Network Working Group
Internet-Draft
Intended status: Informational
Expires: 14 September 2023

S. Nandakumar
Cisco
C. Jennings
cisco

13 March 2023

Media Over QUIC Media and Security Architecture
draft-nandakumar-moq-arch-00

Abstract

This document defines the media and security architecture for Media Over QUIC, a protocol suite intended to run in browsers and other non-browser endpoints and support unified architecture and protocol for data delivery that enables a wide range of media applications with different resiliency and latency needs without compromising the scalability and cost effectiveness associated with content delivery networks.

Note to readers: This document as it stands might not reflect accurately all decisions that are in active discussions in the WG, but it lays out a foundation for expected architectural requirements. The specification will be updated based on the decision made in the WG.

Table of Contents

1. Introduction

Recently new use cases have emerged requiring higher scalability of delivery for interactive realtime applications and much lower latency for streaming applications and a combination thereof. On one side are use cases such as normal web conferences wanting to distribute out to millions of viewers and allow viewers to instantly move to being a presenter. On the other side are uses cases such as streaming a soccer game to millions of people including people in the stadium watching the game live. Viewers watching an e-sports event want to be able to comment with minimal latency to ensure the interactivity aspects between what different viewers are seeing is preserved. All of these uses cases push towards latencies that are in the order of 100ms over the natural latency the network causes.

Interactive realtime applications, such as web conferencing systems, require ultra low latency (< 150ms) delivery. Such applications create their own application specific delivery network over which latency requirements can be met. Realtime transport protocols such as RTP over UDP provide the basic elements needed for realtime communication, both contribution and distribution, while leaving aspects such as resiliency and congestion control to be provided by each application. On the other hand, media streaming applications are much more tolerant to latency and require highly scalable media distribution. Such applications leverage existing CDN networks, used for optimizing web delivery, to distribute media. Streaming protocols such as HLS and MPEG-DASH operates on top of HTTP and gets transport-level resiliency and congestion control provided by TCP.

The architecture defines and uses a unified media delivery protocol that is based on a publish/subscribe metaphor where client endpoints publish and subscribe to named objects that is sent to, and received from relays, that forms an overlay delivery network similar to what CDN provides today. Objects are named such that it is unique for the relay/delivery network and scoped to an application.

The MoQ transport protocol provides services based on application requirements (with the support of underlying transport, where necessary) such as estimation of available bandwidth, resiliency, congestion control and prioritization of data delivery based on data lifetime and importance of data. It is designed to be NAT and firewall traversal friendly and can be fronted with load balancers.

In a sample deployment, Relays can be thought of as arranged in a logical tree (as shown below) where, for a given application, there is an origin Relay at root of the tree that controls the namespace for the objects. Publish messages are sent towards the root of the tree and down the path of any subscribers to that named object. It

is designed to make it easy to implement relays so that fail over could happen between relays with minimal impact to the clients and relays can redirect a client to a different relay.

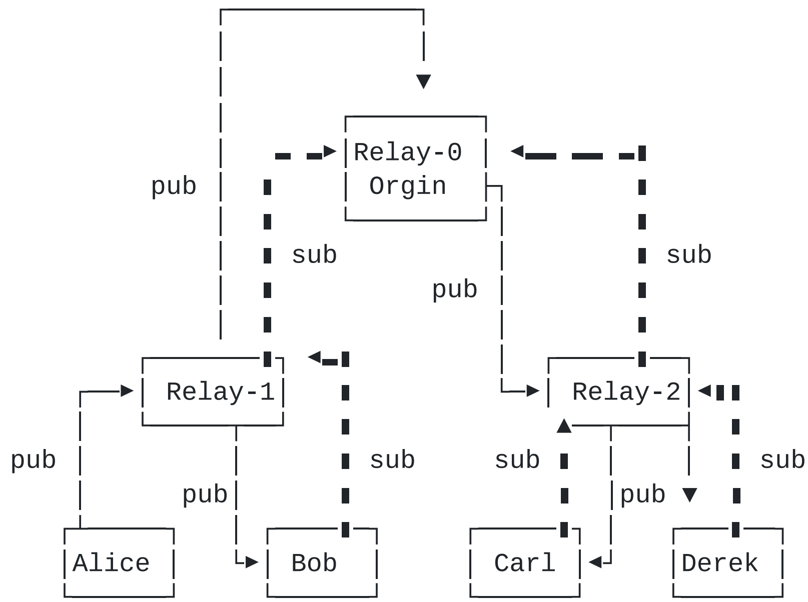


Figure 1: Delivery Tree

The design supports sending named objects between a set of participants in a game or video call with under a hundred milliseconds of latency and meets the needs of conferencing systems. The design can also be used for large scale streaming to millions of participants with latency ranging from a few seconds to under a hundred milliseconds based on applications needs.

