Technical Note

Evaluating Edge-Termination Buffer Effects for Real-Time AV1 over MPEG2-TS over HTTP/3 QUIC

Daisuke Sugisawa *

Posted Date: 31 December 2025

doi: 10.20944/preprints202510.1054.v2

Keywords: QUIC; MPEG-TS; edge POP; ABR; smoothed RTT; congestion control

 doi:10.20944/preprints202510.1054.v2

Technical Note

Evaluating Edge-Termination Buffer Effects for Real-Time AV1 over MPEG2-TS over HTTP/3 QUIC

Daisuke Sugisawa

Xander, LLC. Shibuya, Tokyo, Japan; daisuke.sugisawa.xander@gmail.com

Abstract

In real-time delivery of AV1 over MPEG2-TS using HTTP/3 over QUIC, two fundamental approaches can be identified for improving video quality. The first is bitrate adaptation, represented by Adaptive Bitrate (ABR), which estimates available bandwidth and selects the optimal resolution and frame rate within that range. The second approach focuses on low-latency control, reducing RTT and ACK delay to increase effective throughput. This study emphasizes the latter perspective and, through terminal-side QUIC observation, suggests that user-space buffer saturation occurring in high-RTT mobile environments under nginx-quic HTTP/3 termination implementation is an implementation-originated problem that may degrade real-time video QoS. Furthermore, this study aims to evaluate, through empirical measurements, the feasibility of edge designs that enable packet-level processing (duplication, FEC, pacing) at QUIC termination points near the terminal. Through empirical experiments conducted across domestic and international cloud environments, we evaluate how buffer design at distribution servers affects video quality. This paper shares experimental methods, observations, and guidelines for designing real-time streaming systems using HTTP/3 over QUIC.

Keywords: QUIC; MPEG-TS; edge POP; ABR; smoothed RTT; congestion control

1. Introduction

Research on real-time video delivery using HTTP/3 over QUIC is progressing [2–5], but the influence of server-side buffer design on video quality has not been sufficiently examined. In latency-sensitive environments, buffering can introduce delays and packet loss, degrading quality [14]. While ABR [17] is widely adopted to match bitrate to available bandwidth, reducing RTT and ACK delay can also increase usable throughput. This behavior is closely tied to buffer operation at distribution servers, potentially compensating for impacts beyond ABR's scope. (see Figure 3, and Figure 4).

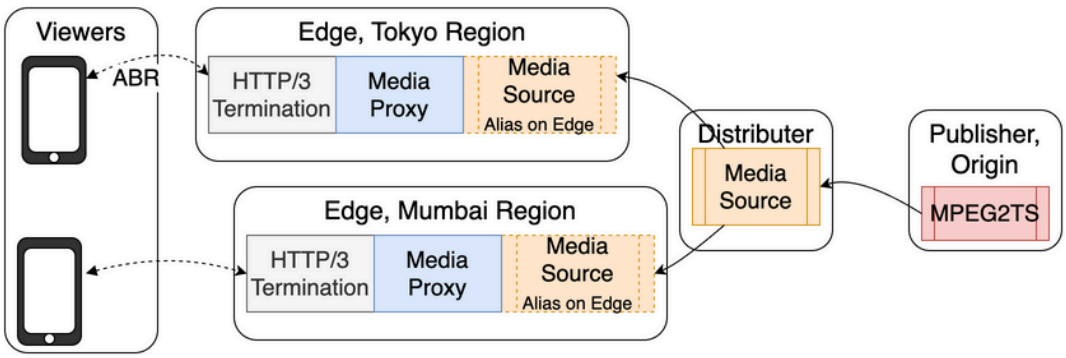

Figure 1. System Overview.

This study aims to verify, through empirical measurements in real network environments, the implicit assumption that short-term buffering at HTTP/3 termination points improves playback continuity in real-time video streaming.

 doi:10.20944/preprints202510.1054.v2

We implemented an AV1 [6,7] over MPEG2-TS streaming system [1] using HTTP/3 over QUIC with ngtcp2 [8], nghttp3 [9], libev [10], and OpenSSL [11]. Cross-region experiments were conducted to analyze the effects of buffer design on quality.

2. System Architecture

Modules and requirements are summarized in Table 1 , while application-level requirements are listed in Table 2

Table 1. Module Specifications.

Video Stream Origin	
Input	2K 30fps(FHD)
Encode	AV1(NVENC)
Output	CBR/ABR (1Mbps, 500Kbps, 200Kbps), multi-rate MPEG-TS Stream
Edge Termination, nginx or CloudFront(sec A.1)	
HTTP3	HTTP3-Endpoints, Backend HTTP2 Proxy
Edge-Module, UDP-MPEGTS to HTTP3-Chunked-Data(sec A.2)	
Video Recieve	MPEG-TS packets received over UDP(BSD socket)
Video Transfer	MPEG-TS packets Transfer, Chunked-Data
Video Sync	MPEG-TS Packet-level fanout frame synchronization
AV1 over MPEG2-TS Realtime Stream Player	
HTTP3 over QUIC	HTTP3 over QUIC, Session establishment, ngtcp2, nghttp3
Chunked Parse	Stream data Receiving, MPEG-TS packet parsed into AV1
ABR Control	Client-side optimization based on RTT, Traffic, and Drop statistics.

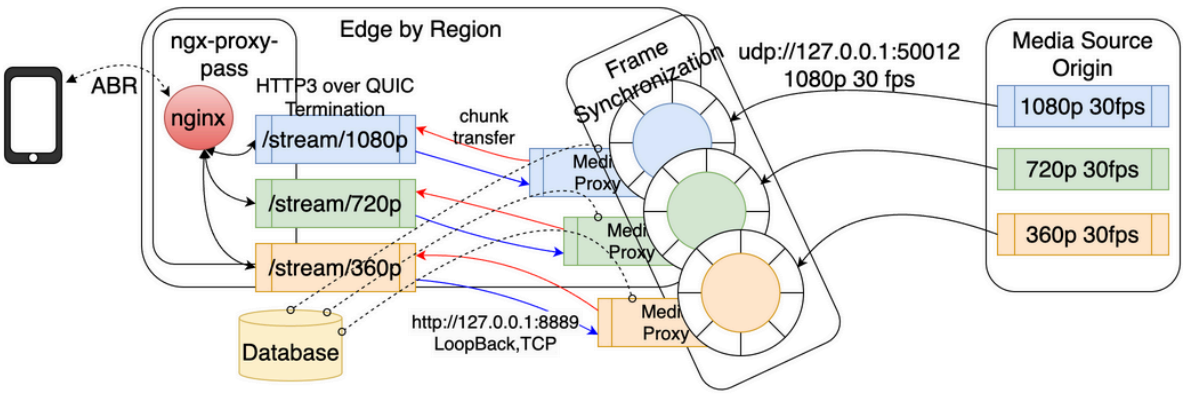

Figure 2. Video Stream System Architecture.

 doi:10.20944/preprints202510.1054.v2

Table 2. Application Requirements.

Technical	Description
HTTP3 over QUIC	Firewall, NAT-friendly UDP protocol
MPEG-TS over HTTP3	Continuous streaming without segmentation
AV1 over MPEG-TS	High compression, low bandwidth, high image quality
Client-sid ABR	Client-side session switching control, last-one-mile optimization
Stateless Edge CDN	A scalable, distributed configuration without session information
FEC	Assign FEC packets to user-defined PIDs
API Endpoint	CloudFront (Edge POP, QUIC termination) + Lambda functions in each region
Stream Endpoint	nginx (QUIC termination) + stream pipeline in each region

2.1. Rationale for Adopting MPEG-TS

The reasons for adopting MPEG-TS in this study are the ease of packet-level shaping and the ability to insert custom metadata and FEC packets using user-defined PIDs.

MPEG-TS has a fixed-length packet structure of 188 bytes, enabling packet-level operations such as duplication, redundancy, and pacing at edge termination points without disrupting the stream structure. This allows the edge-intervention-based QoS control evaluated in this study to be implemented simply and with high reproducibility.

3. Evaluation Experiments

Experiments were performed on AWS CloudFront and Lambda (Tokyo, Mumbai) for API traffic, and EC2 instances (Tokyo, Mumbai) for streaming tests.

