MoQ Production Practice & Multimodal Feedback

Media over QUIC Transport

XQUIC Dev Group

Production practice, experiment data & MoQ Multimodal Feedback draft

Scenarios: MoQ in Production

MoQ (XQUIC-based) deployed in production at scale:

1. Taobao Voice Search

Audio capture + 3A processing (AEC, ANS, AGC) + MoQ low-latency transport. Full-platform: Android / iOS / HarmonyOS.

2. Digital Human (Cloud Rendering)

Real-time cloud-rendered avatar. User gesture -> video frame feedback latency is the core metric. Replaced WebRTC with MoQ.

3. Taobao Live Digital Human

AI-driven 1v1 live-streaming. LLM output -> audio + lip-sync features -> client rendering. 24/7 AI customer service.

4. Alipay AI Assistants

Real-time streaming interactive AI services: Hangzhou Guide / Employment Assistant. Interruptible + low-latency.

Additionally integrating: Amap, Qwen APP, Ant real-time interaction, multimodal model services.

Why MoQ? Comparison with WebSocket & WebRTC

WebSocket

- TCP-based: **head-of-line blocking**
- No built-in multiplexing
- No media-aware CC or priority
- High connection latency (TCP+TLS)
- Simple byte-stream, no media semantics

Not designed for real-time media

WebRTC

- Complex setup (ICE/STUN/TURN/SDP)
- Tightly coupled RTP/RTCP stack
- Hard to customize for AI inference
- Kernel-space UDP, limited CC flexibility
- Designed for peer-to-peer, not pub/sub

Good for video calls, hard to extend

MoQ on QUIC

- QUIC: **no HOL blocking**, multiplexed streams
- **0-RTT** connection, session reuse
- User-space transport, **pluggable CC**
- Pub/Sub model with Track/Object/Group
- Flexible priority & media semantics

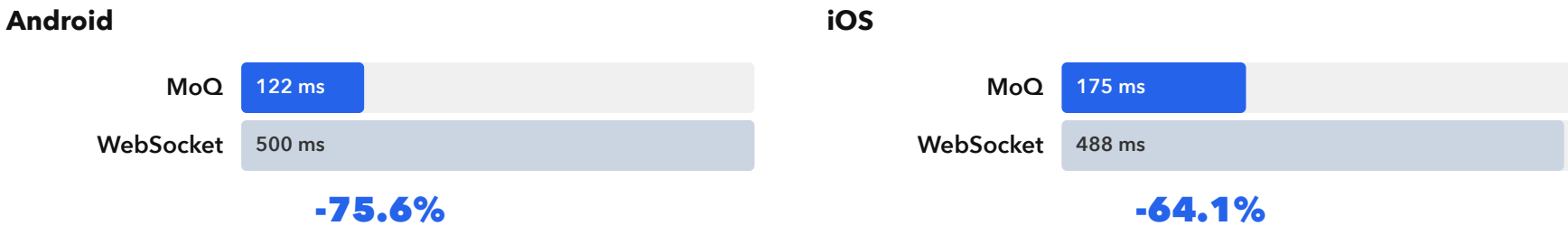
Built for low-latency media + extensible

MoQ inherits QUIC's transport advantages and adds **media-semantic abstractions** (Track / Object / Group) that neither WebSocket nor WebRTC provide natively.

Data: Taobao Voice Search (MoQ vs WebSocket)

Voice search uses MoQ for audio capture + 3A processing + low-latency transport.

Connection Latency Comparison



QUIC 0-RTT + MoQ session reuse significantly reduces connection setup overhead compared to WebSocket (TCP + TLS handshake).

Data: Cloud Rendering Digital Human (MoQ vs WebRTC)

User gesture -> cloud rendering -> video frame feedback. Latency directly affects interaction fluency.

-51.8%

First Frame P90

1900 ms -> 964 ms

-17%

Interaction Delay P90

-25%

Interaction Delay P95

50% online traffic A/B test. MoQ replaces WebRTC for cloud rendering delivery, achieving lower first-frame latency and smoother interaction due to QUIC multiplexing and flexible priority control.

Data: Alipay AI Assistants & More

Ant Alipay: Real-Time AI Interaction

>50%

Connection Latency Reduction
vs traditional pipeline

>50%

Average Delay Reduction
real-time streaming interaction

Scenarios: Hangzhou Guide (city assistant), Employment Assistant. All achieve real-time streaming, interruptible experience.

