Network Working Group
Internet-Draft
Intended status: Informational
Expires: 11 January 2024

C. Jennings
S. Nandakumar
M. Zanaty
Cisco

10 July 2023

MOQ Usages for audio and video applications
draft-jennings-moq-usages-00

Abstract

Media over QUIC Transport (MOQT) defines a publish/subscribe based unified media delivery protocol for delivering media for streaming and interactive applications over QUIC. This specification defines details for building audio and video applications over MOQT, more specifically, provides information on mapping application media objects to the MOQT object model and possible mapping of the same to underlying QUIC transport.

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 11 January 2024.

Table of Contents

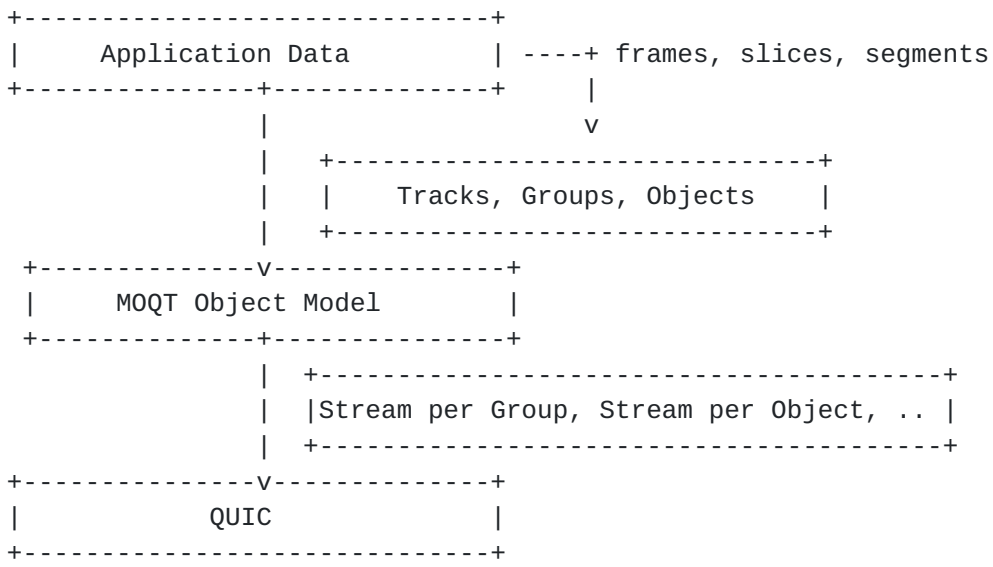
1. Introduction

Media Over QUIC Transport (MOQT) [MoQTransport] allows set of publishers and subscribers to participate in the media delivery over QUIC for streaming and interactive applications. The MOQT specification defines the necessary protocol machinery for the endpoints and relays to participate, however, it doesn't provide recommendations for media applications on using MOQT object model and mapping the same to underlying QUIC transport.

This document introduces MOQT's object model mapping to underlying QUIC transport in Section 2. Section 3 and Section 4 describe various grouping of application level media objects and their mapping to the MOQT object model. Section 5.2 discusses considerations when using multiple quality video applications, such as simulcast and/or layer coding, over the MOQT protocol.Section 8 describes considerations for adaptive bitrate techniques and finally the Section 6 discusses interactions when priorities are used on objects

and tracks.

Below picture captures the conceptual model showing mapping at various levels of a typical media application stack using MOQT delivery protocol.

