CLOUDFLARE

Media over QUIC in Practice

Mike English
Principal Systems Engineer
Cloudflare

IETF 123 MOPS Working Group
2025-07-21 Madrid, Spain

Agenda

- **The "Why Now?"**: The Problem Space & Protocol Goals
- **In the Trenches**: Architectural & Implementation Lessons
- **The Ecosystem**: Ingest, Egress, and Players
- **The Road Ahead**: Next Steps & Call to Action

"Why Now?"

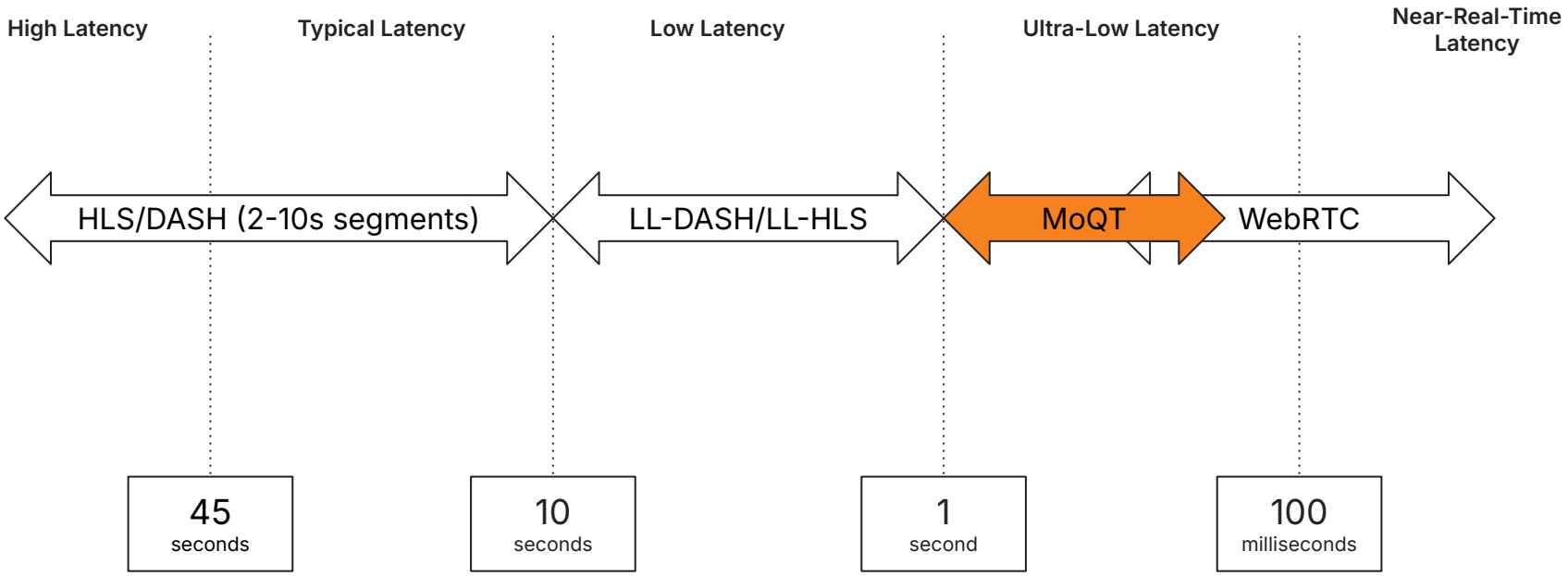
The State of Live Streaming

Segment-Based (HLS/DASH): Proven scale, but latency remains a fundamental architectural challenge.

Session-Based (WebRTC): Excellent latency, but introduces significant complexity in signaling and state management. The ecosystem is also constrained by the dominance of a single core implementation.

The Goal: A protocol that can achieve low latency at scale without the baggage of existing solutions.

The State of Live Streaming



(Based on a slide from Ali Begin)

