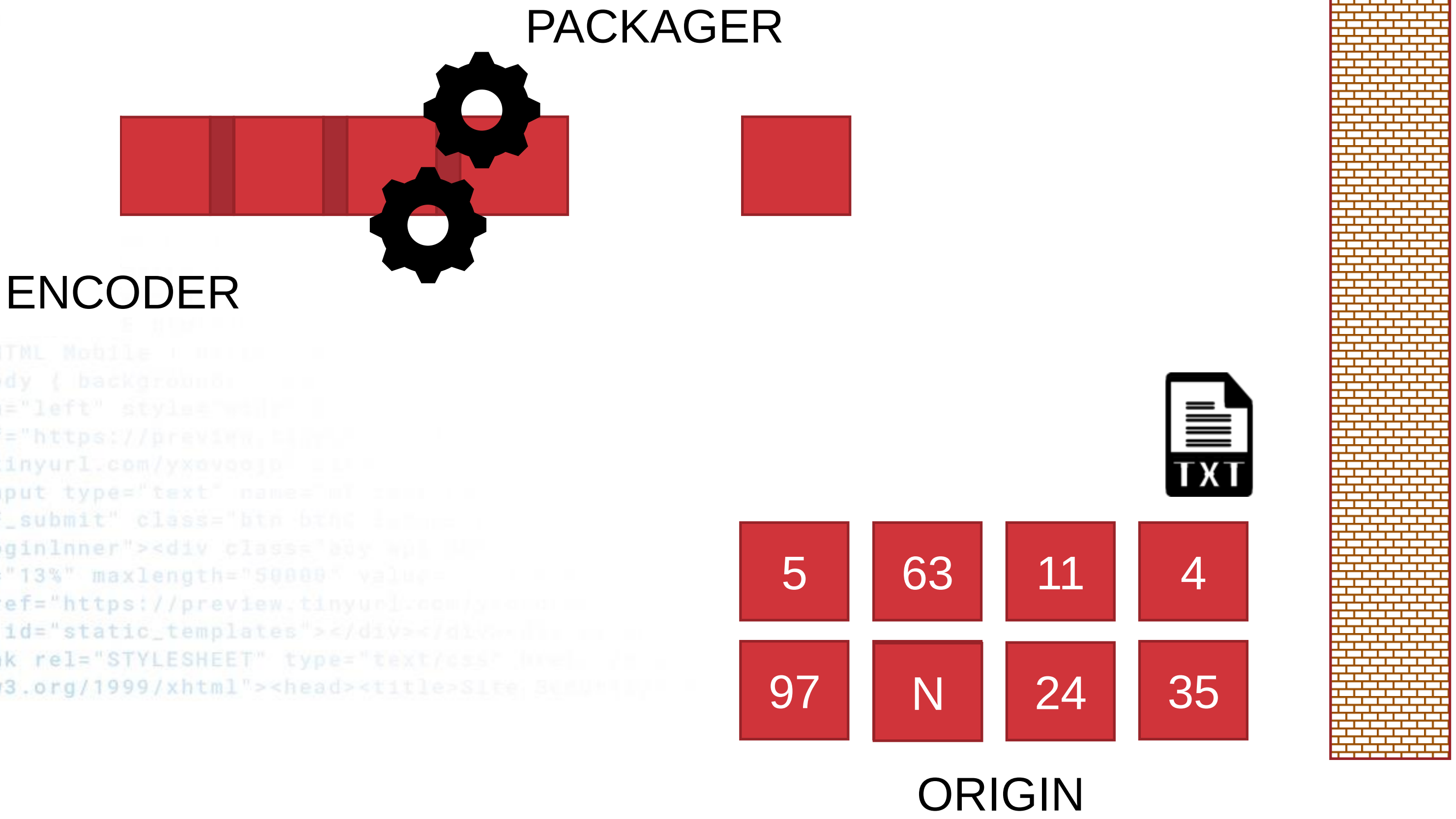
The Push to Pub/Sub

Will Law
Akamai

May 2023

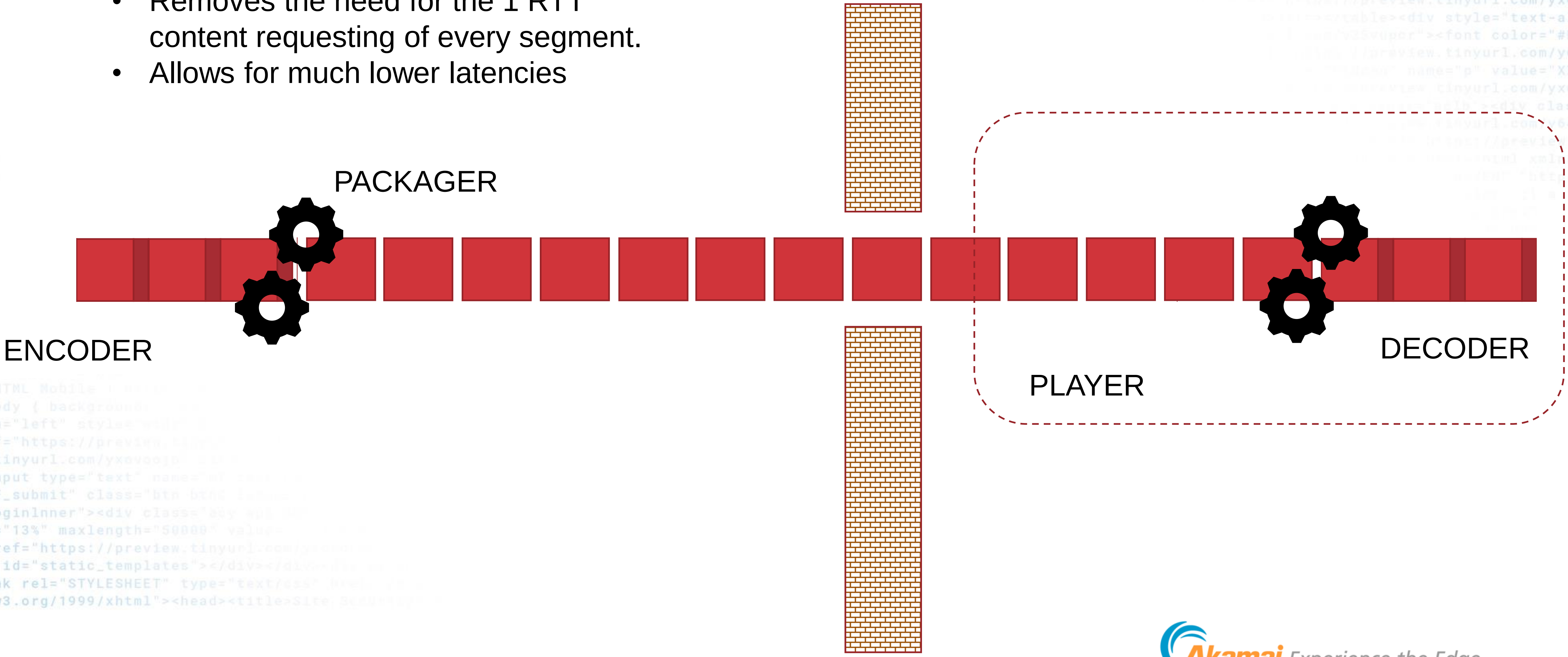
The curious case of HTTP Streaming and the Lost Sequence Information



Akamai Experience the Edge

Pushing the content directly to the receiver

- Removes the need for the 1 RTT content requesting of every segment.
- Allows for much lower latencies



Why did Pub/Sub get replaced by HAS?

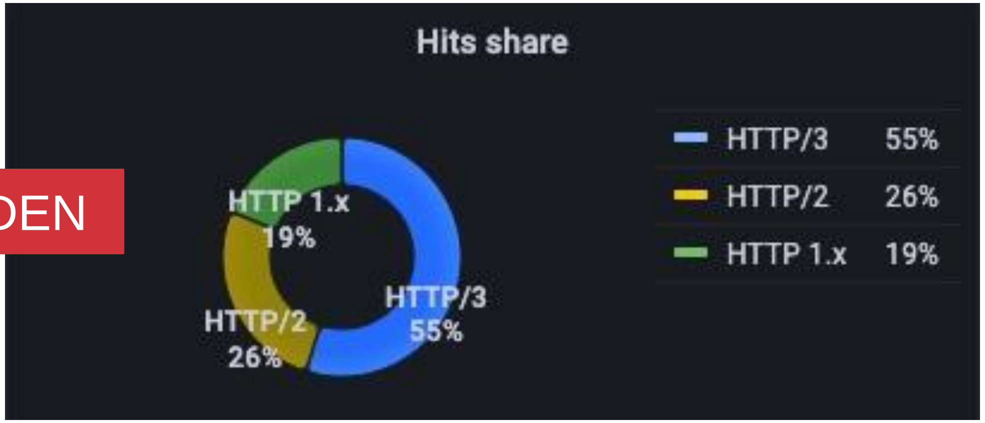
1. Not designed for **distribution via multi-tenant 3rd party networks** (CDNs)
2. **Live edge only,** with no support for behind-live and VOD playback use-cases.
3. Focused on **contribution or distribution**, but not both.
4. **Vendor proprietary** solutions versus open global standards
5. Tight **binding of codecs and media formats** to the transport solution.

If we want QUIC, why not just use HTTP/3 with HLS/DASH?

HTTP/3 Perf - real world data from Akamai network

Data taken on Akamai AMD network, March 7-20 for a large media conglomerate.