1.1. Advantages of MoQ Media Transport Protocol

The architecture for MoQ uses similar concepts and delivery mechanisms to those used by streaming standards such as HLS and MPEG-DASH. Specifically the use of a CDN-like delivery network, the use of named objects and the receiver-triggered media/data delivery. However, there are fundamental characteristics that the MoQ Transport provides to enable ultra low latency delivery for interactive applications such as conferencing and gaming.

* To support low latency the granularity of the delivered objects, in terms of time duration, need to be quite small making it complicated for clients to request each object individually. MoQ Transport protocol uses a publish and subscription semantic to simplify and speed object delivery for low latency applications. For latency-tolerant applications, larger granularity of data, aka group of objects, can be individually requested and delivered without instantiating state in the backend.

* Some realtime applications operating in ultra low latency mode require objects delivered as and when they are available without having to wait for previous objects due to network loss or out of order network delivery. Pipelining of object delivery with delivering media bitstream as and when they appear is supported, for this purpose.

* Published objects have associated max-age that specifies the validity of such objects. max-age influences relay's drop decisions and can also be used by the underlying QUIC transport to cease retransmissions associated with the named object.

* Unlike traditional streaming architectures where media contribution and media distribution are treated differently, MoQ Transport can be used for both object contribution/publishing and distribution/subscribing as the split does not exist for interactive communications.

* "Aggregation of subscriptions" at the relay functions allows "short-circuited" delivery of published objects when there is a match at a given relay function. This further enables local error recovery where applicable.

* Publishers can associate a priority with objects. These can help the delivery network and the subscribers to make decisions about resiliency, latency, drops etc. Priorities can be used to set relative importance between different qualities for layered video encoding, for example.

* MoQ Transport is designed so that objects are encrypted end-to-end and will pass transparently through the delivery network. Any information required by the delivery network, will be included as part of the metadata that is accessible to the delivery network for further processing as appropriate.

2. Entity Roles and Trust Model

This section specifies various roles that make up an application architecture using MoQ Transport.

2.1. Roles

This section defines various roles that can exist within the MoQ Architecture and the associated trust model. A given MoQ entity (physical or logic component) can play one or more roles depending on their utility within the architecture.

2.1.1. Emitter

Entities that perform Emitter role are trusted with E2E encryption keys for the media and operate on one or more uncompressed and unencrypted media. They compress and possibly encrypt and transmit over one or more Data Streams.

Each such encoded and/or encrypted stream corresponds to a Track within the MoQ transport protocol.

2.1.2. Publisher

Entities with Publisher role publish MoQ Objects belonging to a given track using the MoQ Transport Protocol.

2.1.3. Subscriber

Entities with Subscribe role subscribe to receive MoQ Objects corresponding to a track using the MoQ Transport Protocol.

2.1.4. Relay

Entities performing the Relay role are not trusted with E2E encryption keys.

Entities performing Relay role implement store and forward behavior. Such entities receives subscriptions and publishes data to the other endpoints that have subscribed to the named data. They may cache the data as well for optimizing the delivery experience. Since these entities are not trusted with the E2E keys, they cannot access unencrypted MoQ Object Payload, however they are allowed to access the MoQ Object Header.

2.1.5. Catalog Maker

Entities performing Catalog Maker role compose or aggregate tracks from multiple emissions to form a new emission. Akin to the role of entities with the Relay role, Catalog Maker role entities are not trusted with the E2E keys and they perform publisher and subscriber roles. Catalog Makers are allowed to publish tracks with a new name without changing the media content of the received tracks.

2.1.6. Receiver

Entities performing the Receiver role are trusted with E2E keys and can transform the Encrypted Stream into Encoded Stream and decode the encoded stream to source stream for further consumptions

2.1.7. Key Manager

Entities with Key Manager role are responsbile for setting up the needed protocol machinery for distributing keys needed for end to end encryption. Key Manager role doesn't provide access to the E2E encryption keys.