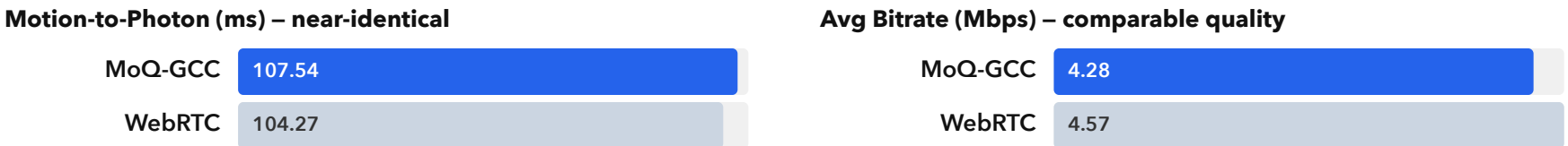
Also Integrating MoQ

Amap (Navigation AI) | Qwen APP | Ant Real-Time Interaction | Multimodal Model Services

Data: Cloud Rendering — MoQ+GCC vs WebRTC (After CC Optimization)

After implementing 5 CC algorithms (BBR/BBRv2/Cubic/New Reno/GCC) on XQUIC and optimizing for real-time scenarios, **MoQ-GCC** achieves latency comparable to WebRTC, with similar bitrate. Under normal network conditions, we can deliver performance on par with WebRTC.

Latency & Bitrate (MoQ-GCC vs WebRTC)



Insight: CC optimization achieves comparable latency + fewer stalls, but **transport-layer CC alone still cannot solve the media quality problem** – we need application-layer feedback.

Ref: *JitBright: Towards Low-Latency Mobile Cloud Rendering through Jitter Buffer Optimization*; *MARC: Motion-Aware Rate Control for Mobile E-Commerce Cloud Rendering*

Open Source & Interoperability

Open-source QUIC and HTTP/3 implementation.

Foundation for our MoQ transport layer.

XQUIC participates in the **MoQ Interop Runner**.

Stable support for multiple draft versions, with **draft-16** adaptation in progress.

draft-14 **draft-05** **draft-16 (WIP)**

We welcome contributions and feedback from the community.

`github.com/alibaba/xquic`

Complementary Feedback: Transport + Application

QUIC provides excellent **transport-layer** feedback. MoQ needs to add **application-layer** feedback on top of it:

QUIC Layer (already works well)

- ACK: per-packet loss detection
- receive-ts: per-packet delay gradient
- RTT / bandwidth estimation
- Drives CC: pacing_rate, cwnd

Scope: **packet-level, transport-level**

+

MoQ Layer (what we need to add)

- Per-Object delivery status (complete? late? lost?)
- Frame-level arrival timing (inter-Object delta)
- Playback buffer headroom
- Drives App: bitrate, FPS, ABR, inference tuning

Scope: **Object-level, media-semantic**

Together they form a **complete dual-layer feedback system** – this is why we propose **MoQ Multimodal Feedback**.

Why Does Feedback Improve Performance?

Without Feedback (Baseline)

1. Sender always sends at **max bitrate** (~7000 kbps)
2. When network degrades, packets queue / drop
3. Sender is **unaware** – keeps sending high bitrate
4. Receiver: FPS collapses (30 -> 4 fps), severe stalling
5. QUIC ACK detects packet loss, but **has no frame-level knowledge** – cannot tell encoder to reduce bitrate

With Feedback

1. Receiver reports per-Object delivery status back to sender
2. Sender receives: Object loss rate, late rate, buffer level
3. **App-layer adapts**: encoder reduces bitrate to match bandwidth
4. **CC-layer adapts**: supplementary signals help CC converge faster
5. Sender stays within bandwidth -> stable FPS, low stalling

Key insight: **QUIC CC alone adjusts pacing_rate/cwnd, but the encoder needs explicit media-semantic signals to adapt bitrate.**

Experiment: Video Feedback – Overview

Preliminary experiment – controlled environment, initial validation of the feedback mechanism.

Both baseline and experiment groups are **MoQ-based**.