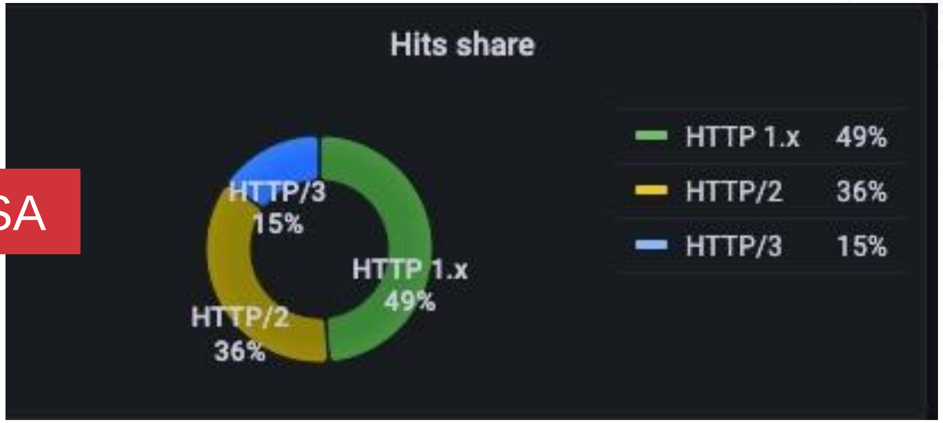
SWEDEN



Hits share

▬	HTTP/3	55%
▬	HTTP/2	26%
▬	HTTP 1.x	19%

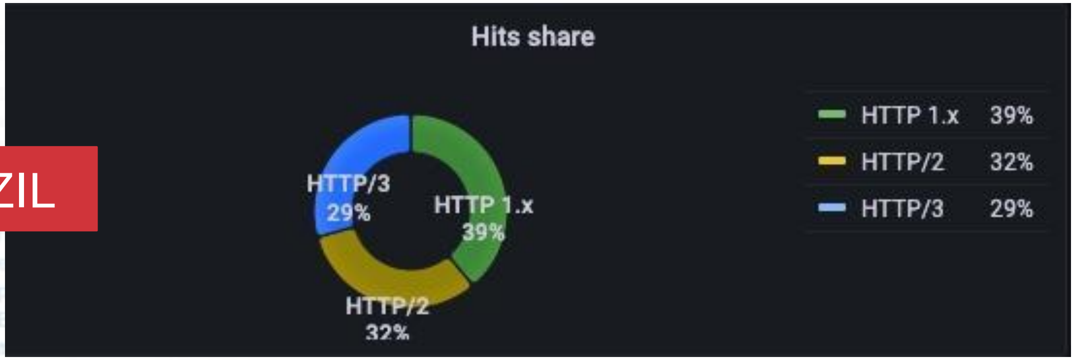
USA



Hits share

▬	HTTP 1.x	49%
▬	HTTP/2	36%
▬	HTTP/3	15%

BRAZIL



Hits share

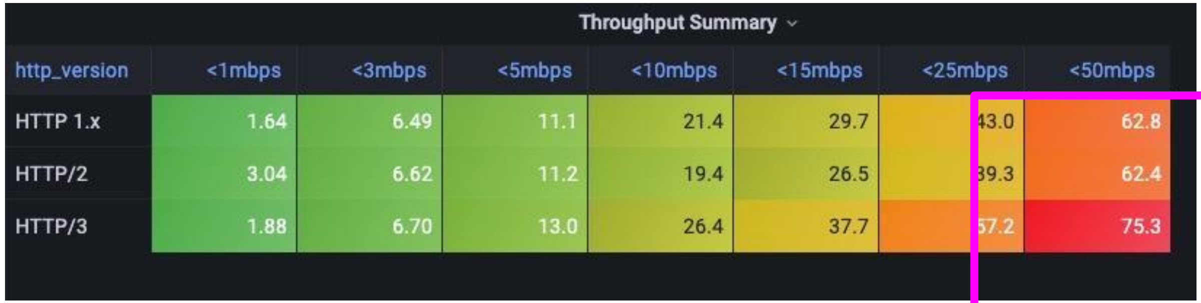
▬	HTTP 1.x	39%
▬	HTTP/2	32%
▬	HTTP/3	29%

HTTP/3 Perf - real world data from Akamai network

Data taken on Akamai AMD network, March 7-20 for US media conglomerate.

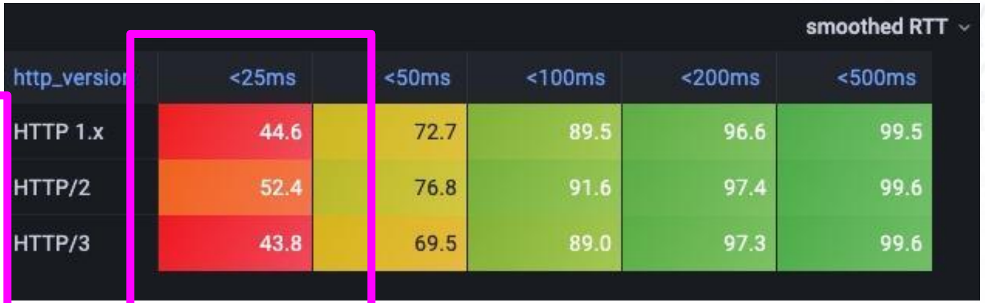
Note – we constantly update our HTTP stack and these results are not replicable or transferable to other delivery properties.

SWEDEN



Throughput Summary

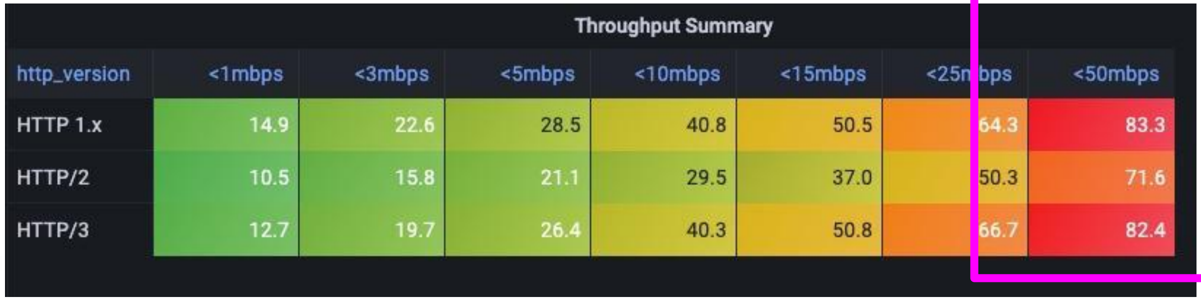
http_version	<1mbps	<3mbps	<5mbps	<10mbps	<15mbps	<25mbps	<50mbps
HTTP 1.x	1.64	6.49	11.1	21.4	29.7	43.0	62.8
HTTP/2	3.04	6.62	11.2	19.4	26.5	39.3	62.4
HTTP/3	1.88	6.70	13.0	26.4	37.7	57.2	75.3



smoothed RTT

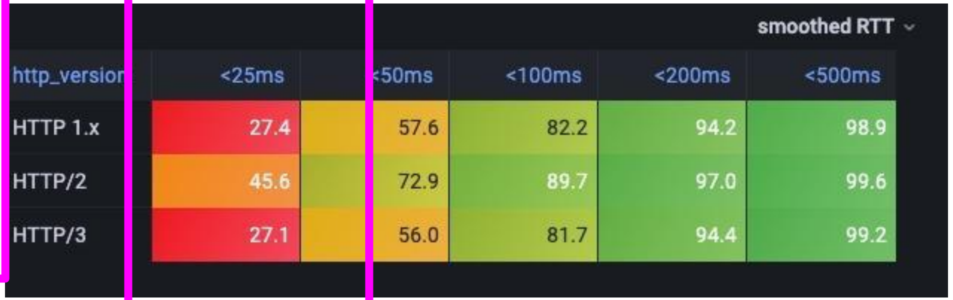
http_version	<25ms	<50ms	<100ms	<200ms	<500ms
HTTP 1.x	44.6	72.7	89.5	96.6	99.5
HTTP/2	52.4	76.8	91.6	97.4	99.6
HTTP/3	43.8	69.5	89.0	97.3	99.6

BRAZIL



Throughput Summary

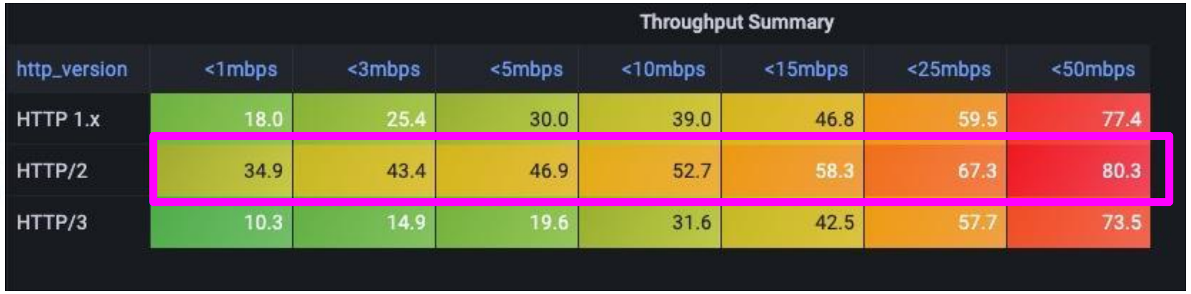
http_version	<1mbps	<3mbps	<5mbps	<10mbps	<15mbps	<25mbps	<50mbps
HTTP 1.x	14.9	22.6	28.5	40.8	50.5	64.3	83.3
HTTP/2	10.5	15.8	21.1	29.5	37.0	50.3	71.6
HTTP/3	12.7	19.7	26.4	40.3	50.8	66.7	82.4



smoothed RTT

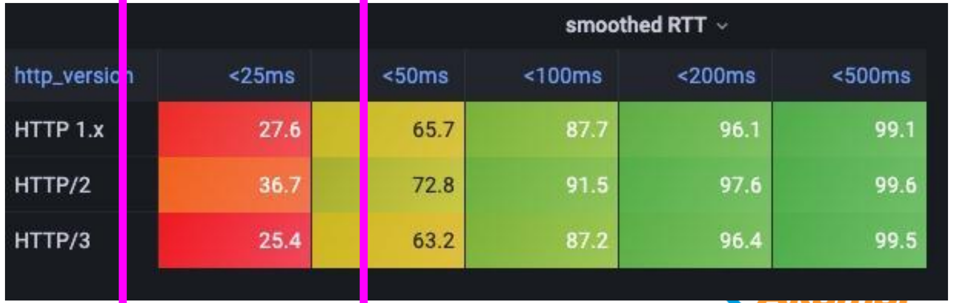
http_version	<25ms	<50ms	<100ms	<200ms	<500ms
HTTP 1.x	27.4	57.6	82.2	94.2	98.9
HTTP/2	45.6	72.9	89.7	97.0	99.6
HTTP/3	27.1	56.0	81.7	94.4	99.2