The Proposition of MoQ

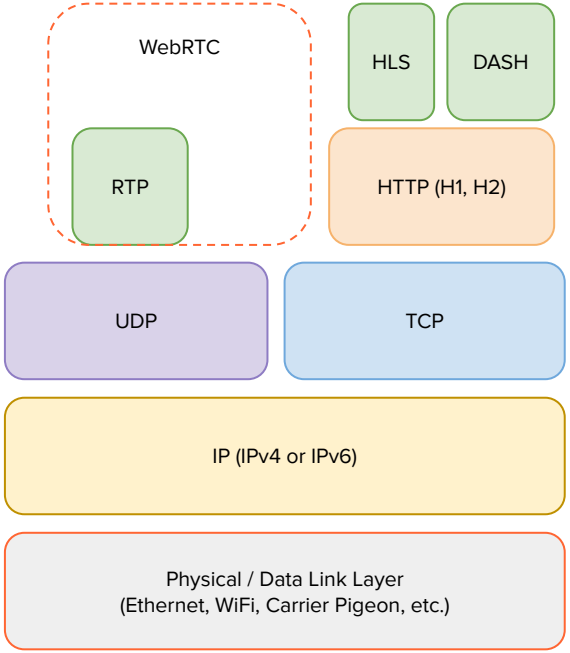
Why not just use HTTP/3? H3 solves transport-level Head-of-Line blocking, but not the media-level blocking inherent to manifest-and-segment-based delivery. H3 is not media-aware.

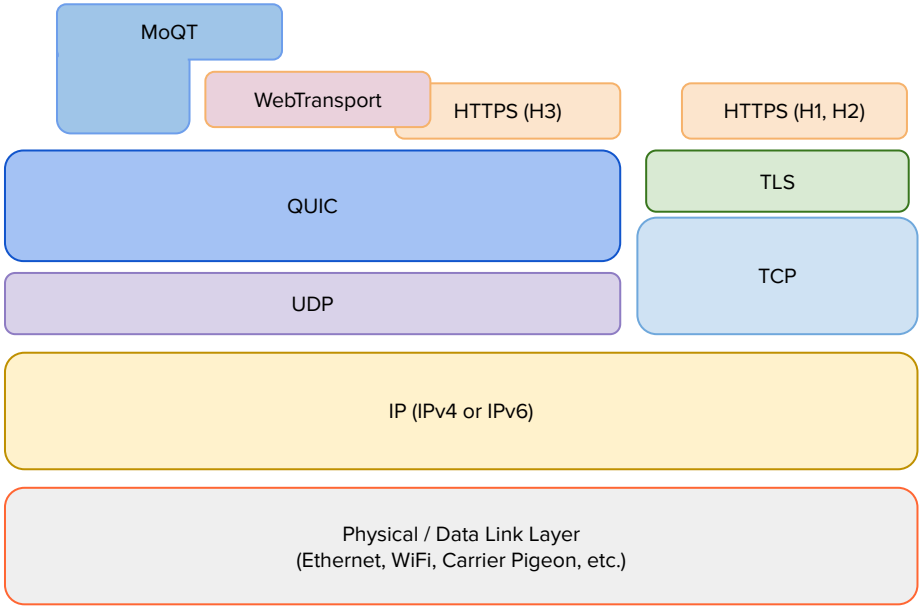
MoQ's Role: A simple mapping of encoded media onto the QUIC transport.

It's not a codec.

It defines just enough semantics for media delivery (tracks, priorities).

Where MoQ Fits in the Ecosystem





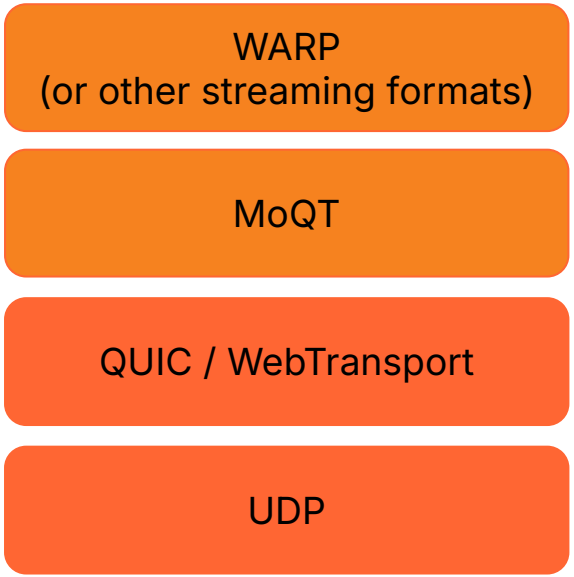
Where MoQ Fits in the Ecosystem

Ingest: A modern replacement for RTMP/SRT.

Distribution: A simpler, scalable alternative for one-to-many distribution, improving on WebRTC's complexity.