2.2. Media Transformer

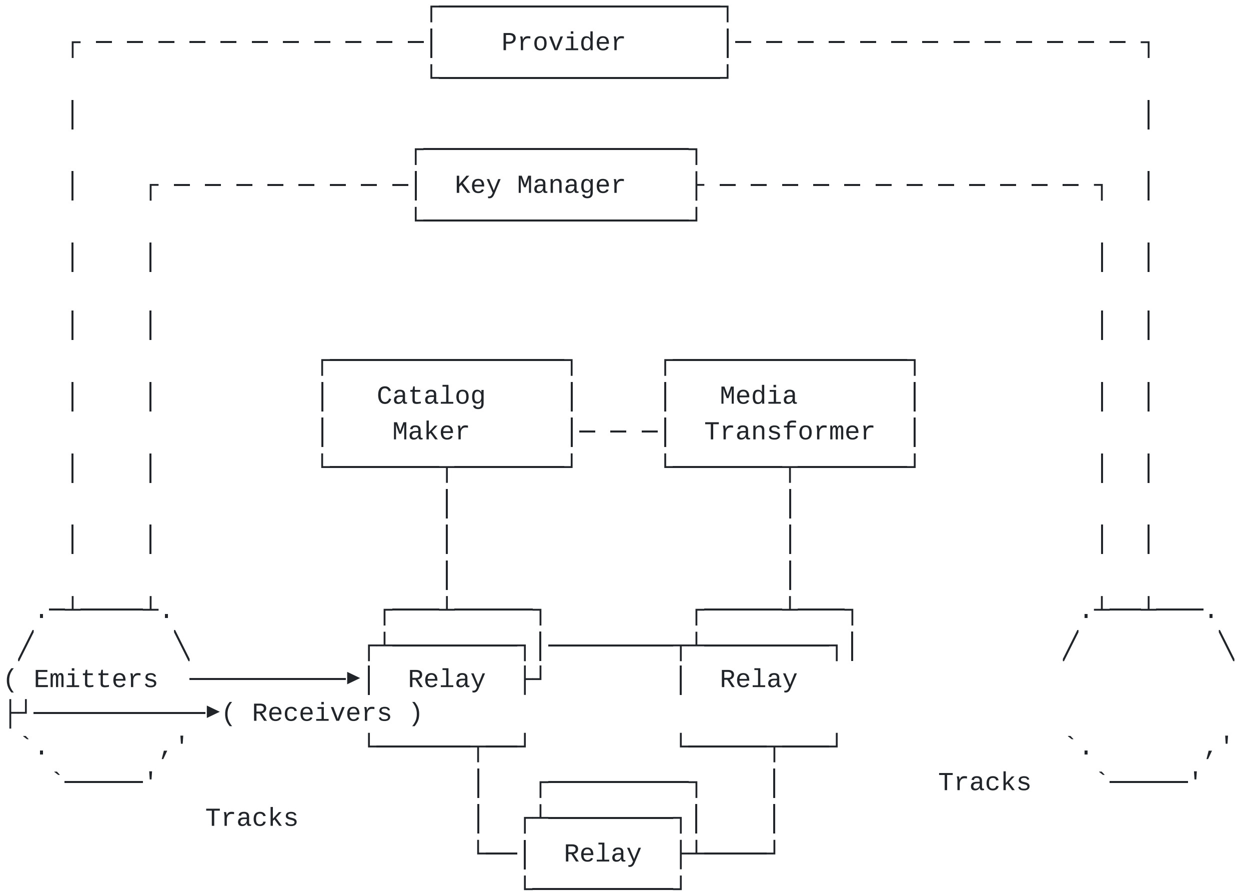
Entities with Media Transformer roles are trusted with E2E encryption keys and thus have access to the media. They combine the following roles - Subscriber, Receiver, Emitter and publisher. They subscribe to media tracks, decrypt and transform the received media, encrypt and publish the transformed media as new media tracks.

2.3. Provider

TODO

3. Reference Architecture

Below picture depicts reference architecture involving entities playing various roles.



4. Overview

4.1. Streaming Architectures

Streaming scenarios typically separate "content ingestion" and "content distribution". In a reference live streaming architecture shown below, the emitter live streams on or more tracks as part of the application operated under a provider domain, which gets eventually distributed to multiple clients by some form of distribution server operating under the provider domain, over a content distribution network.

```
                        +-------------+
                        |Relay        |
       +----------------> |Orgin:tw.com |-----+
       |                +-------------+     |
       |                              ^     |
       |pub:tw.com/ch22/audio-1       |     |
       |pub:tw.com/ch22/audio-2       |     |
       |        sub:tw.com/ch22/audio-1 |     |
       |                              |     |pub:tw.com/ch22/audio-1
       |                              |     v
   +-----------+                    +----------+
   | Emitter   |                    | Receiver |
   +-----------+                    +----------+
```

The above picture shows MoQ Transport based delivery network for an hypothetical streaming architecture rooted at the Origin Relay, akin to Ingest server/Distribution Server (for the domain tw.com). In this architecture, the media contribution is done by publishing named tracks for audio corresponding to channel-22, at 2 different qualities, to the ingest server at the Origin Relay. Media consumption happens via subscribes sent to the Origin Relay to the name (tw.com/ch22/audio-1) to receive the right quality track.

