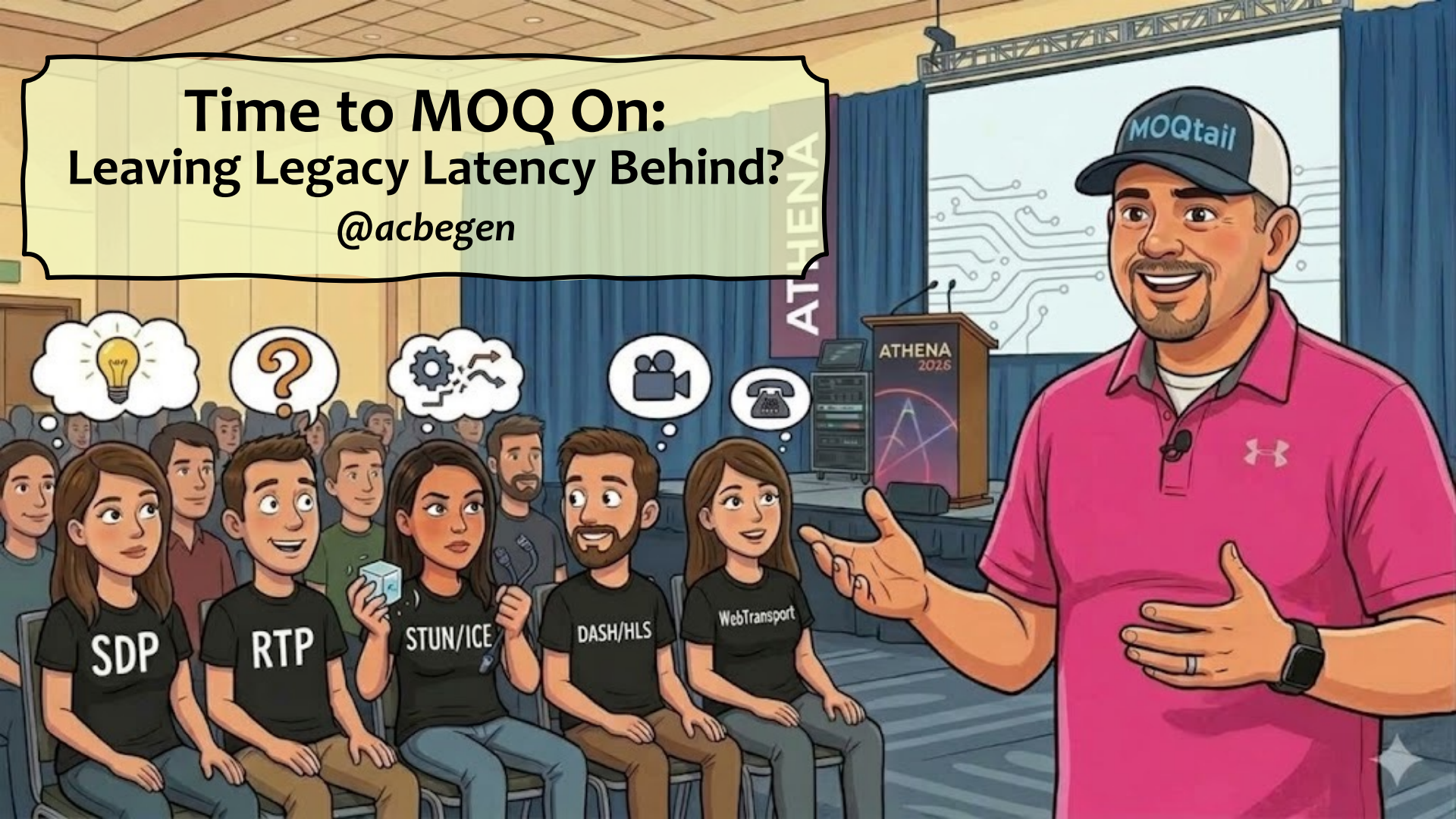
Time to MOQ On:
Leaving Legacy Latency Behind?

@acbegen





Using LL-DASH/LL-HLS for scalability, WebRTC for low latency and Sye for synchronicity



Using MOQ for scalability, low latency and synchronicity

The Real Cost of WebRTC



SDP Complexity

Developing with SDP means fighting chaotic O/A state machines and debugging endless browser interoperability quirks



STUN / ICE Pitfalls

NAT traversal remains unpredictable; blocked ports, firewall delays and recurring ICE restarts introduce significant connection drop-off rates



SFU Infrastructure

Scaling stateful SFUs globally is complex and financially punitive, demanding specialized servers and custom routing logic

Introducing MOQ Transport (MOQT)

