Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: March 5, 2018

J. Ott
TUM
R. Even
Huawei
C. Perkins
University of Glasgow
V. Singh
callstats.io
September 1, 2017

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

Abstract

QUIC is a UDP-based protocol for congestion controlled reliable data transfer, while RTP serves carrying (conversational) real-time media over UDP. This draft discusses design aspects and issues of carrying RTP over QUIC.

Status of This Memo

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

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

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

This Internet-Draft will expire on March 5, 2018.

Table of Contents

1. Introduction

The Real-time Transport Protocol (RTP) [RFC3550] provides a framework for delivery of audio and video data for telephony, teleconferencing, video streaming, TV distribution, and other real-time applications. Previous work has defined the RTP data transfer protocol, along with numerous profiles, payload formats, and other extensions.

The QUIC transport protocol [I-D.ietf-quic-transport] [I-D.ietf-quic-tls] [I-D.ietf-quic-recovery] [I-D.ietf-quic-manageability] [I-D.ietf-quic-applicability] [I-D.ietf-quic-http] is a UDP-based, stream-multiplexing, encrypted transport protocol, primarily targeting web applications. When compared to the combination of TCP and TLS, QUIC reduced connection set-up times, improved congestion control, and stream multiplexing without head-of-line blocking.

RTP has typically been run over UDP or DTLS [RFC5763] [RFC5764], to leverage timely but unreliable data transfer as part of interactive application frameworks such as SIP [RFC3261] and WebRTC [I-D.ietf-rtcweb-overview] [I-D.ietf-rtcweb-rtp-usage], or to build on UDP/IP multicast support for large-scale managed TV distribution. A mapping of RTP onto TCP [RFC4571] has been widely used for video on demand applications using RTSP [RFC7826], although with relaxed delay bounds [Delay-TCP]. There is also an experimental mapping of RTP onto DCCP [RFC5762]. This memo explores how RTP can be run over QUIC. It has four main purposes:

1. to document use cases for RTP over QUIC, and to help understand when it's appropriate to use RTP and QUIC together (Section 2;
2. to understand and define a sensible mapping of RTP sessions onto one (or more) QUIC connections (Section 4);
3. to derive a wish-list for additional QUIC functionality to be fed into the QUIC WG (Section 5); and
4. to define the necessary signalling extensions to allow negotiation of RTP over QUIC (Section 6).

Editor's note: Section 5 is intended to document requirements for now and may disappear later if those are met or formally folded into a separate document. Also Section 4 and Section 6 may ultimately become separate drafts for consideration by different working groups (e.g., AVTCORE and MMUSIC).

2. Use Cases for RTP over QUIC

We identify the following possible use cases for RTP over QUIC:

1. Interactive peer-to-peer applications, such as telephony or video conferencing. Such applications operate in a trapezoid topology using a client-server signalling channel running SIP or WebRTC, and an associated peer-to-peer media path and/or data channel. Mappings of SIP and WebRTC onto QUIC are possible, but outside the scope of this memo. It might be desirable to transport the peer-to-peer RTP media path and data channel using QUIC, to leverage QUIC's security, stream demultiplexing, and congestion control features running over a single UDP port. This would simplify media demultiplexing, and potentially obviate the need for the congestion control work being done in the RMCAT working group. The design of QUIC makes it difficult however, since QUIC does not support peer-to-peer NAT traversal using STUN and ICE (and indeed uses a packet format that conflicts with STUN). These applications require low latency congestion control, and would benefit from unreliable delivery modes.
2. Interactive client-server applications. For example, a "click here to speak to a representative" button on a website that

starts an interactive WebRTC call. Such applications avoid the NAT traversal issues that complicate peer-to-peer use of QUIC, and can benefit from stream demultiplexing and (if appropriate algorithms are provided) congestion control. They would benefit from unreliable delivery modes to reduce latency.

3. Client-server video on demand applications using WebRTC or RTSP. These benefit from QUIC stream demultiplexing in the same way as interactive client-server applications, but with relaxed latency bounds that make them fit better with existing congestion control algorithms and reliable delivery.
4. Live video streaming from a server can also benefit from stream demultiplexing. If designed carefully, it should be easier to gateway RTP over QUIC into multicast RTP for scalable delivery than to gateway HTTP adaptive video over QUIC into multicast.