3.1. Stream, HTTP/3 Neighbor - Far Comparison

AV1 encapsulation in MPEG TS is not yet standardized, requiring trial-and-error for parser implementation. Implementation differences in HTTP/3 over QUIC (such as ngx_http3_module eBPF implementations) resulted in inconsistent connectivity and stability of HTTP/3 over QUIC termination modules.

Listing 1: packet capture at Neighbor docker

```
tcpdump -n tcp port 8080  -i docker0 -c 5000 -w /tmp/neighbor-docker-upstream.pcap
```

Figure 3 and Figure 4 visualize and analyze the first 5000 received packets when requesting the same media origin over an HTTP3 over QUIC session.

 doi:10.20944/preprints202510.1054.v2

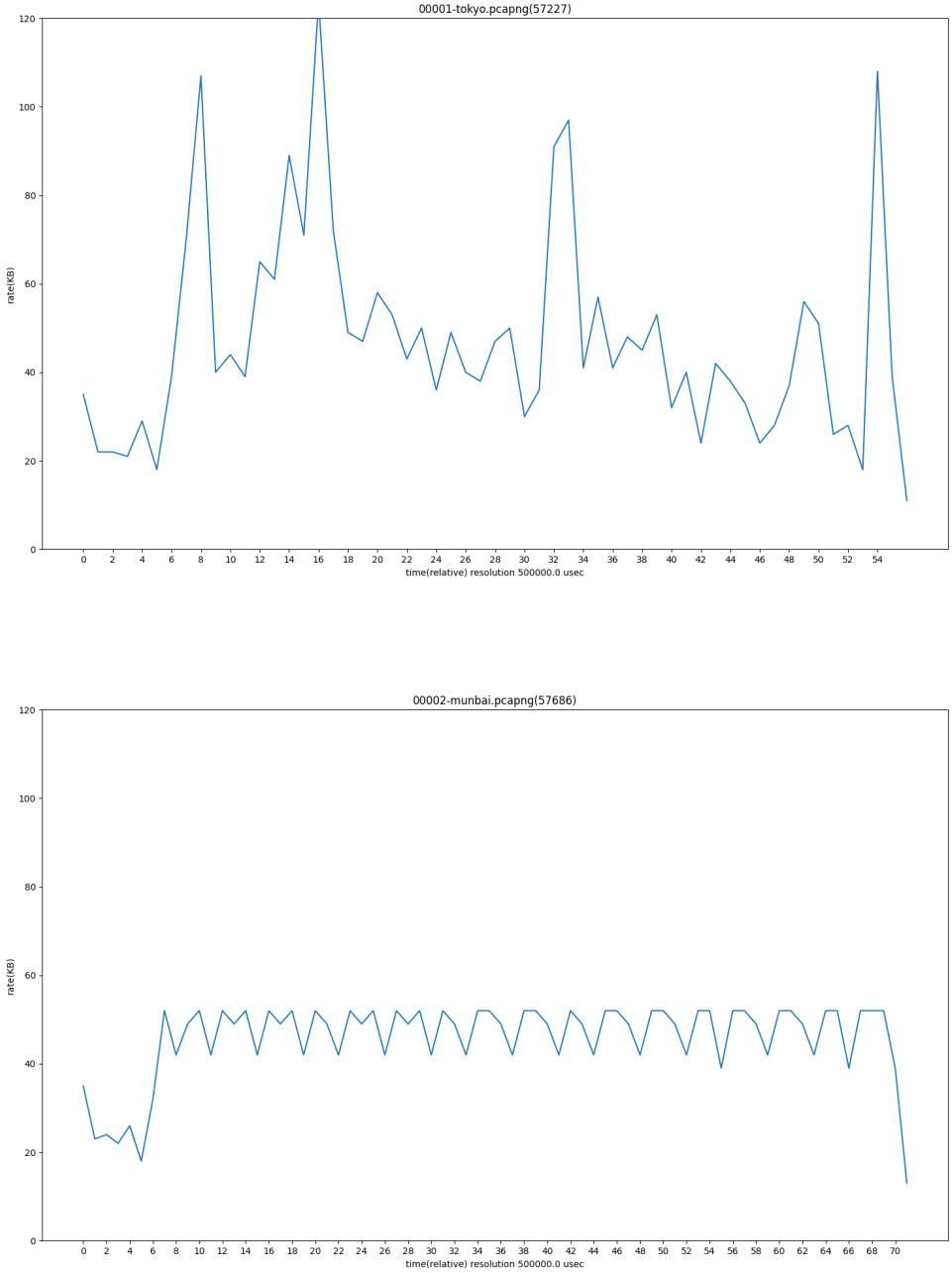

Figure 3. Packet Analysis Statistics: Neighbor (Tokyo, up), Far (Mumbai, down) CUBIC.

 doi:10.20944/preprints202510.1054.v2

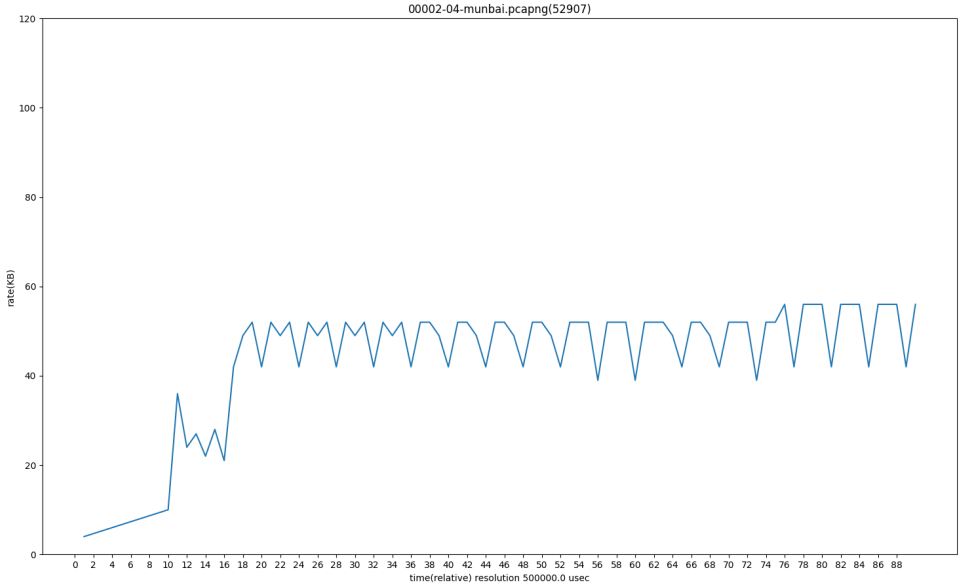

Figure 4. Packet Analysis Statistics: Far (Mumbai) BBR, proxy_buffering on.

3.2. Edge-Termination Proximity

Comparisons between neighbor (Tokyo) and distant (Mumbai) terminations revealed significant RTT effects. As shown in Figures 3, 4, 5, Smoothed RTT in Mumbai (133 ms) led to degraded quality (5 FPS), with nginx internal buffer analysis (Table 3) suggesting that ACK delay and silent packet drops may be contributing factors.

 doi:10.20944/preprints202510.1054.v2

Figure 5. Packet Analysis Statistics: Far(Mumbai,up),Neighbor(Tokyo, down)Origin to Docker.

We analyzed the downstream TCP packets (port 8080) destined for nginx-quic within the Docker container (Figure 5) and confirmed no significant difference. This narrowed the root cause to buffer processing within the HTTP/3 termination module (ngx_http3_module).

 doi:10.20944/preprints202510.1054.v2

Table 3. nginx analyze.

TCP upstream nginx receive buffer (nginx 3739fe94d1c3c844708b219776f1921bff16b56f)	
ngx_event_connect_peer()	Arrival at the BSD socket buffer
nginx internal HTTP3, HTTP write filter transmission control	
ngx_http_write_filter_module.c L:299	send_chain()
ngx_event_quic_streams.c L:952	ngx_quic_stream_send_chain()
	flow= qs.acked + qc.conf.stream_buffer_size - qs.sent;
flow formula	Available-Buffer= (ACKed bytes + Buffer size) - Sent bytes
	In high RTT environments, qs.acked updates are delayed by 133ms.
	When flow == 0, immediately return at lines 976-978 and skip transmission.