Unified Transport Logic

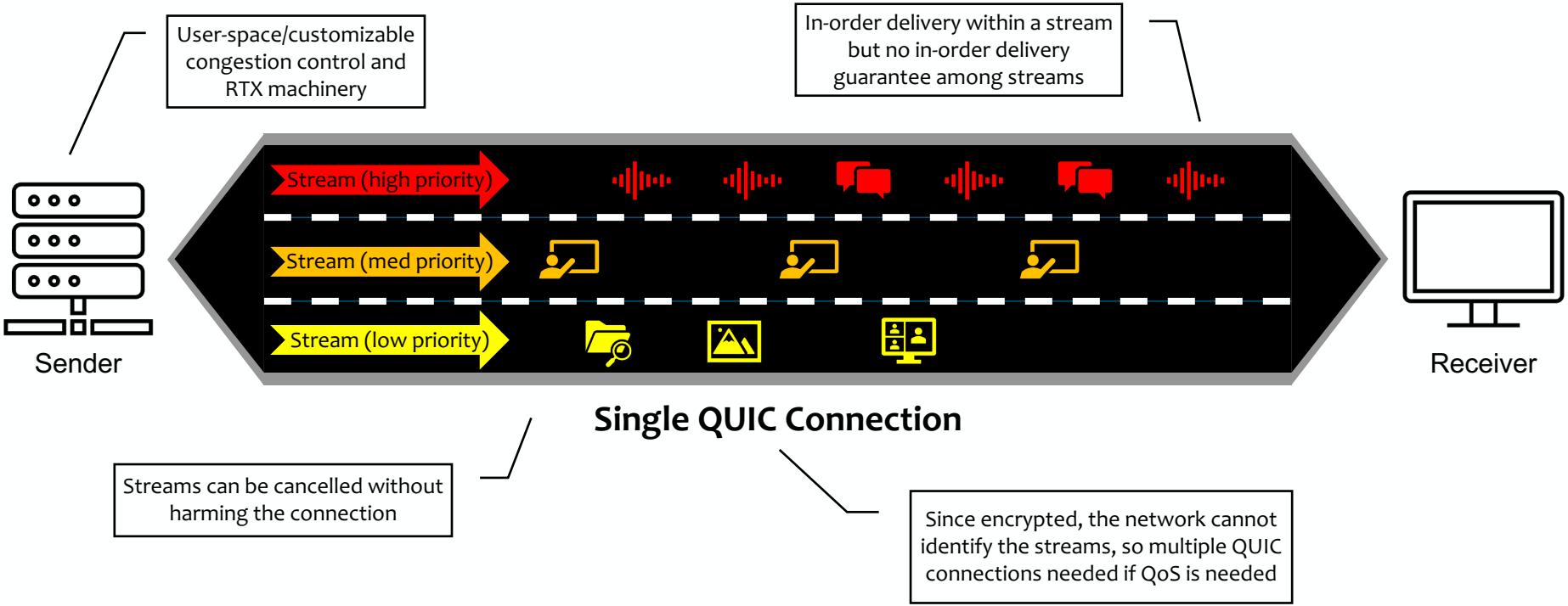
MOQT leverages the UDP-based QUIC protocol, natively integrating connection multiplexing, flow control and end-to-end encryption without requiring WebRTC's complex DTLS/SRTP setup

Clean Pub/Sub Paradigm

Ditch legacy signaling; MOQT introduces simple pub/sub concepts to control channels; developers interact with clean, app-level streams instead of wrestling with heavy negotiation formats

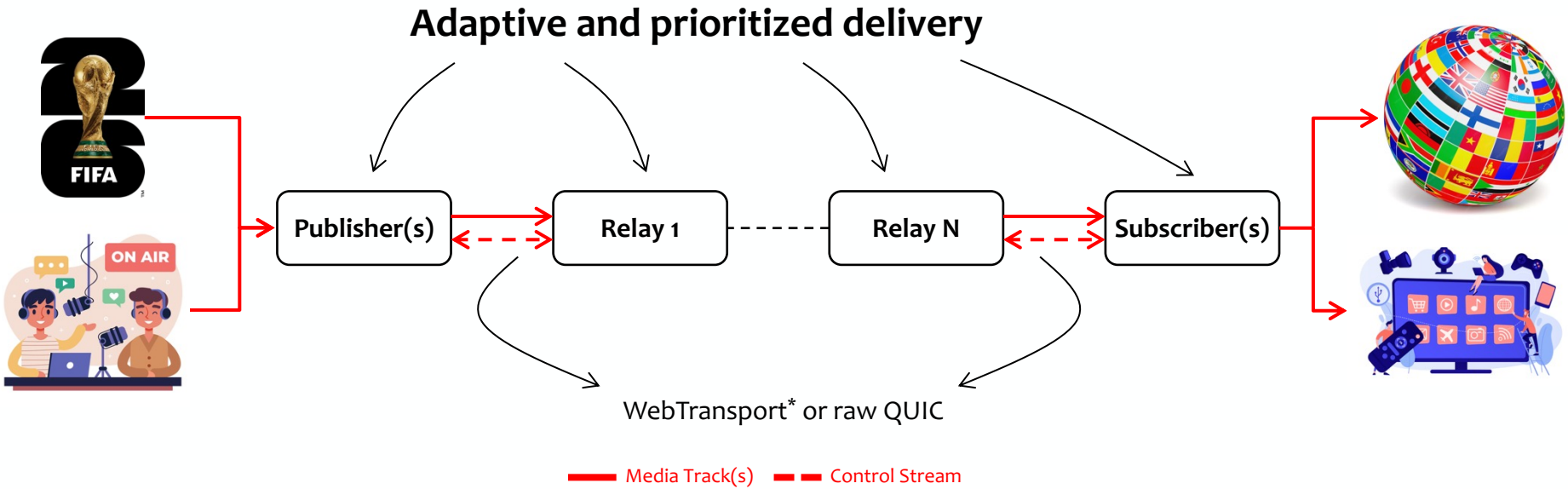
QUIC Is a Multi-Lane Highway

Partition the data into multiple streams with respective priorities → Helps avoid HoL blocking

