RESEARCH-ARTICLE

FrameTrace: Frame-Level Telemetry for Media over QUIC

BIRKAN DENIZER, University of Kiel, Kiel, Schleswig-Holstein, Germany

LASSE WINKEL, Technical University of Lübeck, Lubeck, Schleswig-Holstein, Germany

OLAF LANDSIEDEL, Hamburg University of Technology, Hamburg, Germany

Open Access Support provided by:

Hamburg University of Technology

University of Kiel

Technical University of Lübeck

Published: 08 September 2025

Citation in BibTeX format

SIGCOMM '25: ACM SIGCOMM 2025 Conference
September 8 - 11, 2025
Coimbra, Portugal

Conference Sponsors:
SIGCOMM

EMS '25: Proceedings of the 3rd Workshop on Emerging Multimedia Systems (September 2025)
https://doi.org/10.1145/3746441.3748231
ISBN: 9798400720611

FrameTrace: Frame-Level Telemetry for Media over QUIC

Birkan Denizer
Kiel University
Kiel, Germany
birkan.denizer@cs.uni-kiel.de

Lasse Winkel*
Technical University of Applied Sciences Lübeck
Lübeck, Germany
lasse.winkel@stud.th-luebeck.de

Olaf Landsiedel
Hamburg University of Technology
Hamburg, Germany
olaf.landsiedel@tuhh.de

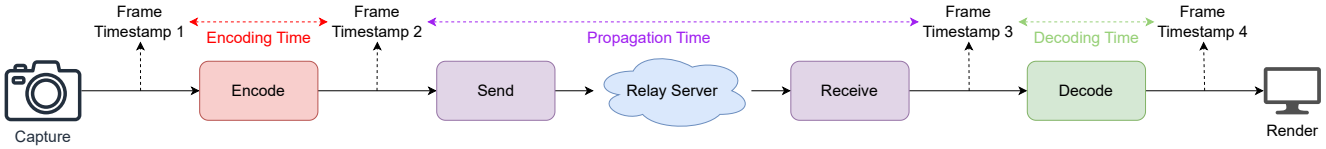

Figure 1: We collect four timestamps during the video processing pipeline to calculate processing times for the encoding, propagation, and decoding stages. FrameTrace gives the end-to-end latency for MoQ by using all stages.

Abstract

Live-streaming and remote control applications increasingly demand low end-to-end latency. Yet, evaluation tools of today give only connection-level or segment-level metrics. Coarse telemetry obscures the distinction between stalls experienced during encoding, propagation, and rendering, thereby concealing system bottlenecks. For example, under lossy conditions, long Group-of-Pictures (GoP) sizes increase stall duration as delayed I-frames block decoding of dependent frames. Prior studies of Media over QUIC (MoQ) use average segment times or logs from video players, leaving the frame-level latency and quality trade-off largely unexplored.

In this paper, we introduce FrameTrace, a frame-level logger for analyzing Quality of Experience telemetry. Using FrameTrace, we demonstrate that MoQ itself cuts latency by 83% and 63% at 5% packet loss and 10 Mbps bandwidth limitation, respectively, compared to Low-Latency DASH (LL-DASH). As a case study, we investigate how dynamically adapting GoP size changes stall duration and perceived quality. A shorter GoP size improves the VMAF score by up to 30% compared to a longer one. Driven by FrameTrace, our lightweight real-time GoP adaptation controller reduces latency by an additional 10% while increasing VMAF by 3.65%.

*The work was carried out while at Kiel University, Germany.

EMS '25, Coimbra, Portugal

ACM ISBN 979-8-4007-2061-1/2025/09
https://doi.org/10.1145/3746441.3748231

CCS Concepts

• **Information systems** → **Multimedia streaming**.

