MOQ POC

IETF 116

Jordi Cenzano

Disclaimer

- **Early stages Proof of concept (POC)**
  - Just A way (NOT the way) of implementing MOQ

- **Goal:**
  - Learn about proposed technologies
  - Provide as much information / metrics possible

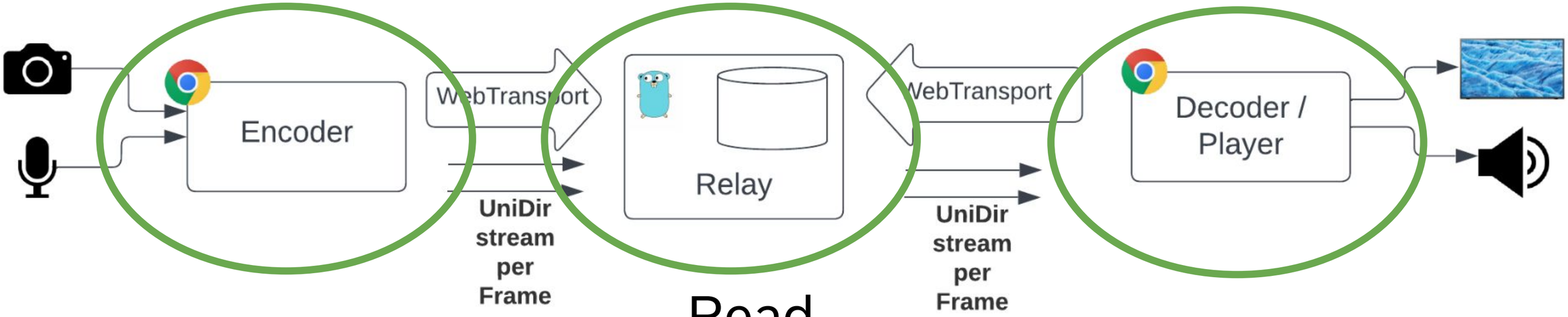
- **Future:**
  - Hope could help us to have a better discussion about MOQ options
  - Perhaps give some light over some trade-offs to make

Use cases

- **Ingest**
  - ULL live
- **Egress**
  - ULL live (live edge)
  - Live rewind
  - Highlights / VOD

- **Not yet**
  - Priorities (sendOrder)
  - ABR

03 What it is?



Read while write

Test Ultra low latency with Webcodecs + WebTransport: PLAYER

server -> Demux -> Decode -> Play

(Encoder audio sampling frequency should be the same than audioContext (player) sampling frequency, this is almost guaranteed if you use same brov

Data needed

WT server: https://moq-test.oregon.jordicenzano.dev:4433/moqdelivery

Stream type: Live edge StreamID: streamtest

Packager type: v2 - Binary

Player buffer (ms): 100 (it waits until audio buffers this amount to start playback)

Audio jitter buffer buffer for this player (ms): 200 Video jitter buffer buffer for this player (ms): 100

Start Stop

Test Ultra low latency with Webcodecs: ENCODER

WebCam(v+a) -> Encode -> Mux -> Send -> Server

Data needed

WT server: https://moq-test.oregon.jordicenzano.dev:4433/moqingest

StreamID: 20230328042624 Old StreamID: -

Packager type: v2 - Binary

Max audio sending buffer allowed (ms): 300

Max video sending buffer allowed (ms): 150

Max inflight audio requests: 100

Max inflight video requests: 50

Expiration time for media chunks (except init) (in secs): 120

Start Stop

Main features

- **Uses QUIC streams over WebTransport (WT)**
- **QUIC stream per frame**

- Encoder
  - Opens WT session against relay
  - Pushes data to relay

- Player
  - Opens WT session against relay
  - Relay pushes data to player

Demo setup


~8000Km (4970 miles)
RTT ~ 140ms (ping)
Sea of Okhotsk
Yelizovo
Елизово
Sapporo
札幌
HEILONGJIANG
JILIN
LIAONING
North Korea
Sea of Japan
South Korea
Osaka
大阪
Yellow Sea
East China Sea
JIANGSU
ANHUI
ZHEJIANG
FUJIAN
Taipei
台北
Taiwan
Philippine Sea
Manila
Philippines
Palawan
Negros
Mindanao
Basilan Island
Celebes Sea
Danipei
North Pacific Ocean
Honolulu
Lahaina
Hilo
HAWAII
BRITISH COLUMBIA
Edmonton
SASKATCHEWAN
Calgary
Vancouver
Seattle
NEVADA
UTAH
COLORADO
CALIFORNIA
Las Vegas
Los Angeles
ARIZONA
NEW MEXICO
San Diego
Gulf of California