The only difference is whether Delivery Feedback is enabled:

Without Feedback

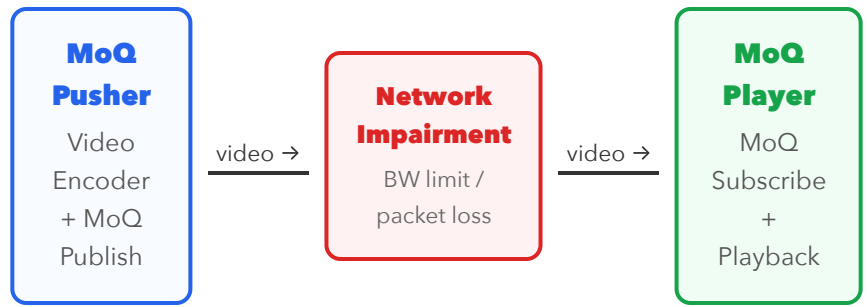
Sender always sends at max bitrate.
No adaptation.

With Feedback

Receiver reports delivery status.
Sender adapts bitrate accordingly.

Key finding: Under bandwidth-limited (≤5Mbps) and high-loss (≥30%) scenarios, the feedback version shows significantly better performance – lower latency and fewer stalls.

Test Setup



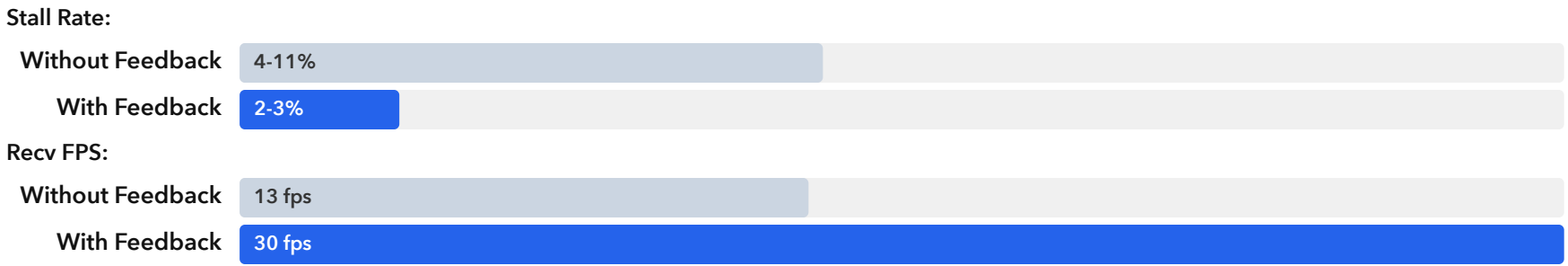
Both groups use MoQ transport. Impairment applied on the MoQ link only.

AntRTC 蚂蚁集团 ANT GROUP Experiment contributed by AntRTC, Ant Group.

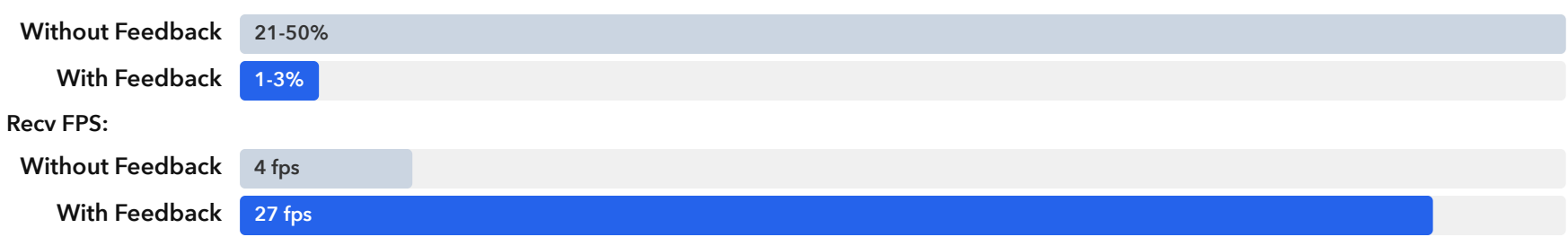
Experiment: Bandwidth-Limited Scenarios

Video: 1080p, Mac camera. Stall = frame interval > 200ms within 5s window.

