A Practical Assessment Approach of the Interplay Between WebRTC and QUIC

David Baldassin[a,b], Ludovic Roux [b], Guillaume Urvoy-Keller[a] and Dino Martin Lopez-Pacheco[a]

[a] Université Côte d'Azur, CNRS, I3S, France
[b] CoSMoSoftware, Singapore

Motivation

- Objective: efficiently combine QUIC and WebRTC
- Contribution: a flexible container-based testbed featuring
  - WebRTC with Legacy QUIC
  - WebRTC over unreliable datagram extension of QUIC
- Open questions:
  - Which QUIC implementation?
  - Which congestion control (CC) algorithm at the QUIC and WebRTC level?
  - Interplay between QUIC and WebRTC level CC

Related Work

- Work by J.Ott and M.Engelbart
  - QUIC unreliable extension (IETF draft) [3]
  - RTP over QUIC experiments [1]

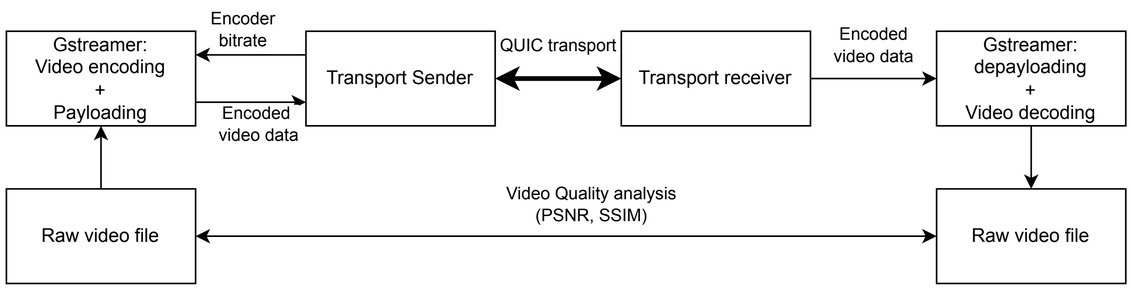

Figure 1: Engelbart and Ott setup

- Results [1]:
  - Two CC algorithms: one for RTP (SCReAM) and one for QUIC (QUIC NewReno)
  - The bitrate collapses when mixing two CC algorithms

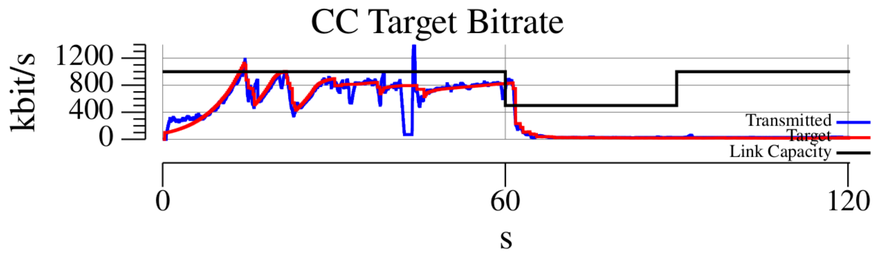

Figure 2: SCReAM and QUIC NewReno

A QUIC tunnel-based tested for WebRTC

- No native QUIC support in WebRTC → interconnect two WebRTC peers through a light QUIC tunnel
- Signaling: replace ICE candidate ports with QUIC tunnel listening port
- Media: listen for RTP packets and forward them in the QUIC tunnel

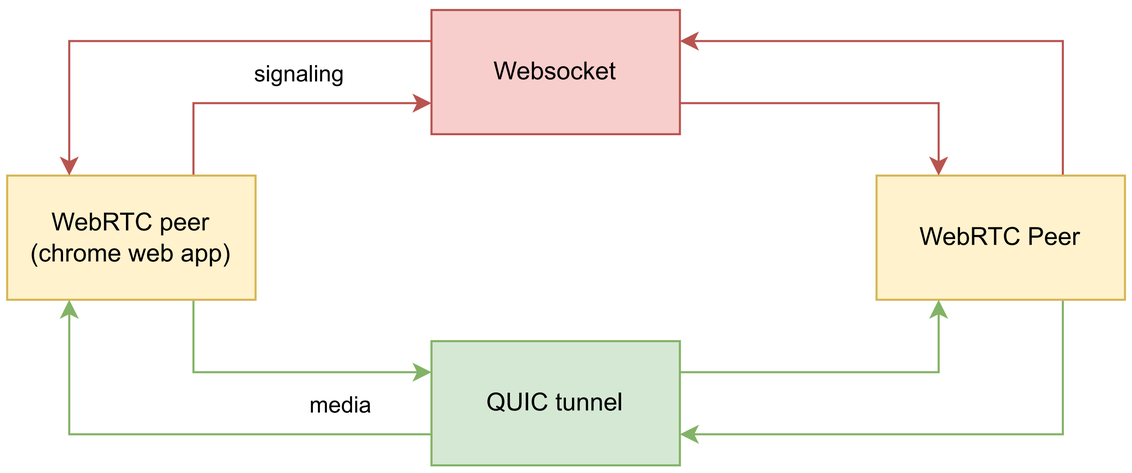

Figure 3: Webrtc over QUIC overview

- Use Medooze WebRTC media server [2] to send video and a Chrome web application as receiver
- Use qlog to log QUIC statistics and Qvis to visualize them.