Keywords

frame-level telemetry, real-time GoP adaptation, MoQ, DASH

ACM Reference Format:
Birkan Denizer, Lasse Winkel, and Olaf Landsiedel. 2025. FrameTrace: Frame-Level Telemetry for Media over QUIC. In *3rd Workshop on Emerging Multimedia Systems (EMS '25), September 8–11, 2025, Coimbra, Portugal.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3746441.3748231

1 Introduction

The increase in live-streaming, conferencing, and remote-control applications has made ingest and delivery at both low-latency and adaptivity essential [5, 13]. To mitigate the impact of dynamic network conditions, such as fluctuating packet delivery rates, latency, and bandwidth, applications must continuously adjust parameters, such as bitrate and Group-of-Pictures (GoP) size, to sustain high quality. HTTP-based DASH dominates video streaming of today [11]. Its low-latency version (LL-DASH) still incurs more than a second delay in typical configurations, which is unsuitable for real-time use cases that target latencies of a few hundred milliseconds [3, 4]. To address the latency limitations of these protocols, the Internet Engineering Task Force (IETF) is developing Media over QUIC (MoQ) [6]. MoQ provides a design for one-to-many scenarios like adaptive, server-based streaming, as opposed to peer-to-peer streaming protocols such as WebRTC. This makes MoQ particularly well-suited for highly interactive applications, such as online gaming,

live commentary, and the remote control of hardware. Existing tools expose only connection-level or segment-level telemetry; no logger captures per-frame capture-to-render QoE metrics for MoQ or any QUIC-based media protocol [7]. Therefore, it is challenging to understand in which part of the video pipeline stalls occur and degrade QoE.

In this paper, we introduce FrameTrace, a frame-level performance logger for MoQ. FrameTrace timestamps precise capture, encode, arrival, and decode steps to expose end-to-end behaviour of MoQ. Using FrameTrace, we analyze the relationship between varying GoP size, stall duration, and perceived quality. On top of FrameTrace, we add a threshold-based real-time GoP adaptation that optimizes for low latency and high quality by reducing stall time. We compare MoQ and LL-DASH across QoE metrics such as latency, packet loss, and stall events. Also, we measure VMAF for fixed and adaptive GoP sizes in MoQ.

Overall, this paper makes the following contributions:

(1) Design of FrameTrace, a frame-level performance logger for MoQ to track and analyze QoE telemetry.
(2) Real-time GoP adaptation, driven by FrameTrace.
(3) Evaluation of MoQ that shows up to 83% latency reduction compared to LL-DASH, shorter GoP sizes have a VMAF advantage of up to 30% compared to longer GoP sizes, and GoP adaptation cuts the latency by up to 10% and increases VMAF by up to 3.65%.

The rest of this paper is structured as follows: Section 2 presents the background and related work in the field of media delivery and GoP adaptations in streaming, Section 3 introduces the design of FrameTrace, Section 4 evaluates the performance of MoQ and LL-DASH using FrameTrace, and Section 5 concludes the paper.

2 Background and Related Work

In this section, we first provide the background information on LL-DASH and MoQ. Then, we look at the related work in media delivery and GoP adaptations in streaming.

2.1 LL-DASH and Media over QUIC (MoQ)

LL-DASH reduces the segment duration to minimize the end-to-end delay compared to DASH [9]. Chunked encoding and delivery features enable quick video delivery to the viewer. However, HTTP/2 head-of-line blocking negatively affects LL-DASH under lossy network conditions [8].

The Media over QUIC Transport (MOQT) builds on independent streams of QUIC for real-time media [6]. It supports existing encoding and packaging technologies within a MoQ stream. Sending data on a separate QUIC stream enables granular control over data transmission for improved QoE.

2.2 Related Work

Media Delivery: Zhang et al. [14] benchmark the performance of LL-DASH and LL-HLS players in metrics such as average stream bitrate, the amount of media data downloaded, and streaming latency. Gurel et al. [2] benchmark MoQ with LL-DASH in terms of latency, bandwidth, and stall duration. Wang et al. [12] propose an extension to standardize QoE telemetry, enabling the export of metrics into cloud analytics platforms. Nguyen et al. [7] benchmark MoQ with LL-DASH, reporting segment-level latency and throughput.

Unlike previous works, FrameTrace logs per-frame capture-to-render QoE telemetry for MoQ. We reveal metrics on latency, stalls, GoP size, number of skipped frames, frame delivery rate (FDR), frame render rate (FRR), and VMAF.

