

MOQT PRs and Issues

Boulder

#1446 Remove subgroup ID from priority tiebreakers

3. If two objects in response to the same request have the same subscriber and publisher priority and belong to the same group of the same track, ...

Old:

the one with the lowest Subgroup ID (*Subgroups*), or the lowest Object ID (*Datagrams*) is scheduled to be sent first. If the two objects have different Forwarding Preferences the order is implementation dependent.

New:

the one with **the lowest Object ID** is scheduled to be sent first. For objects with forwarding preference Subgroup, *the first Object ID in the Subgroup is used* for comparison, providing a stable ordering for the entire Subgroup.

Rationale:

1. Easier to compare subgroups and datagrams
2. Matches existing behavior (stable subgroup comparison)
3. This is the 3rd tiebreaker

#1447 Add RENDEZVOUS_TIMEOUT parameter for lingering SUBSCRIBE

Duration a relay should wait for a publisher to materialize or reconnect before failing subscription

Reconnect case can create un signaled gaps - even for “high-fidelity” tracks

→ Don't do that or be prepared to handle it?

Should join and reconnect timeouts be the same or different parameters?

#83 Improve startup latency

#1370 Does the Client have to speak first

Current (1-RTT):

1-RTT: C->S: CLIENT_SETUP

1.5-RTT: S->C: SERVER_SETUP
(MAX_REQUEST_ID)

2-RTT: C->S: SUBSCRIBE or PUBLISH

Current (0-RTT):

0-RTT: C->S: CLIENT_SETUP

0.5-RTT: S->C: SERVER_SETUP

1-RTT: C->S: SUBSCRIBE or PUBLISH

Proposed (1-RTT):

0.5-RTT: S->C: SERVER_SETUP
(MAX_REQUEST_ID)

1-RTT: C->S: CLIENT_SETUP

1-RTT: C->S: SUBSCRIBE or PUBLISH

SUBSCRIBE or PUBLISH in 0-RTT is
fraught because of auth replay

Pair of Uni Control Streams

Used by HTTP/3 - widely deployed protocol based on QUIC with 0-RTT support

Allows either party to speak first

Cuts 1-RTT from first client request from 1-RTT handshakes

0-RTT is not trivial to deploy and get right, optimizing 1-RTT is still worthwhile

Restricts how extensions are negotiated

Each peer unilaterally declares what they support, intersection is run by both

#799 How can end subscriber verify an authorization claim in a PUBLISH*

End Subscriber's can receive PUBLISH*

1) From a direct peer publisher

→ likely contain an auth token / out of band auth

2) From a relay in response to SUBSCRIBE_NAMESPACE

AUTH_TOKEN is hop-by-hop → will not be sent by relay in PUBLISH

NAMESPACE has nowhere to convey auth

Subscriber *already trusts* relay is authoritative for all tracks under SUB_NS?

End-to-End application schemes can use Track Extensions?

What do we need to write?

#1245 Goaway time

“goway message should include a time in seconds that indicates how long the client has to goaway before just getting shut off”

Yes / No?

Could be optional (eg: 0 means I don't know)

Seems fine?

#1376 Text on reordering and Objects going from DNE to Exists is incorrect

Ways an object transitions to DNE:

- Processing a Gap header
- Processing an EOG or EOT
- Processing a FETCH with a gap

Edge cases:

FETCH races a Subscription to OP - OP declares all Objects DNE in FETCH
Normal Object arrives delayed → Malformed Track
Probably wanted to forward that

Two FETCHes race

First got Normal object (delayed), second got DNE (expired)

DNE

A single FETCH is easy to check for Malformed conditions against itself

Multiple FETCHes can race - are conflicts malformed or delayed?