Analysis and status of Figure 3, Figure 4, and Figure 5 Based on detailed packet analysis at the TCP layer, this suggests that temporary ACK delays and spike traffic-induced buffer overflows, and the resulting server-side (ngx_http3_module) silent packet drops are likely contributing factors to degraded video quality (Table 3)

Listing 2: Viewer QUIC Settings(CUBIC)

```
ngtcp2_settings settings;
ngtcp2_settings_default(&settings);
settings.cc_algo → NGTCP2_CC_ALGO_CUBIC;
```

We compared the effects of congestion control differences between CUBIC (Figure 3) and BBRv2 (Figure 4) [12]. No significant difference in video quality was observed under the measurement conditions of this study. This is mainly because the edge buffer was frequently saturated, leading to transmission stalls (flow=0 condition). As a result, the potential differences in congestion control behavior at the terminal side were masked by these buffer-induced stalls at higher layers. This finding is consistent with prior reports indicating that buffer occupancy can dominate system behavior, particularly during RTT spikes and retransmissions [14].

3.3. Stream, Neighbor–Distant Origin Comparison

Even when there is a physical distance between the regional edge and the media origin, we observed that traffic between the origin and the edge remains consistently stable. This stability is achieved through both explicit and implicit buffering — for example, explicit buffering introduced by QUIC termination close to the edge, and implicit buffering provided by proxy modules and similar mechanisms.

 doi:10.20944/preprints202510.1054.v2

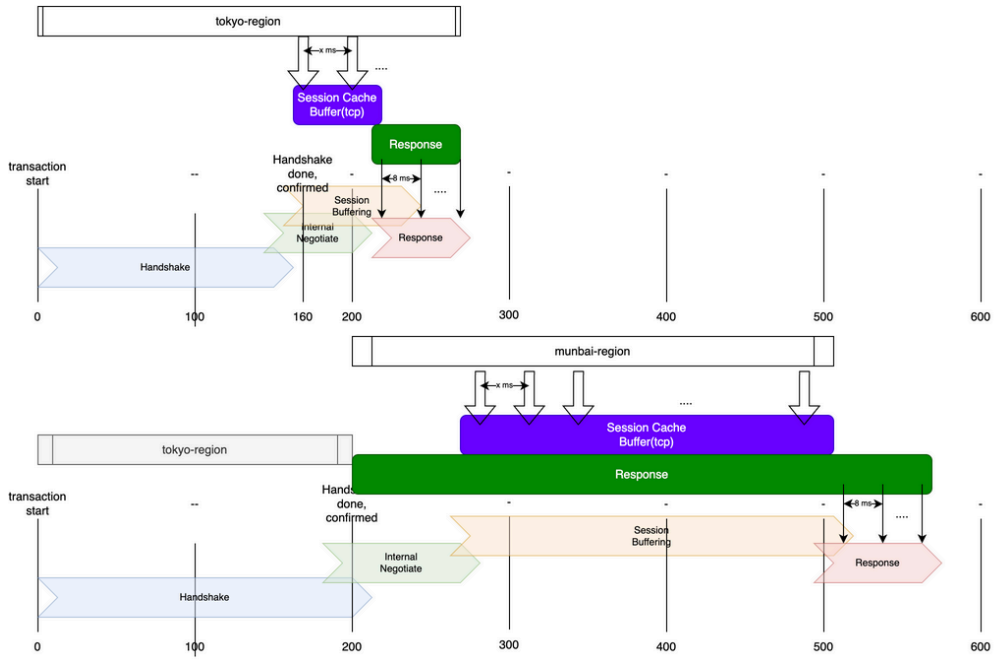

Figure 6. Neighbor, Distant Origin.

3.4. Last-One-Mile Mobile Network Effects

Tests conducted under stressed RAN conditions—specifically at Tokyo-Shibuya Station during the evening rush hour—across three major Japanese carriers revealed variation in packet loss rates and ACK ratios across carriers. However, when the data were analyzed in terms of cumulative packet counts and payload statistics (Figure 7, Figure 8, Figure 9, Figure 10, Table 5, Table 4), the results showed convergence, indicating that aggregate behavior was largely consistent despite carrier-level differences.

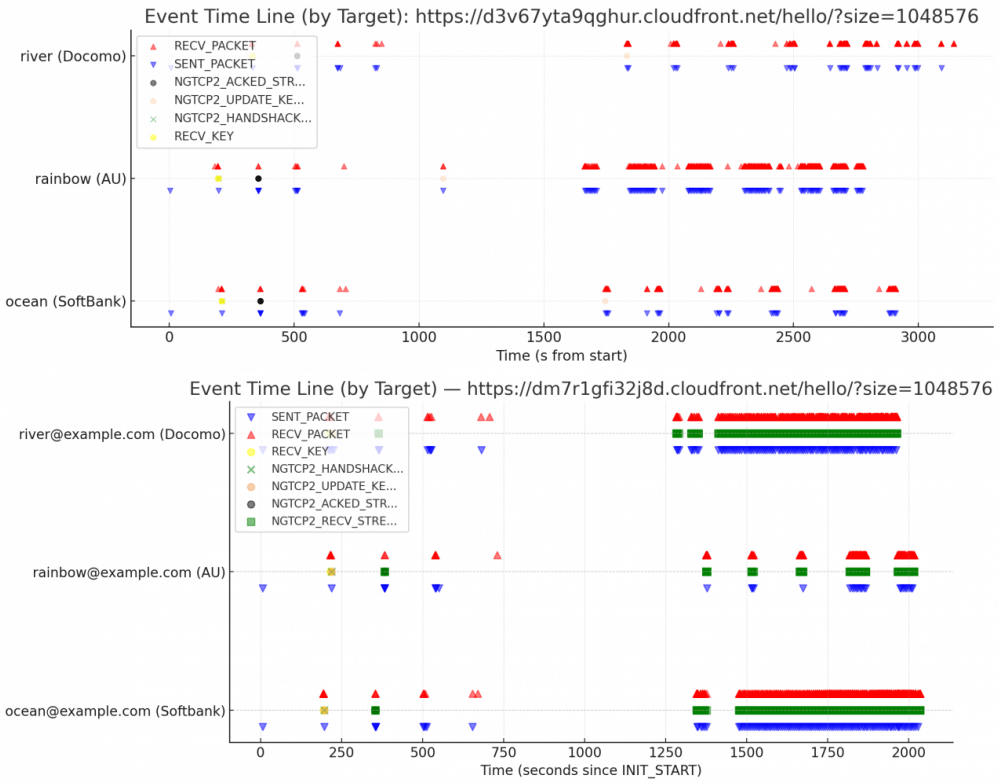

Figure 7. Event Analysis: Far (Mumbai,up),Neighbor (Tokyo, down), 1MB-DL HighBandwidth office-RAN.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 31 December 2025 doi:10.20944/preprints202510.1054.v2