GoP Adaptations in Streaming: The High Efficiency Streaming Protocol (HESP) [10] splits each video into an I-frame only initialization stream and a long-GoP continuation stream. Chen et al. [1] introduce a *pseudo-GOP* for video on demand (VOD) to optimize QoE for longer GoP sizes.

By contrast, our lightweight GoP adaptation based on FrameTrace uses the real-time FDR to detect packet loss and delay, and then shrinks the GoP size when FDR drops.

3 Design

In this section, we first present an overview of the design, then describe it in detail. Our design includes two key parts: per-frame telemetry and real-time GoP adaptation.

3.1 Per-frame Telemetry

FrameTrace exposes per-frame QoE metrics such as latency and stalls for MoQ streams. We tag each frame during the capture, encoding, arrival, and render steps. In Figure 1, we highlight the four places we access individual frames within the video processing pipeline to store a timestamp. FrameTrace takes the first timestamp (#1) after capture and before encoding. Next, it takes the second timestamp (#2) when a frame is written to a segment. The delta between timestamps #1 and #2 gives the per-frame encode time. It also stores this value and the size of each frame to know how many bytes are sent to a subscriber. FrameTrace logs the third timestamp (#3) and the number of received bytes when a frame is received at the client. Using the delta between timestamps #2 and #3, it calculates the propagation time. We consider any frame without a reception timestamp (#3) lost during transmission. FrameTrace calculates FDR by measuring the number of frames sent and received. It captures the fourth timestamp (#4) when a frame is decoded and rendered to the screen. Ascending-order MoQ may drop late frames before render. It calculates FRR by measuring whether each frame has been rendered. By combining timestamps #1 to #4, FrameTrace gives per-frame capture-to-render latency.

3.2 Real-Time GoP Adaptation

Built on top of FrameTrace, we add a light-weight GoP adaptation controller driven by FDR. The controller on the client side dynamically requests adjustments for the GoP size when FDR crosses defined thresholds (i.e., how ABR algorithms work). We use long GoP sizes to improve quality when FDR remains high. We define FDR thresholds as follows:

- **Long GoP size**: Used when FDR is above the upper threshold (e.g., 99%), indicating stable conditions.
- **Medium GoP size**: Used when FDR is between thresholds (e.g., 99% — 95%), allowing for adjustment.
- **Short GoP size**: Used when FDR is below the lower threshold (e.g., 95%), indicating unstable conditions.

When FDR drops below thresholds, FrameTrace shrinks the GoP size. Due to the same target bitrate in our design, a shorter GoP size allows smaller I-frames to be sent more often, thus refreshing the content without waiting for retransmissions during congestion. The relationship between GoP sizes and thresholds is as follows:

$$\begin{aligned} \text{Long GoP size}: &\quad \textbf{FDR} > \text{Threshold 1} \\ \text{Medium GoP size}: &\quad \text{Threshold 1} > \textbf{FDR} > \text{Threshold 2} \\ \text{Short GoP size}: &\quad \text{Threshold 2} > \textbf{FDR} \end{aligned} \tag{1}$$

4 Evaluation

In this section, we first detail our experiment setup and metrics. Next, we compare MoQ and LL-DASH, stressing both with loss, delay, and bandwidth limits. Then, we test our real-time GoP adaptation based on FrameTrace.

4.1 Experiment Setup

Our MoQ setup includes two MoQ publishers for either live camera or VOD, a relay node, and a subscriber. Our live publisher encodes in real-time, creating one QUIC stream per segment, allowing real-time GoP adaptation without the head-of-the-line blocking issues of prior LL-DASH solutions. We use the VOD publisher based on FFmpeg[1] for VMAF tests in Section 4.6. We use the open source moq-js[2] and moq-rs[3] implementations for our publishers, moq-rs for our relay node, moq-js for our subscriber, and dash.js player.

By default, we stream with a resolution of 1920×1080 pixels, 30 FPS, 6 Mbps bitrate, 2 seconds GoP size, 0 ms additional network delay, and 100 Mbps link bandwidth. Latency budget (e.g., buffer size) is kept to a minimum for each protocol. Furthermore, we perform 3 experiments for each streaming setup and calculate the average values. To emulate real-world network conditions, we use tc-netem[4]. We use Precision Time Protocol (PTP) for clock synchronization.