5 Mbps Bandwidth Limit



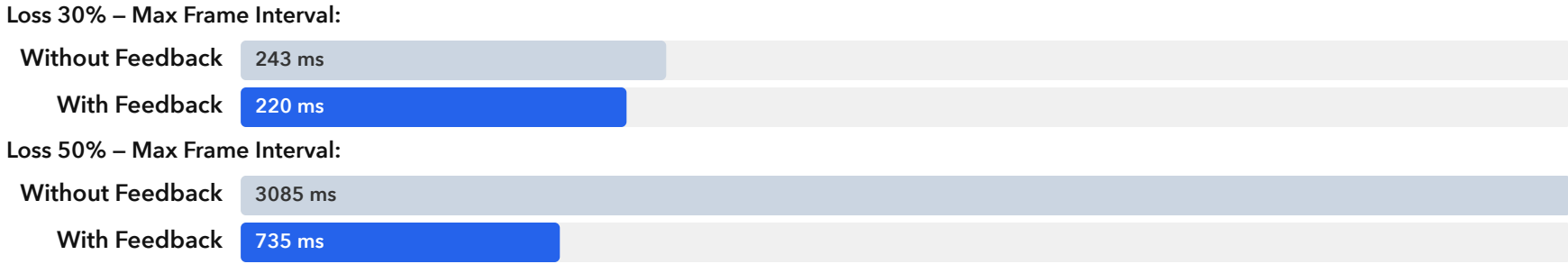
3 Mbps Bandwidth Limit



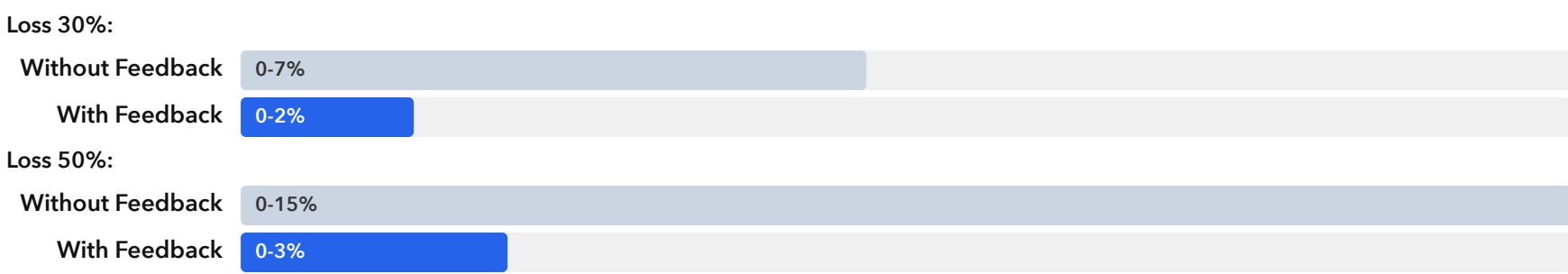
Without Feedback sends at fixed ~7000 kbps. With Feedback adapts to 3871/2232 kbps -> stable.

Experiment: Packet Loss Scenarios

Max Frame Interval Comparison (no bandwidth limit)

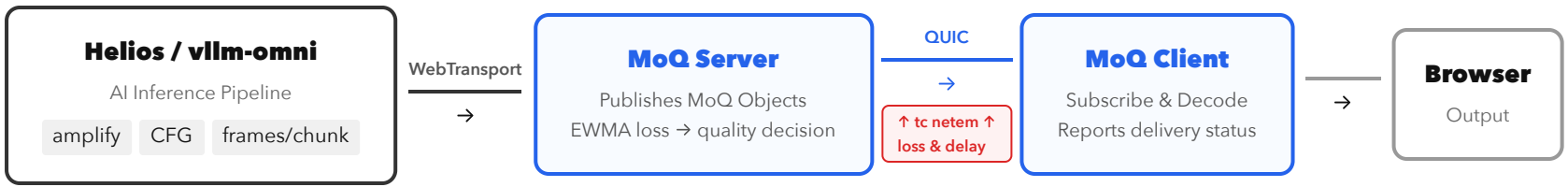


Stall Rate Comparison



Feedback-driven adaptation is most valuable under constrained/lossy conditions.

Experiment: AI Inference Video – Architecture



Feedback Loop

1. MoQ Client reports **Delivery Feedback** per Object
2. MoQ Server detects degradation (EWMA loss) → decides quality level
3. Quality signal sent upstream → Inference pipeline adapts parameters

Inference Adaptation

HIGH	MED	LOW
amplify ON	OFF	OFF
CFG 5.0	1.0	1.0
33 fr/chunk	28	24

Experiment: AI Inference Video — Results

100s per run, 5-phase network pattern. Feedback triggers inference-side adaptation.

