



Evaluation of HTTP/3 for Media Streaming

Philip Nys | Open Distributed Systems | Workshop 2 | June 20th 2023

Overview



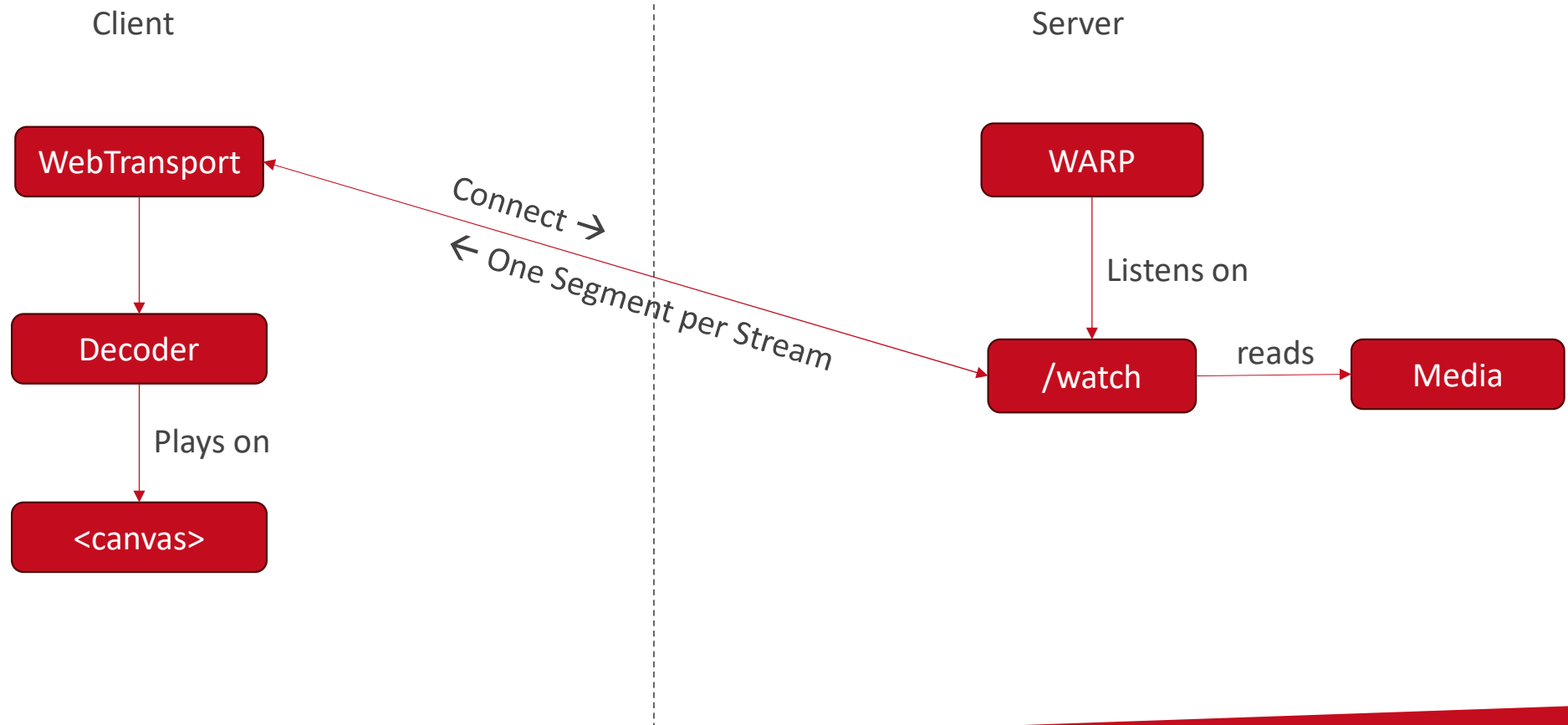
- Recap Problem Statement
- Proposed Solutions and Progress
 - WARP
 - RUSH
 - QUICR
- Challenges/Blockers
- Demo Recording
 - WARP
 - RUSH
- Next Steps
- References

Recap Problem Statement

- Media Streaming makes up the majority of all Internet traffic
 - In 2017: **72%** ^[1]
- HTTP/3 aims to reduce latency, head of line (HOL) blocking and improve the user experience
 - Better stream multiplexing
 - Faster verification handshake ^[2]