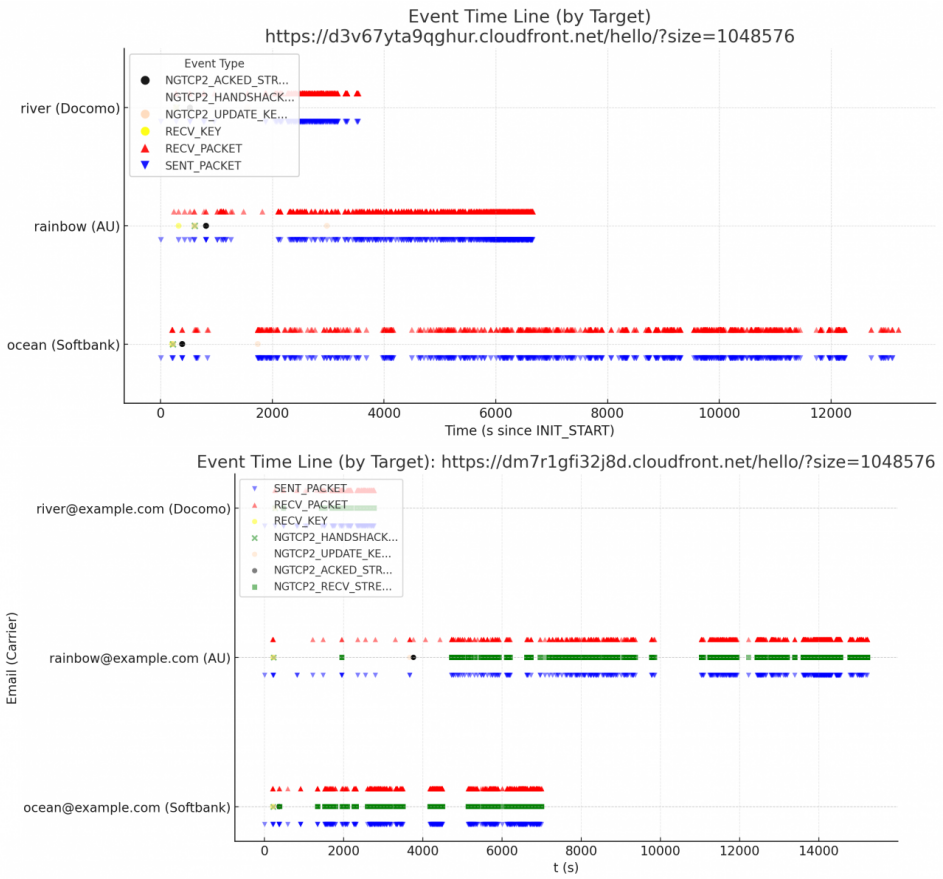

Figure 8. Event Analysis: Far(Mumbai,up),Neighbor(Tokyo, down), 1MB-DL Public-RAN.

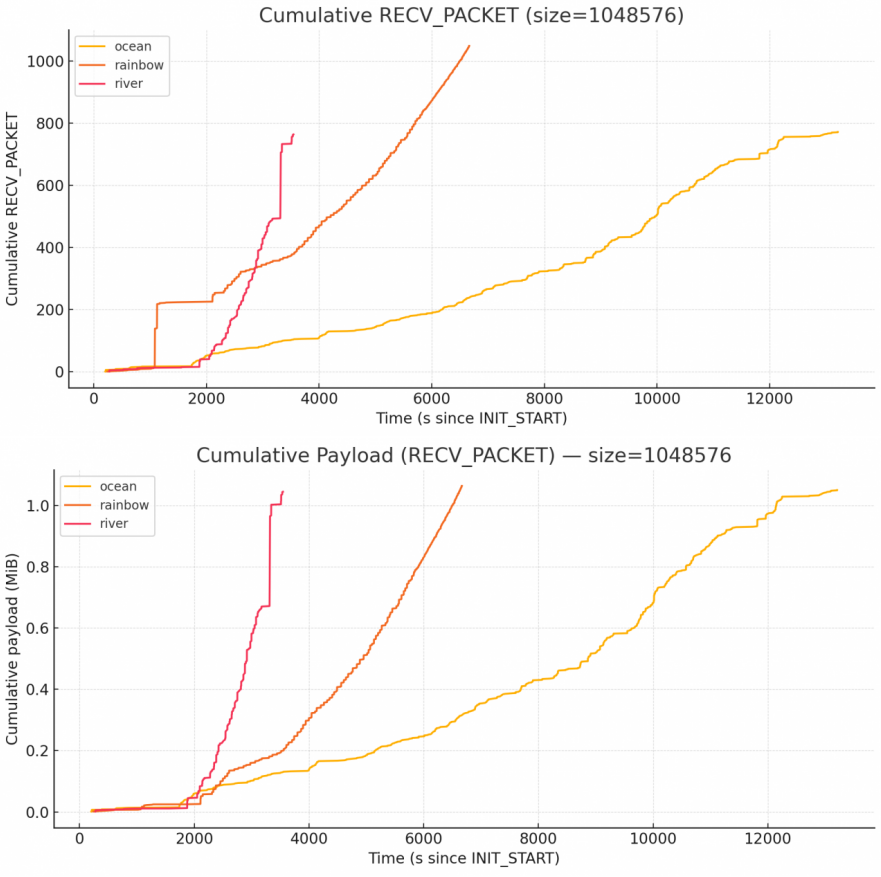

Figure 9. Event Analysis 1MB-DL, Receive: packet-count, payload-size, cumulative.

 doi:10.20944/preprints202510.1054.v2

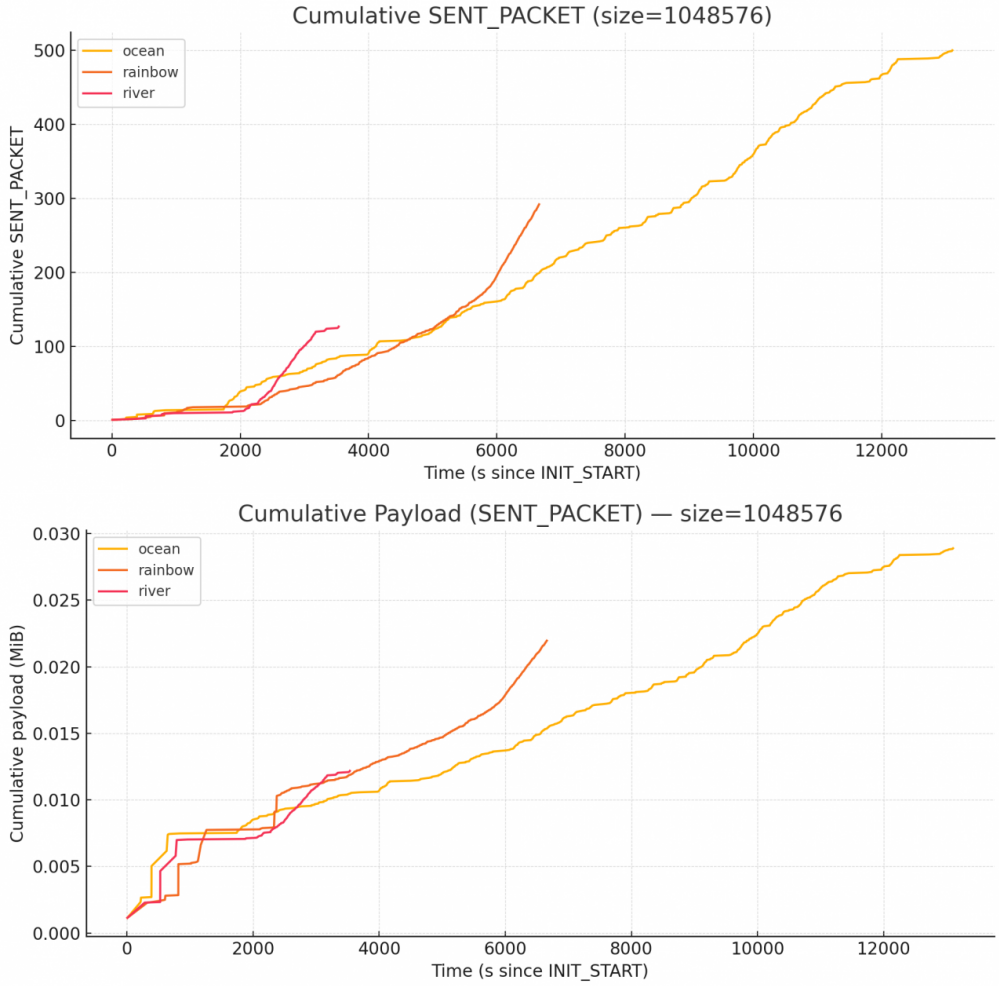

Figure 10. Event Analysis 1MB-DL, Send: packet-count, payload-size, cumulative.

Table 4. Event Details: Far (Mumbai) 1MB-DL.

Event	MNO(A)	MNO(B)	MNO(C)
	Far(Mumbai) Public-RAN		
Drops NGTCP2_STREAM _LOSS_COUNT	0	1	0
ACK Ratio SENT_PACKET/ RECV_PACKET	127/764 16.6 %	292/1049 7.8 %	500/772 64.7 %
MNO(A) MNO(C)	Almost uninterrupted continuous transmission and receive (transmission consists only of ACKS) Almost no gaps Streaming-like		
MNO(B)	Burst+Intermittent Bulk processing		
Resources RAN	Total timeCost Same all carriers		

 doi:10.20944/preprints202510.1054.v2

Table 5. Event Details: Neighbor (Tokyo) 1MB-DL.