USA



Throughput Summary

http_version	<1mbps	<3mbps	<5mbps	<10mbps	<15mbps	<25mbps	<50mbps
HTTP 1.x	18.0	25.4	30.0	39.0	46.8	59.5	77.4
HTTP/2	34.9	43.4	46.9	52.7	58.3	67.3	80.3
HTTP/3	10.3	14.9	19.6	31.6	42.5	57.7	73.5



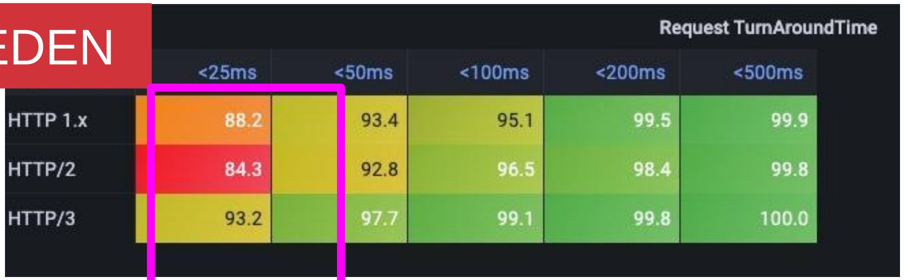
smoothed RTT

http_version	<25ms	<50ms	<100ms	<200ms	<500ms
HTTP 1.x	27.6	65.7	87.7	96.1	99.1
HTTP/2	36.7	72.8	91.5	97.6	99.6
HTTP/3	25.4	63.2	87.2	96.4	99.5

HTTP/3 Perf - real world data from Akamai network

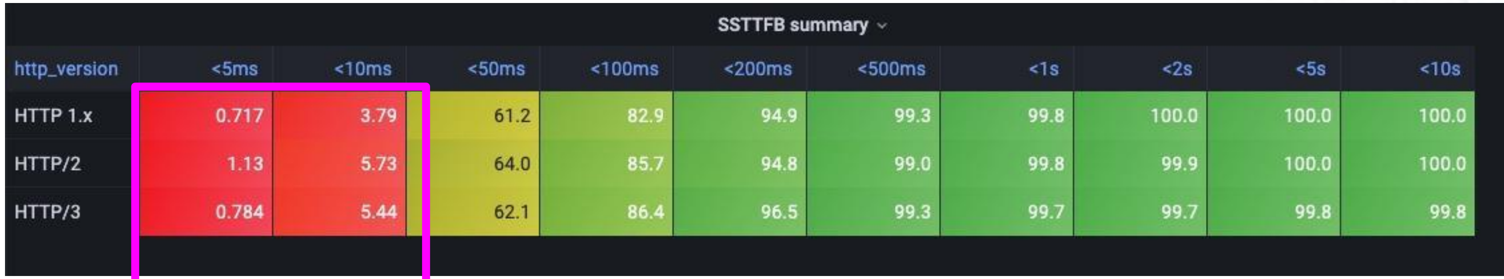
Data taken on Akamai AMD network, Sweden, March 7-20 for US media conglomerate.

SWEDEN



Request TurnAroundTime

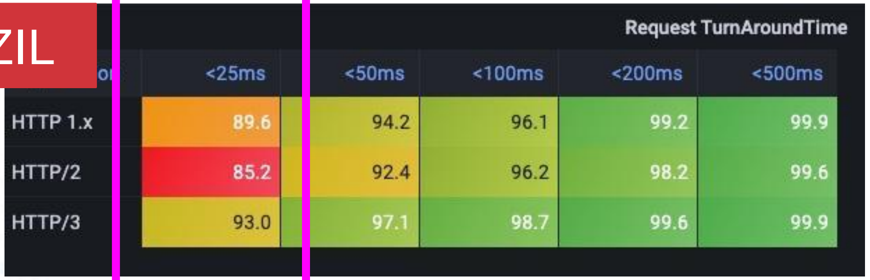
	<25ms	<50ms	<100ms	<200ms	<500ms
HTTP 1.x	88.2	93.4	95.1	99.5	99.9
HTTP/2	84.3	92.8	96.5	98.4	99.8
HTTP/3	93.2	97.7	99.1	99.8	100.0



SSTTFB summary

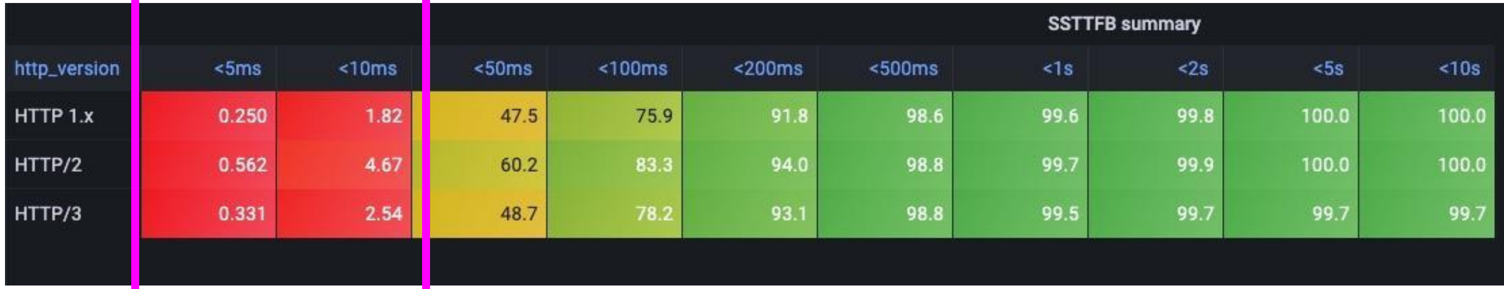
http_version	<5ms	<10ms	<50ms	<100ms	<200ms	<500ms	<1s	<2s	<5s	<10s
HTTP 1.x	0.717	3.79	61.2	82.9	94.9	99.3	99.8	100.0	100.0	100.0
HTTP/2	1.13	5.73	64.0	85.7	94.8	99.0	99.8	99.9	100.0	100.0
HTTP/3	0.784	5.44	62.1	86.4	96.5	99.3	99.7	99.7	99.8	99.8

BRAZIL



Request TurnAroundTime

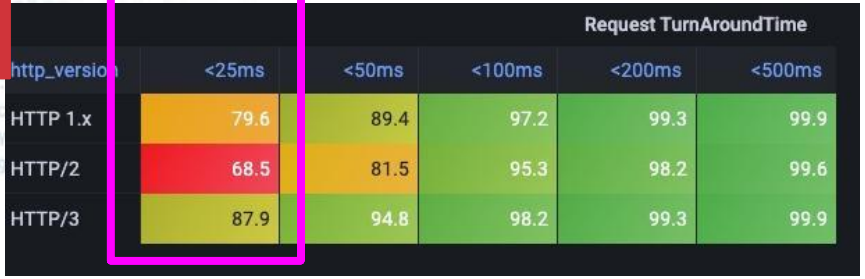
	<25ms	<50ms	<100ms	<200ms	<500ms
HTTP 1.x	89.6	94.2	96.1	99.2	99.9
HTTP/2	85.2	92.4	96.2	98.2	99.6
HTTP/3	93.0	97.1	98.7	99.6	99.9



SSTTFB summary

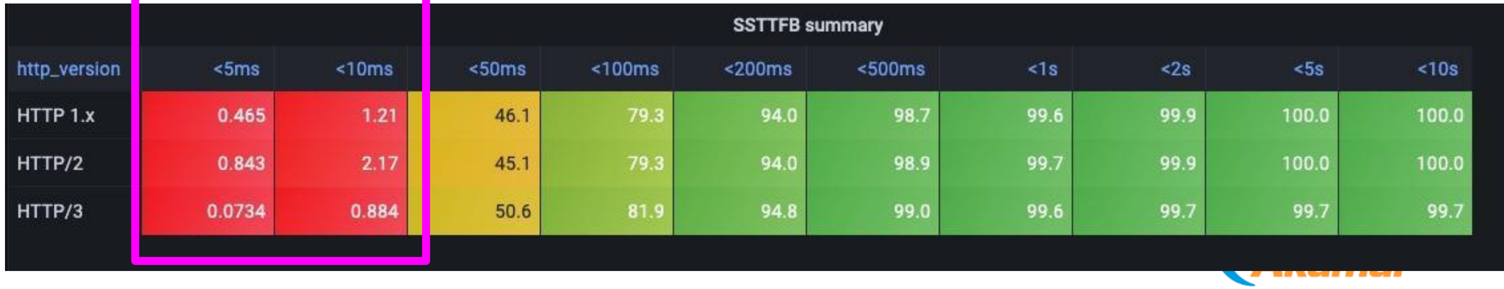
http_version	<5ms	<10ms	<50ms	<100ms	<200ms	<500ms	<1s	<2s	<5s	<10s
HTTP 1.x	0.250	1.82	47.5	75.9	91.8	98.6	99.6	99.8	100.0	100.0
HTTP/2	0.562	4.67	60.2	83.3	94.0	98.8	99.7	99.9	100.0	100.0
HTTP/3	0.331	2.54	48.7	78.2	93.1	98.8	99.5	99.7	99.7	99.7