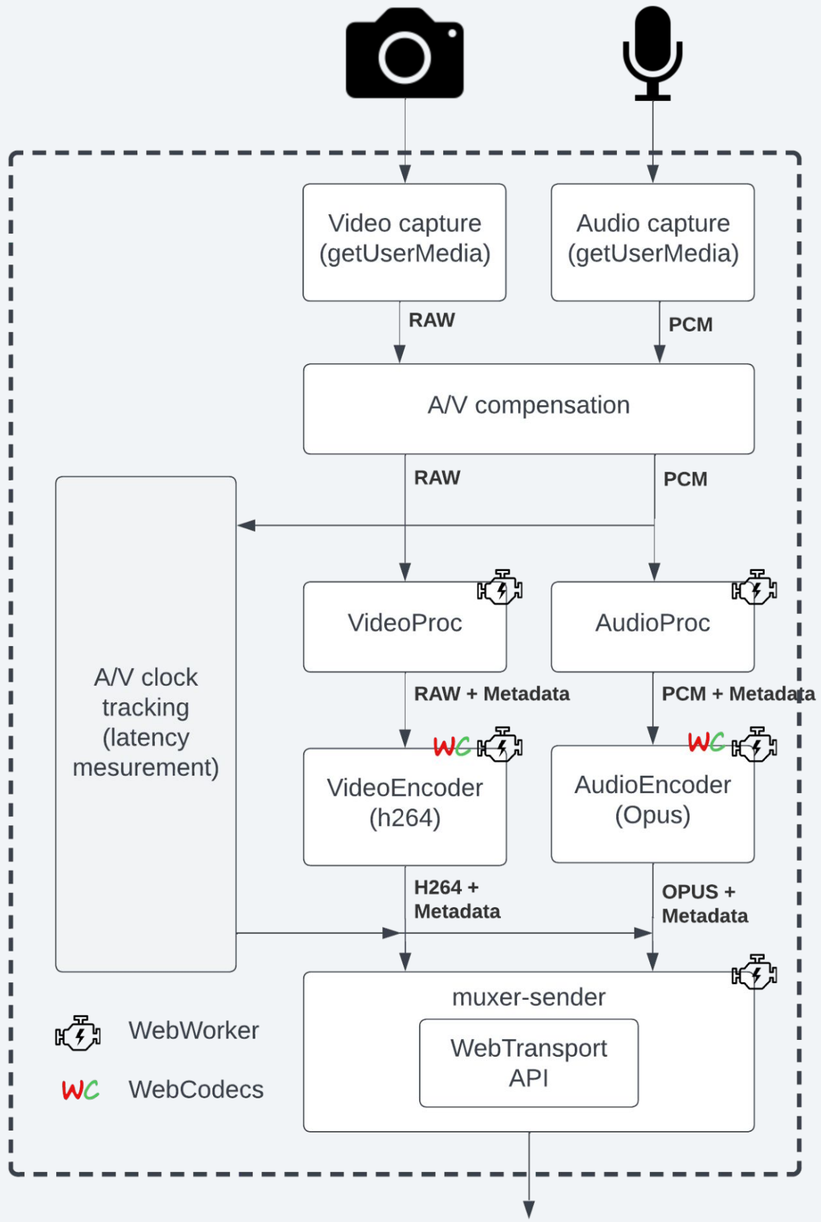
DEMO

DEMO Results

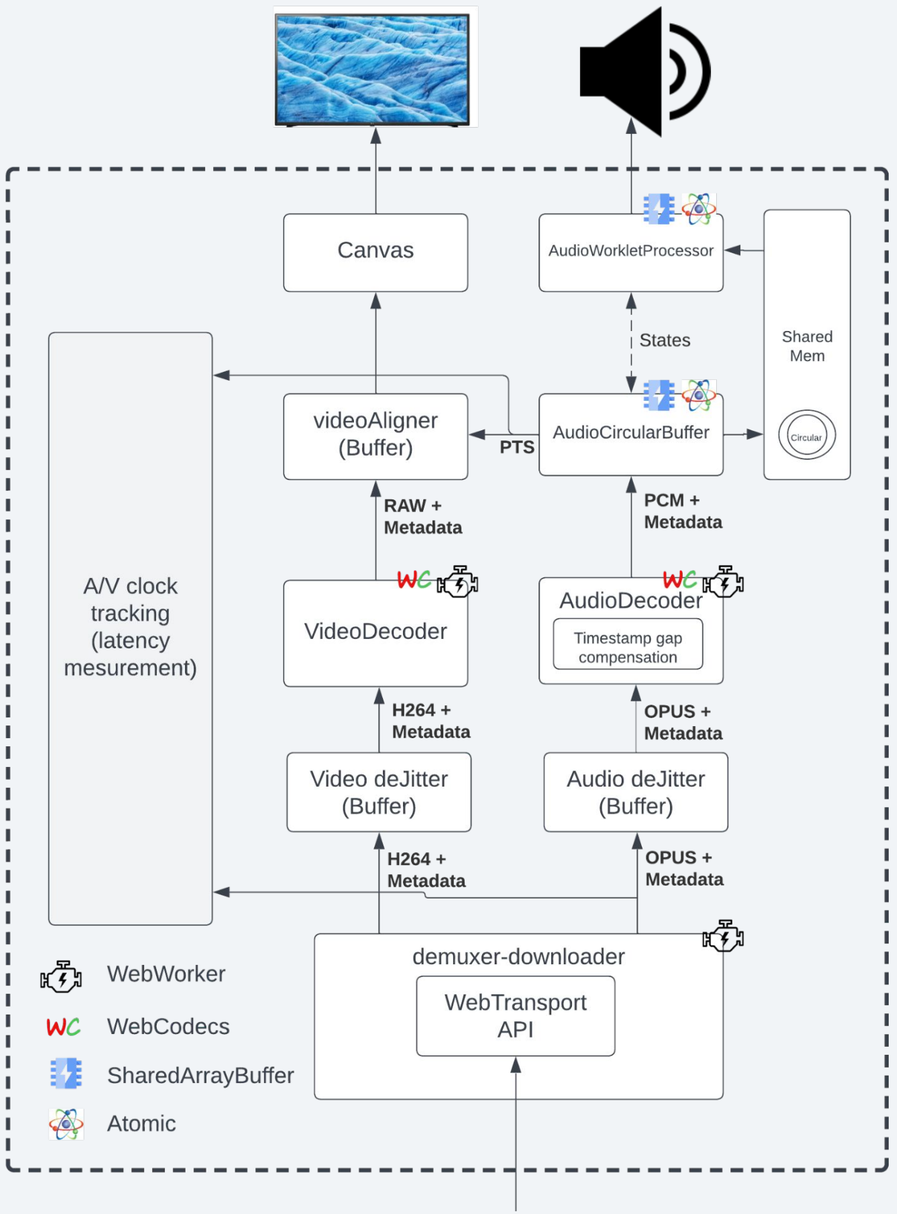
- H264 500Kbps + AAC 32Kbps
- Jitter buffer: 200ms

- **E2E latency: ~ 400ms**
- QOE: VC+ (IMHO)

