Workgroup: AVTCORE/MMUSIC Working Groups
Internet-Draft:
draft-gruessing-moq-requirements-00
Published: 25 October 2021
Intended Status: Informational
Expires: 28 April 2022
Authors: J. Gruessing S. Dawkins
Tencent America LLC

QUIC Encapsulation for Media over RTP - Requirements and Use Cases

Abstract

This document describes the uses cases, requirements, and considerations that should guide the design of the encapsulation of a real-time media transport protocol as a payload in the QUIC protocol.

Note to Readers

RFC Editor: please remove this section before publication

Source code and issues for this draft can be found at https://github.com/fiestajetsam/draft-gruessing-moq-requirements.

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

This document describes the uses cases, requirements, and considerations that should guide the design of the encapsulation of a real-time media transport protocol as a payload in the the QUIC protocol [RFC9000].

Protocol developers have been considering the implications of the QUIC protocol ([RFC9000]) on media transport for several years, but the initial focus on QUIC in the IETF was to support web applications that used the HTTP/3 protocol [I-D.draft-ietf-quic-http]. The completion of the initial versions of the QUIC specifications, and the adoption of [I-D.draft-ietf-quic-datagram], have cleared the way for proposals to use QUIC as a media transport. This document considers a number of proposals for "Media Over QUIC", and analyzes them to understand requirements and considerations.

2. Terminology

For the purposes of this document, it is assumed that we are starting with a protocol stack that looks like this:

```
Media Transport Protocol | Media
```

and the goal is to provide a protocol stack that looks like this:

```
QUIC | Media Transport Protocol | Media
```

3. Prior and Existing Specifications

Several existing draft specifications and protocols already exist which base their implementation around using existing Media Transport Protocols on top of QUIC, or define their own. With the exception of RUSH (Section 3.3), it is unknown if the other specifications have had any deployments or interop with multiple implementations.

3.1. QRT: QUIC RTP Tunnelling

[I-D.draft-hurst-quic-rtp-tunnelling]

QRT encapsulates RTP and RTCP and define the means of using QUIC datagrams with them, defining a new payload within a datagram frame which distinguishes packets for a RTP packet flow vs RTCP.

3.2. RTP over QUIC

[I-D.draft-engelbart-rtp-over-quic]

This specification also encapsulates RTP and RTCP but unlike QRT which simply relies on the default QUIC congestion control mechanisms, it defines a set of requirements around QUIC implementation's congestion controller to permit the use of separate control algorithms.

3.3. RUSH - Reliable (unreliable) streaming protocol

[I-D.draft-kpugin-rush]

RUSH uses its own frame types on top of QUIC as it pre-dates the datagram specification; in addition individual media frames are given their own stream identifiers to remove HoL blocking from processing out-of-order.

It defines its own registry for signalling codec information with room for future expansion but presently is limited to a subset of popular video and audio codecs and doesn't include other types (such as subtitles, transcriptions, or other signalling information) out of bitstream.

3.4. Tunnelling SRT over QUIC

[I-D.draft-sharabayko-srt-over-quic]

Secure Reliable Transport (SRT) ([I-D.draft-sharabayko-srt]) itself is a general purpose transport protocol primarily for contribution transport use cases and this specification covers the encapsulation and delivery of SRT on top of QUIC using datagram frame types. This specification sets some requirements regarding how the two interact and leaves considerations for congestion control and pacing to prevent conflict between the two protocols.

3.5. Comparison of Existing Specifications

* Both QRT and the Engelbart draft attempt to use existing payloads of RTP, RTCP, and SDP unlike RUSH and SRT, as well as using existing Datagram frames
* RUSH introduces new frame types as its development pre-dates Datagram frames
* All drafts take differing approaches to flow/stream identification and management; some address congestion control and others just omit the subject and leave it to QUIC to handle
* Both QRT and RUSH specify ALPN identification; the Engelbart and SRT drafts do not.

4. Use Cases

4.1. Use Cases From *[I-D.draft-rtpfolks-quic-rtp-over-quic]*

An early draft in the "media over QUIC" space, [I-D.draft-rtpfolks-quic-rtp-over-quic], defined several key use cases. The following sections are taken from that draft, with minimal editing.

