Media over QUIC (MoQ) Relays

IETF 115
Nov 2022

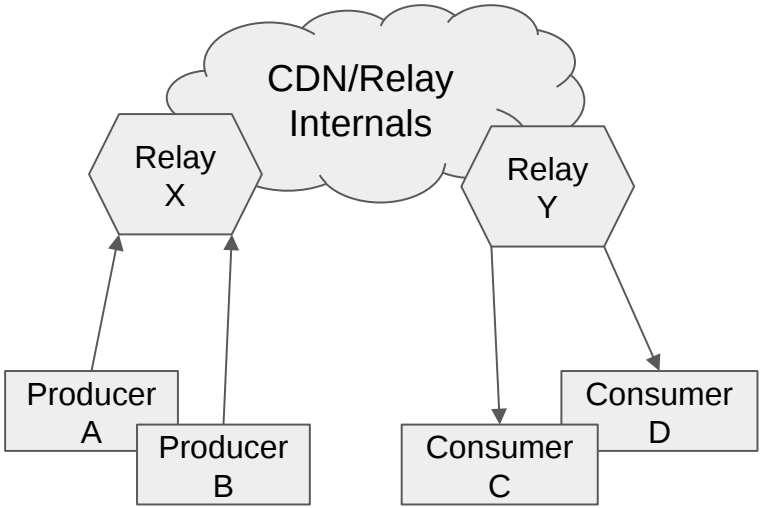
V1

Goals of Presentation

- Common understanding of Relays from:
  - draft-shi-moq-design-space-analysis-of-moq
  - draft-lcurley-warp
  - draft-defoy-moq-relay-network-handling
  - draft-jennings-moq-quicr-arch/protocol
  - draft-kpugin-rush
  - other drafts and slack chat
- Cover similarities and difference at architectural level of Relays in all the MoQ drafts.
- Highlight some of the decisions the working group will need to make and pros/cons.
- Set the stage for an informed technical discussion about Relays.

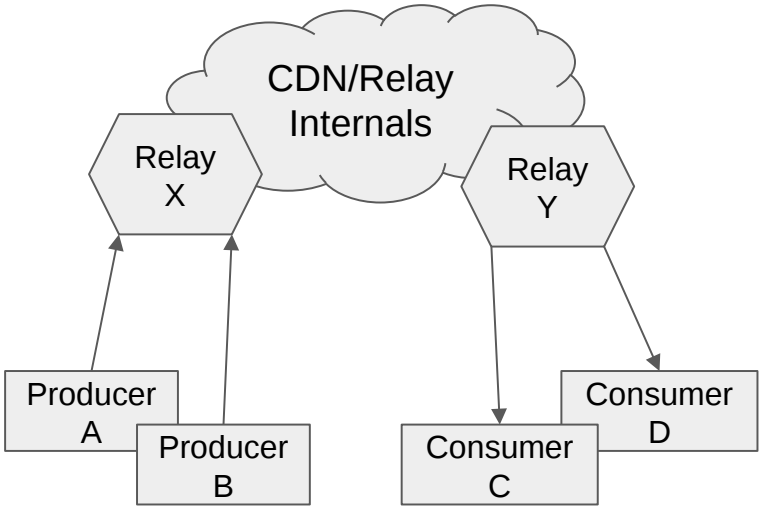
Non-goals: Pick any given solution.

Where do the Relays fit into MoQ?



- Relays take incoming media and forward it to one or more places.
- Relay decides the order of what to send and what gets discarded.
- MoQ defines a media-delivery protocol from Producer to Consumer through the Relays, **but does not** define internal CDN design.
- Does need to make sure enough information is available to Relay to do its job.
- Relays don’t have access to the MoQ media payload.
- Terminology: Producer, Consumers, and things like transcoders are Ends. Relays are middles—not an End. Producers include transcoders in the cloud. Ends can decrypt MoQ media payload.

Who could provides these Relays?



- Like today's CDN market, there will likely be a range of providers.
- Companies like Twitch have existing CDN deployments and they may adapt the same to use MoQ.
- Companies like Cisco/Webex today run their own but would like to buy it as a service
- Companies like Akamai, Cloudflare, and Fastly are interested in running MoQ
- Interdigital is looking at Relays tied to 5G network optimizations

Why do we need Relays?

- Distribution needs fan out. Each Consumer can receive media at different qualities. Relays and Consumers work together to solve this.

- Points close to Consumers and Producers enable rapid recovery from packet losses. (Low Round Trip Time (RTT) allows QUIC streams to quickly recover lost packet with less impact on end-to-end latency.)

- Multi-CDN brings scale (fan out) and reliability.

Relay Operational Requirements

- Minimize CPU load to ensure scalability.
- Allow configurable latency/congestion response.
  - Allow Relays to make forward/drop decisions.
- Ability to cleanly hand-off/transition clients to new Relays without loss of services.
  - Relay redirect
- Low latency (See XR use cases in draft-defoy-moq-relay-network-handling.)
- Rapid recovery from failed Relay.