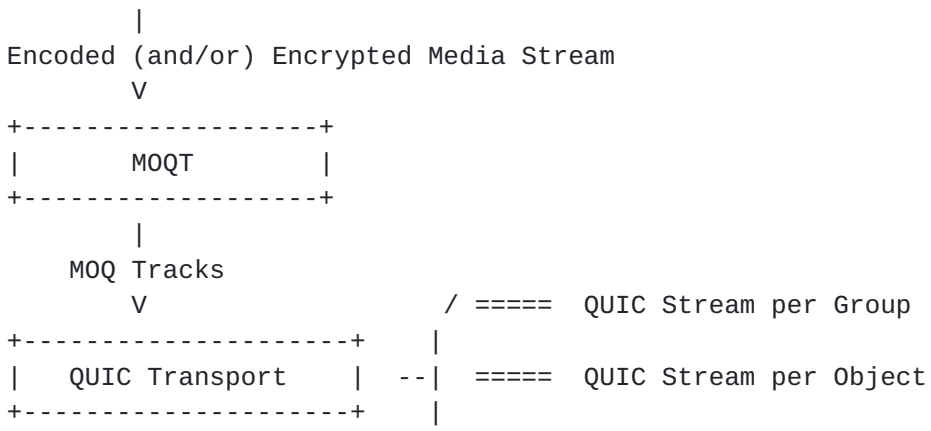


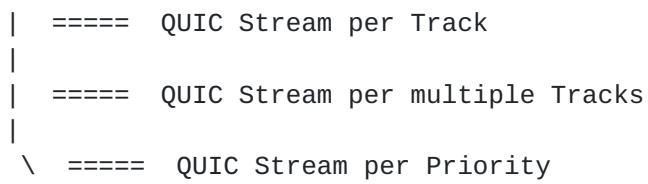
1.1. Requirements Notation and Conventions

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 [RFC2119].

2. MOQT QUIC Mapping

In a typical MOQT media application, the captured media from a media source is encoded (compressed), encrypted (based on the encryption scheme), packaged (based on the container format) and mapped onto MOQT object model. Applications (such as Media producers, Relays) deliver MOQT objects over QUIC transport by choosing the mapping that is appropriate based on the context.





Subsections below describes a few possibilities to consider when mapping MOQT objects to QUIC Streams.

2.1. Stream per MOQT Group

In this mode, an unidirectional QUIC stream is setup per MOQT Group (section 2.2 of [MoQTransport]). Following observations can be made about such a setup:

* MOQT groups typically represent things that share some kind of relationship (eg decodability, priority) and having the objects of a group share the same underlying stream context allows them to delivered coherently.

* Media consumers can map each incoming QUIC stream into decoding context in the order of their arrival per group.

* Since the objects within group share the QUIC stream, there is likelihood of increased end to end latency due to head-of-line blocking under losses.

2.2. Stream per MOQT Object

In this mode, an unidirectional QUIC stream is setup per MOQT Object (section 2.1 of [MoQTransport]). Following observations can be made about such a setup:

* Using a single stream per object can help reduce latency at the source especially when objects represent smaller units of the application data (say a single encoded frame).

* The impact of on-path losses to the end to end latency is scoped to the object duration. The smaller the object duration, lesser the impact on the end to end latency.

* One stream per object may end up in creating large number of streams, especially when the objects durations is small.

* Media consumers may need to re-organize the incoming streams for handling objects within a group, since streams may arrive out of order.

2.3. Stream per MOQT Track

In this mode, there is one unidirectional QUIC stream per MOQT Track (section 2.3 [MoQTransport]). Following observations can be made

about such a setup:

* This scheme is the simplest in its implementation and the streams stays active until the track exists. Endpoints need to maintain just one stream context per track.

* Since all the objects within the track share the same stream, the may be impact on end to end latency due to HOL blocking under loss.

2.4. Stream per Priority

In this mode, there is one unidirectional QUIC stream per MOQT Track/ Object priority. Following observations can be made about such a setup:

* This scheme is relatively simpler in its implementation and number of stream contexts match the number of priority levels.

* Such a scheme can be used at Relays when forwarding decisions can be naturally mapped to priorities carried in the object header.

2.5. Stream per multiple MOQT Tracks

This mode is similar Section 2.3 but with more than one MOQT track delivered over a single unidirectional QUIC stream, thus allowing implementations to map multiple incoming QUIC streams to a few outgoing QUIC Streams. Similar to Section 2.3, this mode is relatively simple in its implementation and at the same time may suffer from latency impacts under losses.

3. MoQ Audio Objects

Each chunk of encoded audio data, say 10ms, represents a MOQ Object. In this setup, there is one MOQT Object per MOQT Group, where, the Group Sequence in the object header is increment by one for each encoded audio chunk and the Object Sequence is defaulted to value 0. When mapped to the underlying QUIC Stream, each such unitary group is sent over individual unidirectional QUIC stream (similar to Section 2.2/Section 2.1).

Future sub 2 kbps audio codecs may take advantage of a rapidly updated model that are needed to decode the audio which could result in audio needing to use groups with multiple objects to ensure all the objects needed to decode some audio are in the same group.