3. RTP-to-Transport Interface

The Real-time Transport Protocol defines the notion of RTP sessions to describe an elementary communication relationship between two or more parties. An RTP session comprises a uni-, bi-, or multidirectional flow of RTP packets carrying media as well as flows of RTCP packets providing feed forward from RTP senders to receivers and feedback from RTP receivers to senders.

Each media source is identified by a 32-bit Synchronization Source (SSRC) identifier, unique within an RTP session. An RTP session comprise the set of media sources that have the same view of the SSRC space. A single endpoint may use multiple SSRC identifiers (e.g., one for audio and one for video). Multiple media streams of a single endpoint are tied together by means of a common Canonical Name (CNAME) carried as part of the RTCP Source Description (SDES) packets. This allows receivers to, e.g., determine which media streams to synchronize.

Originally, in an RTP session the RTP and RTCP streams each used different port numbers, so that a single RTP session would use two port numbers (historically, when used with multicast conferencing, these were adjacent port numbers, RTP on the even and RTCP on the next higher odd port number). However, the use of unicast RTP has, (not just) due to the presence of NATs, motivated the multiplexing of both RTP and RTCP on a single port number [RFC5761]. The payload structure and number spaces used for RTP and RTCP packets were designed to support this easily.

The bundle framework [I-D.ietf-mmusic-sdp-bundle-negotiation] allows multiplexing of multiple RTP streams on a single address:port combination. All the RTP streams in a bundled group are part of a single RTP session sharing a single SSRC number space [RFC3550].

These two efforts also reduce the number of ICE candidates to be validated as part of a multimedia call or conference setup procedure. They are particularly required in conjunction with WebRTC to reduce the signaling and resource requirements, which would affect NATs as well as STUN and TURN servers. We note, however, that ICE is not currently usable with QUIC, since QUIC and STUN packets are not readily distinguished on a single UDP port, due to poor choice of packet formats.

WebRTC deserves particular consideration because its potential close relationship to QUIC: WebRTC uses HTTP/1.1 (possibly using WebSockets), or HTTP/2 to connect to web servers, and thus will likely use QUIC in the future as a signaling transport. Moreover, WebRTC supports peer-to-peer data channels, which currently target using SCTP over UDP over DTLS: SCTP for stream multiplexing within a connection and UDP for better NAT traversal properties. Since QUIC would seem to support these two functions, it could be a natural choice to be used for the data channel as well - although this would require changes to the QUIC packet formats to allow demultiplexing with STUN for NAT traversal.

For the actual media transmission, RTP use codec-specific payload formats that define how a piece of encoded media is broken down into data units that can fit into an MTU-sized packet for transmission. One important goal of RTP payload format design is allowing decoding packets as much as possible independent of each other as some may be lost due to the best-effort nature of the underlying UDP [RFC2736]. This implies, on the one hand, that RTP senders have to perform codec-level fragmentation in a semantically meaningful manner and, on the other hand, that are in control of packet boundaries and transmission scheduling and timing as well as retransmission decisions.

On the receiving side, RTP expects a detailed understanding of packet reception timing, possible reordering, and losses, as this information is used to ensure smooth media play-out, and is reported in the RTCP feedback statistics.

4. RTP-to-QUIC Mapping

This section address the necessary considerations to realize _one_ possible way of carrying RTP-over-QUIC.

Editor's note: At this point, this section is intended to explore the design space and briefly describe a number of different options without making specific recommendations about which option(s) to choose. Future revisions of this document move towards taking concrete decisions.

4.1. Mapping Semantic Units

RTP payload formats define a mapping of media data units (e.g., video or audio frames, audio samples, etc.) to packets. Assuming that we will preserve the structure of RTP header, optional header extension, and payload, there are two obvious options:

- Preserve the previous RTP assumptions about semantic fragmentation at MTU size boundaries; i.e., use the same packetization mechanism as before, just then drop the resulting RTP packet into a QUIC payload. Note that the MTU size may be smaller since QUIC packet headers are larger than plain UDP headers. This approach is most effective if the QUIC implementation allows the application to provide hints on where to fragment the QUIC stream into UDP packets at the sender side.
- Operate solely on semantic units such as video frames, and map each semantic unit to a QUIC payload. This approach leaves the final packetization decision to QUIC. In this case, our "MTU size" would not be defined by the IP layer but by QUIC. It is possible in this case for video frame composed of multiple RTP packets to use one RTP header for the whole video frame; no need to break the video frame to multiple RTP packet, put all payload as one RTP packet whose size may be bigger than MTU and send it as QUIC payload.