Definition of Envelopes and Payload

MoQ data is made up of **[Envelope] and [Payload].**

Envelope: What a Relay can read and use. Relays don’t modify the Envelope.

Payload: Opaque to Relay and often end-to-end encrypted.

Rough list of envelope data in various proposals

- Warp envelope: id for stream, deliver order aka priority, dependency list
- Defoy envelope: id for segment/pdu set, first/last indicator, seq number, group size, priority / group relative importance , target delay, target error rate, burst periodicy, discarable/reliability,
- QuicR envelope: stream id and seq num (name), Priority, Time To Live (best before), Time stamp, discardable

Priority and Delivery Order

Key thing a relay does is decide what to send next to any given destination

All the proposals have some form of numeric priority that the Relay can read:

- If a scalable video codec is sending a base layer at 360p and upscaled layer at 720p, the Relay needs to understand that the 360p layer is more important than the 720p layer.

Warp and QuicR both have a numeric counter that indicates decoder delivery order.

- All the proposals also have some stream or segment ID and something like a sequence number inside of that. This allows a temporal ordering of the chunks of data.
- Think of this sort of like the sequence number in RTP.
- If some data is very old, it may not make sense to send it.
- This field is called Priority/Order in Warp and is part of the objectID in QuicR.
- This can be used to ensure things like not bothering with very old data
- Warp uses a prioritization idea that more recent data in the temporal ordering has higher priority. QuicR uses Time To Live to effectively remove data that is too old.

Dependency Indication

Dependency List: Warp specifies decode dependency order via optional dependency list in the Envelope.

Dependency by Priority: QuicR uses priority in Envelope for expressing dependency information and assumes that they would be assigned in a way where dependent segments had a lower priority than anything they depended on.

Relays need a media-agnostic way to act on dependency information for making forward/drop choices.

Open question of it priority is enough or if we need an explicit dependency list.

- Dependency list may add significant computation cost to relays

Caching

- Certain use cases need the Relays to also be able to act as short-term caches.
  - Would drive need for some sort of Time-To-Live hints in the Envelope data.

- Dealing with multiple qualities:
  - Consumer can ask for the specific quality.
  - Consumer can provide a “hint” to the Relay to pre-fetch alternate qualities enabling quick transition between them.

- Ability for a client to request a sub-range within a named object to speed up join-to-live edge times.

Security and Privacy

The Relays can only read the Envelope data and may not be able to read the rest of the Payload data.

- Minimize Envelope to only information the relay needs.

The Payload can be end to end encrypted with techniques such as MLS groups keying for all the consumers or with existing DRM techniques.

- The QUIC TLS connections are terminated by the Relays (like HTTPS on today's CDNs). The QUIC layer protects all the content from network snooping.
- The distribution side needs to fan out so end-to-end QUIC streams or TLS is not an option because the state management in QUIC and TLS only allows for 1:1 connection not 1:n.

Authentication and Authorization Options

1. Origin is always the final decision maker.
    a. All authorizations will go to the origin, proxied via Relays.

2. Optional proxied authorization at the edge via tokens and pre-shared secrets between edge and the origin.
    a. Highly scalable.
    b. Http media streaming uses this scheme.
    c. Pre-shared secret provisioning is done out-of-band (is out of scope for MoQ).
    d. Tokens can be oauth, jwt, cat (common access token).
    e. May be useful to inherit this.

Warp assumes authentication (auth) token and user identifier in CONNECT.

QuicR assumes auth token and URI in message when forming connection to Relay.

Time Permitting …

Modifying Envelope Data

- Question: Can Relays modify the Envelope data?
- None of current proposal have need for this.
- Leads to complication of later putting redundant Envelope data in payload.
- Current architecture is, this can be end-to-end integrity protected using techniques such as MLS to set up the keys.

Routing Options

1. Consumers tell Relays what each Consumer wants to receive and Relays send it to them. (This is the same as today's CDNs that support HLS/DASH). This is what QuicR does, though it collapses the many HTTP GETs into one GET with a wildcard.

2. Consumer tells a central service what they want to consume, and the central service tells the Producer to tag media with a flow path tag for each Consumer that wants it. The Relays can then look at the tags to decide how to forward the media to the next next hop in the flow path. (See paragraph 3, section 4, Shi draft).

Forwarding Options

TODO: Stream vs Store & Forward

Normal Operation

Forward the packets as they arrive (pipelining) to keep latencies minimal

OR

Can wait until have the full media segment / GOP / frame whatever and then forward.

Under Congestion

May need to drop objects/layers based on Envelope information

Relay Topologies

- Relays topologies can typically be deployed as: Tree, Meshes, Dynamic Meshes or a combination of these.
- MoQ doesn’t enforce a single topology.
- MoQ defines an end-to-end media delivery protocol from Producers to Consumers through Relays across various topologies.