[1]Available at https://www.ffmpeg.org
[2]Available at https://github.com/kixelated/moq-js
[3]Available at https://github.com/kixelated/moq-rs

Table 1: MoQ performance results

Metric	0% Loss	5% Loss	10% Loss
FRR (%)	99.98	99.46	98.58
# stall events	272.67	467	490.33
Stall duration (s)	2.43	8.53	12.67
Encoding time (ms)	45.55	46.82	48.56
Propagation time (ms)	5.2	22.13	50.15
Decoding time (ms)	7.29	7.53	7.79
Total time (ms)	66.99	84.87	113.13

Table 2: LL-DASH performance results

Metric	0% Loss	5% Loss	10% Loss
FRR (%)	94.93	97.67	14.94
# stall events	73	27.67	31.33
Stall duration (s)	0.975	2.771	41.348
Time behind live (ms)	67.3	268.38	28449.7
Buffer time (ms)	74.85	246.5	57.88
Total time (ms)	142.14	514.89	28506.58

4.2 Metrics

We calculate the following metrics for MoQ and LL-DASH:

- The Frame Render Rate (FRR) (%)
- The number of skipped frames
- The number of stall events and stall duration (s)
- Total time (end-to-end latency) (ms)
- **MoQ-only:** Encode, propagation, decode time (ms)
- **MoQ-only:** GoP size
- **LL-DASH-only:** Time behind live and buffer time (ms)

We measure visual quality with VMAF [5]. Using individual timestamps, we reconstruct the video using the most recently rendered frame, taking any frame stall into account.

4.3 Stable Network Conditions

In this section, we detail the system performance of MoQ vs. LL-DASH under stable network conditions. Compared to the LL-DASH, the MoQ achieves 5% higher FRR and lowers the latency by half, while having more stalls in Tables 1 and 2.

Next, we compare stall durations under different loss rates for MoQ in Figure 2. Most stalls are short, despite contributing to more stall events and a longer total stall duration.

[4]https://man7.org/linux/man-pages/man8/tc-netem.8.html
[5]Available at https://github.com/Netflix/vmaf

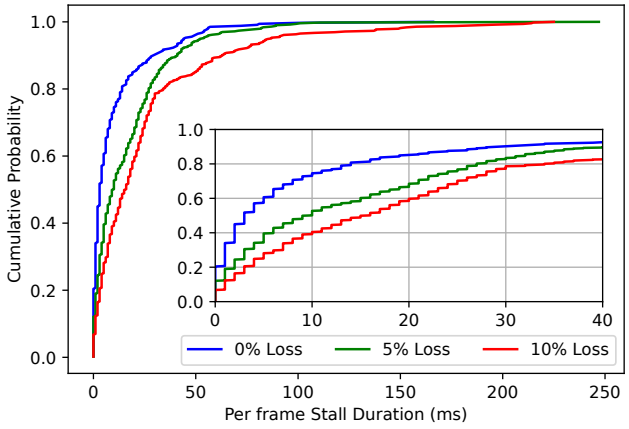

Figure 2: Comparison of per-frame stall durations under loss rates for MoQ: Most stalls occur in the sub-40-ms range, with longer stalls on higher packet losses.

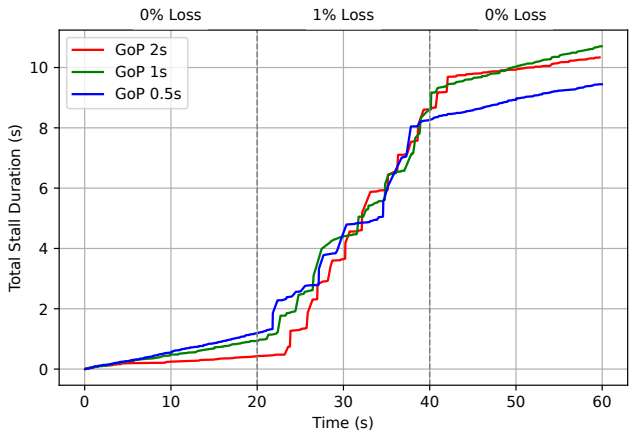

Figure 3: We compare the latency and stall duration of fixed GoP sizes of 2, 1, and 0.5 seconds under dynamic packet loss. Under loss, shorter GoP sizes skip content more often and keep the stall duration lower than higher GoP sizes. We apply a 500 ms delay.