If we assume that semantic units are to be received and processed (and released to the application) atomically for best performance results, then option 2) would be preferred. If we consider that subunits are meaningful (e.g., slices in case of video), then option 1) may be preferred. This is heavily dependent on how tightly coupled are the application, RTP stack, and QUIC transport, and on what visibility and control is provided into the QUIC stream fragmentation, reception, and reception timing. In any case, however, it would be up to the payload definition to determine what a semantic unit.

4.2. Encapsulating Media Units

QUIC streams do not preserve packet boundaries, but rather offer a stream abstraction similar to that of TCP. Therefore, if multiple identifiable media units are to be transmitted on the same stream, the encapsulation mechanisms MUST provide boundaries for media data units, e.g., similar to the approach chosen for carrying RTP in TCP.

[Editor's note: QUIC requires a stream abstraction on the wire, but does it require the API offered to the application to provide a stream abstraction? Could a QUIC implementation that's tightly

integrated into the application provide more control without violating the on-the-wire protocol?]

The exception would be if only a single frame is ever transmitted across a single stream (see option 3 in section 3.3) so that stream termination signifies the end of the respective packet.

4.3. Mapping Media to Streams

There are (at least) three basic distinct options for mapping media to streams:

- Map an RTP session to a QUIC stream. In this case, all media packets of the RTP session would be carried within a single QUIC stream.
- Map an RTP stream to a QUIC stream. In case, as presently discussed in the QUIC WG, the QUIC stream would be unidirectional and we will have one QUIC stream per transmission direction.

Note that both options would map, e.g., FEC or retransmission sessions to different QUIC streams. Note also that both 1 and 2 implicitly create the problem of head-of-line blocking since QUIC streams are reliable and order preserving. This would thus not serve the real-time nature of RTP packets well. [Editor's note: to what extent are reliability and ordered required in the QUIC API? Provided the retransmission is made on the wire, is there anything stopping a QUIC implementation releasing data to the application out-of-order?]

- Map each independently decodable groups of frames, video frame, or even packet, depending on the encapsulation chosen to an individual QUIC stream. This is independent of whether streams, would be uni- or bi-directional.

Option 3 eliminates the head of line blocking problem of options 1. and 2. because QUIC does not provide any ordering across different streams. Using larger semantic units (e.g., GOPs) for stream mapping, would provide for more efficient stream number usage. However, all stream frames are still transmitted reliably. This implies that QUIC will perform retransmissions even for packets that would be too late already.

Mapping each video frame or packet to a different stream would raise an issue with stream numbering unless all RTP sessions are multiplexed on a single UDP socket anyway and then all RTP packets would simply be mapped to different streams.

An open question here would be how to deal with additional data channels that don't use RTP. Ideally, it should be possible that those be within the same QUIC connection (if QUIC is used as transport) to avoid consuming again more port numbers. Since, on the one hand, data channels can be set up and torn down at any time and, on the other hand, media packets are transmitted continuously, a need arises to set aside streams for data channels. One option would be "reserving" those streams in some form. But then, how many to reserve? Moreover, this would be incompatible with the slides stream number window being used by QUIC. Alternatively, one would need to synchronize the use of QUIC streams in real-time between the signaling and application channels and the media packet transmission. This may be hard to achieve and also suffers from the problem of the stream id window moving fast with frame transmissions. A third option would be adding another demultiplexing structure (e.g., to different RTP headers from data packets) and use a similar scheme of one application data unit (ADU) per stream for other applications. While feasible, this appears somewhat cumbersome in the mapping.

We finally need to consider inter RTP stream synchronisation and how/ if this would be affected by use of multiple QUIC streams.

None of the above schemes appear truly satisfactory from a system design perspective. This may call for some refined design considerations for QUIC, which we will begin discussing in section 4.

4.4. Mapping RTCP packets

RTCP is a bi-directional stream unlike RTP streams which are unidirectional. There can be for example a video stream receiver that only receives video content but will send and receive RTCP messages.

The current discussion on uni-directional streams direction will allow both uni- and bi-directional QUIC streams in the same QUIC connection. Such a solution will allow multiplexing of RTP and RTCP streams in the same QUIC connection.

