MoQ Draft-01 Interop Status
## Interop Test Cases So Far

<table>
<thead>
<tr>
<th>Test Cases</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 = Exchange Setup</td>
<td>Send a ClientSetup and receive a ServerSetup</td>
</tr>
<tr>
<td>2 = Exchange Subscribe/Ok</td>
<td>Send a Subscribe and receive a Subscribe Ok</td>
</tr>
<tr>
<td>3 = Receive Object</td>
<td>Send or Receive an Object on an active subscription</td>
</tr>
<tr>
<td>4 = Exchange Announce/Ok</td>
<td>Send an Announce and receive an AnnounceOK</td>
</tr>
<tr>
<td>5 = Subscribe Hint</td>
<td>Send a Subscribe Hint for non-live edge and receive correct object</td>
</tr>
<tr>
<td>6 = Unsubscribe</td>
<td>Unsubscribe from an active subscription, terminating object flow</td>
</tr>
<tr>
<td>7 = Goaway</td>
<td>Drain a connection after receiving a Goaway</td>
</tr>
<tr>
<td>Q = RawQUIC also</td>
<td>Can do it over RawQUIC as well</td>
</tr>
<tr>
<td>q = RawQUIC only</td>
<td>Can only do the above on Raw QUIC</td>
</tr>
</tbody>
</table>
## Notable Results:

- Alan/Jordi: Ran a smooth video & audio stream Prague (J) -> Ohio (A) -> Prague (J)
- Luke: Streaming video at quic.video
- Cisco: Ran a smooth video & audio stream Prague -> London -> Prague, Prague → Oregon → Prague
MetaMoQLib (Alan)

**Language**: C++  
**Transport**: WebTransport Only (proxygen/mvfst)  
**Endpoints**: Client, Server, Relay

**Applications**:

Generic Relay  
Matching announce to subscribe and forwarding  
Deduplication of upstream subscribes

MoQ Clock Server (client/server or client/relay/client)  
Subscribe Hints allow fetching historical time back to 1970  
Chat (client/server only)

Text Client (subscribes to one track and prints every object)
MetaMoQBrowser (Jordi)

**Language**: Javascript  
**Transport**: WebTransport Only (Chrome)  
**Endpoints**: Client - Publisher (encoder) / Client - Subscriber (Player)

**Applications**:  
LiveStreaming / VideoConference (aligned audio & video)  
- QUICStream per object, object = frame (video frame or audio frame)  
- Priorities implemented  
- Sending audio & video info and playing them back aligned  
- Fully configurable: Encoding, settings, dejitter buffers, etc

Text Client (AKA: Chat)  
- QUICStream per object, object = text message
Rust!

Native code is written in Rust and is available at kixelated/moq-rs. It's split into a few crates:

<table>
<thead>
<tr>
<th>crate</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>moq-transport</td>
<td>A library that mostly implements the MoqTransport draft... with a few strong opinions.</td>
</tr>
<tr>
<td>moq-relay</td>
<td>A MoQ server that connects publishers to subscribers, caching any duplicate subscriptions.</td>
</tr>
<tr>
<td>moq-pub</td>
<td>A MoQ client that integrates with ffmpeg to publish fMP4.</td>
</tr>
<tr>
<td>moq-clock</td>
<td>It's a clock! Just to demonstrate that MoqTransport can do more than media.</td>
</tr>
<tr>
<td>webtransport-quinn</td>
<td>A WebTransport client and server utilizing Quinn.</td>
</tr>
</tbody>
</table>
Web code is written in Typescript and is available at [kixelated/moq-js](kixelated/moq-js). It’s split into a few folders:

<table>
<thead>
<tr>
<th>folder</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>transport</td>
<td>Mostly implements <a href="https://www.moq.org">MoQ Transport</a> draft... with same said strong opinions.</td>
</tr>
<tr>
<td>contribute</td>
<td>Captures media, encodes via <a href="https://www.w3.org/2020/webcodecs">WebCodecs</a> and transmits over <a href="https://www.w3.org/transport">WebTransport</a>.</td>
</tr>
<tr>
<td>playback</td>
<td>Receives media over <a href="https://www.w3.org/transport">WebTransport</a>, decodes via <a href="https://www.w3.org/2020/webcodecs">WebCodecs</a>, and renders via <a href="https://www.w3.org/OffscreenCanvas">OffscreenCanvas</a> and <a href="https://www.w3.org/2020/worklet">WebWorklet</a>.</td>
</tr>
</tbody>
</table>

🚀flammable Time to rewrite it in Rust 🚀flammable
Google Quiche (Martin, renaming will happen someday)

**Language:** C++  **Transport:** WebTransport Only  **Endpoints:** Client

**Applications:**

Chat (client only)
moqtransport (Mathis)

**Language:** Go    **Transport:** QUIC and WebTransport
**Endpoints:** Client, Server

[https://github.com/mengelbart/moqtransport](https://github.com/mengelbart/moqtransport)

**Applications:**

- Chat (client/server)
- Clock
  - Server sending timestamps in objects every second
  - Client printing timestamps to stdout
- **Gstreamer video streaming**
  - Server sending video test stream in a single QUIC/WebTransport stream
Demo - 2 Way A/V Call

Relay (Akamai-London)

A (Prague)
B (Prague)

~160 ms
QuicR - Realtime Media Over QUIC (Cisco)

Language: C++
Transport: Raw QUIC (Picoquic)
Endpoints:
Mac Client: h264 1080p/720p, Opus 48khz
             Simulcast Transmission
             Tx Modes: Track, Datagram
Relay:      Object Cache
            TTL Expiration
            Priorities
Catalog Server: Custom Catalog Format

moqt-01 Features Supported
(a) Announce/Ok/Error/Unannounce
(b) Setup (Client/Server)
(c) Subscribe/Ok/Error/Unsubscribe

Deviating/Missing moqt-01/02 Features
(a) GoAway
(b) Original publisher can publish without a Subscribe
(c) moqt-catalog-00
(d) Subscribe Hints