4.4 Variable Network Conditions

In this section, we give details on MoQ and LL-DASH under variable network conditions, namely, packet loss, bandwidth constraints, and additional delay. Table 1 shows MoQ under 5% and 10% loss. While the average encoding and decoding times remain similar and FRR stays above 98%, the latency rises from 67 ms to 85 ms and 113 ms. FRR of LL-DASH collapses from 97% to 15% under 5% and 10% loss in Table 2, while latency increases from 142 ms to 514 ms and 28 s. Comparing the total times, we see 142.14 ms at 0% loss, 514.89 ms at 5% loss, and 28 s at 10% loss. When we compare MoQ against LL-DASH under packet loss conditions, at 5% loss both have high FRR, while at 10% loss only MoQ produces usable results. While the number of stall events increases for MoQ, Figure 2 confirms that MoQ stalls are micro-stutters.

We compare 2 s, 1 s, and 0.5 s GoP sizes under dynamic packet loss rates in Figure 3. 0.5 s GoP size cuts average latency by around 20 ms compared to 2 s GoP size, but produces more stalls. However, under loss and delay, the shorter GoP size skips ahead, reducing the stall duration compared to the longer GoP sizes. The **key insight** is that long GoPs are preferable under stable conditions, while short GoPs work best under congestion. In Figure 4, MoQ achieves an average of 67 ms and 85 ms at 0% and 5% loss, respectively, compared to 142 ms and 515 ms of LL-DASH. At 50 Mbps and 10 Mbps limits, MoQ achieves 78 ms and 132 ms, respectively, compared to 156 ms and 361 ms of LL-DASH in Figure 5. Even at a 5% loss or a 10 Mbps bandwidth limitation, the total latency of MoQ stays below the best case of LL-DASH. As we increase the additional delay in Figure 6, the 2-second GoP size collapses to under 50% FRR while both 0.5-second and 1-second GoP sizes maintain around 66% FRR.

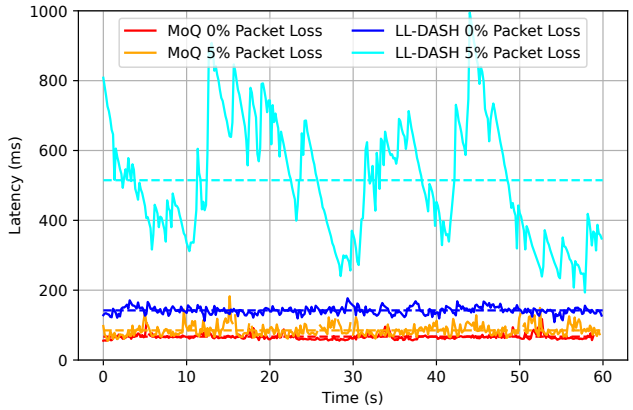

Figure 4: MoQ and LL-DASH latency under stable and packet loss conditions. We show the specific per-frame latency and the average latency over the duration.

4.5 Real-Time GoP Adaptation

Table 3 provides empirical FDR ranges for three GoP size threshold sets with a 2-second default. For example, set 1 switches to 1-second and 0.5-second GoP sizes when FDR drops below 99% and 95%, respectively. We compare the performance of adaptive GoP size with threshold set 1 under 0%, 10%, and 20% loss in Table 4. Under no loss, adaptive GoP behaves the same as fixed GoP size. As we increase the loss, FRR and the average GoP size drop to 91% and less than a second, respectively, and the average latency increases by almost 70 ms. We leave threshold tuning for future work.