Event	MNO(A)	MNO(B)	MNO(C)
	Neighbor(Tokyo) Public-RAN		
Drops NGTCP2_STREAM _LOSS_COUNT	0	1	0
ACK Ratio SENT_PACKET/ RECV_PACKET	71/765 9.3 %	345/839 41.1 %	324/769 42.1 %
Received Packet Interval NGTCP2_RECV _STREAM	p95 6.5 p99 45.2	p95 71.1 p99 277	p95 23.3 p99 137
MNO(A)	No problem		
MNO(C)	The unusually high frequency of ACK packets and the large gaps observed in packet reception intervals suggest that retransmissions were triggered by factors other than actual packet loss.		
MNO(B)	NGTCP2_STREAM_LOSS_ COUNT==1 Drop retransmission was detected		

In mobile networks, congestion typically arises from fluctuations in available radio capacity and from new users joining the cell, not solely from traffic spikes as in fixed networks.

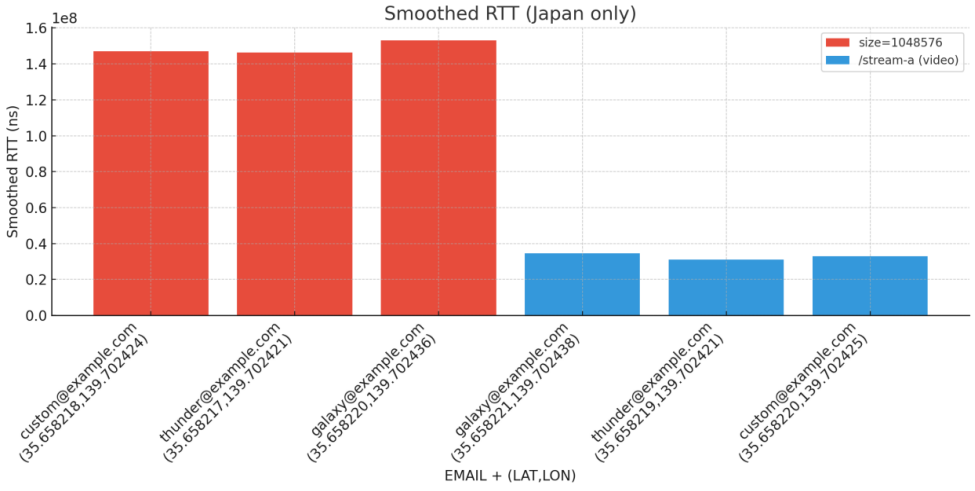

Figure 11. Smoothed RTT (Japan)

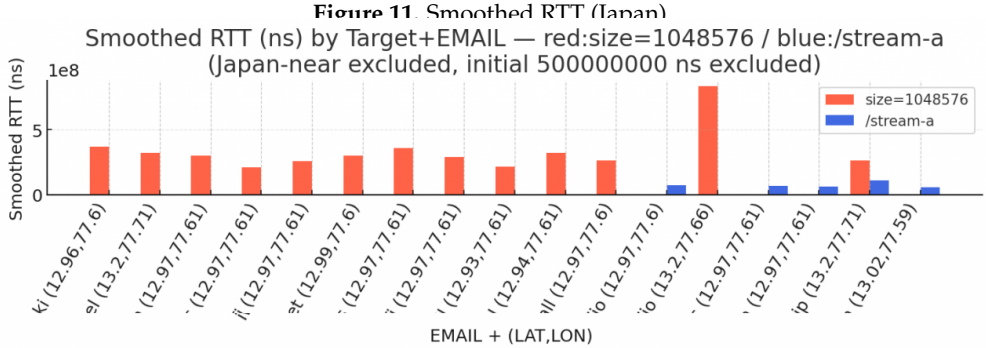

Figure 12. Smoothed RTT (Mumbai)

 doi:10.20944/preprints202510.1054.v2

The video stream in this study has a low bitrate of CBR 1Mbps. However, on the last-mile path (mobile network), buffer saturation can occur temporarily due to channel allocation fluctuations caused by UE joining or leaving the cell. When this occurs, the Smoothed RTT observed at the terminal increases. By leveraging this phenomenon, the effective bandwidth limit for each session can be estimated from Smoothed RTT measurements at the terminal.

Furthermore, in Tokyo and Mumbai, comparisons between 1 Mbps stream-based CBR traffic and 1 MB API downloads confirmed that Smoothed RTT for stream-based reception remained consistently lower. This held true even under differing conditions of mobile core bandwidth, RAN technologies (e.g., TD), and congestion control methods. This result suggests that continuous low-bitrate streams (application/octet-stream) are less likely to saturate last-mile buffers compared to bursty API calls (application/json GET). While bursty downloads temporarily occupy buffers and cause RTT spikes, CBR streams minimize buffer occupancy through uniform packet intervals, thereby maintaining stable RTT characteristics.

These findings indicate that, despite diverse end-to-end influences such as mobile core variations and RAN channel allocation, application-layer priority-based FIFO control and buffering strategies exert a direct and significant impact on QoS. In other words, beyond the inherent complexity of wireless and core networks, control design at the application layer emerges as a decisive factor for effective QoS.

Based on empirical observations from this study, we propose the following heuristic for estimating the optimal bitrate for each session using the Smoothed RTT observable at the terminal. RTT_{min} represents the ideal network state (buffer unsaturated), while RTT_{smoothed} reflects the current state including buffer delays. The ratio $RTT_{\text{min}}/RTT_{\text{smoothed}}$ serves as an indicator of current network degradation. By multiplying this ratio by the current receive rate S_{meas} and applying a margin η for spikes, we can estimate the optimal bitrate that avoids buffer saturation:

$\eta \in [0.8, 0.95]$	Spike coefficient
S_{meas}	Current Receive Rate (bps)
RTT_{min}	`NGTCP2_CONN_INFO_MIN_RTT` initial value (excluding $5e8$) Ideal network state (buffer unsaturated)
RTT_{smoothed}	`NGTCP2_CONN_INFO_SOOTHED_RTT` Current state including buffer delays

$$BR_{\text{opt}} \approx \eta \cdot S_{\text{meas}} \cdot \frac{RTT_{\text{min}}}{RTT_{\text{smoothed}}} \qquad \text{(Optimal Adaptive Bitrate per Session)}$$

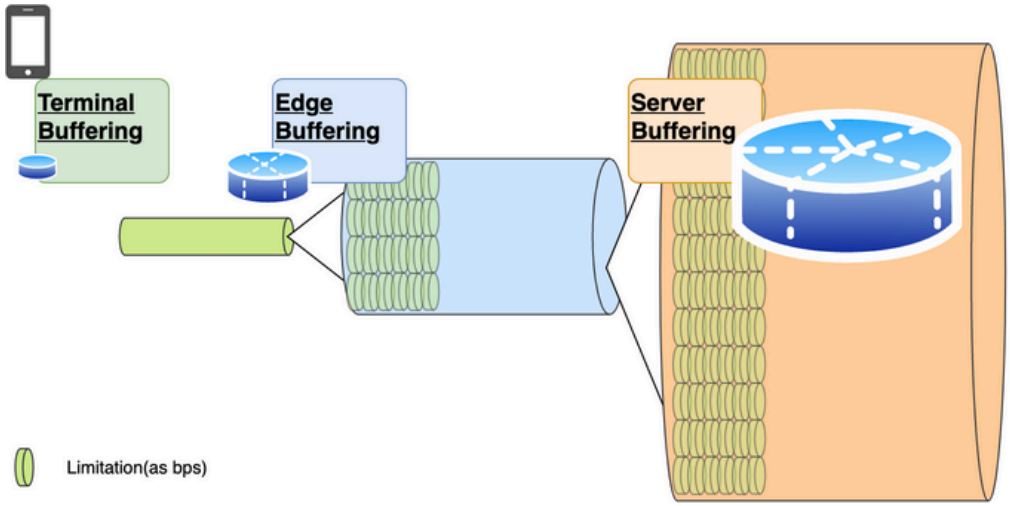

Figure 13. Asymmetric Bandwidth and Buffering.

 doi:10.20944/preprints202510.1054.v2

3.5. Compatibility of Asymmetric Bandwidth/Buffering with Caching Strategies

In mobile networks, the uplink (terminal to base station) and downlink (base station to terminal) bandwidths are asymmetric. Furthermore, the available bandwidth and buffer sizes differ significantly between the last-mile segment (edge node to terminal) and the backhaul segment (origin to edge node) (Figure 13). This asymmetry affects file-segment caching and packet-level caching differently.