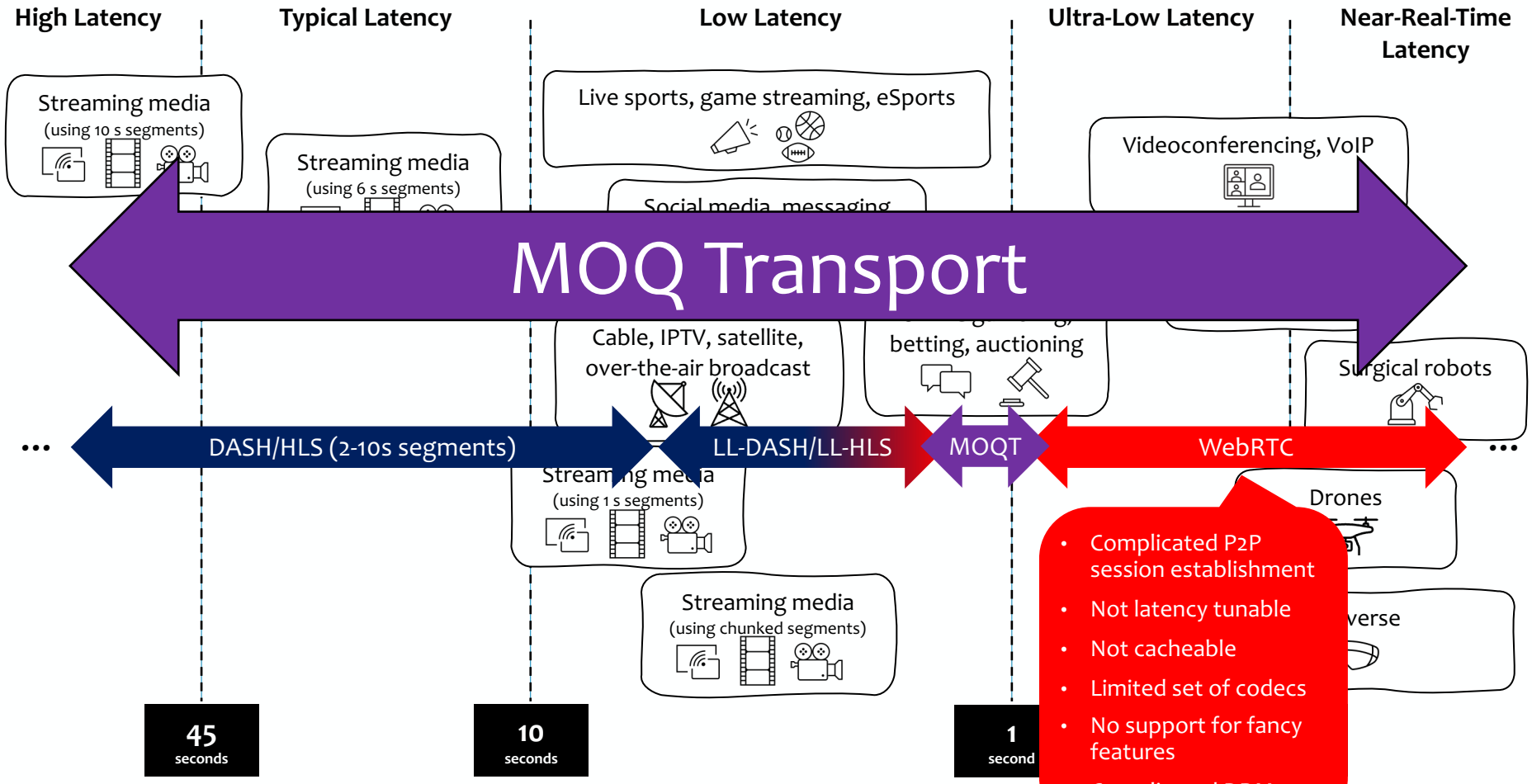


MOQ Transport

Latency-tunable, cache-friendly pub/sub protocol for ingest and distribution



*: Secure, multiplexed, bi-directional real-time transport; supports sending data both unreliably and reliably



Reading: Toward one-second latency: evolution of live media streaming – IEEE CS&T (2026)

Let's MOQ On

	Traditional WebRTC Stack	Modern MOQT Stack
Session Negotiation	SDP (heavy, complex text blobs)	Direct application-level APIs
Handshake Overheads	Heavy O/A cycles	Immediate pub/sub commands
Flexible Latency	No	Yes
Infrastructure State	Highly stateful	Lighter weight (no messing with the payload)
Caching Paradigm	No standard network caching	Native compatibility with MOQT CDNs
Codec Support	Limited	Virtually unlimited
Fancy Features/DRM	Limited	Virtually unlimited

MOQ Transport Supports



Tunable Latency

From real time to shifted-live and on-demand

Adaptive and Prioritized Delivery

Subscriber-side track switching
Sender-side track switching
Selective discard policies