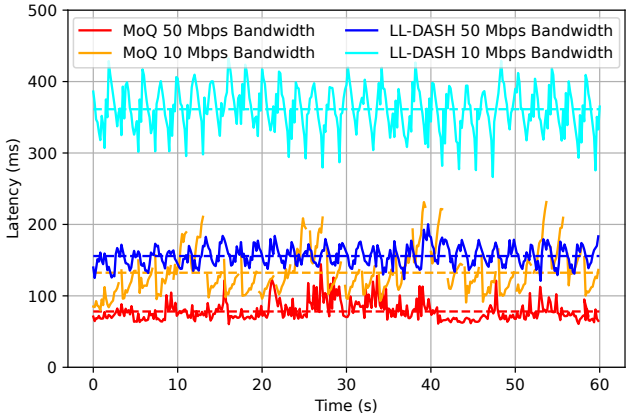

Figure 5: MoQ and LL-DASH latency with bandwidth limitation. We display the specific per-frame latency and the average latency over the whole duration.

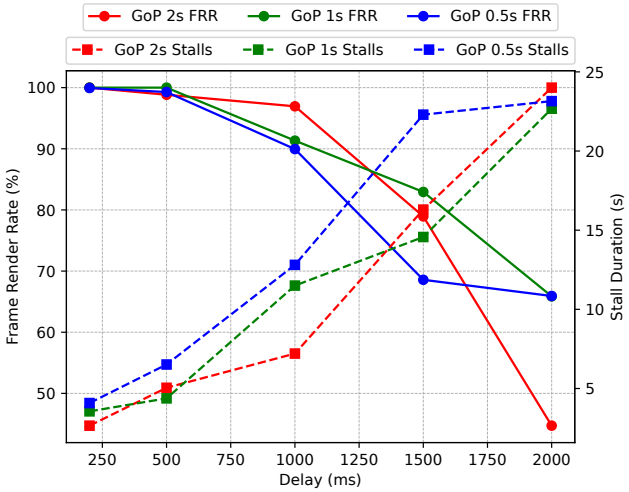

Figure 6: FRR and stall duration with fixed GoP sizes of 2, 1, and 0.5 seconds under 200 to 2000 ms dynamic network delays during the middle 20 seconds of the experiment and 50 ms network delay otherwise.

Table 3: FDR ranges for three GoP size threshold sets. Based on FDR feedback, we adapt the GoP size.

Threshold Set	1 s threshold (FDR)	0.5 s threshold (FDR)
1	99%	95%
2	95%	90%
3	90%	85%

We plot per-frame latency of adaptive vs fixed GoP sizes under 30% loss in Figure 7. Adaptive GoP shortens the segment duration during the loss window to skip to newer content rather than wait for retransmissions. As a result, the overall latency drops by 10% from 168 ms to 151 ms.

Table 4: Comparison of real-time GoP adaptation performance results with threshold set 1 under 0%, 10%, and 20% packet losses. We apply a 50 ms delay.

Metric	0% Loss	10% Loss	20% Loss
FRR (%)	100	98.41	91.35
# skipped frames	0	28.67	155.67
GoP size (s)	2.02	1.56	0.88
Propagation time (ms)	56.39	97.57	133.41
Total time (ms)	122.64	158.16	191.28
Stall duration (s)	0.97	23.29	31.44

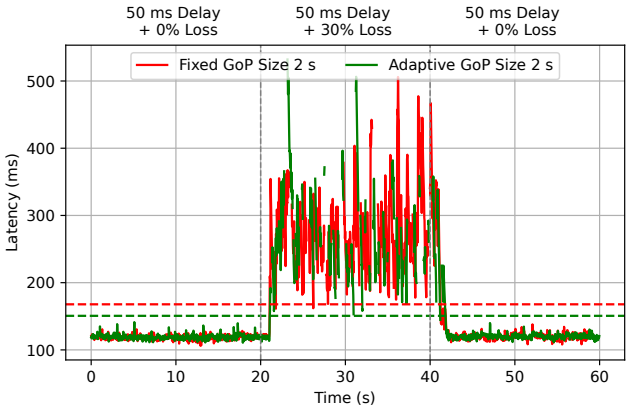

Figure 7: Per-frame and average latency for adaptive GoP size against fixed GoP size under dynamic packet loss rates. During packet loss duration, adaptive GoP video uses shorter segments to skip content more often rather than waiting for retransmissions. The adaptive GoP latency averages 151 ms compared to 168 ms for fixed GoP size, reducing the latency by 10%.