4.2. Interactive/Conferencing Architectures

A interactive conference typically works with the expected operating glass-to-glass latency to be around 200ms and is made up of multiplicity of participant with varying capabilities and operating under varying network conditions.

A typical conferencing session comprises of:

* Multiple emitters, publishing on multiple tracks (audio, video tracks and at different qualities)

* A media switch, sourcing tracks that represent a subset of tracks from across all the emitters. Such subset may represent tracks representing top 5 speakers at higher qualities and lot of other tracks for rest of the emitters at lower qualities.

* Multiple receivers, with varied receiving capacity (bandwidth limited), subscribing to subset of the tracks

```
                     +--------+
              +----> |   SFU  | <----------------+
              |      +--------+                  |
              |       ^       |                  |sub:alice-low
    pub:alice-hi      |       pub:alice-hi       |sub:alice-hi
    pub:alice-low     |       pub:alice-low      |
              | sub:alice-low |                  |
              |       |       |                  |
             +---------+      |  +------------------------+
     +------>| Relay-A |      +->|    Relay-B             |
     |       +---------+         +------------------------+
     |           |  ^              |    ^          |    ^
    pub1:alice-hi|  |              |    |          |    |
    pub2:alice-low  |              |    |          |    |
     |           |  |              |    |          |    |
     |          pub:alice-low     pub:alice-hi,low pub:alice-hi,low
     |           |  |              |    |          |    |
     |           | sub:alice-low   |   sub:alice.. |   sub:alice..
     |           v  |              v    |          v    |
     +------+    +---+              +----+         +-----+
     | Alice|    |Bob|              |Carl|         |Derek|
     +------+    +---+              +----+         +-----+
```

The above picture shows a MoQ Transport based media delivery tree formed with multiple relays in the network. The example has 4 participants with Alice being the publisher and rest being the subscribers. Alice's is capable of publishing video streams at 2 qualities identified by their appropriate names. Bob subscribes to a low resolution video track from alice, whereas Carl/Derek subscribe to all the qualities of video feed published by Alice. All the subscribes are sent to the Origin Relay and are saved at the on-path Relays, this allowing for "short-circuited" delivery of published data at the relays. In the above example, Bob gets Alice's published data directly from Relay-A instead of hair pinning from the Origin Relay. Carl and Derek, however get their video stream relayed from Alice via Origin Relay and Relay-B.

5. Tracks, Objects and Groups

Tracks form the central concept within the MoQ Transport protocol for delivering media. A Track identifies the namespace and the authorization scope under which MoQTransport objects are delivered.

A track is a transform of a uncompressed media using a specific encoding process, a set of parameters for that encoding, and possibly an encryption process.

The MoQTransport is designed to transport tracks.

The binary content of a track is composed of a sequence of objects. An Object is the smallest unit that makes sense to decode and may not be independently decodable. An Object MUST belong to a group.

Objects MUST be uniquely identifiable within the MoQ delivery system. Objects carry associated header/metadata containing priority, time to live, and other information aiding the caching/forwarding decision at the Relays. Objects MAY be optionally cached at Relays. The content of the Objects are opaque to Relays and delivered on the strict priority order.

An Object MUST belong to a group. Groups are composition of objects and the objects within the group carry the necessary dependency information needed to process the objects in the group.

A typical example would be a group of pictures/video frames or group of audio samples that represent synchronization point in the video conferencing example.

6. Relays and Scalability

The Relays play an important role for enabling low latency media delivery within the MoQ architecture. This specification allows for a delivery protocol based on a publish/subscribe metaphor where some endpoints, called publishers, publish media objects and some endpoints, called subscribers, consume those media objects. Some relays can leverage this publish/subscribe metaphor to form an overlay delivery network similar/in-parallel to what CDN provides today. While this type of overlay is expected to be a major application of relays, other types of relays can also be defined to offer various types of services.

Relays provide several benefits including

* Scalability – (U+2013) Relays provide the fan-out necessary to scale up streams to production levels (millions) of concurrent subscribers.

* Reliability - relays can improve the overall reliability of the delivery system by providing alternate paths for routing content.