An issue to consider is the encryption of RTCP messages. The RTP secure profiles RTP/SAVP [RFC3711] and RTP/SAVPF [RFC5124] allow NULL cipher for RTCP with message integrity. Using a NULL cipher allow RTP middleboxes to monitor the RTP delivery quality (the QUIC connection is encrypted as normal, this relates to whether the data within the QUIC connection is itself encrypted; c.f. the PERC working group).

Whether to use a single stream for forward RTCP and another for reverse could be a function of the streams being uni- or

bidirectional in the end. Another question to answer is if there should be one stream per SSRC per direction for RTCP. Finally, RTCP packets may also be lost and they contain timing information. Avoiding HoL blocking may thus also be important.

4.5. Mapping of RTP header extensions

QUIC provides a reliable protocol which addresses the requirement in [I-D.ietf-avtcore-rfc5285-bis] to transmit the RTP header extension in a couple of RTP packets to provide better reliability. Still if we will adopt mapping option 3 each RTP packet or media frame will use a separate QUIC stream. If a packet with RTP header extension is blocked the consecutive RTP packet will continue to arrive; in this case it will be beneficial to transmit the RTP header extensions more than once to allow for its arrival by the receiver. Using QUIC as a transport for RTP will have all RTP header extensions encrypted allowing only entities that terminate a QUIC connection to decode them. RTP header extension as defined in [I-D.ietf-avtcore-rfc5285-bis] can be sent in the clear and provide information to RTP middleboxes enabling them to route encrypted RTP packets. Currently the following header extensions are used for routing of encrypted RTP streams. Client to mixer audio level [RFC6464]. Frame marking [I-D.ietf-avtext-framemarking] and splicing interval [I-D.ietf-avtext-splicing-notification].

Editor's note: need to be clearer about the role of RTP middleboxes as specified in RTP topologies [RFC7667] connected by QUIC connections, and what is encrypted/authenticated end-to-end across the mesh of QUIC connections in that topology, and what is only protected hop-by-hop by QUIC.

5. Design considerations for QUIC

This section will address design implications for QUIC and the interaction with QUIC of both RTP and RTCP. In this version, this section is still very rudimentary and only identifies some of the aspects we expect to discuss in the future:

5.1. Reliability (or restransmission) control for stream frames

RTP packets are usually transmitted over unreliable UDP transport, with RTP being in full control of timing and, as applicable, of error resilience mechanisms.

QUIC supports only full reliability at this point and would retransmit lost packets even if they are no longer needed. While using indepdent streams for different media units could prevent head-of-line blocking, retrasmissions would appear to still happen. To

deal with this "issue", it may be worthwhile considering ways to control (re)transmission in a fine-grained fashion, e.g., by means of supporting partial realibility or by providing access to QUIC buffers for (re)transmission control.

5.2. Congestion control adaptation

QUIC defines a congestion control mechanism the feasibility of which for real-time media streams is yet to be understood. Media codecs have their own constraints for adapting the media transmission rate (in terms of reactivity and granularity) and the RMCAT working group is currently considering a number of options for real-time media congestion control (e.g., [I-D.ietf-rmcat-scream-cc] [I-D.ietf-rmcat-gcc] [I-D.ietf-rmcat-nada] [I-D.ietf-rmcat-coupled-cc] [I-D.singh-rmcat-adaptive-fec]), in addition to the basic circuit breaker mechanism [RFC8083]). It is an open question the extent to which these congestion control algorithms, or approaches inspired by them, ought to be incorporated into QUIC.

5.3. RTCP mapping

RTCP provides feed forward and feedback about the media channel, with extensions supporting very detailed per packet reporting. The reception statistics partly overlap with what QUIC ACKs provide (especially ACK/NACK ranges and per-packet timestamps). RTCP algorithms could benefit from obtaining access to these statistics via a local API to avoid redundancy.

RTCP packets must also be mapped to QUIC frames (and streams). Since RTP and RTCP can be multiplexed on the same transport address, as long as payload boundaries are preserved, RTCP packets could go onto any stream. However, since RTCP packets are used for RTT measurements, they should be transmitted independent of the RTP packets and ideally without blocking, so that head-of-line blocking by other packets should be avoided. If RTT measurements can be imported from QUIC (see above), exact timing control of RTCP packets won't be necessary; yet RTCP packets contain other information that require timely delivery. Similar to RTP, RTCP does not require reliable delivery.