4.6 Latency-Aware Quality

We compare fixed GoP sizes under dynamic packet loss in Figure 8. We use *Big Buck Bunny* [6] with 1280×720 pixels at 24 FPS, and re-encode with target GoP sizes. VMAF scores are 72.6 and 55.7 for GoP sizes of 0.5 and 2 seconds, respectively. The shorter GoP size recovers faster than others, as it refreshes content more often during the lossy section.

In Figure 9, we compare the adaptive and fixed GoP size of 2 seconds. When network conditions are stable, adaptive and fixed GoP behave similarly. Under 30% loss, adaptive GoP size improves VMAF score by 3.65% from 41.9 to 43.5. By switching to shorter GoP sizes during congestion, adaptive GoP improves the delivery of newer segments rather than waiting for retransmissions for longer GoP sizes, and thus improves perceived video quality at the same target bitrate.

[6] Available at https://peach.blender.org

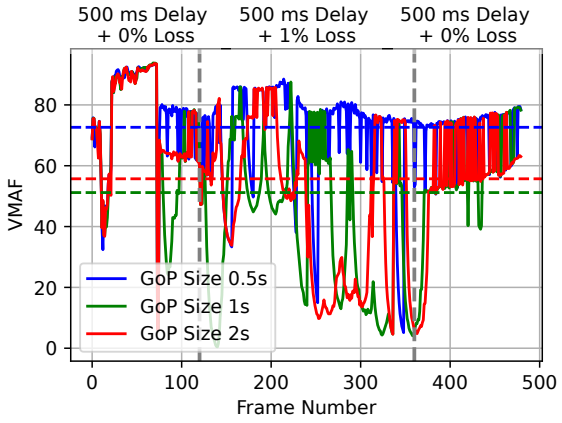

Figure 8: VMAF scores of fixed GoP sizes under 500 ms delay and dynamic 1% packet loss: Under packet loss with additional delays, the shorter GoP size of 0.5 seconds recovers quality quicker than others.

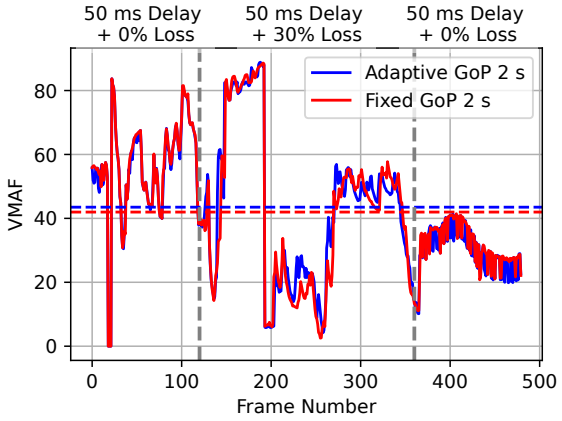

Figure 9: VMAF scores of adaptive and fixed GoP sizes under 50 ms delay and dynamic 30% packet loss: Adaptive GoP begins with a 2-second GoP size, switches to a 0.5-second GoP size under loss, and switches back to a 2-second GoP size when network conditions stabilize. Under loss, the shorter GoP size refreshes the video content more often, which increases the VMAF score.

5 Conclusion

In this paper, we introduce FrameTrace, a frame-level QoE analysis logger for MoQ. Our results show that MoQ reduces latency by 83% at 5% loss and 63% under a 10 Mbps bandwidth limitation compared to LL-DASH. We analyze the dynamic between GoP sizes and QoE, showing that shorter GoP sizes increase VMAF by up to 30% under high latencies and losses. We design a real-time GoP adaptation based on FrameTrace, which uses long GoP sizes when network conditions are stable, thereby reducing latency by up to 10% and improving VMAF by up to 3.65% under high latencies and losses.

Acknowledgments

This research has been partially funded by the German Ministry of Transport and Digital Infrastructure within the project CAPTN Förde Areal II (45DTWV08D).