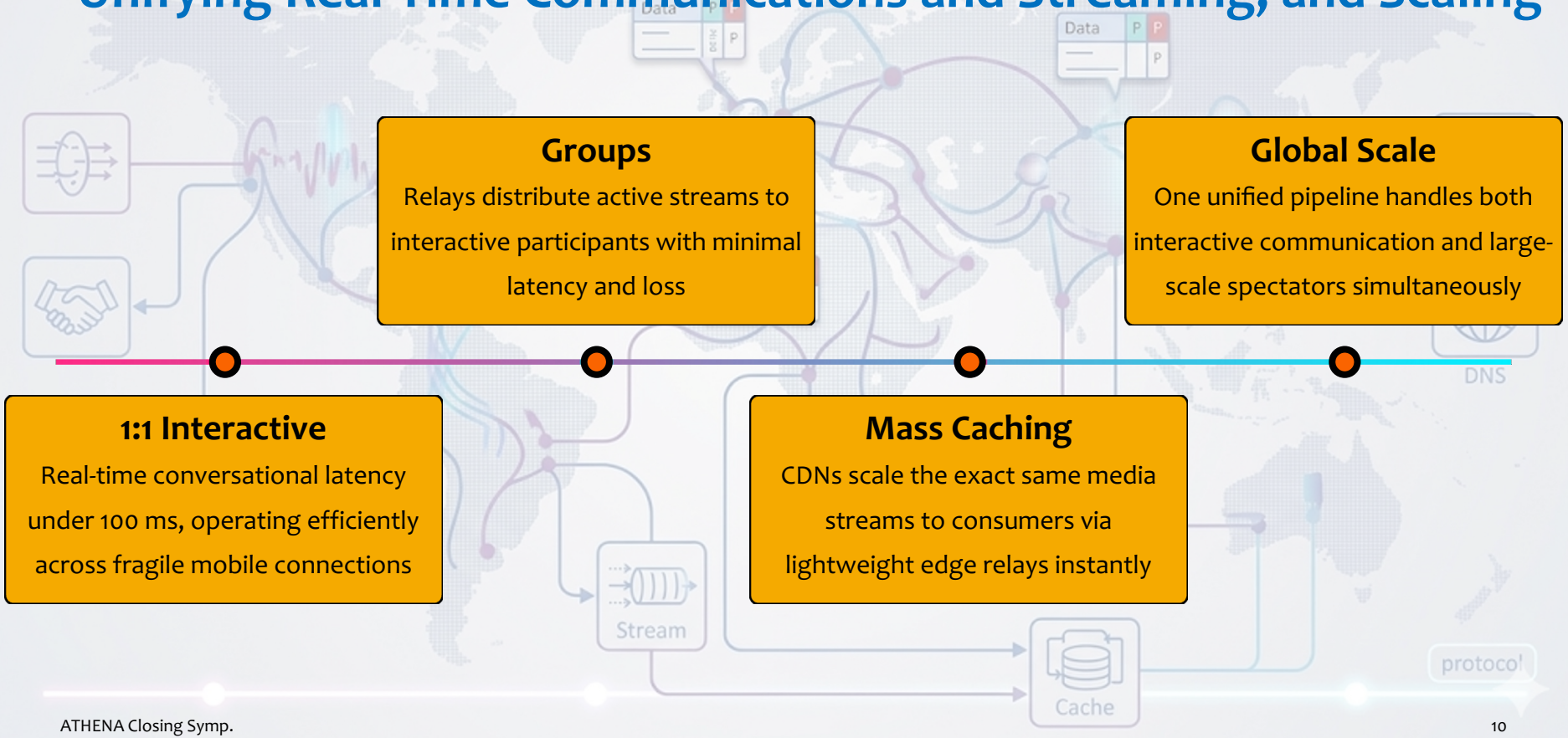
Synchronicity

Synchronized playback across subscribers for both live and on-demand

Scalability

CDN-like caching
No-media specific tasks at relays

Unifying Real-Time Communications and Streaming, and Scaling



1:1 Interactive

Real-time conversational latency under 100 ms, operating efficiently across fragile mobile connections

Groups

Relays distribute active streams to interactive participants with minimal latency and loss

Mass Caching

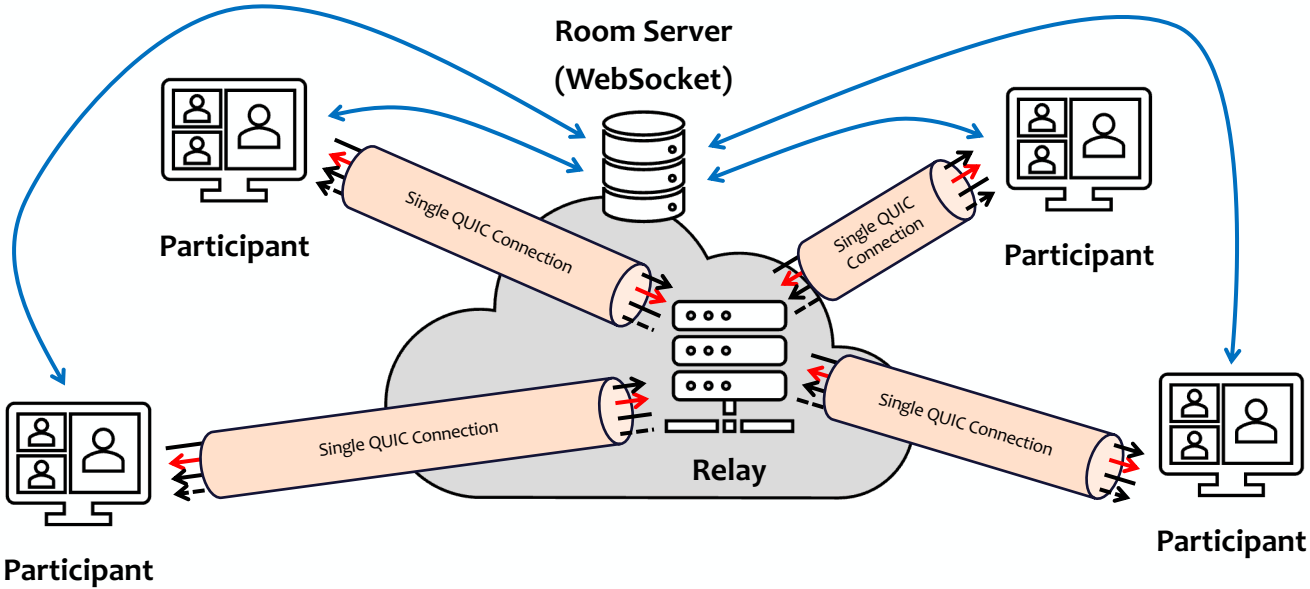
CDNs scale the exact same media streams to consumers via lightweight edge relays instantly