5.4. API

We will need to understand how (if at all) a QUIC API could (and if it should) provide the necessary support for RTP/RTCP transmission and reception. This could include transmission timing control; providing transmission and reception timestamps; supporting retransmission control and/or buffer managements, among others. The

extent to which the principle of application level framing [ALF] should be incorporated into QUIC implementations, and how tightly coupled those implementations can be to the RTP stack and application, is unclear. How example, can a QUIC implementation deliver data out-of-order or allow control over stream fragmentation, both of which would improve performance for real-time media over RTP, provided it keeps the wire format unchanged? The service model needs to become clearer.

5.5. Multiparty

RTP is explicitly a group communication protocol, even when unicast. If we assume multicast QUIC is undesirable, there needs to be a scoping discussion around topologies.

Usual web-based conferencing services use one or more central system(s) for mixing or forwarding. Whenever the media streams do not require processing at such an entity but are merely forwarded, SRTP can provide the necessary end-to-end encryption. In contrast, QUIC "just" provides a secure channel between the endpoints and the central entities. To this end, SRTP could be applied inside QUIC for certain scenarios.

6. SDP Extensions for Negotiating RTP-over-QUIC

TBD

7. Security Considerations

RTP is used as a plain payload for QUIC, exploiting its multiplexing capabilities. To this end, the RTP packets are protected (confidentiality) by the QUIC security mechanisms. Hence, the security considerations pertinent to QUIC apply.

QUIC is by its very nature a transport layer security mechanisms. RTP traffic will thus be protected on a single transport hop only. As soon RTP topologies more complex than a point-to-point connection are used (e.g., [RFC7667]), RTP traffic will lose its end-to-end protection as transport connections are terminated at the intermediary, even if this acts just as a relay.

8. IANA Considerations

There are no IANA considerations at this point.

9. Acknowledgments

10. References

10.1. Normative References

[I-D.ietf-quic-recovery]
Iyengar, J. and I. Swett, "QUIC Loss Detection and Congestion Control", draft-ietf-quic-recovery-05 (work in progress), August 2017.

[I-D.ietf-quic-tls]
Thomson, M. and S. Turner, "Using Transport Layer Security (TLS) to Secure QUIC", draft-ietf-quic-tls-05 (work in progress), August 2017.

[I-D.ietf-quic-transport]
Iyengar, J. and M. Thomson, "QUIC: A UDP-Based Multiplexed and Secure Transport", draft-ietf-quic-transport-05 (work in progress), August 2017.

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

10.2. Informative References

[ALF] Clark, D. and D. Tennenhouse, "Architectural Considerations for a New Generation of Network Protocols", Proceedings of ACM SIGCOMM, 1990.

[Delay-TCP]
Brosh, E., Baset, S., Rubinstein, D., and H. Schulzrinne, "The Delay-Friendliness of TCP", Proceedings of ACM SIGMETRICS, 2008.

[I-D.ietf-avtcore-rfc5285-bis]
Singer, D., Desineni, H., and R. Even, "A General Mechanism for RTP Header Extensions", draft-ietf-avtcore-rfc5285-bis-14 (work in progress), August 2017.

[I-D.ietf-avtext-framemarking]
Berger, E., Nandakumar, S., and M. Zanaty, "Frame Marking RTP Header Extension", draft-ietf-avtext-framemarking-05 (work in progress), July 2017.

[I-D.ietf-avtext-splicing-notification]
Xia, J., Even, R., Huang, R., and D. Lingli, "RTP/RTCP extension for RTP Splicing Notification", draft-ietf-avtext-splicing-notification-09 (work in progress), August 2016.

[I-D.ietf-mmusic-sdp-bundle-negotiation]
Holmberg, C., Alvestrand, H., and C. Jennings, "Negotiating Media Multiplexing Using the Session Description Protocol (SDP)", draft-ietf-mmusic-sdp-bundle-negotiation-38 (work in progress), April 2017.

[I-D.ietf-quic-applicability]
Kuehlewind, M. and B. Trammell, "Applicability of the QUIC Transport Protocol", draft-ietf-quic-applicability-00 (work in progress), July 2017.

[I-D.ietf-quic-http]
Bishop, M., "Hypertext Transfer Protocol (HTTP) over QUIC", draft-ietf-quic-http-05 (work in progress), August 2017.