References

[1] Cheng Chen, Wenpei Yin, Zhexiong Huang, and Shu Shi. 2024. AGiLE: Enhancing Adaptive GOP in Live Video Streaming. In *Proceedings of the 15th ACM Multimedia Systems Conference*. 34–44.

[2] Zafer Gurel, Tugce Erkilic Civelek, Deniz Ugur, Yigit K Erinc, and Ali C Begen. 2024. Media-over-quic transport vs. low-latency dash: a deathmatch testbed. In *Proceedings of the 15th ACM Multimedia Systems Conference*. 448–452.

[3] Craig Gutterman, Brayn Fridman, Trey Gilliland, Yusheng Hu, and Gil Zussman. 2020. Stallion: video adaptation algorithm for low-latency video streaming. In *Proceedings of the 11th ACM Multimedia Systems Conference* (Istanbul, Turkey) *(MMSys '20)*. Association for Computing Machinery, New York, NY, USA, 327–332. doi:10.1145/3339825.3397044

[4] Theo Karagkioules, Rufael Mekuria, Dirk Griffioen, and Arjen Wagenaar. 2020. Online learning for low-latency adaptive streaming. In *Proceedings of the 11th ACM Multimedia Systems Conference* (Istanbul, Turkey) *(MMSys '20)*. Association for Computing Machinery, New York, NY, USA, 315–320. doi:10.1145/3339825.3397042

[5] Xiangbo Li, Mahmoud Darwich, Mohsen Amini Salehi, and Magdy Bayoumi. 2021. Chapter Four - A survey on cloud-based video streaming services. Advances in Computers, Vol. 123. Elsevier, 193–244. doi:10.1016/bs.adcom.2021.01.003

[6] Suhas Nandakumar, Victor Vasiliev, Ian Swett, and Alan Frindell. 2025. *Media over QUIC Transport.* Internet-Draft draft-ietf-moq-transport-13. Internet Engineering Task Force. https://datatracker.ietf.org/doc/draft-ietf-moq-transport/13/ Work in Progress.

[7] Minh Nguyen, Philip Nys, Stefan Pham, Daniel Silhavy, Stefan Arbanowski, and Stephan Steglich. 2025. Streaming Face-Off: A Testbed Analysis of Media-over-QUIC and Low-Latency DASH. In *Proceedings of the 16th ACM Multimedia Systems Conference*. 317–321.

[8] Michael Scharf and Sebastian Kiesel. 2006. NXG03-5: Head-of-line Blocking in TCP and SCTP: Analysis and Measurements. In *IEEE Globecom 2006*. 1–5. doi:10.1109/GLOCOM.2006.333

[9] Iraj Sodagar. 2011. The mpeg-dash standard for multimedia streaming over the internet. *IEEE multimedia* 18, 4 (2011), 62–67.

[10] Pieter-Jan Speelmans. 2024. *HESP - High Efficiency Streaming Protocol.* Internet-Draft draft-theo-hesp-06. Internet Engineering Task Force. https://datatracker.ietf.org/doc/draft-theo-hesp/06/ Work in Progress.

[11] Thomas Stockhammer. 2011. Dynamic adaptive streaming over HTTP–standards and design principles. In *Proceedings of the second annual ACM conference on Multimedia systems*. 133–144.

[12] Yifan Wang, Minzhao Lyu, and Vijay Sivaraman. 2024. Standardizing Multimedia QoE Telemetry from Telecommunications Networks for Open Analytics. In *Proceedings of the 2024 SIGCOMM Workshop on Emerging Multimedia Systems*. 14–20.

[13] Abid Yaqoob, Ting Bi, and Gabriel-Miro Muntean. 2020. A survey on adaptive 360 video streaming: Solutions, challenges and opportunities. *IEEE Communications Surveys & Tutorials* 22, 4 (2020), 2801–2838.

[14] Bo Zhang, Thiago Teixeira, and Yuriy Reznik. 2021. Performance of low-latency HTTP-based streaming players. In *Proceedings of the 12th ACM Multimedia Systems Conference*. 356–362.