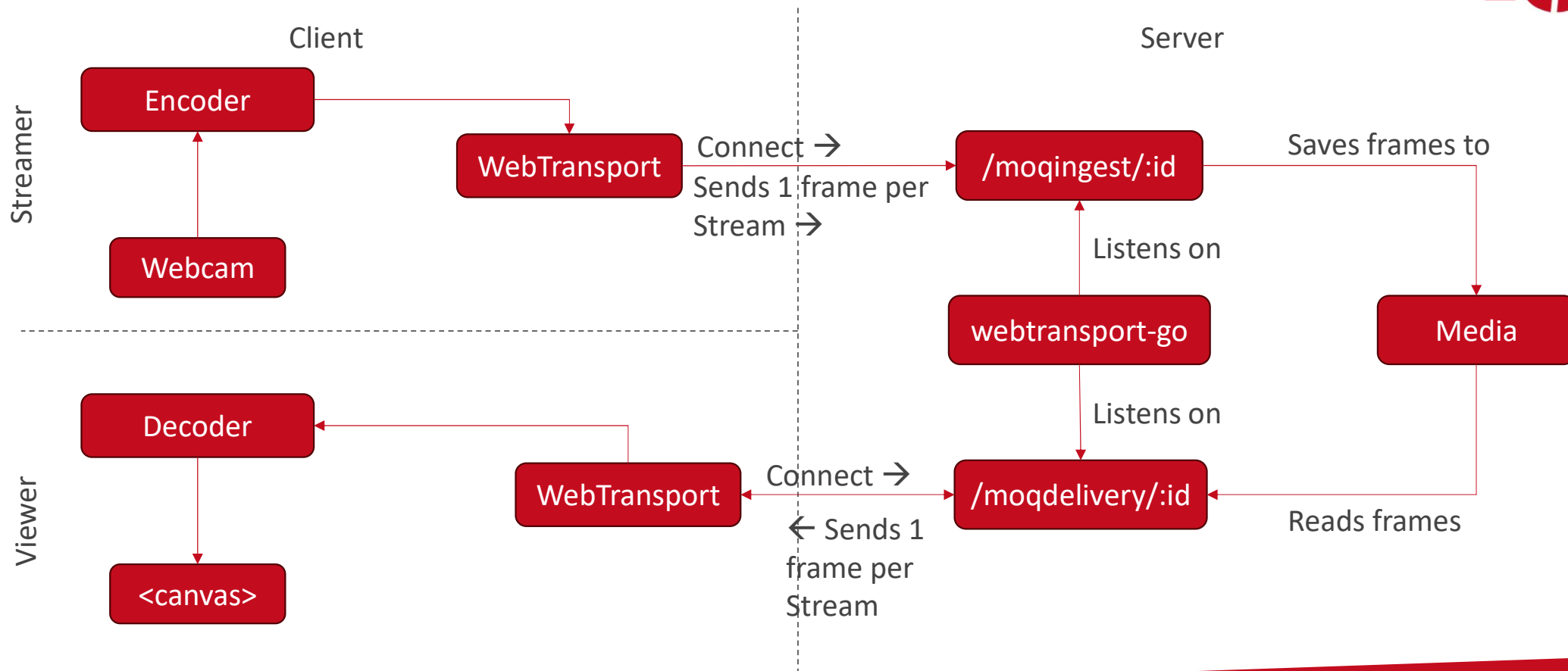
Proposed Solution

WARP – Streaming media (vod / live stream)



Proposed Solution

RUSH – 1 Streamer / N Viewer



Proposed Solution

QUICR – 2-way Video Call or 1 Streamer / N Viewer

- Demo runs a dedicated macOS client
- Server uses libquicr ^[7] a API implementation of quicrq ^[8] library
- Demo 1:
 - 2-way call: <https://user-images.githubusercontent.com/947528/201170693-629525d4-211e-4849-98c5-57b883bccba7.mp4>
 - Server: Akamai's Atlanta Network, USA
 - Client: London, UK
- Demo 2:
 - 1 streamer / 3 viewers: <https://user-images.githubusercontent.com/947528/181114950-400c22da-f623-4bc5-a8d6-7f4e3188a9c5.mp4>
 - Server: AWS Ohio, USA
 - Client: San Jose California, USA

