides Negative Acknowledgments [RFC4585]. Acknowledgment and loss notifications are already provided by the QUIC connection.

* PT=205, FMT=8, Name=RTCP-ECN-FB: Provides RTCP ECN Feedback [RFC6679]. If supported, ECN may directly be exposed by the used QUIC implementation.

* PT=205, FMT=11, Name=CCFB: RTP Congestion Control Feedback which contains receive marks, timestamps and ECN notifications for each received packet [RFC8888]. This can be inferred from QUIC as described in Section 6.

* PT=210, FMT=all, Name=Token, [RFC6284] specifies a way to dynamically assign ports for RTP receivers. Since QUIC connections manage ports on their own, this is not required for RTP over QUIC.

9. Enhancements

The RTT statistics collected by QUIC may not be very precise because it can be influenced by delayed ACKs. An alternative to the RTT is to explicitly measure a one way delay. [QUIC-TS] suggests an extension for QUIC to implement one way delay measurements using a timestamp carried in a special QUIC frame. The new frame carries the time at which a packet was sent. This timestamp can be used by the receiver to estimate a one way delay as the difference between the time at which a packet was received and the timestamp in the received packet. The one way delay can then be used as a replacement for the receive time estimation derived from the RTT as described in Section 6 to create the pkt_delay_list.

Editor's note: Even with one-way delay measurements it is still

not possible to identify exact timestamps for individual packets, since the timestamp may be sent with an ACK that acks more than one earlier packet.

10. Discussion

10.1. Impact of Connection Migration

11. Security Considerations

TBD

12. IANA Considerations

This document has no IANA actions.

13. References

13.1. Normative References

[H3-DATAGRAM]
Schinazi, D., Ed., "Using QUIC Datagrams with HTTP/3", Work in Progress, Internet-Draft, draft-schinazi-quic-h3-datagram-05, <https://datatracker.ietf.org/doc/html/draft-schinazi-quic-h3-datagram-05>.

[QUIC-DATAGRAM]
Pauly, T., Ed., Kinnear, E., Ed., and D. Schinazi, Ed., "An Unreliable Datagram Extension to QUIC", Work in Progress, Internet-Draft, draft-ietf-quic-datagram-02, <https://datatracker.ietf.org/doc/html/draft-ietf-quic-datagram-02>.

[QUIC-TS] Huitema, C., Ed., "Quic Timestamps For Measuring One-Way Delays", Work in Progress, Internet-Draft, draft-huitema-quic-ts-05, <https://datatracker.ietf.org/doc/html/draft-huitema-quic-ts-05>.

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

[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>.

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

[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>.

[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>.

[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>.

[RFC8122] Lennox, J. and C. Holmberg, "Connection-Oriented Media Transport over the Transport Layer Security (TLS) Protocol in the Session Description Protocol (SDP)", RFC 8122, DOI 10.17487/RFC8122, March 2017, <https://www.rfc-editor.org/rfc/rfc8122>.

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

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

[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>.

[RFC8843] Holmberg, C., Alvestrand, H., and C. Jennings, "Negotiating Media Multiplexing Using the Session Description Protocol (SDP)", RFC 8843, DOI 10.17487/RFC8843, January 2021, <https://www.rfc-editor.org/rfc/rfc8843>.

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

[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>.

[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>.

[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>.

13.2. Informative References

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

Acknowledgments

TODO acknowledge.

Authors' Addresses

Jörg Ott
Technical University Munich

Email: ott@in.tum.de

Mathis Engelbart
Technical University Munich

Email: mathis.engelbart@gmail.com