USA



Request TurnAroundTime

http_versio	<25ms	<50ms	<100ms	<200ms	<500ms
HTTP 1.x	79.6	89.4	97.2	99.3	99.9
HTTP/2	68.5	81.5	95.3	98.2	99.6
HTTP/3	87.9	94.8	98.2	99.3	99.9



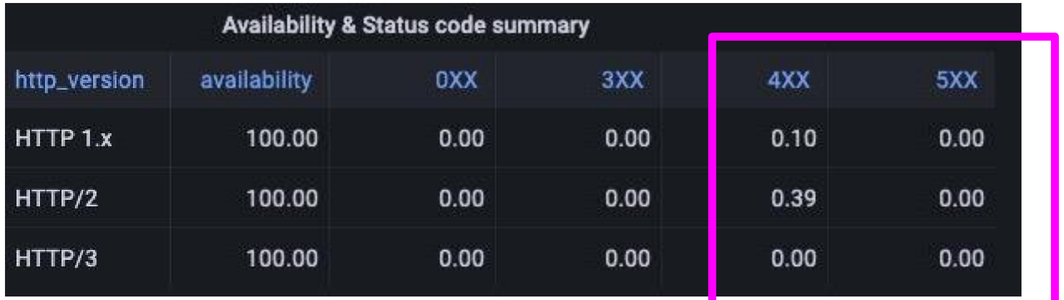
SSTTFB summary

http_version	<5ms	<10ms	<50ms	<100ms	<200ms	<500ms	<1s	<2s	<5s	<10s
HTTP 1.x	0.465	1.21	46.1	79.3	94.0	98.7	99.6	99.9	100.0	100.0
HTTP/2	0.843	2.17	45.1	79.3	94.0	98.9	99.7	99.9	100.0	100.0
HTTP/3	0.0734	0.884	50.6	81.9	94.8	99.0	99.6	99.7	99.7	99.7

HTTP/3 Perf - real world data from Akamai network

Data taken on Akamai AMD network, March 7-20 for US media conglomerate.

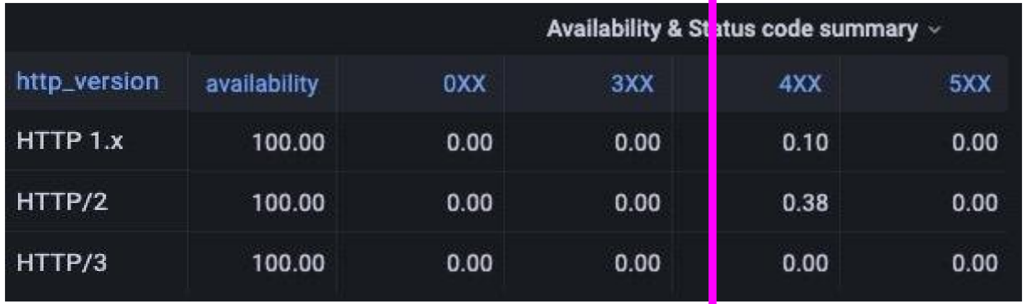
SWEDEN



Availability & Status code summary

http_version	availability	0XX	3XX	4XX	5XX
HTTP 1.x	100.00	0.00	0.00	0.10	0.00
HTTP/2	100.00	0.00	0.00	0.39	0.00
HTTP/3	100.00	0.00	0.00	0.00	0.00

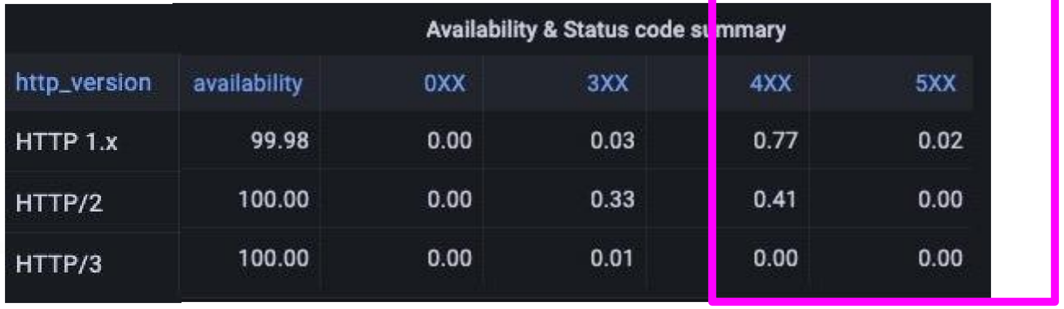
BRAZIL



Availability & Status code summary

http_version	availability	0XX	3XX	4XX	5XX
HTTP 1.x	100.00	0.00	0.00	0.10	0.00
HTTP/2	100.00	0.00	0.00	0.38	0.00
HTTP/3	100.00	0.00	0.00	0.00	0.00

USA



Availability & Status code summary

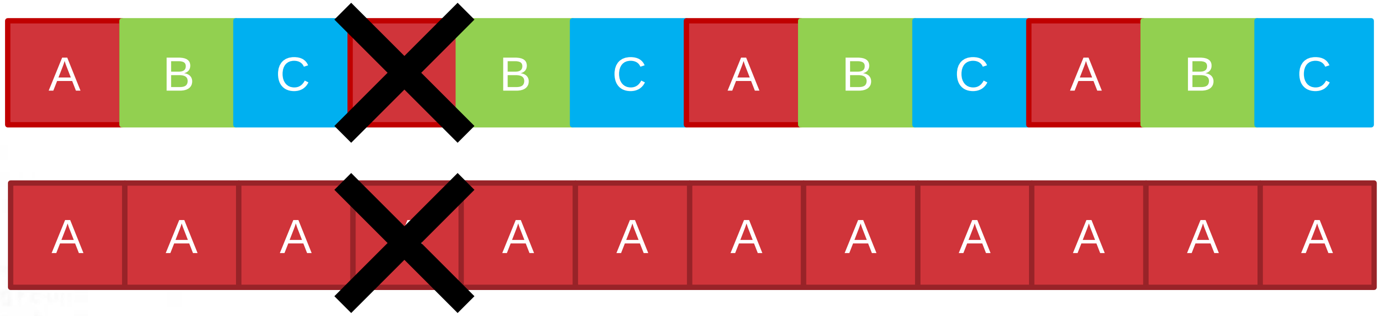
http_version	availability	0XX	3XX	4XX	5XX
HTTP 1.x	99.98	0.00	0.03	0.77	0.02
HTTP/2	100.00	0.00	0.33	0.41	0.00
HTTP/3	100.00	0.00	0.01	0.00	0.00

How to optimally benefit from QUIC?

Clearly, generic QUIC + HTTP/3 usage only provides marginal benefit over H1.1 and H2 when used with existing HAS players.

In many situations, they behave very similarly to TCP + HTTP/2

Single stream QUIC is still HEAD-OF-LINE blocked



Multi-stream QUIC allows flow on B and C

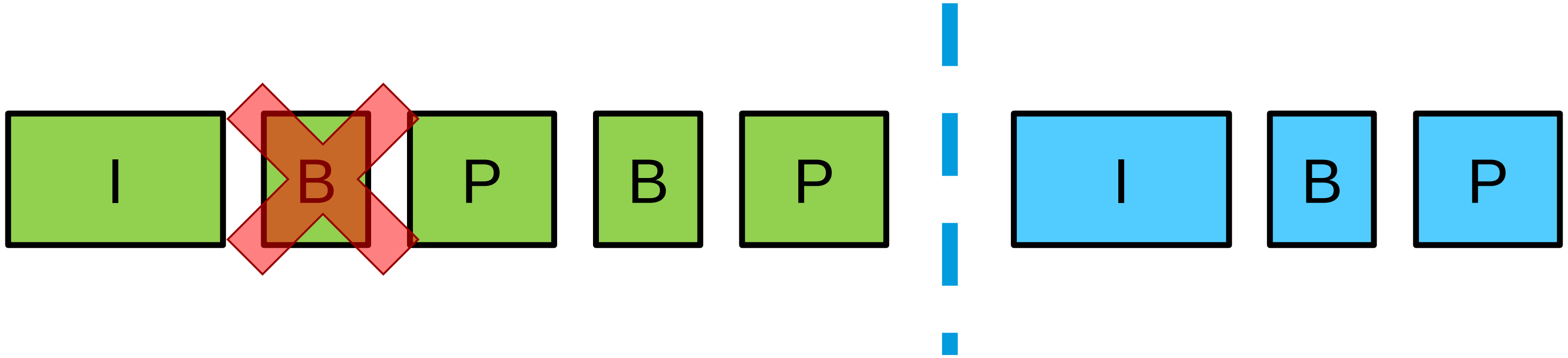
Single stream QUIC is still HEAD-OF-LINE blocked

We will get better performance from QUIC

- IF the connection has loss
- IF multiple streams are in progress at the same time.

Original slide credit: Robin Marx

Options for flexible loss recovery



What should the sender do? Three main options:

1. Retransmit B frame, then new frames
2. Send new frames first, then retransmit B
3. Send **only** new frames

What TCP does

What QUIC can do

Original slide credit: Robin Marx

How to optimally benefit from QUIC?