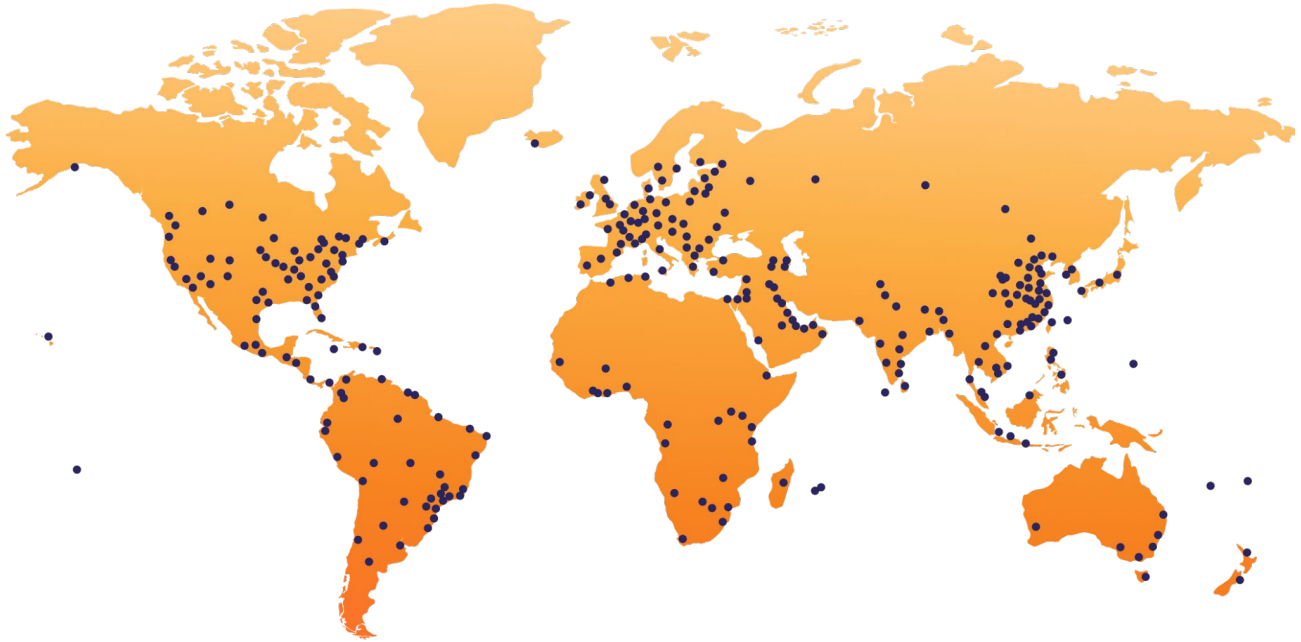
Economies of Scale: Because MoQ decouples the media details from the transport relay, the relay infrastructure can be simpler and more generic, potentially serving a wide variety of media and even non-media use cases.