4. MoQ Video Objects

The decision on what constitutes a MOQ object/group and its preferred mapping to the underlying QUIC transport for video streams is governed by the granularity of encoded bitstream, as chosen by the application. The smallest unit of such an application defined

encoded bitstream will be referred to as "Video Atom" in this specification and they are mapped 1:1 to MOQ Objects.

The size and duration of a video atom is application controlled and follows various strategies driven by application requirements such as latency, quality, bandwidth and so on.

Following subsections identify various granularities defining the video atoms and their corresponding mapping to MOQT object model and the underlying QUIC transport.

4.1. Encoded Frame

In this scheme, the video atom is a single encoded video frame. The Group Sequence is incremented by 1 at IDR Frame boundaries. The Object Sequence is increment by 1 for each video frame, starting at 0 and resetting to 0 at the start of new group. The first video frame (Object Sequence 0) should be IDR Frame and the rest of the video frames within a MOQT group are typically dependent frames (delta frames) and organized in the decode order.

When using QUIC mapping scheme defined in Section 2.2, each unidirectional QUIC stream is used to deliver one encoded frame. In this mode, the receiver application should manage out of order streams to ensure the MOQ Objects are delivered to the decoder in the increasing order of the Object Sequence within a group and then in the increasing order of the Group Sequence.

When using QUIC mapping as defined in {spg}, one unidirectional QUIC stream is setup to deliver all the encoded frames (objects) within a group.

4.2. Encoded Slice

In Slice-based encoding a single video frame is “sliced” into separate sections and are encoded simultaneously in parallel. Once encoded, each slice can then be immediately streamed to a decoder instead of waiting for the entire frame to be encoded first.

In this scheme, the video atom is a encoded slice, starting with the IDR frame as Object Sequence of 0 for that slice and followed by delta frames with Object Sequence incremented by 1 successively. A MOQT Group is identified by set of such objects at each IDR frame boundaries. To be able to successfully decode and render at the media consumer, the identifier of the containing video frame for the slice needs to be carried end to end. This would allow the media consumer to map the slices to right decoding context of the frame being processed.

> Note: The video frame identifier may be carried either as part of encoded object's payload header or introduce a group header for conveying the frame identifier.

When using QUIC mapping scheme defined in Section 2.2, each unidirectional QUIC stream is used to deliver one encoded slice of a video frame. When using QUIC mapping as defined in {spg}, each unidirectional QUIC stream is setup to deliver all the encoded slices (objects) within a group.

4.3. CMAF Chunk

CMAF [CMAF] chunks are CMAF addressable media objects that contain a consecutive subset of the media samples in a CMAF fragment. CMAF chunks can be used by a delivery protocol to deliver media samples as soon as possible during live encoding and streaming, i.e., typically less than a second. CMAF chunks enable the progressive encoding, delivery, and decoding of each CMAF fragment.

A given video application may choose to have chunk duration to span more than one encoded video frame. When using CMAF chunks, the video atom is a CMAF chunk. The CMAF chunk containing the IDR Frame shall have Object Sequence set to 0, with each additional chunk with its Object Sequence incremented by 1. The Group Sequence is incremented at every IDR interval and all the CMAF chunks within a given IDR interval shall be part of the same MOQT Group.

When using QUIC mapping scheme defined in Section 2.2, each unidirectional QUIC stream is used to deliver a CMAF Chunk. When using QUIC mapping as defined in {spg}, each unidirectional QUIC stream is setup to deliver all the CMAF chunks (objects) within a group. When using QUIC mapping defined in Section 2.3 CMAF chunks corresponding to CMAF track are delivered over the same unidirectional QUIC stream.

4.4. CMAF Fragment

CMAF fragments are the media objects that are encoded and decoded. For scenarios, where the fragments contains one or more complete coded and independently decodable video sequences, each such fragment is identified as single MOQT Object and it forms its own MOQT Group. There is one unidirectional QUIC stream per such an object Section 2.2. Media senders should stream the bytes of the object, in the decode order, as they are generated in order the reduce the latencies.

5. MOQT Track

MOQT Tracks are typically characterized by having a single encoding and optionally a encryption configuration. Applications can encoded a captured source stream into one or more qualities as described in the sub sections below.