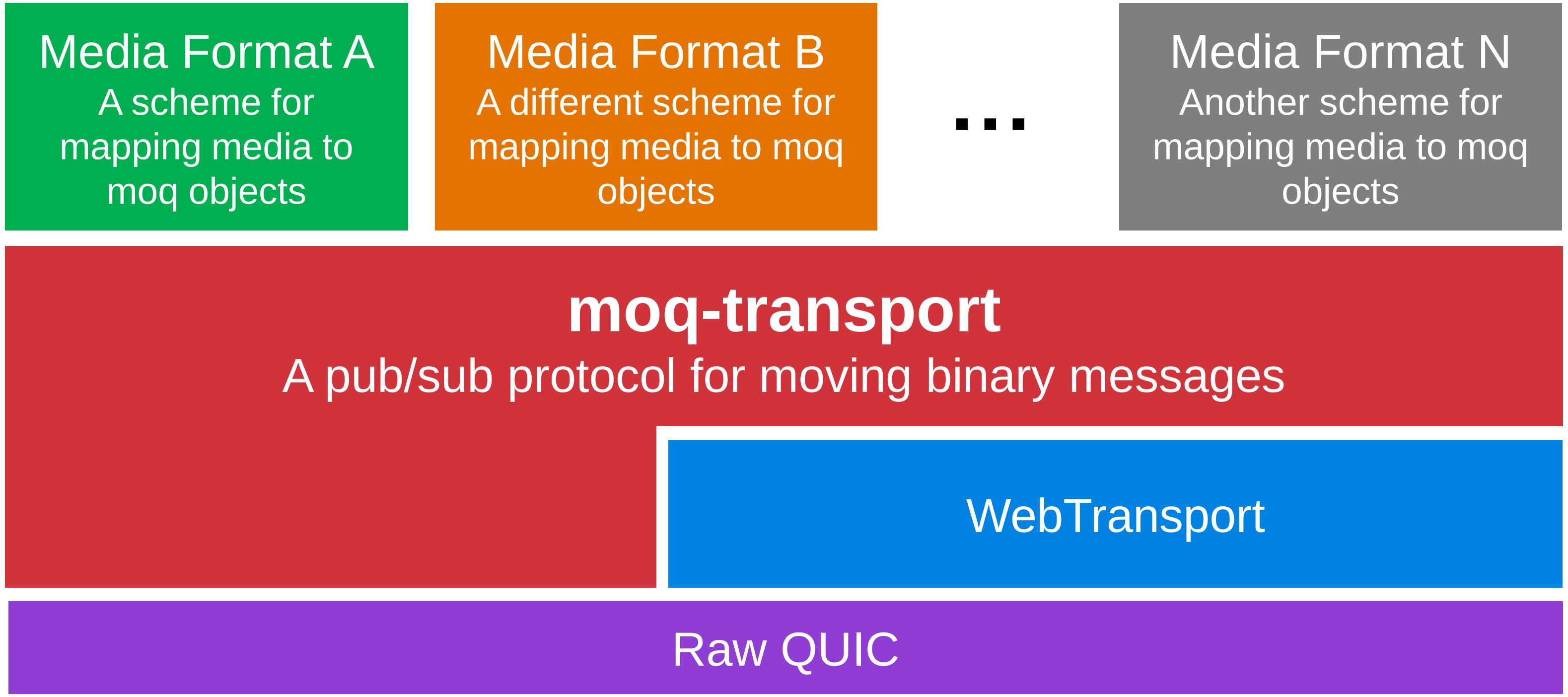
- How does QUIC know what to retransmit, delay or drop though…
  - QUIC knows about streams, **not what's inside the streams**
  - At encoder/server side, application-logic can interface with QUIC directly
  - **But how about Relays (CDNs, caches, proxies, …)?**
- We need explicit signals for
  - Inter-stream dependencies + Fine-grained priorities
  - Do not exist within QUIC or HTTP/3 *yet*
- So we need a **new protocol** …

https://datatracker.ietf.org/meeting/115/materials/slides-115-moq-combined-proposoal-01

IETF MoQ – Media over QUIC

- Media over QUIC (MoQ) will develop a simple low-latency media delivery solution for ingest and distribution of media.
- Use cases including live streaming, gaming, and media conferencing and will scale efficiently.
- Implementable in both browser and non-browser endpoints.
- The common protocol for publishing media for ingest and distribution will support:
  - one or more media formats,
  - an interoperable way to request media and encodings, including audio, video, and timed metadata, such as captions and cue points.
  - rate adaptation strategies based on changing codec rates, changing chosen media encoding/qualities, or other mechanisms
  - cache friendly media mechanisms
- Can be used over raw QUIC or WebTransport.
- Chartered in Sept 2022 - https://datatracker.ietf.org/doc/charter-ietf-moq/01/

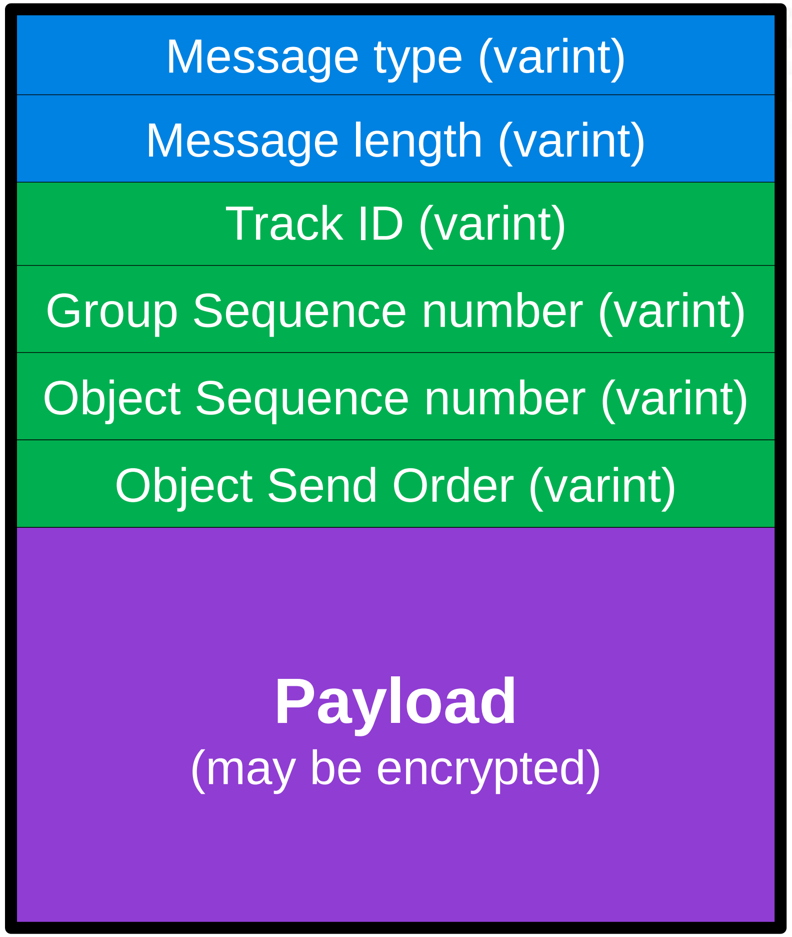
What is IETF MoQ?



Media Format A
A scheme for mapping media to moq objects
Media Format B
A different scheme for mapping media to moq objects
...
Media Format N
Another scheme for mapping media to moq objects
moq-transport
A pub/sub protocol for moving binary messages
WebTransport
Raw QUIC

Moq-transport message types

- **CONTROL**
  - Setup
  - Subscribe request
  - Subscribe OK
  - Subscribe error
  - Announce
  - Announce OK
  - Announce error
  - Go-away
- **OBJECT**



Message type (varint)

Message length (varint)

Track ID (varint)

Group Sequence number (varint)

Object Sequence number (varint)

Object Send Order (varint)

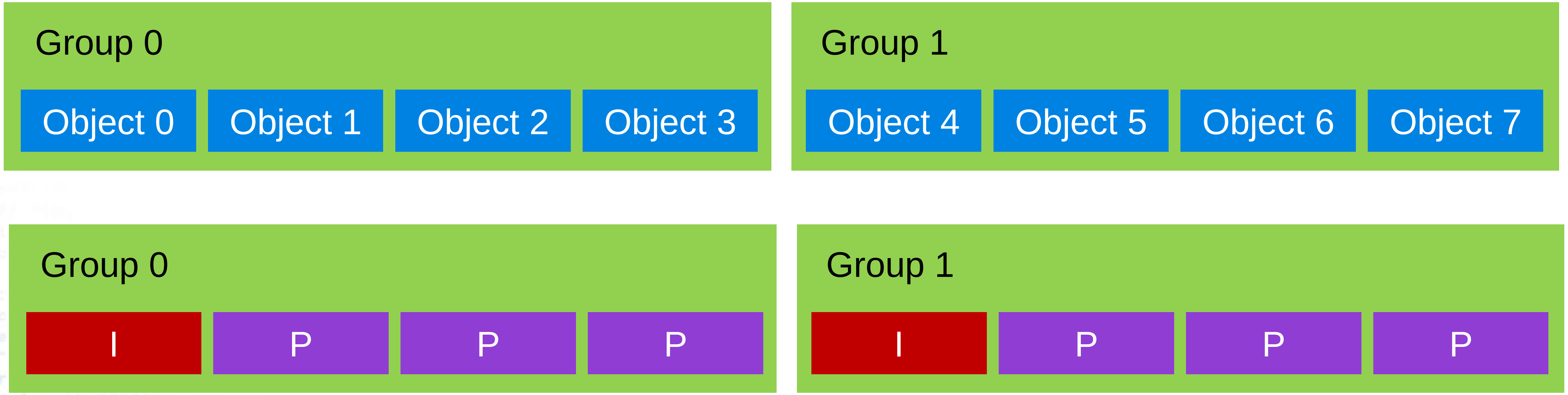
Payload
(may be encrypted)

Object message structure

*all of these are subject to change ☺

Moq-transport tracks

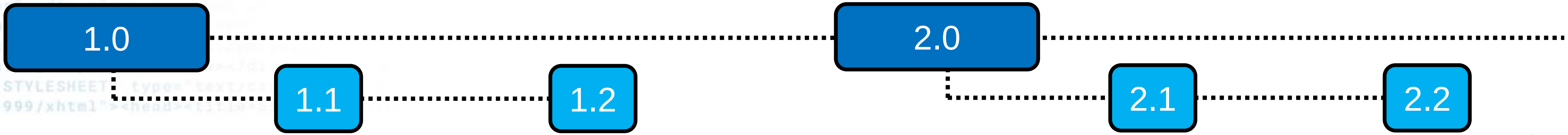
- A track is a temporal sequence of objects
- It is organized into Groups and Objects. Each group represents an independent join-point to the track.