Global Scale

One unified pipeline handles both interactive communication and large-scale spectators simultaneously

Time Travel in MOQ Conferencing

https://timetravel.moqtail.dev/

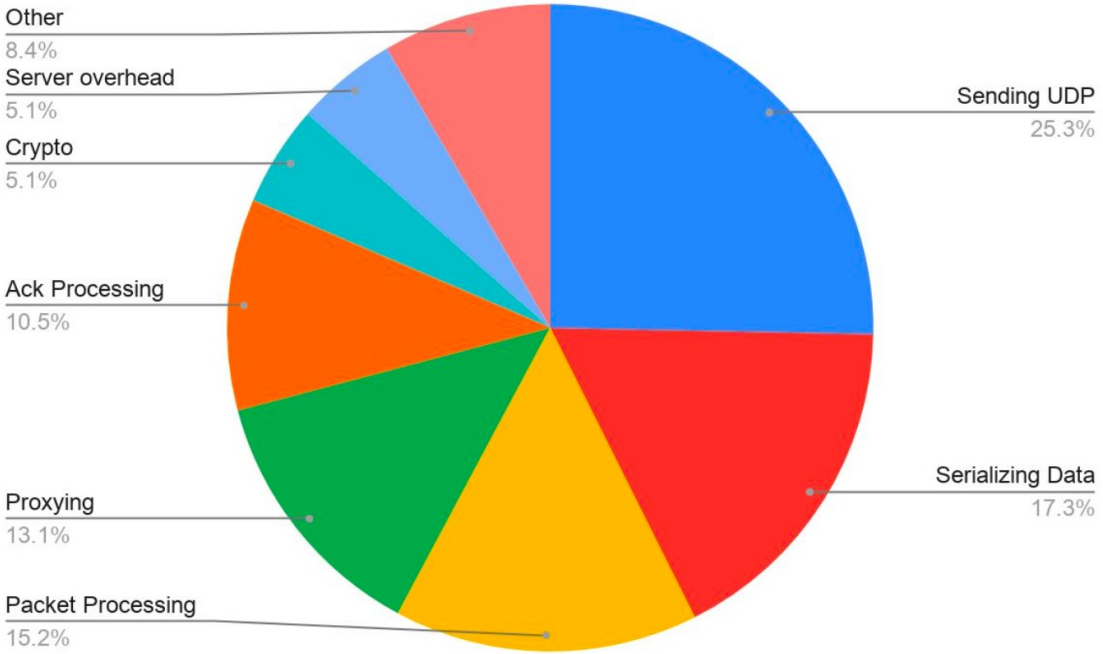


CPU Usage Reported by Google

January 2017 at 2x

Only sendmsg and one memcpy are obviously costly

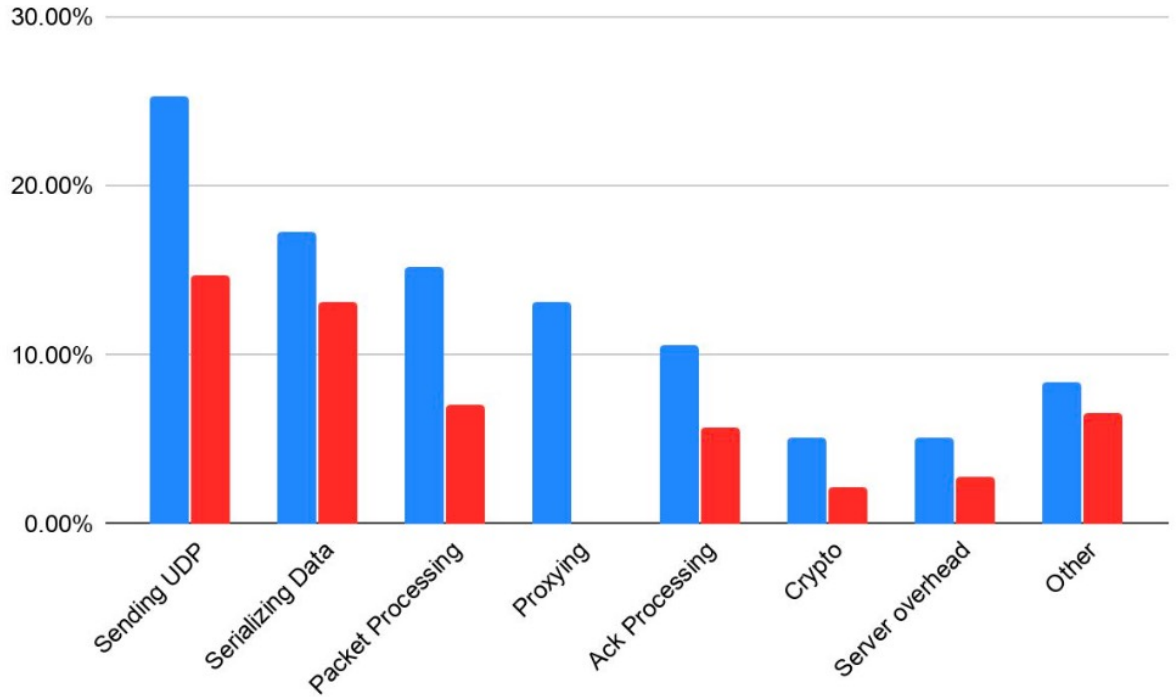
Other CPU users are tiny



Source: QUIC CPU Performance (SIGCOMM EPIQ 2020)