3.5.1. File-Segment Caching

- **Bursty Segment Retrieval:** HLS and DASH retrieve multi-second segments in bulk, causing instantaneous bandwidth spikes during acquisition. In asymmetric buffering environments, these spikes readily cause buffer overflow in bottleneck segments.
- **Buffer Size Mismatch:** The mismatch between large-capacity buffers on the origin-edge path and small-capacity buffers on the edge-terminal path causes congestion concentration in the last mile during segment-level cache transfers.
- **Cumulative ACK Delays:** Waiting for complete segment transfers causes ACK delays to accumulate in high-RTT environments, delaying the start of subsequent segment retrieval. This leads to playback stalls (rebuffering).
- **Difficult Bandwidth Estimation:** ABR bandwidth estimation tends to be inaccurate in asymmetric environments, leading to either buffer exhaustion from over-estimated bitrate selection or quality degradation from under-estimation.

3.5.2. Packet-Level Caching

- **Fine-Grained Traffic Smoothing:** Packet-by-packet transfer distributes bandwidth usage over time. This makes buffer overflow less likely even in segments with small buffer capacity.
- **Adaptive Flow Control:** Per-packet ACK responses enable real-time adaptation to the effective bandwidth of bottleneck segments. Transmission rates automatically adjust to asymmetric buffer sizes.
- **Delay Minimization:** Short-lived caching via ring buffers minimizes delay from packet arrival to terminal delivery. Since there is no need to wait for complete segment accumulation, low latency is maintained even in asymmetric environments.
- **Precise Control via Smoothed RTT:** Terminal-side Smoothed RTT measurement enables accurate characterization of the effective bandwidth and delay properties of asymmetric last-mile segments, allowing optimal adaptive bitrate calculation.

Thus, in asymmetric bandwidth and buffering environments, file-segment caching faces challenges for low-latency delivery due to its bursty traffic patterns, although segment-level cache efficiency works effectively for VOD and time-shift viewing. In contrast, packet-level caching has high compatibility with asymmetric environments for real-time delivery due to its fine-grained, smoothed traffic patterns. The silent drop issues caused by buffer saturation in high-RTT environments at edge termination points, as observed in this study, are largely attributable to this asymmetry.

4. Discussion: Design Trade-offs between HLS and Packet-Level Caching

File-segment caching employed in HLS and related systems, and the packet-level caching in this study, have fundamentally different design objectives.

HLS is primarily intended for VOD (Video on Demand) and time-shift viewing, providing rewind, fast-forward, and archive functionality through segment-level storage retention. In contrast, the architecture in this study is specialized for ultra-low-latency real-time delivery with emphasis on playback continuity, maximizing fanout efficiency to viewer terminals by using ring buffers with only a few seconds of capacity. The two are not substitutes for each other, but represent design trade-offs according to use case.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 31 December 2025 doi:10.20944/preprints202510.1054.v2

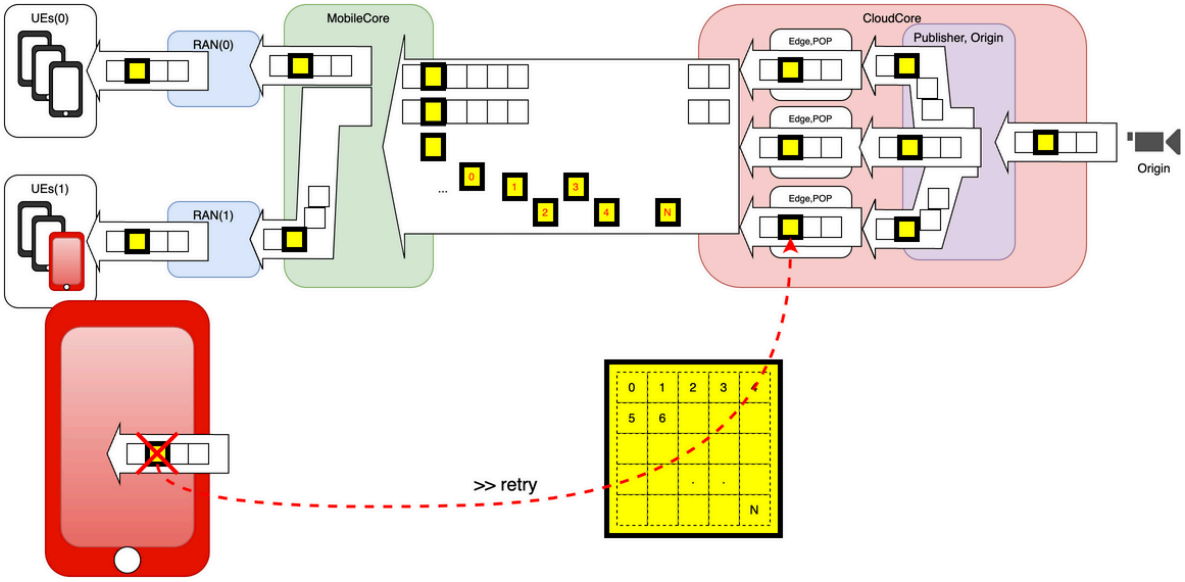

Figure 14. Conventional File Caching: System Overview.

As illustrated in Figure 2, Figure 15, the packet-level caching approach goes beyond merely functioning as relay nodes. It has potential as a foundational technology for achieving inter-terminal synchronization at the video frame level. In particular, by enabling playback synchronization at the unit of packet caches across terminals, it becomes possible to construct a low-latency, distributed live streaming system. Such a design is expected to contribute to scalable real-time delivery in edge environments and mobile networks [18].

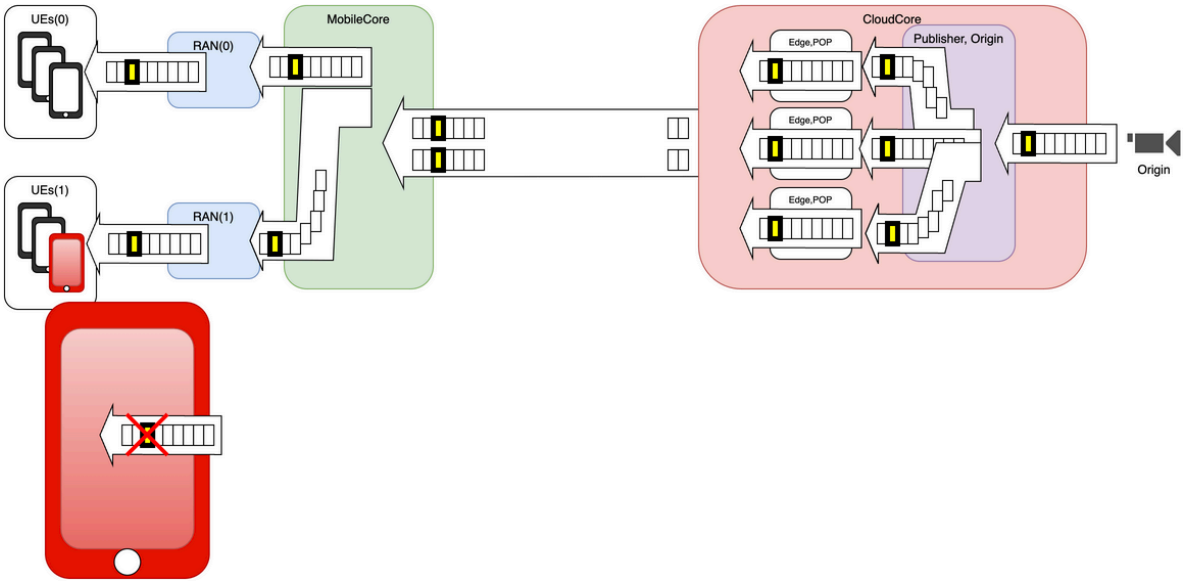

Figure 15. This Study's Packet Caching: System Overview.

 doi:10.20944/preprints202510.1054.v2

Table 6. POP Edge - Origin Alias Design.