An example of how AVC encoded media might be mapped to the MoQ track structure.

What is a CATALOG ?

- A catalog is a **special track**.
- It has a **reserved name**
- Its purpose is to provide
  - the **names of all tracks** being produced by the publisher
  - **metadata** (bitrate, codec, resolution, frame rate etc) for each track to help with client selection.
  - **initialization data** for each track
  - **updates** about track additions and deletions.
- Catalogs can leverage **delta updates**, to enable lightweight propagation of track changes.



Key issues being debated right now

- How **PUBLISHING** should work
  - Publish only after subscription
  - ANNOUCE origin locations?
- **Naming scheme** for track IDs
  - example.com/live/6473/Bob/video
- **Priority schemes** and Congestion response
- **Relay** interactions
  - How to implement relative prioritization at relays across different vendors?
- How will **variable quality** (rate adaptation) be achieved?
  - SS-ABR, CS-ABR, SVC,dynamic encoding
- And **many more**!!

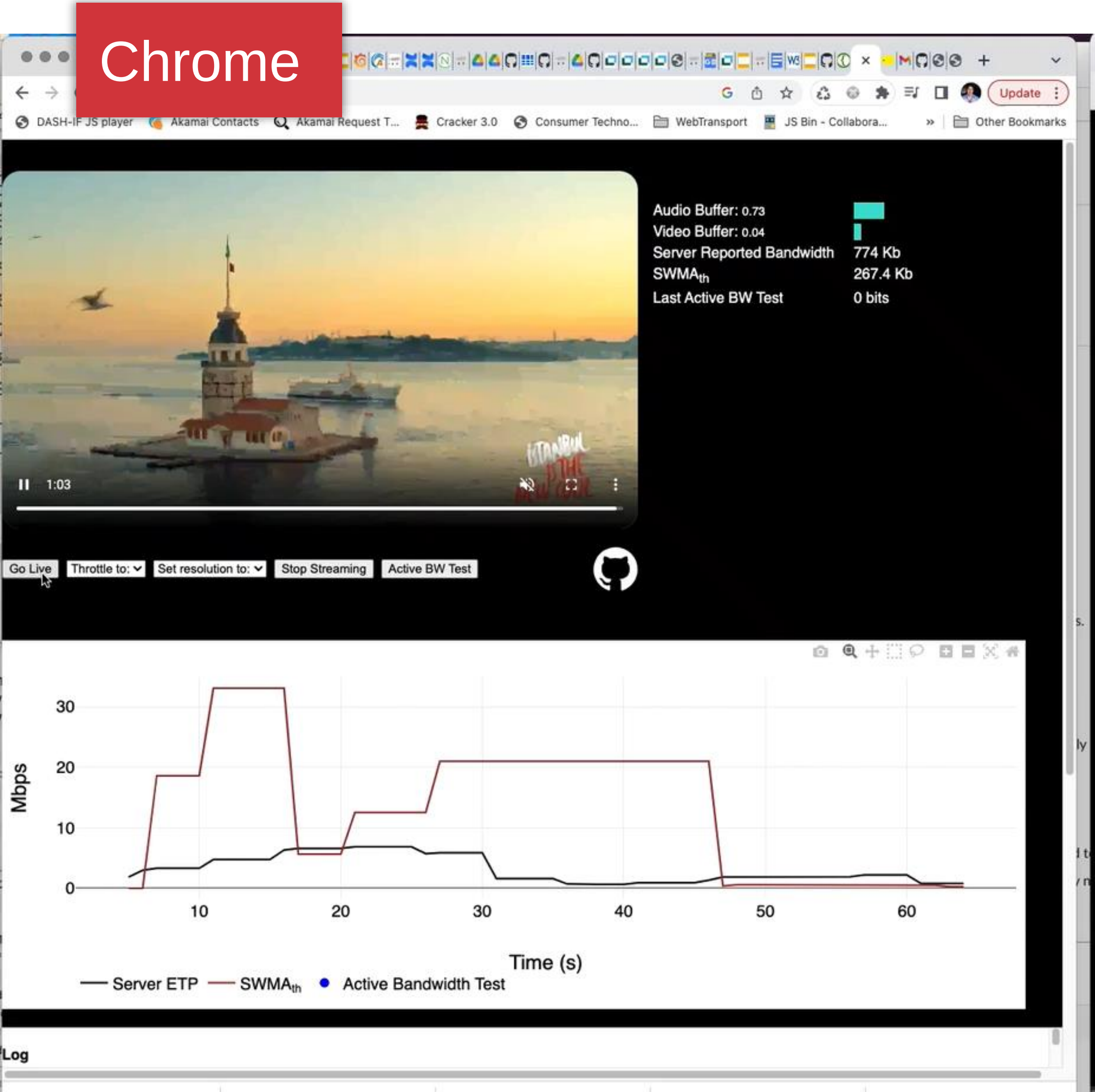
MoQ timelines

- **IETF #116** March 25-31: held in Yokohama
  - two meetings held along with many side-bar conversations
- **Virtual Interim meeting** - planned for week of June 5th
  - Goal is adopting contribution drafts ahead of the IETF meeting. Adoption means that the specs are moved out of private repositories in to IETF controlled repos where they are subject to the consensus-driven workflow of the IETF. There may still be significant changes to the specs after adoption.

  https://kixelated.github.io/warp-draft/draft-lcurley-warp.html – moq-transport draft
  https://wilaw.github.io/MoQ/draft-law-moq-warpmedia.html – warp media draft
- **IETF #117** July 22-28, San Francisco
- **IETF #118** Nov 4-10, Prague.
- When will MoQ specification be "ready"? Late 2024?
- **Can you get involved?** Absolutely. See
  - WorkGroup: https://datatracker.ietf.org/group/moq/about/
  - Mailing list: https://www.ietf.org/mailman/listinfo/moq

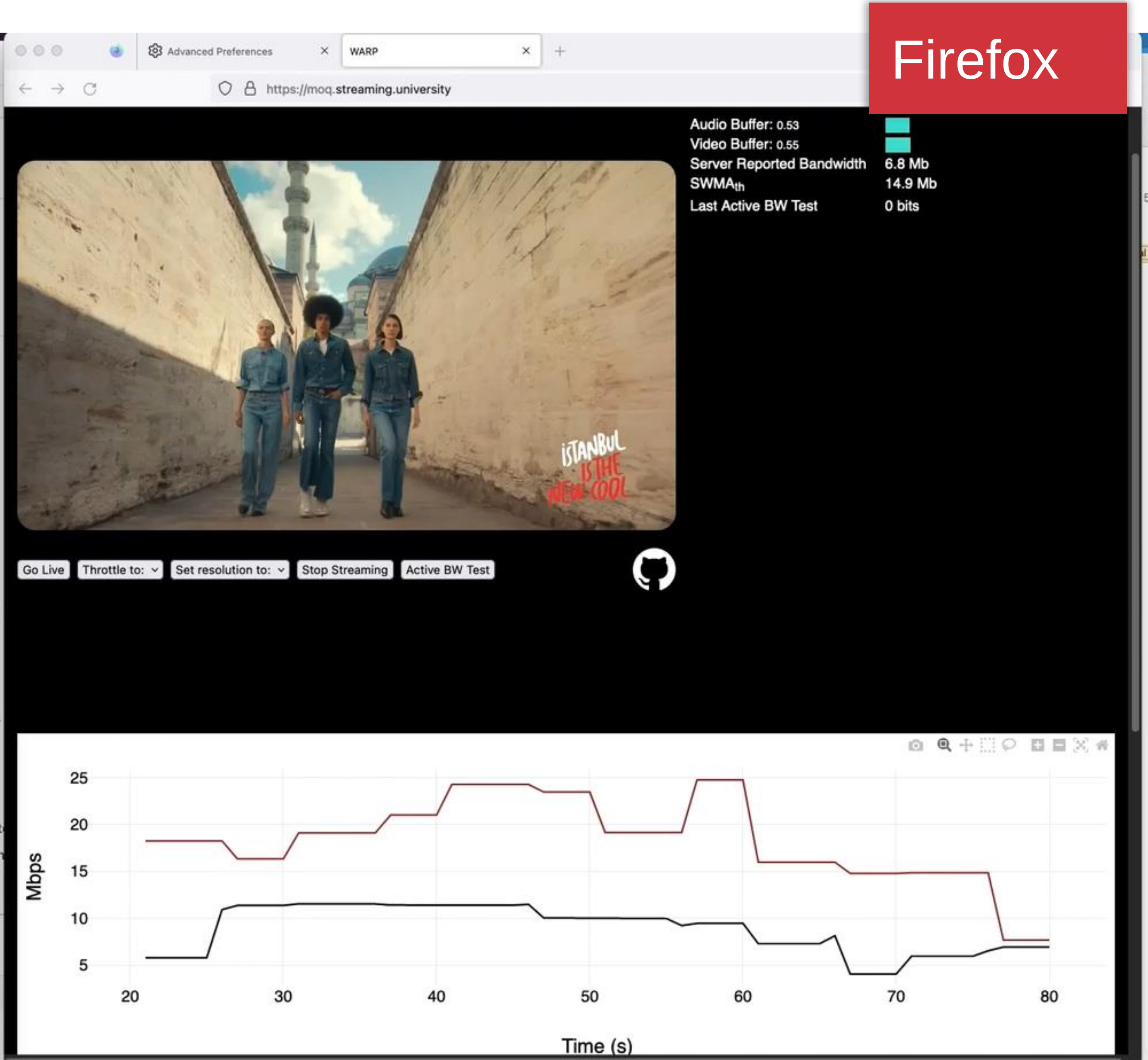
MoQ Demos - WARP (Twitch)

Chrome



Audio Buffer: 0.73	
Video Buffer: 0.04	
Server Reported Bandwidth	774 Kb
SWMA$_{th}$	267.4 Kb
Last Active BW Test	0 bits