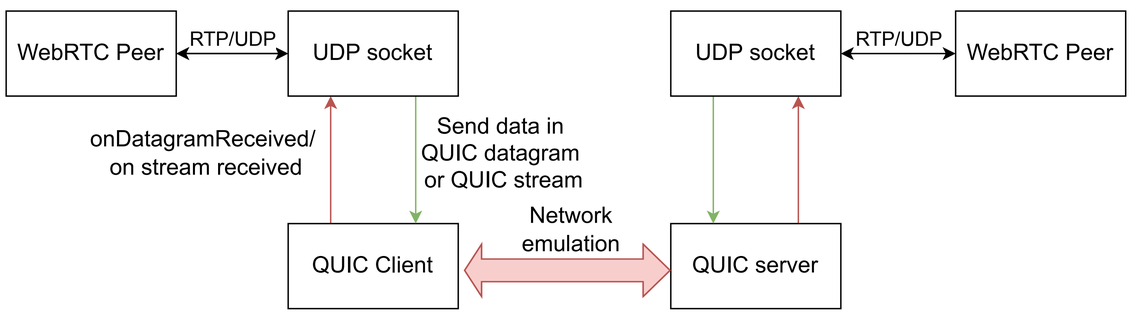

Figure 4: QUIC tunnel overview

- Mvfst (Meta) and quic-go implementations supported
- Tune bandwidth, losses and latency in network tunnel with Netem

First experiments

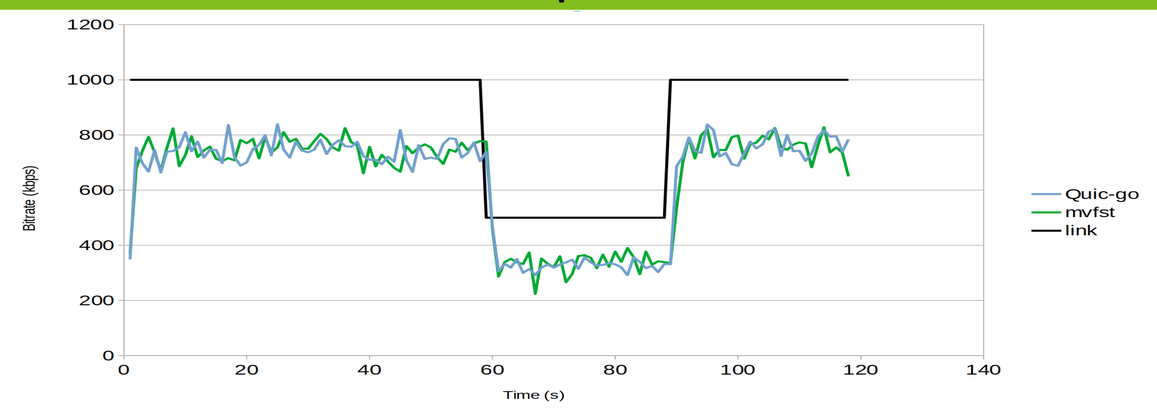

Figure 5: Comparison between mvfst and quic-go (both with NewReno)

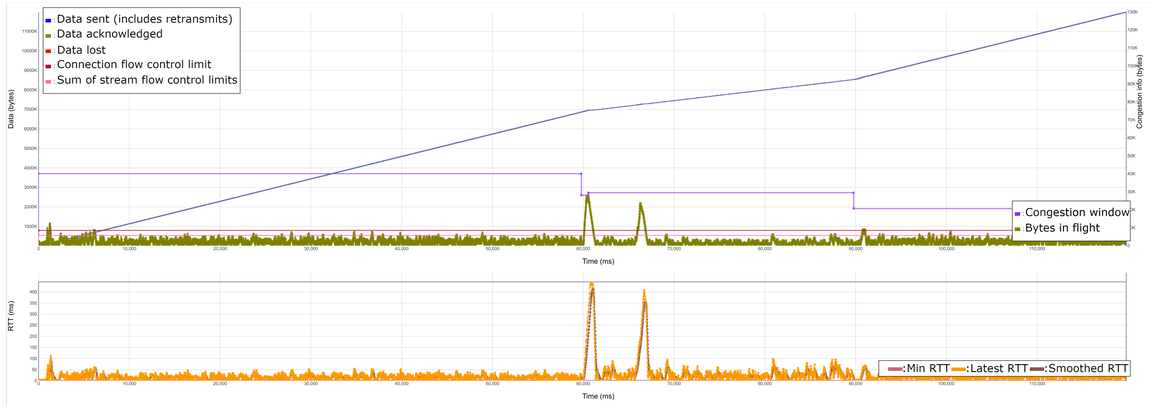

Figure 6: Qvis visualization (quic-go with NewReno)