5.1. Single Quality Media Streams

For scenarios where the media producer intents to publish single quality audio and video streams, applications shall map the objects from such audio and video streams to individual tracks enabling each track to represent a single quality.

5.2. Multiple Quality Media Streams

It is not uncommon for applications to support multiple qualities (renditions) per source stream to support receivers with varied capabilities, enabling adaptive bitrate media flows, for example. We describe 2 common approaches for supporting multiple qualities(renditions/encodings) - Simulcast and Layered Coding.

5.2.1. Simulcast

In simulcast, each MOQT track is an time-aligned alternate encoding (say, multiple resolutions) of the same source content. Simulcasting allows consumers to switch between tracks at group boundaries seamlessly.

Few observations:

* Catalog should identify time-aligned relationship between the simulcasted tracks.

* All the alternate encodings shall matching base timestamp and duration.

* All the alternate encodings are for the same source media stream.

* Media consumers can pick and choose the right quality by subscribing to the appropriate track.

* Media consumers react to changing network/bandwidth situations by subscribing to different quality track at the group boundaries.

5.2.2. Scalable Video Coding (SVC)

SVC defines a coded video representation in which a given bitstream offers representations of the source material at different levels of fidelity (spatial, quality, temporal) structured in a hierarchical manner. Such an organization allows bitstream to be extracted at lower bit rate than the complete sequence to enable decoding of pictures with multiple image structures (for sequences encoded with spatial scalability), pictures at multiple picture rates (for sequences encoded with temporal scalability), and/or pictures with multiple levels of image quality (for sequences encoded with SNR/ quality scalability). Different layers can be separated into different bitstreams. All decoders access the base stream; more capable decoders can access enhancement streams.

5.2.2.1. All layers in a single MOQT Track

In this mode, the video application transmits all the SVC layers under a single MOQT Track. When mapping to the MOQT object model, any of the methods described in Section 4 can be leveraged to mapped the encoded bitstream into MOQT groups and objects.

When transmitting all the layers as part of a single track, following properties needs to be considered:

* Catalog should identify the SVC Codec information in its codec definition.

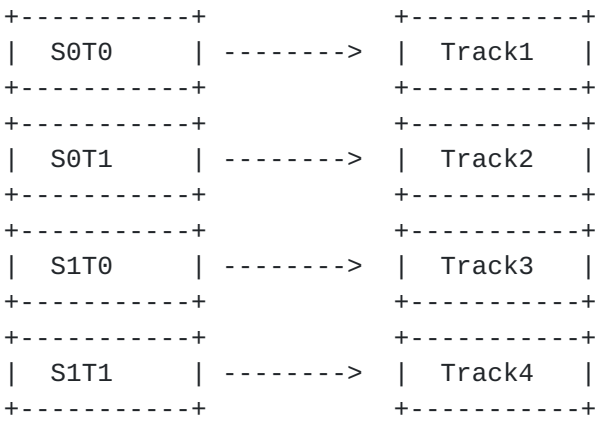
* Media producer should map each video atom to the MOQ object in the decode order and can utilize any of the QUIC mapping methods described in Section 2.

* Dependency information for all the layers (such as spatial/ temporal layer identifiers, dependent descriptions) are encoded in the bitstream and/or container.

The scheme to map all the layers to a single track is simple to implement and allows subscribers/media consumers to make independent layer drop decisions without needing any protocol exchanges (as needed in Section 5.2.1). However, such a scheme is constrained by disallowing selective subscriptions to the layers of interest.

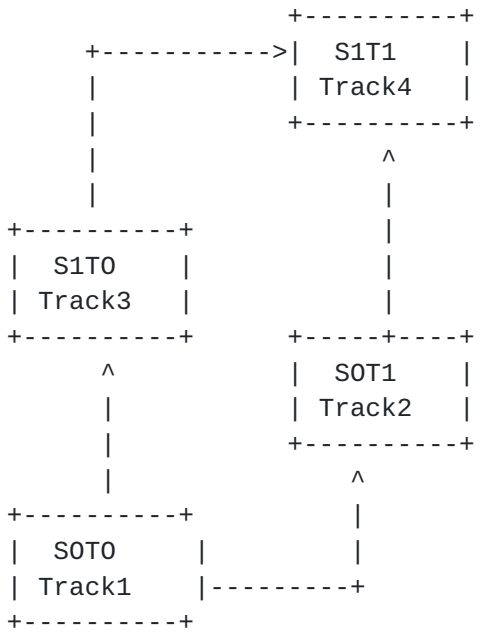
5.2.2.2. One SVC layer per MOQT Track

In this mode, each SVC layer is mapped to a MOQT Track. Each unique combination of fidelity (say spatial and temporal) is identified by a MOQT Track (see example below).