Encoder

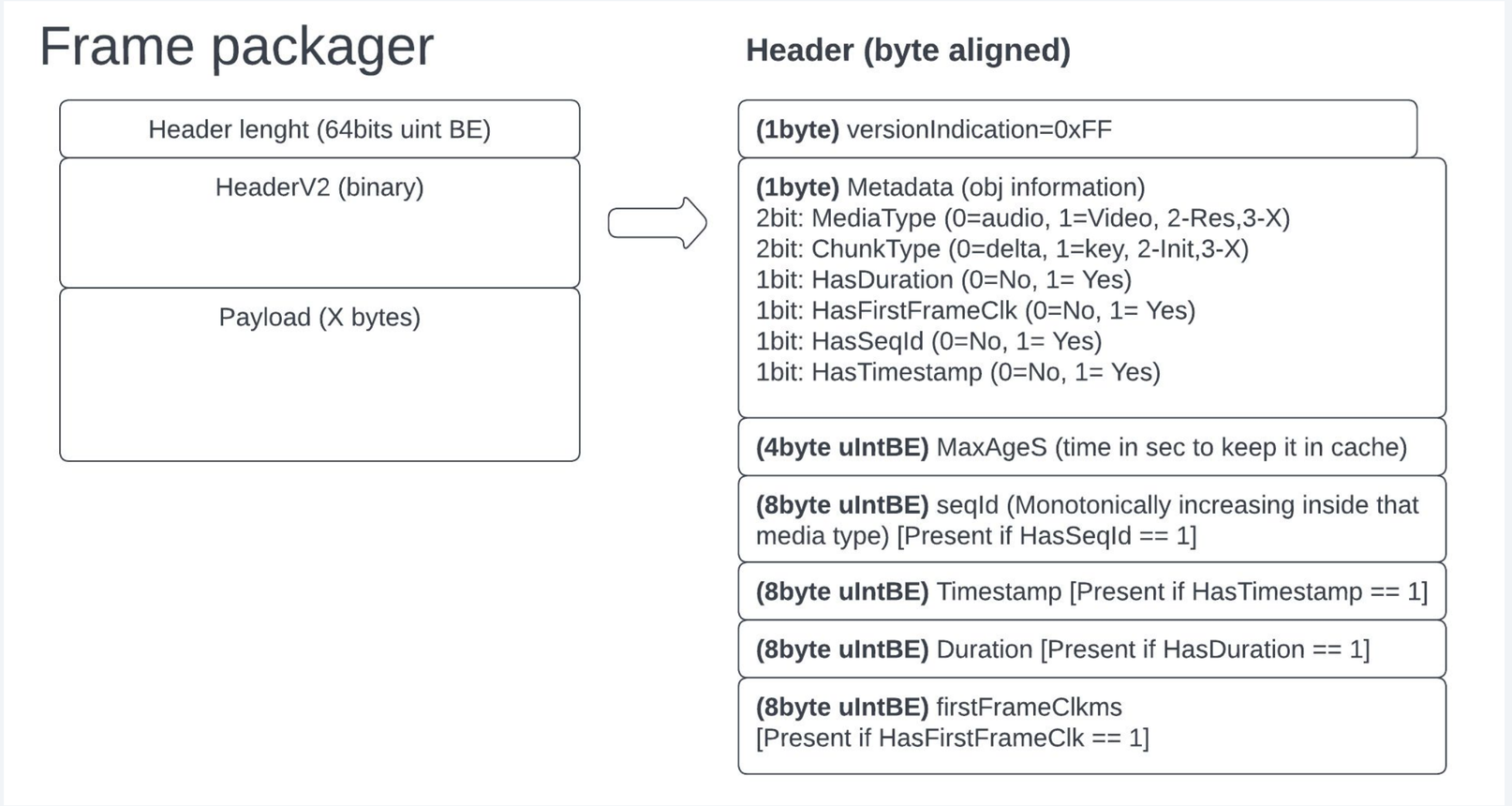

Video capture (getUserMedia)
Audio capture (getUserMedia)
RAW
PCM
A/V compensation
RAW
PCM
A/V clock tracking (latency mesurement)
VideoProc
AudioProc
RAW + Metadata
PCM + Metadata
VideoEncoder (h264)
AudioEncoder (Opus)
H264 + Metadata
OPUS + Metadata
muxer-sender
WebTransport API
WebWorker
WebCodecs

Player


Canvas
AudioWorkletProcessor
States
Shared Mem
Circular
videoAligner (Buffer)
PTS
AudioCircularBuffer
A/V clock tracking (latency mesurement)
RAW + Metadata
PCM + Metadata
VideoDecoder
AudioDecoder
Timestamp gap compensation
H264 + Metadata
OPUS + Metadata
Video deJitter (Buffer)
Audio deJitter (Buffer)
H264 + Metadata
OPUS + Metadata
demuxer-downloader
WebTransport API
WebWorker
WebCodecs
SharedArrayBuffer
Atomic

Media Packager

VERY EXPERIMENTAL!!!!



