QUIC+FEC

Some results for low latency video streaming
Loss recovery in classical transport protocols (SR-ARQ)

Sending “hello, world !”

- Pkt 1 [“hel”]
- Pkt 2 [“lo, “]
- Pkt 3 [“worl”]
- Pkt 4 [“d !”]
- Pkt 5 [“worl”]

Loss detection
Retransmission

> 1*RTT

ACK {1,2,4}

Deliver “hello, “
to the application

Store “d !” in
receive buffer

Deliver “world !”
to the application

Sending “hello, world !”

Deliver “hello, “
to the application

Store “d !” in
receive buffer

Deliver “world !”
to the application

ACK {1,2,4}
Forward Erasure Correction in the transport

Sending “hello, world!”

- Pkt 1: [“hel”]
- Pkt 2: [“lo, “]
- Pkt 3: [“worl”]
- Pkt 4: [“d !”]
- Pkt 5: [“hel” ⊕ “lo, “ ⊕ “worl” ⊕ “d !”]

Pkt 1, 2, 3, 4 contain source symbols
Pkt 5 contains a repair symbol

Reconstruct “worl”
Deliver “hello, world!” to the application
QUIRL: implementing draft-michel-quic-fec-01

Based on Cloudflare’s quiche implementation.

- the patch for adding FEC is around 1500 lines (the rest sits in external crates)
- can be integrated with curl for HTTP/3 queries
- can probably be integrated with MOQT as well

Forward Erasure Correction for QUIC loss recovery

Abstract
This documents lays down the QUIC protocol design considerations needed for QUIC to apply Forward Erasure Correction on the data sent through the network.
Using QUIRL for FFmpeg/GStreamer

- Every **RTP packet** is placed into a dedicated **QUIC stream**
  - large RTP packets cannot fit in DATAGRAM frames
- Repair symbols are sent regularly to protect one or more video frames
- We want to minimize frames **lateness** to improve video fidelity (SSIM)
Sending drone videos over Starlink

Analyzing Real-time Video Delivery over Cellular Networks for Remote Piloting Aerial Vehicles

Aygün Baltaci
aygun.baltaci@tum.de
Technical University of Munich & Airbus
Germany

Hendrik Cech
hendrik.cech@tum.de
Technical University of Munich
Germany

Nitin R Mohan
mohanc@in.tum.de
Technical University of Munich
Germany

Fabien Geyer
fabien.geyer@airbus.com
Airbus
Germany

Vaibhav Bajpai
bajpai@cispa.de
CISPA Helibolten Center for Information Security
Germany

Jörg Ott
ott@in.tum.de
Technical University of Germany

Dominic Schupke
dominic.schupke@airbus.com
Airbus
Germany

ABSTRACT
Emerging Remote Piloting (RP) operations of electrified Unmanned Aerial Vehicles (UAVs) demand low-latency and high-quality video delivery to conduct safe operations in the low-altitude airspace. Although cellular networks are one of the prominent candidates to provide connectivity for such operations, their ground-centric nature limits their capabilities in achieving seamless and reliable aerial connectivity. In this paper, we study the feasibility of supporting RP operations with low latency and high-quality video delivery over commercial cellular networks. By setting up an adaptive bitrate video transmission pipeline with the Google Congestion Control (GCC) and Self-Checked Rate Adaptation for Multimedia (SCREAM) Congestion Control (CC) algorithms, we analyze the video delivery performance for the RP application requirements and compare the performance of GCC and SCREAM against constant bitrate video delivery. Our results show that low-latency video delivery with < 300 ms playback latency between full-HD and 4K resolution can be maintained up to about 95% of the time in the air and meeting the service requirements using cellular networks. We release our collected traces and the video transmission pipeline as open-source to facilitate research in this field.

CCS CONCEPTS
• Networks → Network performance evaluation; Network performance analysis.

KEYWORDS
Adaptive streaming, real-time video, cellular networks, LTE, ULP, RTP, UAV, drone, eVTOL, flying taxi, UAM, AAM

ACM Reference Format:
Average SSIM per video over Starlink

![Graph showing the average SSIM per video over Starlink. The x-axis represents the GStreamer playback buffer (ms) ranging from 50 to 500, and the y-axis represents the average SSIM per video ranging from 0.9 to 1.0. The graph includes data for different protocols: RTP, RTP_FEC, QUIC, and QUIRL.](image)
If time: Improving curl’s Transfer Completion Time (TCT)

We can send repair symbols during quiescence periods, when the cwin allows it.

50kB transfers using curl with QUIRL

No impact when no loss occurs
Lower completion time upon losses, especially in the last flight
If time: Improving curl’s Transfer Completion Time (TCT)

We can send repair symbols during quiescence periods, when the cwin allows it.

10MB file transfers using `curl` with QUIRL
Summary

All our work and code will soon be open source. More details can be found in:

- The QUIRL paper (soon)

If your use-cases may benefit from QUIC-FEC,

- Discuss on slack and the mailing list
- Send us an e-mail to collaborate: francois.michel@uclouvain.be
- Implement draft-michel-quic-fec