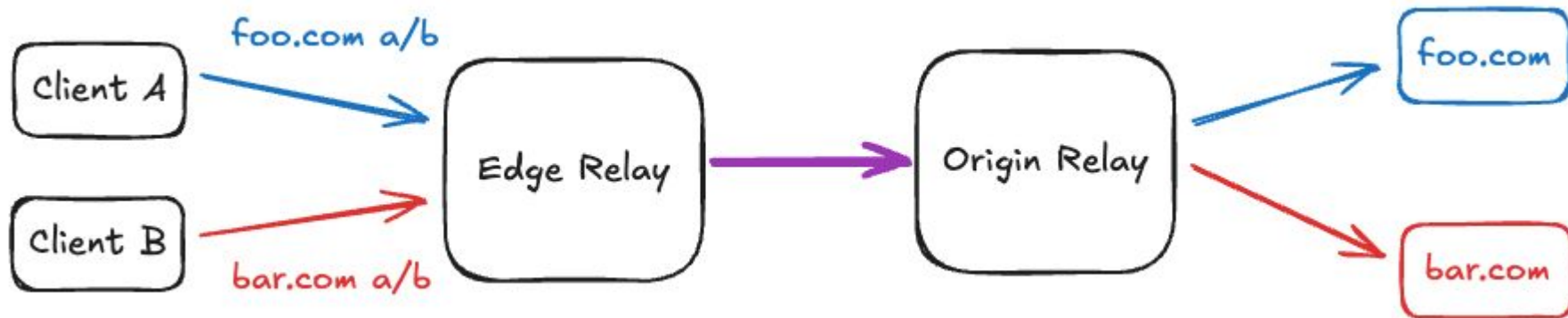
Subscriptions can't be checked against a cache without knowing *how* the DNE got there (Gap/EOG/EOT → Malformed; FETCH → Forward)

Tracking that is expensive - is it worth it

Or we can just say ͇_(\ツ)_/͇

What do we want?

#1432 Define session reuse rules and improve scope definition



Edge Proxy serves two hostnames: [foo.com](#) and [bar.com](#)

Both define a track ns=a name=b - this meets MOQT Scope requirements

Can these subscriptions share a QUIC connection between Edge and Origin?

How about a MOQT session?

Options

Option 1: WebTransport Pooling - preserves the connection URL - no MOQT changes or special namespace requirements. MOQT session is *not* shared.
Bonus - relative isolation in the connection w.r.t. Flow Control and Prioritization.

Option 2: CDN makes namespace requirements that guarantee track uniqueness in their infra (eg: connection URI or unique customer ID embedded in ns[0])

Option 3: Message Parameter to convey authority/path - could be custom per CDN.

Option 4: Upstream connection per authority/path - this is infeasible for most CDNs given large numbers of small domains.

#1433 Can multiple REQUEST_UPDATES be collapsed into one, and then applied?

If yes, receivers can conserve work by merging them and applying once (e.g. per flight)

However, each needs their own REQUEST_OK.

It can save work: eg re-prioritizing $1 \rightarrow 2 \rightarrow 1 \rightarrow 2 \Rightarrow \text{pri}=2$.

#1453 Send Rate Parameter

Seems useful for FETCH

Should we express in bytes or groups/objects?

#1451 Allow multiple Subscriptions to a Track

If two subscribers have different, sparse filters to the same track, a relay must either:

1. Subscribe upstream with no filters
2. Open separate connections

However, if we allow multiple subscriptions to the same track, either:

1. Each subscription needs a unique data-plane component (alias, request id, etc)
 - a. And send duplicate object for each matching subscription
2. The subscriber and publisher must BOTH run the filters to re-sort objects
3. Something else?

This problem already exists with Range filters but could be more pronounced with Object filters (#1401).

Placeholder for more issues/PRs we should discuss

- #1439 What is Implicit in the SUBSCRIBE_NAMESPACE stream?
- #1458 Edge Cases around updating SUBSCRIBE_NAMESPACE
- #1457 Track Namespaces can have zero fields now
- #1467 Inconsistency on Message Parameters

#1316 Possible improvements on MoQ support of VOD applications

Is there a specific ask here? Would a Send Rate parameter (Issue #1453) be sufficient?

#667 DELIVERY_TIMEOUT is unimplementable

What are we going to do here?