4.1.1. Interactive peer-to-peer applications

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.

4.1.2. Interactive client-server applications

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.

4.1.3. Client-server video on demand applications using WebRTC or RTSP

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.

4.1.4. Live video streaming from a server

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.

4.2. Suggested Use Cases for "Media Over QUIC" Going Forward

The use cases that are most applicable today given the existing and known future capabilities of QUIC are included in this section.

Editor Note: this section is a work in progress, and is based on the opinions of the draft authors. We are happy to be guided by discussion about other use cases.

4.2.1. Unidirectional live stream contribution

Unidirectional live stream contribution. Two immediate scenarios that best describe this is firstly users on a streaming platform in a remote scenario from their phone live streaming an event or going on to an audience in real time in relatively low bitrates (~1-5Mbit). The second scenario is larger bitrate contribution feeds in broadcast. This can be an OB feed "back to base" into playout gallery, or from playout facilities to online distribution platforms.

4.2.2. Distribution from platform to audience

Distribution from platform to audience. Whilst use of WebRTC or RTSP today for On-Demand media streaming is not typical with adaptive streaming like HLS and DASH being predominantly used as WebRTC is more applicable in latency sensitive contexts such as live sporting events. Instead use cases where there is live streaming of TV linear output, or live streaming such as Twitch or Facebook, or non-UGC services like OTT offerings made by broadcasters.

5. Requirements

Even a cursory examination of the existing proposals listed in Section 3 shows that there are fundamental differences in the approaches being used - for instance, whether a proposal uses RTP as its Media Transport Protocol.

In this section, we attempt to focus on high-level requirements for real time media streaming over a QUIC connection, recognizing that

* additional analysis will be required, and
* we are starting with requirements that are apparent for RTP-based proposals

5.1. Codec Agility

When initiating a media session, both the sender and receiver should be able to negotiate the codecs, bitrates and other media details based on capabilities and preferences.

5.2. Support a range of Latencies

TODO: confirm requirements for latency

[I-D.draft-ietf-mops-streaming-opcons] describes these latency requirements for streaming media.

- *ultra low-latency (less than 1 second)
- *low-latency live (less than 10 seconds)
- *non-low-latency live (10 seconds to a few minutes)
- *on-demand (hours or more)

5.3. Congestion Control

TODO: Confirm these requirements, consider looking at how RFC 8836 applies to this requirement.

5.4. Support Lossless and Lossy Media Transport

TODO: confirm scope of this draft to describe lossless media transport, lossy media transport, or both lossless and lossy transport.

5.5. Flow Directionality

Media should be able to flow in either direction from client to server or vice-versa, either individually or concurrently.

5.6. WebTransport

TODO: Unsure if this should be a requirement. If it is, we have to consider two things: WebTransport supports HTTP/2, are we going to explicitly exclude it? Also, WebTransport [I-D.draft-ietf-webtrans-overview] has normative language around congestion control which may be at odds with our potential requirements.

5.7. Authentication

The encapsulation SHOULD have capabilities beyond what QUIC provides to allow hosts to authenticate one another, this should be kept simple but robust in nature to prevent attacks like credential brute-forcing.

TODO: More details are required here

6. Non-requirements

This section covers topics that are explicitly out of scope for the time being.

6.1. NAT Traversal

From Section 8.2 of [RFC9000]:

> Path validation is not designed as a NAT traversal mechanism. Though the mechanism described here might be effective for the creation of NAT bindings that support NAT traversal, the expectation is that one endpoint is able to receive packets without first having sent a packet on that path. Effective NAT traversal needs additional synchronization mechanisms that are not provided here.

Although there are use cases that would benefit from a mechanism for NAT traversal, a QUIC protocol extention would be required to support those use cases today.

6.2. New Media Transport Protocols

The creation of new media transport protocols should be avoided, and instead we should make use of RTP [RFC3550] and the existing ecosystem of payload formats and methods of signalling where possible. Work on QUIC encapsulation may reveal a need to extend these specificiations; in which case we should work with the relevant working groups and present our use-cases.