Good Phase (0-20s, no impairment)

FPS	FB **9.6** / NoFB **9.4**	**Avg Interval**	FB **72ms** / NoFB **72ms**	**Stalls** (>200ms)	FB **0** / NoFB **0**

Both groups identical. **Feedback introduces zero overhead.**

Bad Phase (60-80s, delay=80ms, loss=5%)

+33%	**+33%**	**-29%**	**-33%**
Frame Arrival	FPS	Avg Interval	Severe Stalls >500ms
176 vs 132	**8.8 vs 6.6**	**107 vs 150ms**	**4 vs 6**

Max Stall Duration:

Proposal: MoQ Multimodal Feedback

draft-moq-multimodal-feedback – An extension to MOQT

Design Principles

- **Feedback Track** (regular MoQ Track) carries feedback reports – no new QUIC frames or control messages
- **Per-Object, per-Track** media-semantic feedback (not per-packet)
- **Dual consumption**: upward to app layer + downward to CC algorithms
- Complements **QUIC Receive Timestamps** – dual-layer feedback model

Key Feedback Report Fields

Field	Purpose
Object Status	RECEIVED / RECEIVED_LATE / NOT_RECEIVED / PARTIALLY_RECEIVED
Receive Timestamp Delta	Per-Object arrival time delta chain (similar to TWCC)
Summary Stats	Windowed: Objects Lost, Late, Avg Inter-Arrival Delta
PLAYOUT_AHEAD_MS	Peer's playback buffer headroom

Feedback-Driven Adaptation: How It Works

Sender Side

Encoder / Inference
Video: bitrate, FPS, QP, resolution
Audio: Opus bitrate, codec params
Inference: chunk_size, flush, quality, output token rate

MoQ Publish
Audio | Video | Text

QUIC CC: ACK + receive-ts -> pacing_rate, cwnd

► media Objects ►

Receiver Side

MoQ Subscribe -> Decode -> Playback
Records per-Object: arrival time, status, buffer level

◄ Feedback Track ◄

Feedback Signals & Actions

Receiver reports:

- Object status (received / late / lost)
- Receive timestamp delta chain
- PLAYOUT_AHEAD_MS (buffer)
- Estimated bandwidth

↑ App-layer adjusts:

- Video: bitrate, FPS, resolution, QP, ABR switch
- Audio: Opus bitrate (6-128 kbps)
- Inference: chunk_size, flush strategy, generation quality, output token rate, vocoder params
- Priority: pause video Track, keep audio

↓ CC-layer supplements:

- Object loss rate -> stop probing
- Late rate -> treat as equiv. loss
- Inter-arrival delta -> cross-validate
- Buffer low -> conservative pacing

Dual-Layer Feedback: QUIC receive-ts + Multimodal Feedback

Layer	Granularity	CC Role	App Role
QUIC receive-ts	per-packet (~us)	**Primary** (delay gradient, BW est.)	None
Multimodal Feedback	per-Object (~ms)	**Supplement** (loss, late, buffer)	**Primary** (bitrate/ABR/inference)

How They Cooperate

1. **QUIC receive-ts**: per-packet inter-arrival deltas for delay-based CC (= WebRTC TWCC)
2. **Multimodal Feedback** adds what receive-ts cannot:
   - Media semantics: Object completeness, deadline awareness
   - App metrics: playback buffer headroom (PLAYOUT_AHEAD)
   - Compared to GCC+TWCC, additionally provides **Object-semantic feedback**

> **Signal Priority**: Local QUIC ACK/receive-ts > Peer Feedback. Use the more conservative estimate on conflict.

Summary & Discussion

What We Shared

- **Production scenarios** at scale
- **Measurable gains** vs WebSocket / WebRTC
- **QUIC feedback blind spots**
- **draft-moq-multimodal-feedback**

Open Questions

- **Consensus?** Do we need more feedback for real-time transport scenarios?
- **Placement?** MoQ-layer Track vs extending QUIC frames?
- **Scope?** Media-generic or include AI inference?
- **Next steps:** submit draft, welcome co-authors

Thank you – looking forward to your feedback and discussion.