Where MoQ Fits in the Ecosystem

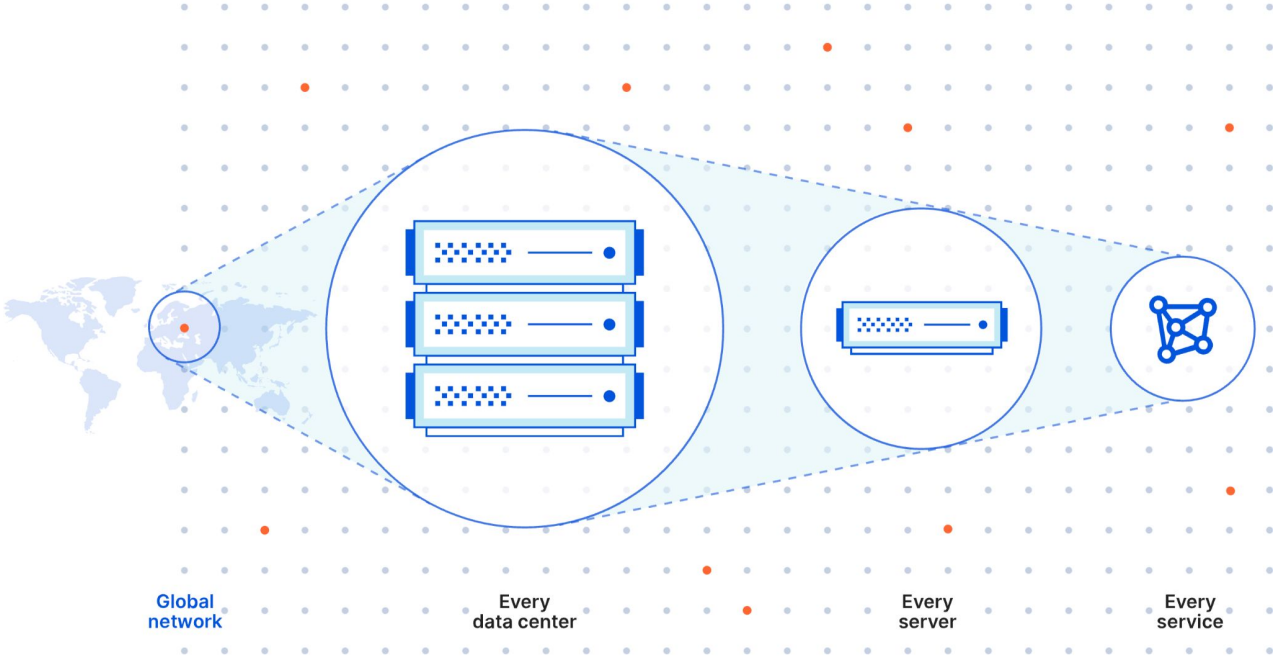


"In the Trenches"

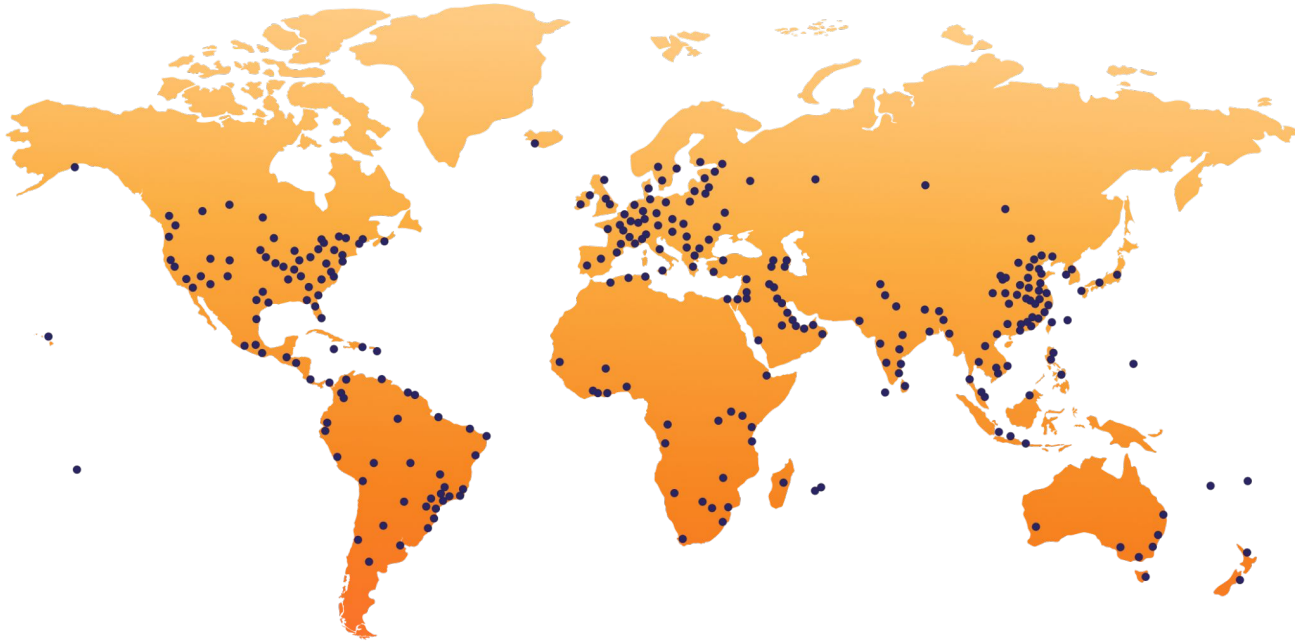
Architecture for a Global Anycast Network



Architecture for a Global Anycast Network



Largest ever MoQ relay deployment?



Architecture for a Global Anycast Network

Core Components & Library Choices

Language Preference: Strong preference for memory-safe languages. This work is primarily in Rust for its performance and safety guarantees.

QUIC/MoQ Library: Based on a fork of the moq-rs open-source implementation, which uses quinn for its QUIC layer.

Key Learning: Integrating an open-source implementation into a large-scale production environment often requires significant refactoring. (e.g., Plans to migrate from quinn to quiche to better align with internal tooling).

Implementing the "Glue" Logic

Spec Evolution: A major challenge is keeping an implementation in sync with a rapidly evolving specification. At times, the cumulative changes can make a rewrite more practical than continuous patching.

System Integration: The core MoQ logic is only part of the story. The "glue" involves integrating with existing proprietary systems for tasks like service discovery and internal routing.