[I-D.ietf-quic-manageability]
Kuehlewind, M., Trammell, B., and D. Druta, "Manageability of the QUIC Transport Protocol", draft-ietf-quic-manageability-00 (work in progress), July 2017.

[I-D.ietf-rmcat-coupled-cc]
Islam, S., Welzl, M., and S. Gjessing, "Coupled congestion control for RTP media", draft-ietf-rmcat-coupled-cc-06 (work in progress), March 2017.

[I-D.ietf-rmcat-gcc]
Holmer, S., Lundin, H., Carlucci, G., Cicco, L., and S. Mascolo, "A Google Congestion Control Algorithm for Real-Time Communication", draft-ietf-rmcat-gcc-02 (work in progress), July 2016.

[I-D.ietf-rmcat-nada]
Zhu, X., Pan, R., Ramalho, M., Cruz, S., Jones, P., Fu, J., and S. D'Aronco, "NADA: A Unified Congestion Control Scheme for Real-Time Media", draft-ietf-rmcat-nada-04 (work in progress), March 2017.

[I-D.ietf-rmcat-scream-cc]
Johansson, I. and Z. Sarker, "Self-Clocked Rate Adaptation for Multimedia", draft-ietf-rmcat-scream-cc-10 (work in progress), July 2017.

[I-D.ietf-rtcweb-overview]
Alvestrand, H., "Overview: Real Time Protocols for Browser-based Applications", draft-ietf-rtcweb-overview-18 (work in progress), March 2017.

[I-D.ietf-rtcweb-rtp-usage]
Perkins, C., Westerlund, M., and J. Ott, "Web Real-Time Communication (WebRTC): Media Transport and Use of RTP", draft-ietf-rtcweb-rtp-usage-26 (work in progress), March 2016.

[I-D.singh-rmcat-adaptive-fec]
Singh, V., Nagy, M., Ott, J., and L. Eggert, "Congestion Control Using FEC for Conversational Media", draft-singh-rmcat-adaptive-fec-03 (work in progress), March 2016.

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

[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/info/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/info/rfc3711>.

[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/info/rfc4571>.

[RFC5124] Ott, J. and E. Carrara, "Extended Secure RTP Profile for Real-time Transport Control Protocol (RTCP)-Based Feedback (RTP/SAVPF)", RFC 5124, DOI 10.17487/RFC5124, February 2008, <https://www.rfc-editor.org/info/rfc5124>.

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

[RFC5762] Perkins, C., "RTP and the Datagram Congestion Control Protocol (DCCP)", RFC 5762, DOI 10.17487/RFC5762, April 2010, <https://www.rfc-editor.org/info/rfc5762>.

[RFC5763] Fischl, J., Tschofenig, H., and E. Rescorla, "Framework for Establishing a Secure Real-time Transport Protocol (SRTP) Security Context Using Datagram Transport Layer Security (DTLS)", RFC 5763, DOI 10.17487/RFC5763, May 2010, <https://www.rfc-editor.org/info/rfc5763>.

[RFC5764] McGrew, D. and E. Rescorla, "Datagram Transport Layer Security (DTLS) Extension to Establish Keys for the Secure Real-time Transport Protocol (SRTP)", RFC 5764, DOI 10.17487/RFC5764, May 2010, <https://www.rfc-editor.org/info/rfc5764>.

[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/info/rfc6464>.

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

[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/info/rfc7826>.

[RFC8083] Perkins, C. and V. Singh, "Multimedia Congestion Control: Circuit Breakers for Unicast RTP Sessions", RFC 8083, DOI 10.17487/RFC8083, March 2017, <https://www.rfc-editor.org/info/rfc8083>.

Authors' Addresses

Joerg Ott
Technische Universitaet Muenchen
Boltzmannstrasse 3
Garching bei Muenchen
Germany

Email: ott@in.tum.de

Roni Even
Huawei
Israel

Email: roni.even@huawei.com

Colin Perkins
University of Glasgow
School of Computing Science
Glasgow G12 8QQ
UK

Email: csp@csperkins.org
URI: https://csperkins.org/

Varun Singh
CALLSTATS I/O Oy
Annankatu 31-33 C 42
Helsinki 00100
Finland

Email: varun@callstats.io
URI: https://www.callstats.io/about