CPU Usage Reported by Google

Q4 2017 vs. 2020



Source: QUIC CPU Performance (SIGCOMM EPIQ 2020)

The Need for Track Switching

ABR–related reasons:
Info from the QUIC layer and auxiliary signals (e.g., SCONE)

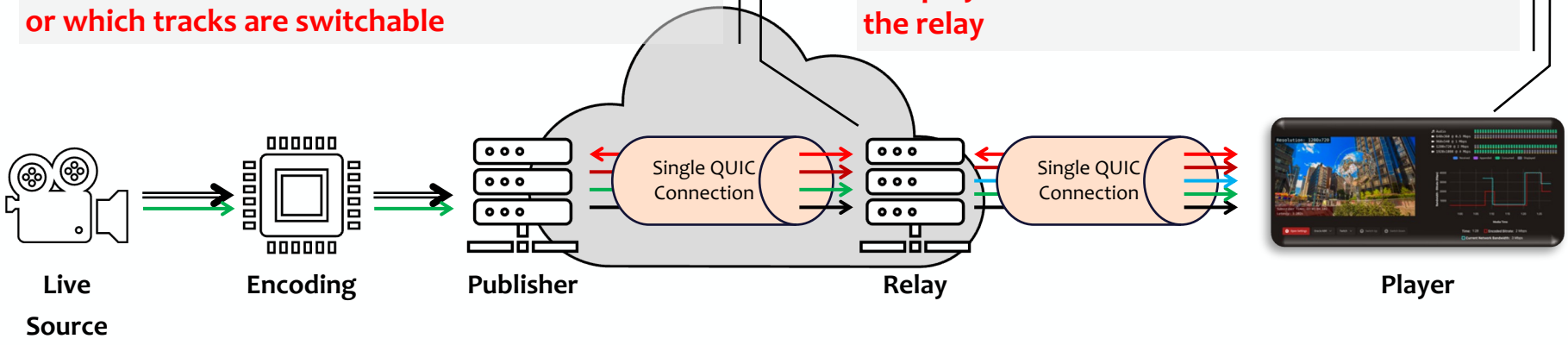
Non-ABR related reasons:
Policy or availability-based switching, centralized switching for thin clients, etc.

The relay does not know the player's conditions or which tracks are switchable

ABR–related reasons:
Measured/predicted bandwidth, buffer level, content type, stall rate, etc.

Non-ABR related reasons:
Viewer-initiated changes such as alternating the cameras in a multi-view event, changing the viewport in 360 video, changing audio language or deliberately reducing quality to minimize data usage

The player does not know the track statuses on the relay



Desired Features of Seamless Track Switching

Zero Stalls

Ensure that the stream never buffers, delivering uninterrupted flow state

Critical for any app

No Gaps or Skips

Play continuously during switching, eliminating noticeable audio/visual alignment dropouts

More critical for streaming

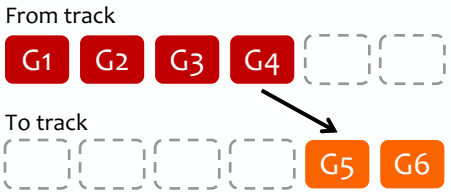
Minimized Waste

Avoid redundant data download and unnecessary data overlap when swapping tracks

Critical for any app







Different Options

Forward State Toggling

- Subscriber pre-subscribes to target track (fwd=0), at switch time: it sends two REQUEST_UPDATEs
- The relay acts on each immediately; no coordination

Risk: Gap or duplicate data if boundary timing is off

Joining FETCH

- Uses a FETCH to pre-buffer past groups of target track
- Subscriber checks LARGEST_OBJECT parameter to decide
- Two downstream legs required

Risk: Fetched (past) data competes with live data under congestion

SWITCH Message

- Single message: subscriber specifies current and target track
- Relay finds next group boundary atomically
- No subscriber feedback loop needed

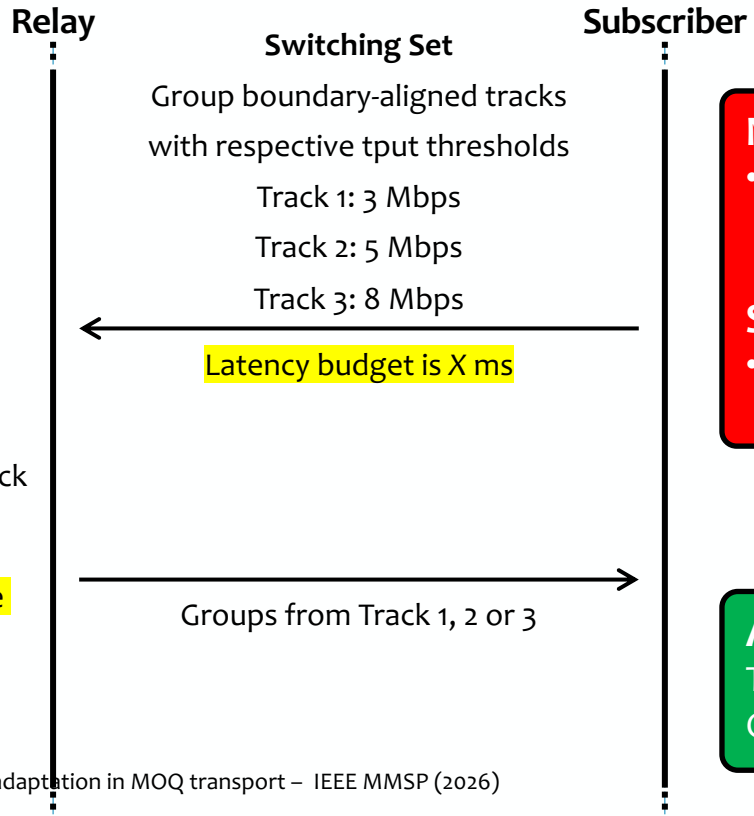
Risk: No-stall switching may be impossible if boundary timing is way off