ex: 2-layer spatial and 2-layer temporal scalability encoding

The catalog should identify the complete list of dependent tracks for each track that is part of layered coding for a given media stream. For example the figure below shows a sample layer dependency structure (2 spatial and temporal layers) and corresponding tracks dependencies.



Catalog Track Dependencies:

Track2 depends on Track1
Track3 depends on Track1
Track4 depends on Track2 and Track3

Within each track, the encoded media for the given layer can follow mappings defined in Section 4 and can choose from the options defined in Section 2 for transporting the mapped objects over QUIC. The bitstream and/or the container should carry the neccessary to capture video frame level dependencies.

Media consumers would need to consider information from catalog to group the the related tracks and gather information from the bistream to establish frame level depedencies. This would allow the consumer to appropriately map the incoming QUIC streams and MOQ objects to the right decoder context.

5.2.3. k-SVC

k-SVC is a flavor of layered coding wherein the encoded frames within a layer depend only on the frames within the same layer, with the exception that the IDR frame in the enhancement layers depends on the IDR frame in the next level lower fidelity layer.

When each layer of a k-SVC encoded bitstream is mapped to a MOQT track, following needs to be taken into consideration:

* Catalog should identify the tracks are related via k-SVC dependency

* MOQT protocol should be extended to propose a group header that enables track for the enhancement layer to identify the group sequence of its dependent track for satisfying IDR Frame dependency.

6. Object and Track Priorities

Media producers are free to prioritize media delivery between the tracks by encoding priority information in the MOQT Object Header for a given track. Relay can utilize these priorities to make forwarding decisions. "draft-zanaty-moq-priority" specifies a prioritization mechanism for objects delivered using the Media over QUIC Transport (MOQT) protocol.

7. Relay Considerations

Relays are not allowed to modify MOQT object header, as it might break encryption and authentication. However, Relays are free to apply any of the transport mappings defined in Section 2 that it sees fit based on the local decisions.

For example, a well engineered Relay network may choose to take multiple incoming QUIC streams and map it to few outgoing QUIC streams (similar to one defined in Section 2.3) or the Relays may choose MOQT object priorities Section 2.4 to decide the necessary transport mapping. It is important to observe that such decisions cam be made solely considering the MOQT Object header information.

8. Bitrate Adaptation

TODO: add considerations for client side ABR and possible options for server side ABR.

9. Usage Mode identification

This specification explores 2 possible usage modes for applications to consider when using MOQT media delivery protocol :

1. Transport Mapping Section 2

2. MOQT Object model mapping

For interoperability purposes, media producers should communicate its usage modes to the media consumers. Same can be achieved in one of the following ways

1. Via out of band, application specific mechanism. This approach limits the interoperability across applications, however.

2. Exchange the usage modes via Catalog. This approach enables consumers of catalog to setup their transport/media stacks appropriately based on the sender's preference.

10. References

10.1. Normative References

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

10.2. Informative References

[MoQTransport] Curley, L., Pugin, K., Nandakumar, S., and V. Vasiliev, "Media over QUIC Transport", Work in Progress, Internet-Draft, draft-ietf-moq-transport-00, 5 July 2023, <https://datatracker.ietf.org/doc/html/draft-ietf-moq-transport-00>.

[CMAF] "Information technology -- Multimedia application format (MPEG-A) -- Part 19: Common media application format (CMAF) for segmented media", March 2020.

Appendix A. Security Considerations

This section needs more work.

Appendix B. IANA Considerations

This document doesn't recommend any changes to IANA registries.

Appendix C. Acknowledgments

Thanks to MoQ WG for all the discussions that inspired this document's existence.

Authors' Addresses

Cullen Jennings
Cisco
Email: fluffy@iii.ca

Suhas Nandakumar
Cisco
Email: snandaku@cisco.com

Mo Zanaty
Cisco
Email: mzanaty@cisco.com