Item	Overview
Packet Granularity Cache	POP Edge maintains MPEG-TS packets in short-lived rings fan-out to concurrent viewwers
ACK Delay	Reduce ACK round trips at edge-near terminations to suppress transmission blocking and session buffer exhaustion at HTTP/3 endpoints
Origin Bandwidth Aggregation Efficiency	Maximize Edge Hit Rate (Fanout) to Minimize Origin Egress Bandwidth
Design Trade-off	Purpose differs from HLS (VOD/time-shift oriented)

Listing 3: Conventional File Segment Cache (Supports HLS)

```
[Origin] → [Storage] → [CDN Cache] → [Viewer]
        ↑
   Retained for several hours-several days
    + Time-shift viewing
    + Rewind/Fast forward
    + Archiving
```

Table 7. Conventional File Segment Cache.

Item	Description
Cache Hit Rate	70 %...from [15] Hit rate linear approximation is $(1-1/C)$ C=Number of concurrent viewers per 100 (per 1POP) Approximately 70 - 90 %
Cache Hit Latency	200 µs
Cache miss Latency	200 ms

Listing 4: This Paper Packet Cache Fanout Ring Buffer

```
[Origin] → [Memory Ring] → [Viewer]
                ↑
     Only a few seconds
    + Live Streaming
    + Buffer reuse
    + Maximized memory efficiency(Band-width)
```

Table 8. This Paper: Packet Cache Fanout.

Item	Summary
Cache Hit Rate	95 % (under the evaluated workload)
Cache Hit EndToEnd Latency	200 µs
Cache Miss EndToEnd Latency	500 µs

 doi:10.20944/preprints202510.1054.v2

4.1. Typical Access Cost [16]

The above equations provide a model for quantitatively evaluating the effect of file-segment caching. As the cache hit rate h increases, the low-latency t_{hit} becomes dominant, reducing the total latency $\mathbb{E}[T]$. For conventional file-segment caching, based on cache architecture and bus width, $t_{\text{hit}} = 200\,\mu$s and $t_{\text{miss}} = 200\,$ms are logically estimated.

Cache Level	Latency	µs Equivalent
L1	4-5 cycles	0.002 µs
L2 Hit	12-15 cycles	0.005 µs
L3 Hit	40-50 cycles	0.017 µs
Main Memory,L2 miss	200-300 cycles	0.100 µs
Redis, memchached, Main Memory miss Includes kernel Protocol stack	xxxx cycles	2-10 ms

S	Number of sessions
P_s	Session s(Packets/Request)
h	Cache Hit Rate
$t_{\text{hit}}, t_{\text{miss}}$	Latency for hits/misses
L	Latency per Packet

$$\mathbb{E}[L] = h \cdot t_{\text{hit}} + (1-h) \cdot t_{\text{miss}} \qquad \text{(Expected Latency per packet)}$$

$$P_n = \Sigma_{s=1}^{S} P_s \qquad \text{(Total Packets)}$$

$$\mathbb{E}[T] = P_n\,\mathbb{E}[L] = P_n\,\left(h \cdot t_{\text{hit}} + (1-h) \cdot t_{\text{miss}}\right) \qquad \text{(Total Latency)}$$

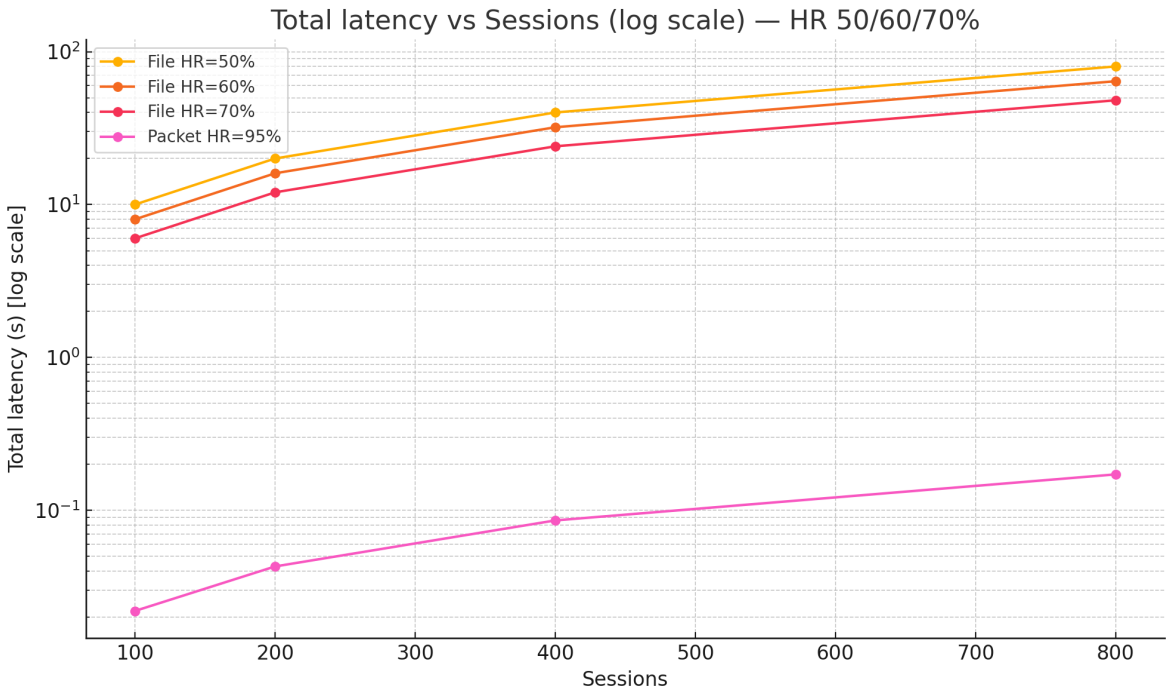

Figure 16. Total Latency (Logical).

 doi:10.20944/preprints202510.1054.v2

In contrast, for the packet-level caching in this study, the request model of "requesting a specific packet" does not exist. Packets flow continuously into the ring buffer, and viewers receive this stream. When packet loss occurs (sequence number skip), the frame containing that packet is excluded from decoding for video, and for audio, only packets hitting the OPUS 20ms boundary are excluded. In other words, rather than waiting for re-retrieval due to cache miss, playback continuity is prioritized through immediate discard of lost packet frames.

5. Discussion: Comparison with Media over QUIC Lite Under the Premises of This Study

Media over QUIC (MoQ) is an upper-layer protocol built on QUIC. In the case of Media over QUIC Lite [13], the relay component is specialized for publish–subscribe routing optimization. It is explicitly designed without last-one-mile optimization mechanisms such as cache retention, which are traditionally provided by CDNs. Accordingly, Media over QUIC Lite aims to enable cloud-native, large-scale distributed routing rather than to maximize QoS. Its objectives and scope therefore differ fundamentally from the unidirectional, real-time video streaming applications considered in this study.

6. Discussion: AV1 over MPEG-TS Real-Time Stream Player in the Context of This Study

Since neither ffplay nor standard web players (HTML5 video, MediaSource Extensions) support the AV1 + MPEG-TS combination—which is itself uncommon—this study implemented a dedicated AV1 + MPEG-TS viewer with built-in analysis functions, available across macOS, iOS, and Android platforms. Beyond playback, the implementation facilitates detailed QoS monitoring and packet-level trace analysis, making it a practical tool for evaluating real-time performance and diagnosing issues in diverse network conditions.