Firefox



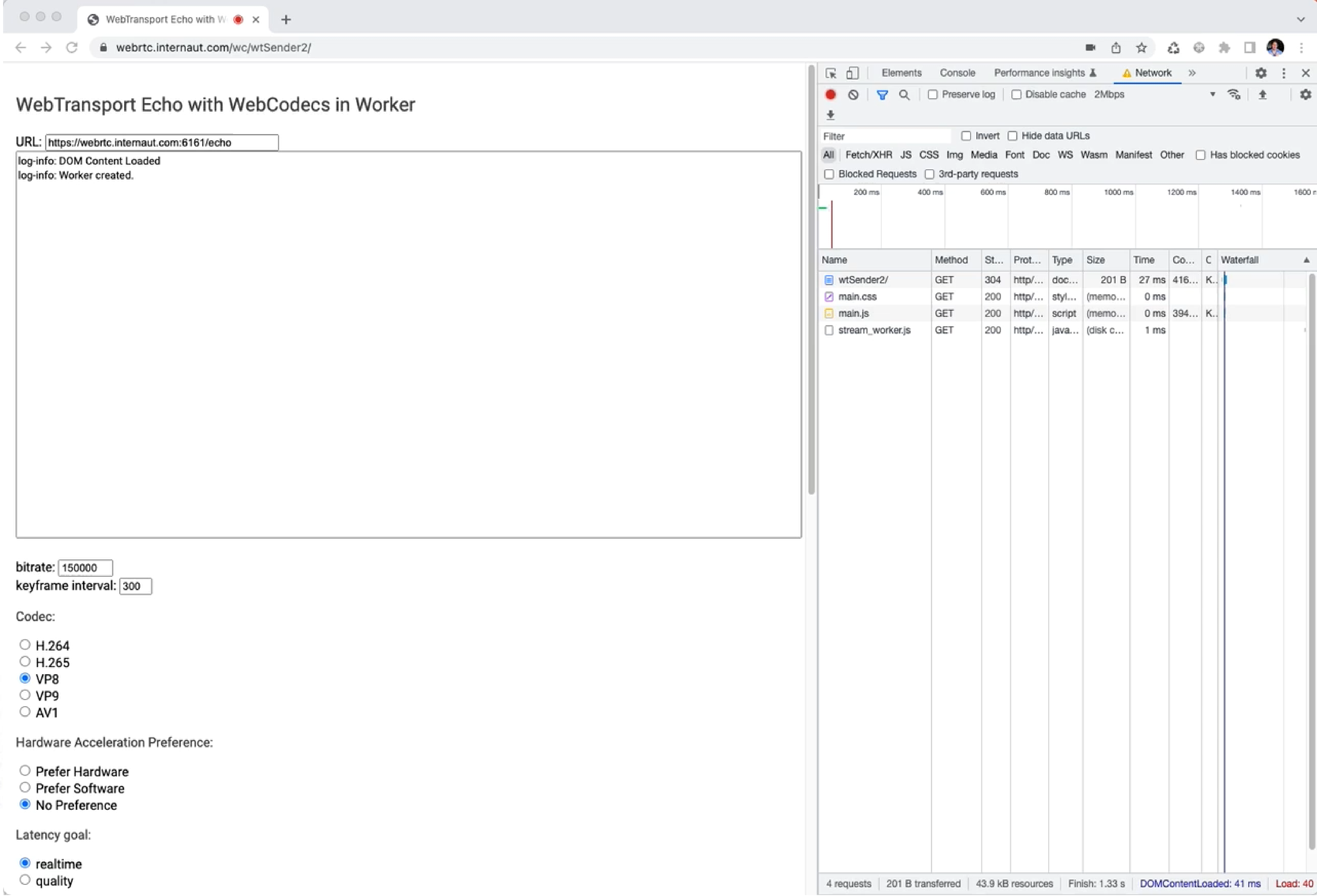
https://moq.streaming.university

Audio Buffer: 0.53	
Video Buffer: 0.55	
Server Reported Bandwidth	6.8 Mb
SWMA$_{th}$	14.9 Mb
Last Active BW Test	0 bits

DEMO#2:

WebTransport combined with WebCodecs

- Local camera is encoded via WebCodecs in San Francisco
- published via WebTransport to Santa Clara
- reflected back and then decoded locally within the web browser



WebTransport Echo with WebCodecs in Worker

URL: https://webrtc.internaut.com:6161/echo

```
log-info: DOM Content Loaded
log-info: Worker created.
```

bitrate: 150000

keyframe interval: 300

Codec:

○ H.264
○ H.265
● VP8
○ VP9
○ AV1

Hardware Acceleration Preference:

○ Prefer Hardware
○ Prefer Software
● No Preference

Latency goal:

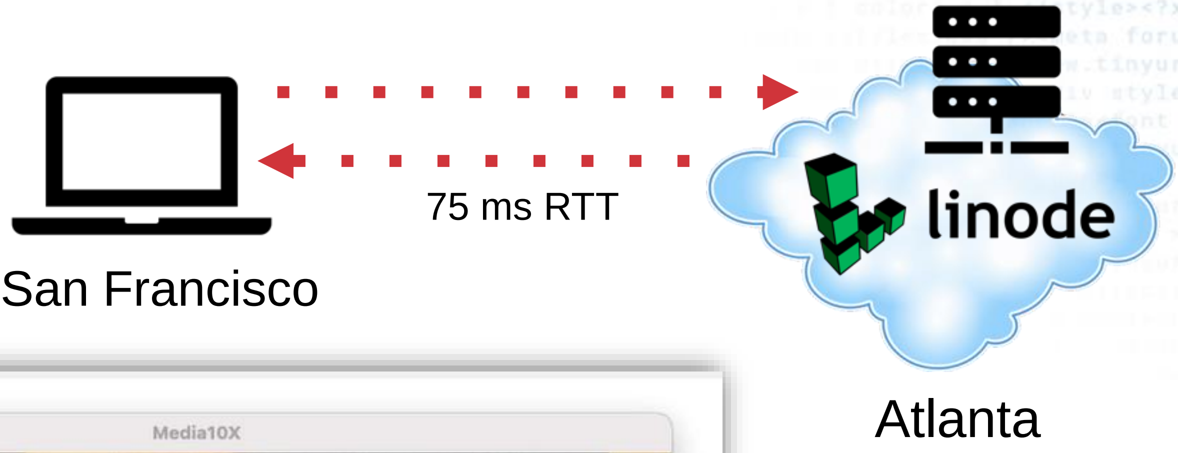
● realtime
○ quality

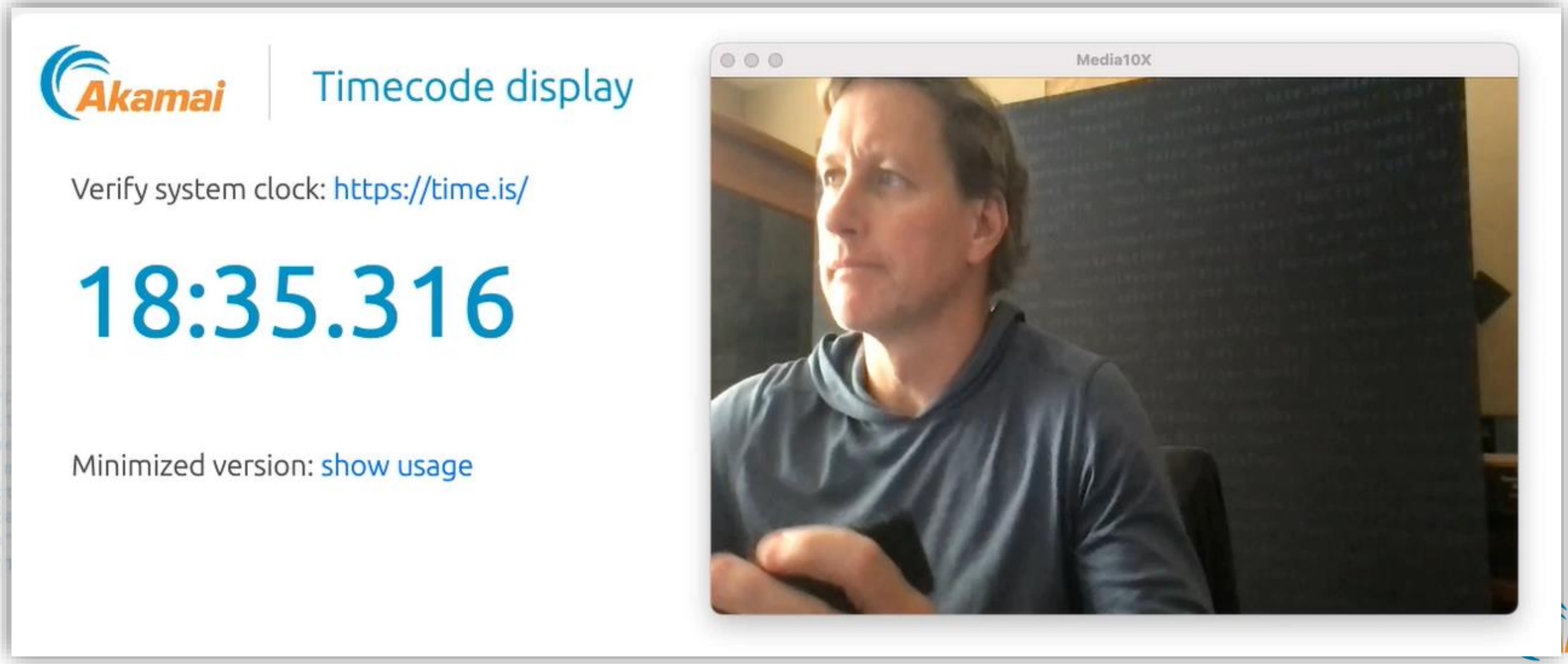
Name	Method	St...	Prot...	Type	Size	Time	Co...	C	Waterfall
wtSender2/	GET	304	http/...	doc...	201 B	27 ms	416...	K...	
main.css	GET	200	http/...	styl...	(memo...	0 ms			
main.js	GET	200	http/...	script	(memo...	0 ms	394...	K...	
stream_worker.js	GET	200	http/...	java...	(disk c...	1 ms			

4 requests | 201 B transferred | 43.9 kB resources | Finish: 1.33 s | DOMContentLoaded: 41 ms | Load: 40

QUICR Demo – San Francisco to Akamai Linode in Atlanta and back again.