Media Complexity (Opt-In): Implementers at the streaming format layer can choose their complexity. A simple player can subscribe to simple tracks, while an advanced player can leverage complex multiplexed tracks for higher performance. This allows for gradual adoption of advanced features.

Practical Tradeoffs for an MVP

Goal: A stable, end-to-end working system.

What we built:

- A forwarding-only relay mesh.
- Basic object ordering for prioritization.
- Integration with a pre-existing discovery service for track announcements.

What we deferred:

- Full implementation of the latest explicit prioritization schemes in the draft
- Updates to the latest draft version
- Some features, e.g. FETCH, SUBSCRIBE_ANNOUNCES
- Transcoding, transmuxing, or other media-level integration

The Debugging Challenge

New Hurdles:

- **Encryption:** tcpdump is less useful.
- **Binary Protocol:** Not human-readable like JSON over WebSockets.
- **Tooling Gaps:** WebTransport lacks the mature browser devtools of WebSockets.

New Solutions:

- **qlog:** Essential for visualizing QUIC connection-level events.
- **MoQ qlog event definitions:** A proposed extension to qlog for MoQ-specific information.
- **Custom Tooling:** The community is building tools to parse and visualize these logs.

The Ecosystem

Publishing: MoQ Encoders

Some commercial options now available: e.g. Norsk, Vindral

Open source demos: moq-pub, moq-encoder-player, mocklivemoq, moqtail

The Playback Opportunity

There is no native browser support for direct MoQ playback.

This is an Opportunity: It avoids the risk of a single implementation controlling the ecosystem.

The Path Forward: A rich ecosystem of JavaScript players using standard APIs.

WebTransport: Early support in Chrome, Firefox, *and* Safari (to varying degrees)

WebCodecs: Provides direct access to platform decoders.

Open source projects like **moq-js**, **moq-encoder-player**, and **moqtail** are already demonstrating this is a viable path. **warp-player** demonstrates an alternative CMAF + MSE based approach.

MoQ + newer web platform APIs allow flexibility so that players can be tailored to specific use cases.

MoQ Relays - Ingest & Egress: A Phased Approach

Our Current Focus:

Prove out a pure MoQ system for both ingest and distribution.

Future Ideas:

Build egress gateways to transmux MoQ for delivery to existing HLS/DASH players.

Build ingest gateways to bridge legacy workflows (e.g., RTMP -> MoQ, WHIP -> MoQ).

Allows MoQ to be integrated into an existing ecosystem gradually.

Open Source MoQ Implementations

See the wiki!

https://github.com/moq-wg/moq-transport/wiki/Interop

- **Interop**
  - Interop Matrix
  - MoQT Implementations
    1. Google Quiche
    2. moxygen
    3. moq-rs
    4. moq-js
    5. moqtransport
    6. moq-wasm
    7. quicr
    8. MOQ Encoder-Player
    9. aiomoqt-python
    10. moqtail (draft-10)
  - Applications built on moqt

The Road Ahead

What We're Measuring Next

With the relay network becoming operational, the next phase is performance characterization.

Key Metrics to Investigate:

- Glass-to-Glass Latency at scale.
- Stream resilience under controlled packet loss.
- Server footprint (CPU/memory per stream).
- Time to First Frame for new viewers.
- ...?

Audience Question: What metrics would be most valuable for you to see from a large-scale testbed?

Requests for the Community 🙏

The MoQ specification needs to be grounded in real-world experience.

Implement & Validate: We need more implementations to test the spec and find interop challenges.

Focus on Simplification: Use implementation experience to provide data-driven feedback to simplify the protocol where possible.

Contribute to the Next Layer: With MoQT stabilizing (🤞), we will soon need more focus on the streaming format layer (e.g., WARP), especially from media experts.

Finish WebTransport: MoQ represents a real use case that depends on broad and robust browser support!

IETF Working Groups and Drafts

Media over QUIC (moq) WG:

Meeting @ IETF 123:

Tuesday 9:30a CET in Auditorio

Thursday 9:30a CET in Castilla

- draft-ietf-moq-transport
- draft-ietf-moq-warp
- draft-pardue-moq-qlog-moq-events

WebTransport (webtrans) WG:

Meeting @ IETF 123:

Wednesday 11:30a CET in Patio 3

- draft-ietf-webtrans-overview
- draft-ietf-webtrans-http3

Thanks!

Questions? Comments? Ideas?

moq@ietf.org

or

Mike English

englishm@cloudflare.com