Challenges and Blockers

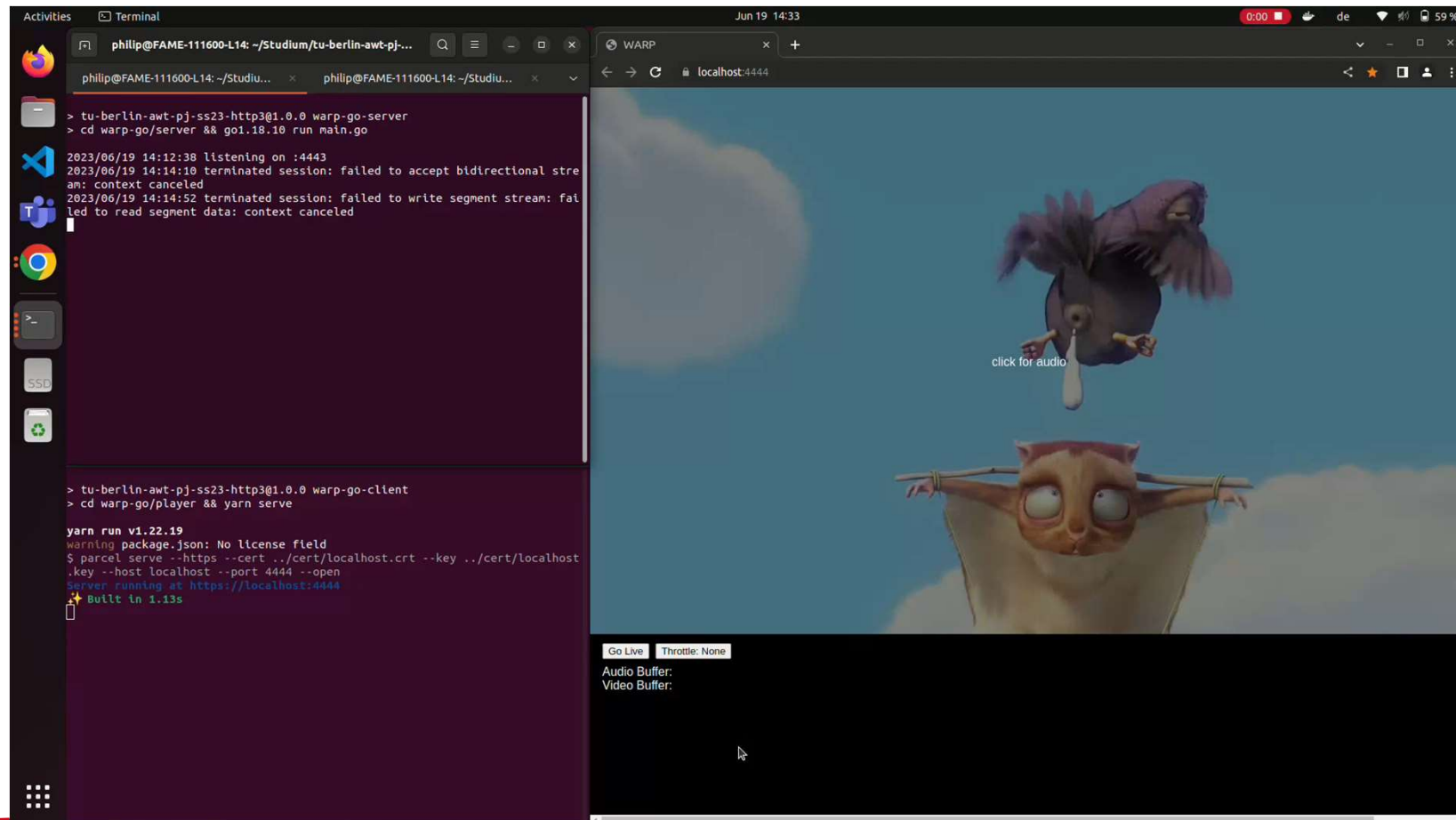
RUSH:

- Viewer doesn't show any video output from the server

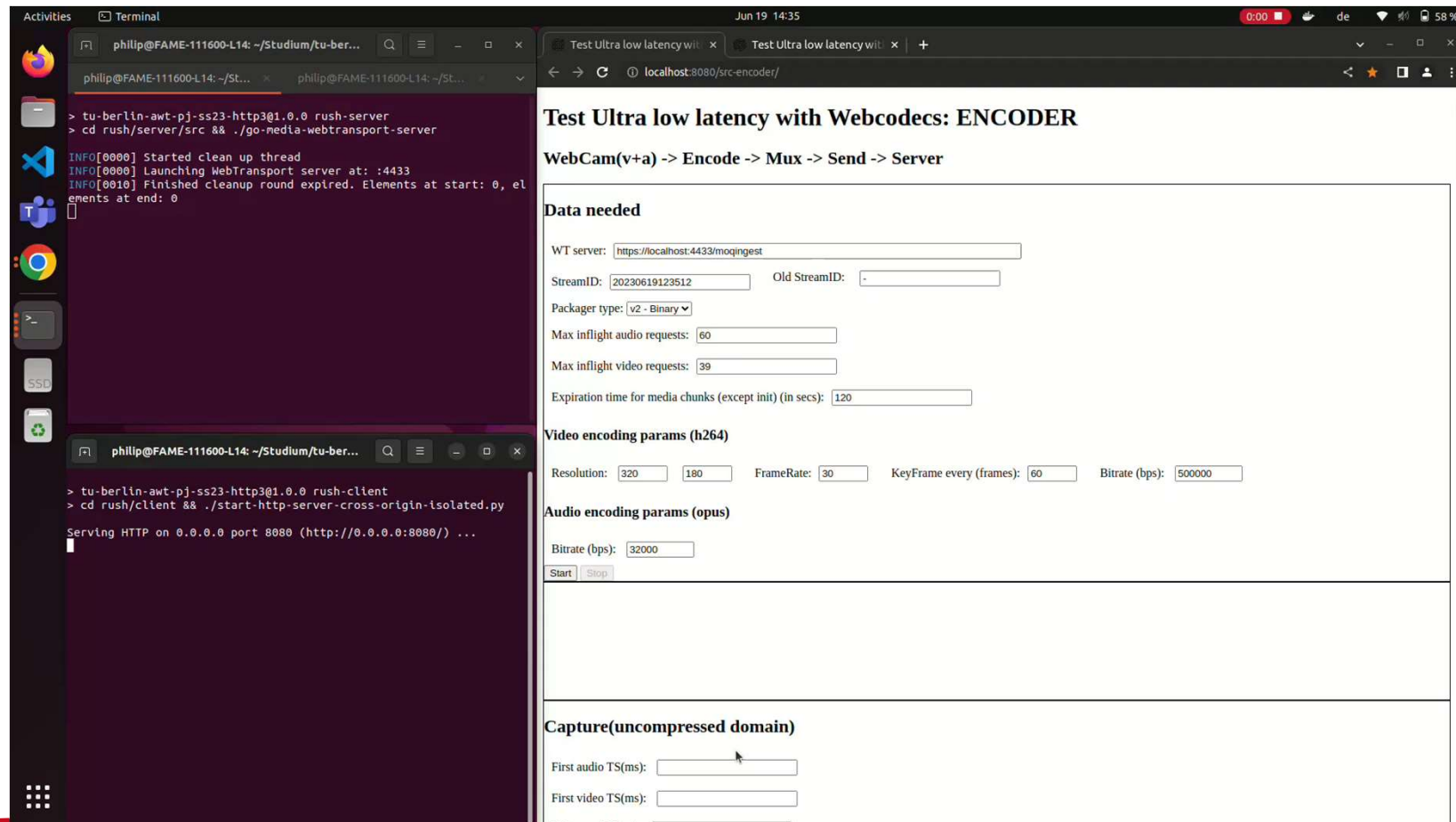
QUCIR:

- Building the code fails

Demo Recording - WARP



Demo Recording - RUSH



The screenshot displays a Linux desktop environment with a terminal window and a web browser window.

Terminal Window:

```

philip@FAME-111600-L14: ~/Stadium/tu-ber...
> tu-berlin-awt-pj-ss23-http3@1.0.0 rush-server
> cd rush/server/src && ./go-media-webtransport-server

INFO[0000] Started clean up thread
INFO[0000] Launching WebTransport server at: :4433
INFO[0010] Finished cleanup round expired. Elements at start: 0, elements at end: 0

philip@FAME-111600-L14: ~/Stadium/tu-ber...
> tu-berlin-awt-pj-ss23-http3@1.0.0 rush-client
> cd rush/client && ./start-http-server-cross-origin-isolated.py
Serving HTTP on 0.0.0.0 port 8080 (http://0.0.0.0:8080/) ...
  
```

Web Browser Window:

Address: `localhost:8080/src-encoder/`

Test Ultra low latency with Webcodecs: ENCODER

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

Data needed

WT server:

StreamID: Old StreamID:

Packager type:

Max inflight audio requests:

Max inflight video requests:

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

Video encoding params (h264)

Resolution: FrameRate: KeyFrame every (frames): Bitrate (bps):

Audio encoding params (opus)

Bitrate (bps):

Capture(uncompressed domain)

First audio TS(ms):

First video TS(ms):

Next Steps

- Dockerize WARP (go) and WARP (rust)
- Integrate Bandwidth Limiter
- Run Benchmarks to compare both implementations
- Try to get RUSH and QUICR to work

References

- [1] M. Nguyen, D. Lorenzi, F. Tashtarian, H. Hellwagner and C. Timmerer, "DoFP+: An HTTP/3-Based Adaptive Bitrate Approach Using Retransmission Techniques," in *IEEE Access*, vol. 10, pp. 109565-109579, 2022, doi: 10.1109/ACCESS.2022.3214827.
- [2] Divyashri Bhat, Rajvardhan Deshmukh, and Michael Zink. 2018. Improving QoE of ABR Streaming Sessions through QUIC Retransmissions. In Proceedings of the 26th ACM international conference on Multimedia (MM '18). Association for Computing Machinery, New York, NY, USA, 1616–1624. DOI:<https://doi.org/10.1145/3240508.3240664>
- [3] <https://github.com/facebookexperimental/webcodecs-capture-play>
- [4] <https://github.com/kixelated/warp>
- [5] <https://github.com/Quicr/qmedia>
- [6] <https://w3techs.com/technologies/details/ce-httpsdefault>
- [7] <https://github.com/Quicr/libquicr>
- [8] <https://github.com/Quicr/quicrq>

Thank you for listening! Any questions?