Efficiency: 92%+

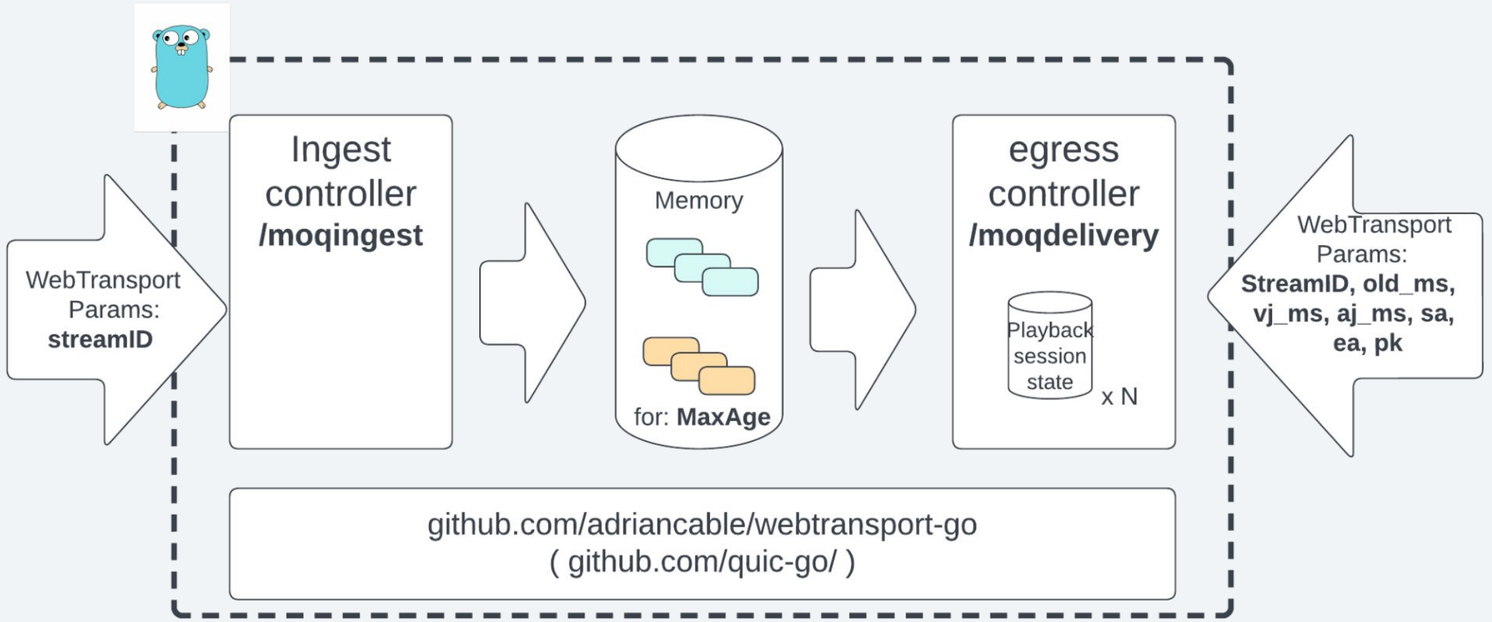
- Efficiency definition:

  Total payload bytes vs packager bytes sent to transport

- Test conditions:

  Video at 1Mbps, and Audio at 32Kbps

Relay



From WT session

From Obj header

Cache key: **streamID/media-type/seqId**

Examples:
- mystreamabc/video/123
- mystreamabc/video/-1 (init)

References

- Player & encoder:
https://github.com/facebookexperimental/webcodecs-capture-play

- Relay
https://github.com/facebookexperimental/go-media-webtransport-server

Q&A