* Performance – (U+2013) Relays are usually positioned as close to the edge of a network as possible and are well-connected to each other and to the Origin via high capacity managed networks. This topography minimizes the RTT over the unmanaged last mile to the end-user, improving the latency and throughput compared to the client connecting directly to the origin.'

* Security – (U+2013) Relays act to shield the origin from DDOS and other

In order to keep the latencies low, Relays don't wait until the entire object is received but rather forward the bitstream/fragments as they are received to downstream subscribers.

7. MoQ Transport Usage Patterns

This section explains usage patterns that can be used to build applications on top of MoQ Transport

7.1. Catalog Objects

TODO

7.2. MOQ Video Objects

Most video applications would use the data identifier component to identity the video stream, as well as the encoding point such as resolution and bitrate. Each independently decodable set of frames would go in a single group, and each frame inside that group would go in a separate named object inside the group. This allows an application to receive a given encoding of the video by subscribing just to the data identifier component of the Name with a wildcard for group and object IDs.

This also allows a subscription to get all the frames in the current group if it joins later, or wait until the next group before starting to get data, based on the subscription options. Changing to a different bitrate or resolution would use a new subscription to the appropriate Name.

The QUIC transport that QuicR is running on provides the congestion control but the application can see what objects are received and determine if it should change it's subscription to a different bitrate data identifier component.

Today's video is often encoded with I-frames at a fixed interval but this can result in pulsing video quality. Future system may want to insert I-frames at each change of scene resulting in groups with a variable number of frames. QuicR easily supports that.

7.3. MoQ Audio Objects

TODO

7.4. MOQ Game Moves Objects

Some games send out a base set of state information then incremental deltas to it. Each time a new base set is sent, a new group can be formed and each increment change as an Object in the group. When new players join, they can subscribe to sync up to the latest state by subscribing to the current group and the incremental changes that follow.

8. Security Considerations

The links between Relay and other Relays or Clients can be encrypted, however, this does not protect the content from Relays. To mitigate this, all the objects need to be end-to-end encrypted with a keying mechanism outside the scope of this protocol. For may applications, simply getting the keys over HTTPS for a particular object/group from the origin server will be adequate. For other applicants keying based on MLS may be more appropriate. Many applications can leverage the existing key managed schemes used in HLS and DASH for DRM protected content.

Relays reachable on the Internet are assumed to have a burstiness relationship with the Origin and the protocol provides a way to verify that any data moved is on behalf of a given Origin.

Relays in a local network may choose to process content for any Origin but since only local users can access them, there is a way to manage which applications use them.

Subscriptions need to be refreshed at least every 5 seconds to ensure liveness and consent for the client to continue receiving data.

9. Protocol Design Considerations

9.1. HTTP/3

It is tempting to base this on HTTP but there are a few high level architectural mismatches. HTTP is largely designed for a stateless server in a client server architecture. The whole concept of the PUB/SUB is that the relays are _not_ stateless and keep the subscription information and this is what allows for low latency and high throughput on the relays.

In today's CDN, the CDN nodes end up faking the credentials of the origin server and this limits how and where they can be deployed. A design with explicitly designed relays that do not need to do this, while still assuming an end-to-end encrypted model so the relays did not have access to the content makes for a better design.

It would be possible to start with something that looked like HTTP as the protocol between the relays with special conventions for wildcards in URLs of a GET and ways to stream non final responses for any responses perhaps using something like multipart MIME. However, most of the existing code and logic for HTTP would not really be usable with the low latency streaming of data. It is probably much simpler and more scalable to simply define a PUB/SUB protocol directly on top of QUIC.

9.2. QUIC Streams and Datagrams

There are pro and cons to mapping object transport on top of streams or on top of QUIC datagrams. The working group would need to sort this out and consider the possibility of using both for different types of data and if there should be support for a semi-reliable transport of data. Some objects, for example the manifest {{#manifest}} would always want to be received in a reliable way while other objects may have to be realtime.

9.3. QUIC Congestion Control

The basic idea in BBR of speeding up to probe then slowing down to drain the queue build up caused during probe can work fine with real time applications. However, the current implementations in QUIC do not appear optimized for real-time applications, resulting in higher jitter (under certain conditions). In order to avoid play-out drops, the jitter buffers add latency to compensate for this. Probing for the RTT has been one of the phases that causes particular problems for this. To reduce the latency of QUIC, this work should coordinate with the QUIC working group to have the QUIC working group develop congestion control optimizations for low latency use of QUIC.

Appendix A. Acknowledgments

TODO

Authors' Addresses

Suhas Nandakumar
Cisco

Email: snandaku@cisco.com

Cullen Jennings
cisco
Canada

Email: fluffy@iii.ca