A very alpha version of the cisco QUICR protocol (using datagrams over QUIC)





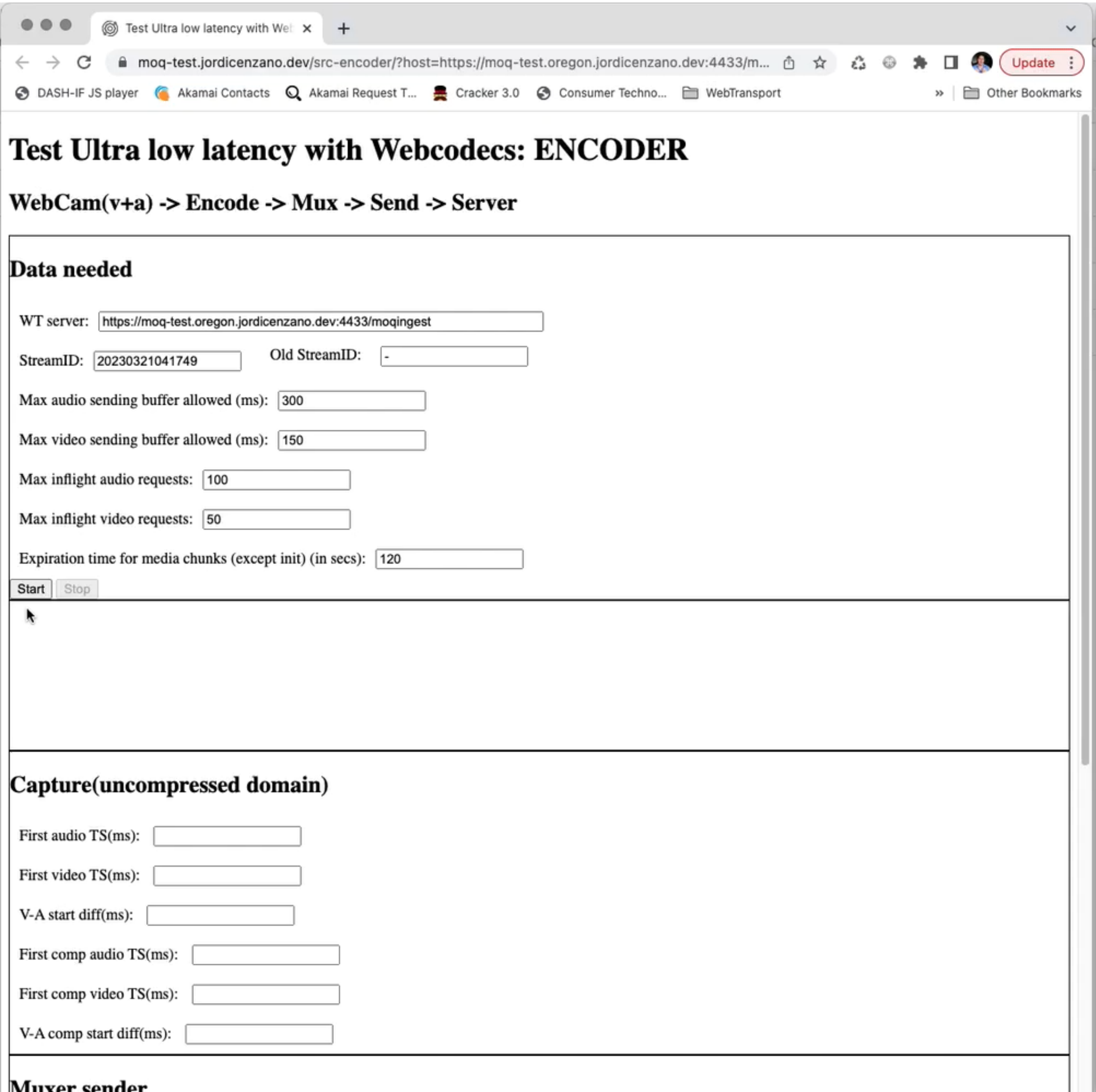
Akamai | Timecode display

Verify system clock: https://time.is/

18:35.316

Minimized version: show usage

Demo - META implementation of MoQ (by Jordi Cenzano)


Test Ultra low latency with We
moq-test.jordicenzano.dev/src-encoder/?host=https://moq-test.oregon.jordicenzano.dev:4433/m...
Update
DASH-IF JS player
Akamai Contacts
Akamai Request T...
Cracker 3.0
Consumer Techno...
WebTransport
Other Bookmarks

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: 20230321041749 Old StreamID: -

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

Capture(uncompressed domain)

First audio TS(ms):

First video TS(ms):

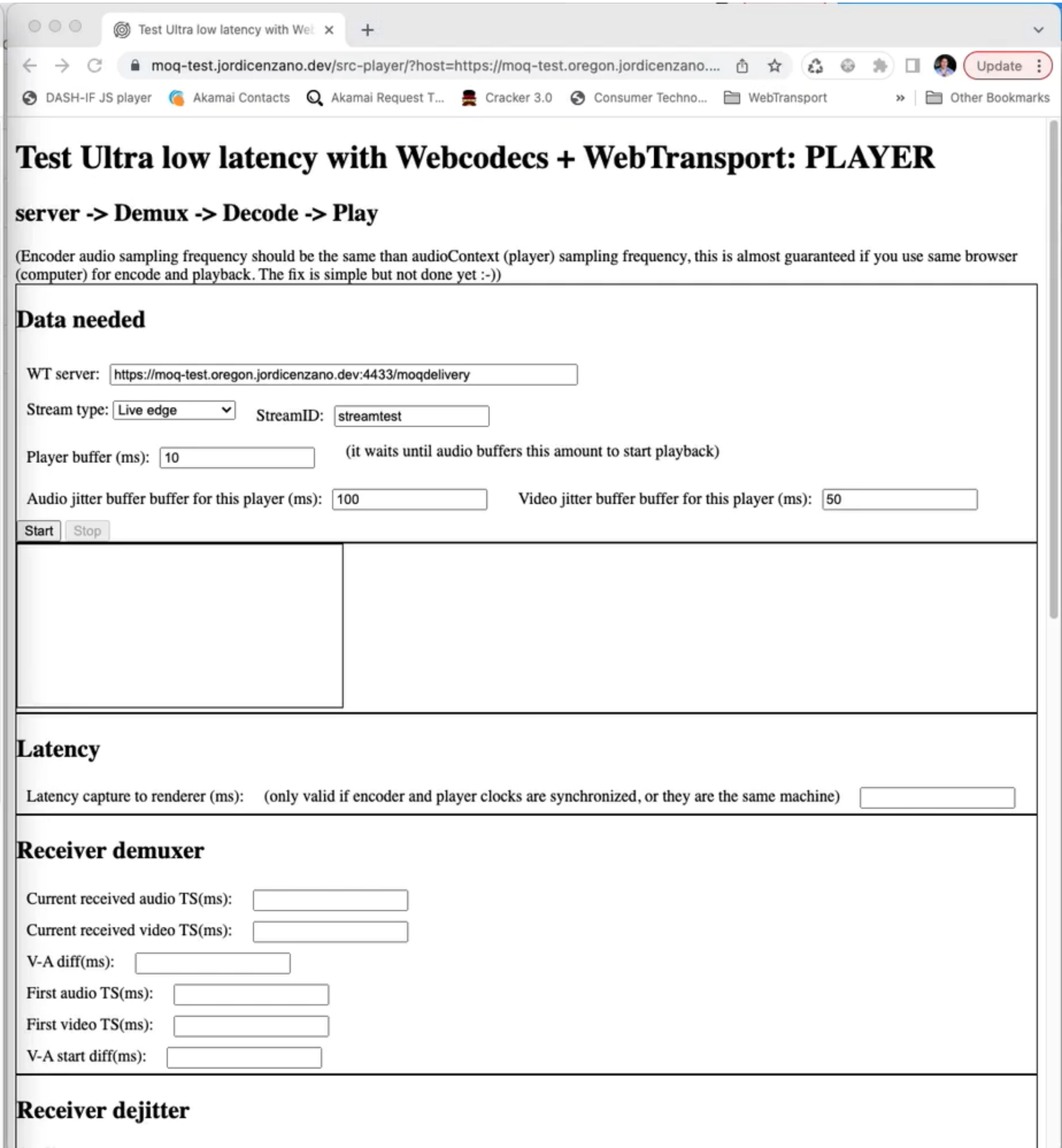
V-A start diff(ms):

First comp audio TS(ms):

First comp video TS(ms):

V-A comp start diff(ms):

Muxer sender


Test Ultra low latency with We
moq-test.jordicenzano.dev/src-player/?host=https://moq-test.oregon.jordicenzano....
Update
DASH-IF JS player
Akamai Contacts
Akamai Request T...
Cracker 3.0
Consumer Techno...
WebTransport
Other Bookmarks

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 browser (computer) for encode and playback. The fix is simple but not done yet :-))

Data needed

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

Stream type: Live edge StreamID: streamtest

Player buffer (ms): 10 (it waits until audio buffers this amount to start playback)

Audio jitter buffer buffer for this player (ms): 100 Video jitter buffer buffer for this player (ms): 50

Start Stop

Latency

Latency capture to renderer (ms): (only valid if encoder and player clocks are synchronized, or they are the same machine)

Receiver demuxer

Current received audio TS(ms):

Current received video TS(ms):

V-A diff(ms):

First audio TS(ms):

First video TS(ms):

V-A start diff(ms):

Receiver dejitter

Akamai