	Forward State Toggling	Joining FETCH	SWITCH Message
# of control messages	2	3–4	**1**
Delivery gap	Maybe	**None**	**None**
Redundant traffic	Maybe	Maybe	**None**
Boundary alignment	Subscriber	Subscriber	Relay

Reading: Zero gap, zero waste: subscriber-initiated, relay-executed track switching in MOQ transport – IEEE MMSP (2026)

Relay-Side Latency-Budget-Aware Switching

Enhancements on https://wilaw.github.io/moqt-ssts/draft-wilaw-moq-moqt-ssts.html



1. Subscribe to Tracks 1, 2 and 3
2. Measure the bandwidth
3. Pick the next group from the best track whose throughput threshold is not higher than the current bandwidth
4. Early discard a subgroup if it won't be delivered within the latency budget

Network telemetry is noisy

- RTT and cwnd estimates are often impractical for app-limited sending regimes

Something less noisy?

- Streams overlap when the network cannot keep up

Active Stream Depth

The count of concurrently open QUIC send streams per media

Reading: Relay-side latency-budget-aware rate adaptation in MOQ transport – IEEE MMSP (2026)



It is demo time

Demos at moqtail.dev/demo

Adaptive Live Streaming (aka ABR) with MOQ

Revamped ABR implementation for MOQtail

f6dfaa8 | 9/3/2026 | Draft 18

Demo | E2E Architecture

Subscriber-Side Fast Switching

The first sub-second fast switching implementation for MOQT

f6dfaa8 | 9/3/2026 | Draft 18

Demo | E2E Architecture

Sender-Side Track Switching

Sender-Side controlled track switching implementation for MOQtail

f6dfaa8 | 9/3/2026 | Draft 18

Demo | E2E Architecture

MOQ-n-Roll: Live Sports Watch Parties

Live sports watch parties using streams and datagrams

a2375af | 9/1/2026 | Draft 16

Demo | E2E Architecture

MOQ Live Streaming with AI-Generated Event Timeline

IBC 2025 demo

c3d0138 | 7/10/2026 | Draft 11

Demo | E2E Architecture

SyncPlay: Synchronized VoD with MOQ

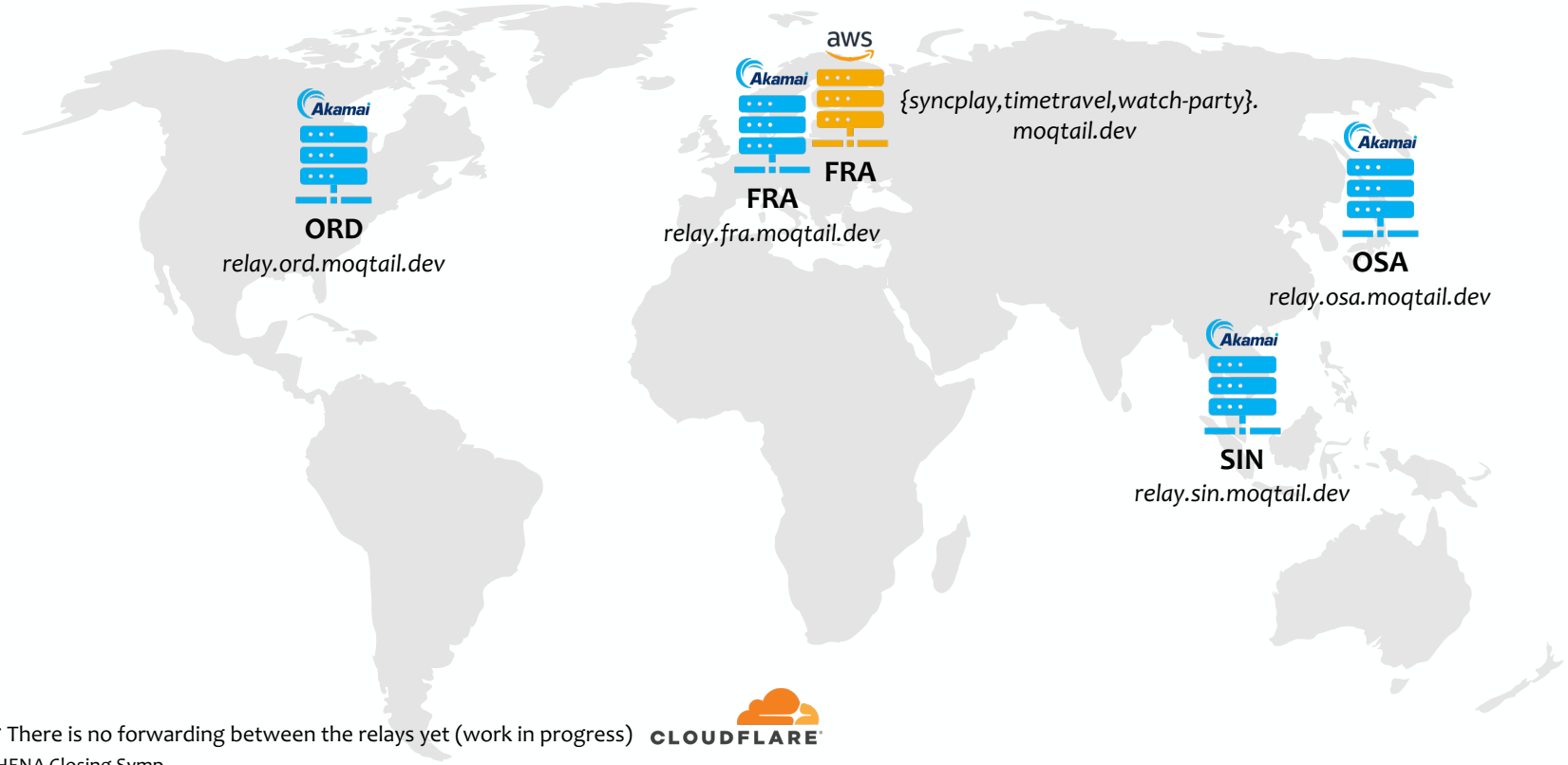
Using MOQ for synchronized viewing of on-demand content