6.3. Multicast

Even if multicast and other network broadcasting capabilities are often used in delivering media in our use cases, QUIC doesn't yet support multicast, and would require a QUIC protocol extension to do so. In addition, the inclusion of multicast would introduce more complexity in both the specification and client implimentations.

7. IANA Considerations

This document makes no requests of IANA.

8. Security Considerations

As this document is intended to guide discussion and consensus, it introduces no security considerations of its own.

9. References

9.1. Normative References

[I-D.draft-ietf-mops-streaming-opcons]
Holland, J., Begen, A., and S. Dawkins, "Operational Considerations for Streaming Media", Work in Progress, Internet-Draft, draft-ietf-mops-streaming-opcons-07, 14 September 2021, <https://datatracker.ietf.org/doc/html/draft-ietf-mops-streaming-opcons-07>.

[I-D.draft-ietf-quic-datagram] Pauly, T., Kinnear, E., and D. Schinazi, "An Unreliable Datagram Extension to QUIC", Work in Progress, Internet-Draft, draft-ietf-quic-datagram-06, 5 October 2021, <https://datatracker.ietf.org/doc/html/draft-ietf-quic-datagram-06>.

[I-D.draft-ietf-webtrans-overview]
Vasiliev, V., "The WebTransport Protocol Framework", Work in Progress, Internet-Draft, draft-ietf-webtrans-overview-02, 28 July 2021, <https://datatracker.ietf.org/doc/html/draft-ietf-webtrans-overview-02>.

[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>.

9.2. Informative References

[I-D.draft-engelbart-rtp-over-quic] Ott, J. and M. Engelbart, "RTP over QUIC", Work in Progress, Internet-Draft, draft-engelbart-rtp-over-quic-00, 12 July 2021, <https://datatracker.ietf.org/doc/html/draft-engelbart-rtp-over-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-quic-http]
Bishop, M., "Hypertext Transfer Protocol Version 3 (HTTP/3)", Work in Progress, Internet-Draft, draft-ietf-quic-http-34, 2 February 2021, <https://datatracker.ietf.org/doc/html/draft-ietf-quic-http-34>.

[I-D.draft-kpugin-rush] Pugin, K., Frindell, A., Cenzano, J., and J. Weissman, "RUSH - Reliable (unreliable) streaming protocol", Work in Progress, Internet-Draft, draft-

kpugin-rush-00, 12 July 2021, <https://datatracker.ietf.org/doc/html/draft-kpugin-rush-00>.

[I-D.draft-rtpfolks-quic-rtp-over-quic] Ott, J., Even, R., Perkins, C., and V. Singh, "RTP over QUIC", Work in Progress, Internet-Draft, draft-rtpfolks-quic-rtp-over-quic-01, 1 September 2017, <https://datatracker.ietf.org/doc/html/draft-rtpfolks-quic-rtp-over-quic-01>.

[I-D.draft-sharabayko-srt] Sharabayko, M., Sharabayko, M., Dube, J., Kim, J., and J. Kim, "The SRT Protocol", Work in Progress, Internet-Draft, draft-sharabayko-srt-01, 7 September 2021, <https://datatracker.ietf.org/doc/html/draft-sharabayko-srt-01>.

[I-D.draft-sharabayko-srt-over-quic] Sharabayko, M. and M. Sharabayko, "Tunnelling SRT over QUIC", Work in Progress, Internet-Draft, draft-sharabayko-srt-over-quic-00, 28 July 2021, <https://datatracker.ietf.org/doc/html/draft-sharabayko-srt-over-quic-00>.

[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>.

Appendix A. Acknowledgements

The authors would like the thank the many authors of of the specifications referenced in Section 3 for their work:

*Alan Frindell
*Colin Perkins
*Jake Weissman
*Joerg Ott
*Jordi Cenzano
*Kirill Pugin
*Maria Sharabayko
*Mathis Engelbart
*Maxim Sharabayko
*Roni Even
*Sam Hurst
*Varun Singh

James Gruessing would also like to thank Francesco Illy and Nicholas Book for their part in providing the needed motivation.

Authors' Addresses

James Gruessing

Email: james.ietf@gmail.com

Spencer Dawkins
Tencent America LLC
United States of America

Email: spencerdawkins.ietf@gmail.com