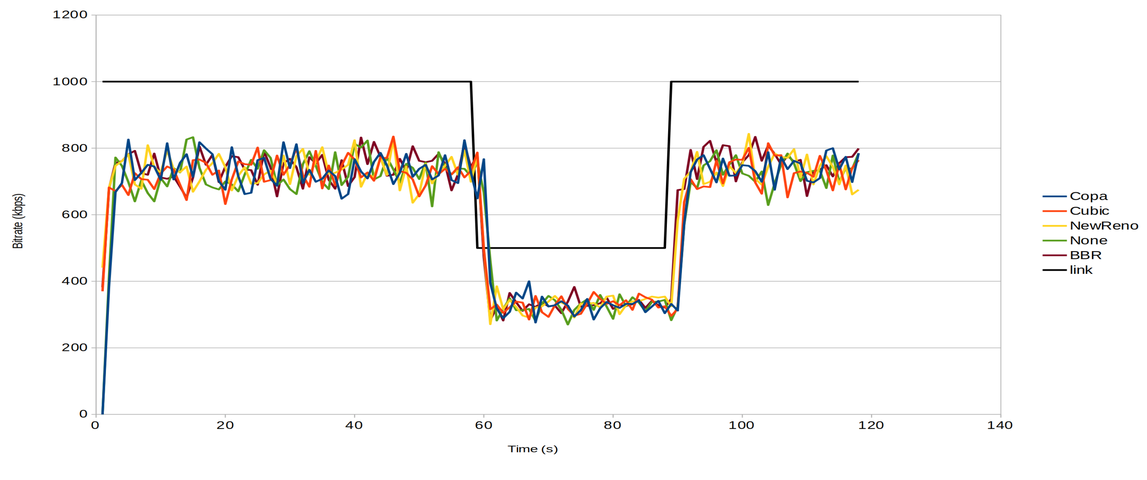

Figure 7: Comparison between congestion control algorithms (mvfst only)

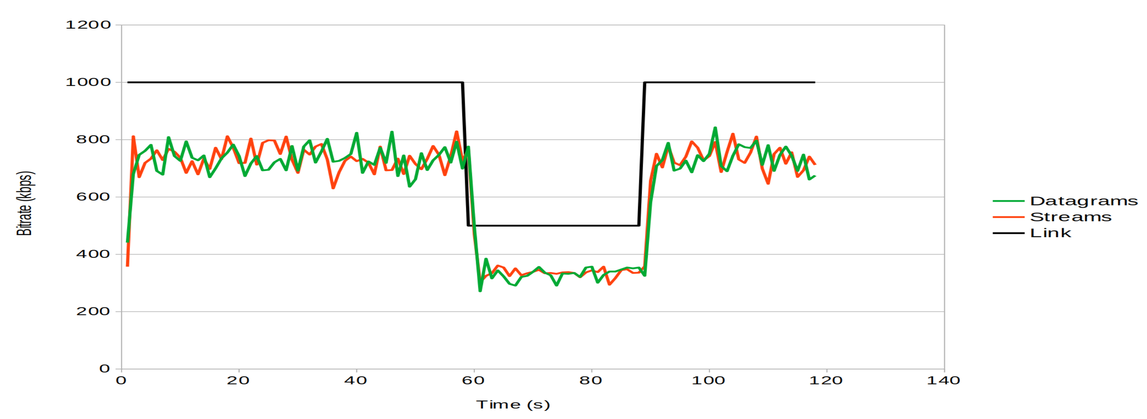

Figure 8: Comparison between datagram and stream usage (mvfst only)

Ongoing Work

- Flexible testbed with mvfst and quic-go implementations
  ... to be released as open source
- Initial tests: upper (WebRTC level) CC algorithm dominates
- Next steps
  - Investigate other network scenarios, e.g. random losses, to further test CC interactions
  - Add other QUIC implementations, e.g. Google or Microsoft
  - Test interoperability
  - Add video quality indicators (SSIM, VMAF)

References

[1] M. Engelbart and J. Ott.
Congestion control for real-time media over QUIC.
In *EPIQ '21: Proceedings of the 2021 Workshop on Evolution, Performance and Interoperability of QUIC, Virtual Event / Munich, Germany, 7 December 2021*, pages 1–7. ACM, 2021.

[2] S. G. Murillo.
Medooze media server.
https://github.com/medooze/media-server, 2022.

[3] J. Ott and M. Engelbart.
RTP over QUIC.
Internet-Draft draft-ietf-avtcore-rtp-over-quic-00, Internet Engineering Task Force, July 2022.
Work in Progress.