1475e67 | 7/14/2026 | Draft 14

Demo | E2E Architecture

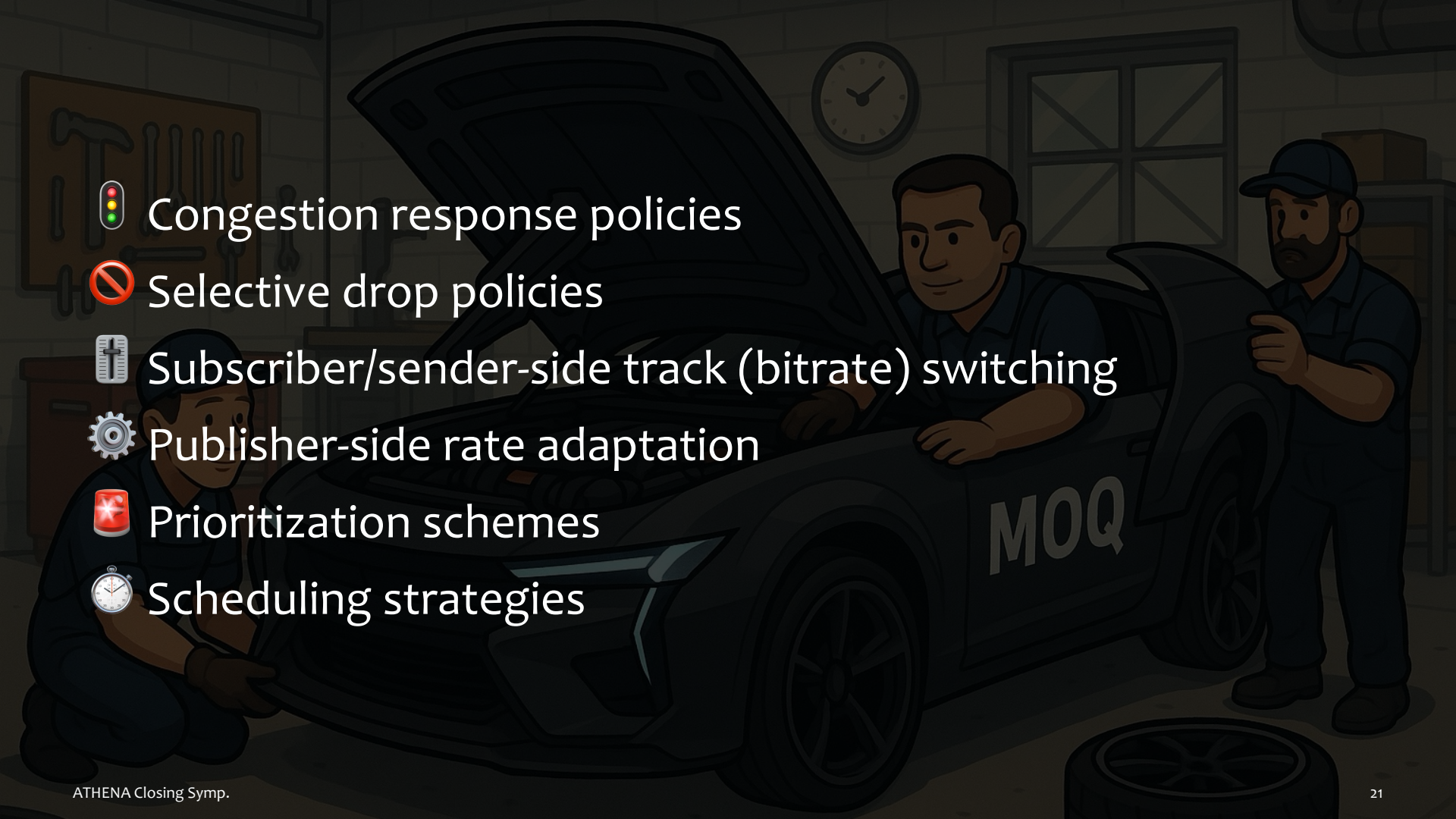
MOQtail Relays

Automatic load balancer at relay.moqtail.dev*



* There is no forwarding between the relays yet (work in progress)

CLOUDFLARE



- Congestion response policies
- Selective drop policies
- Subscriber/sender-side track (bitrate) switching
- Publisher-side rate adaptation
- Prioritization schemes
- Scheduling strategies


MOQ Meetup Japan - Oct. 2026
MOQ ミートアップジャパン - 2026年10月
切
MOQ
KREEE-SH!
爆!
DASH
HLS
WebRTC



Sip yourself a drink at https://moqtail.dev

and please do not abuse the relays ☺

ÖZYEĞİN UNIVERSITY

OpenMOQ