6.1. iOS Application Research Objectives and Key Findings

The primary objectives of implementing a native iOS application with integrated QUIC observation capabilities were as follows:

- **Terminal-Side RTT Estimation:** Enable real-time estimation of Smoothed RTT at the application layer, which is not accessible through standard iOS networking APIs.
- **Last-Mile Visibility:** Capture QUIC events (handshake, ACK, congestion window changes) directly on the device to diagnose issues invisible to server-side logging.
- **Carrier-Specific Behavior Analysis:** Quantify differences in ACK response characteristics and packet loss patterns across mobile network operators (MNOs) under identical application conditions.
- **Validation of Buffer Assumptions:** Empirically test whether edge-side buffering improves or degrades playback continuity under real mobile RAN conditions.

Key findings from the iOS application research include:

1. **Buffering Does Not Always Improve Continuity:** Contrary to the implicit assumption, short-term buffering at HTTP/3 termination points did not consistently improve playback continuity in mobile environments. In high-RTT or congested RAN conditions, buffering induced retransmission bursts and intermittent playback stalls.
2. **Carrier-Level ACK Ratio Variability:** ACK ratios varied significantly across carriers (16.6% to 64.7%), suggesting that radio resource allocation policies differ substantially between MNOs. Higher ACK ratios correlated with reduced transmission efficiency.
3. **Terminal-Side Observation is Essential:** Server-side qlog analysis alone could not identify the root causes of quality degradation. Terminal-side observation enabled detection of silent drops and buffer exhaustion events within the HTTP/3 termination module.

 doi:10.20944/preprints202510.1054.v2

4. **Smoothed RTT as a Quality Indicator:** Continuous monitoring of Smoothed RTT provided a reliable proxy for detecting congestion and predicting quality degradation before visible playback issues occurred.

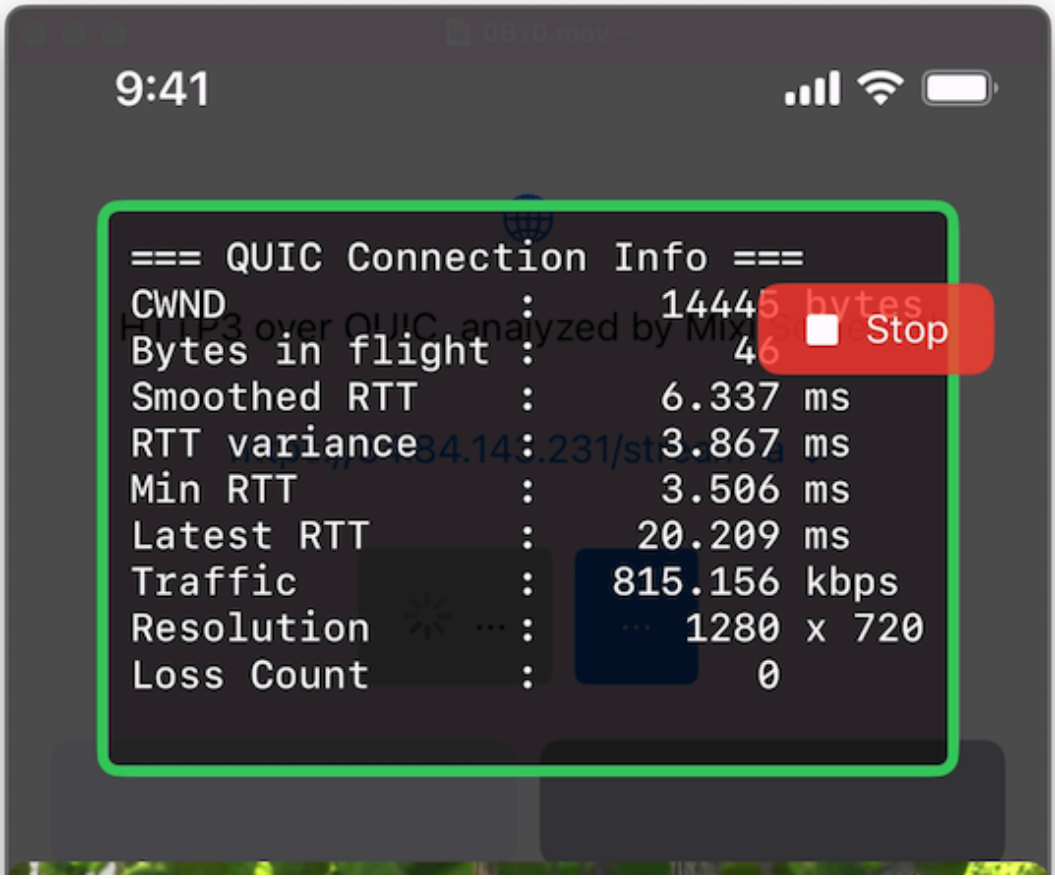

Figure 17. AV1 over MPEG-TS Player iOS (PoC).

The HTTP/3 over QUIC combined with AV1 over MPEG-TS configuration demonstrated the potential for cloud-native, scalable video delivery. Looking ahead, commercial-grade players and origin servers may emerge as cloud infrastructure continues to extend toward the edge.

7. Discussion: Effectiveness of Packet Duplication-Based Edge Computing

The empirical findings of this study suggest the effectiveness of an approach that combines packet duplication with edge computing to improve video quality while suppressing traffic spikes. This approach addresses the fundamental limitation identified in our experiments: that short-term buffering at HTTP/3 termination points can degrade rather than improve playback continuity under mobile network conditions.

7.1. Traffic Spike Suppression Through Proactive Packet Duplication

A primary contributing factor to the quality degradation observed in our experiments was the interaction between edge buffer exhaustion and retransmission bursts. When the buffer becomes saturated, silent packet drops occur, triggering retransmissions that further exacerbate congestion. A proactive packet duplication strategy may be effective in mitigating this behavior:

- **Selective Redundancy:** Rather than duplicating all packets, apply redundancy selectively to keyframes (I-frames) and packets identified as high-priority based on MPEG-TS PID classification. This minimizes bandwidth overhead while protecting critical video data.

 doi:10.20944/preprints202510.1054.v2

- **RTT-Adaptive Duplication:** Adjust the duplication factor dynamically based on Smoothed RTT measurements. When RTT exceeds a threshold (e.g., 100 ms), increase redundancy to preempt retransmission delays.
- **Traffic Smoothing:** By proactively sending redundant packets during low-traffic intervals, the overall traffic pattern becomes smoother, reducing the probability of buffer overflow at edge termination points.

The relationship between packet duplication and traffic spike suppression can be expressed as:

$$P_{\text{loss,effective}} = P_{\text{loss}}^{(1+d)} \qquad \text{(Effective loss probability with duplication factor } d\text{)}$$

where P_{loss} is the baseline packet loss probability and d is the duplication factor. Even modest duplication ($d = 1$) can substantially reduce effective loss rates in environments with moderate packet loss.

7.2. Edge Computing Architecture for Quality Optimization

An edge computing architecture that leverages the packet-level caching approach described in this study may be effective for implementing intelligent traffic management:

Table 9. Edge Computing Functions for Video Quality

Function	Description
Packet-Level FEC Insertion	Edge nodes insert Forward Error Correction packets based on observed loss patterns, reducing reliance on retransmissions
Adaptive Pacing	Edge nodes pace packet transmission to match measured last-mile capacity, preventing buffer overflow at terminals
RTT-Based Session Routing	Route sessions to edge nodes with optimal RTT characteristics based on real-time measurements from terminals
Predictive Prefetching	Prefetch subsequent video segments to edge caches based on playback position, reducing origin-to-edge latency spikes

7.3. Network Traffic Improvement Through Coordinated Edge Behavior

The combination of packet duplication and edge computing enables network-wide traffic improvement:

- **Reduced Retransmission Volume:** Proactive redundancy reduces the need for reactive retransmissions, which typically occur in bursts and exacerbate congestion.
- **Improved Cache Hit Rates:** Edge-level packet caching with intelligent prefetching reduces origin bandwidth consumption and improves fanout efficiency.
- **Temporal Traffic Smoothing:** By distributing redundant packets over time rather than sending bursts after loss detection, the aggregate traffic pattern becomes more predictable and less prone to inducing congestion.
- **Carrier-Aware Optimization:** Edge nodes can adapt their behavior based on observed carrier-specific characteristics (e.g., ACK ratios), optimizing transmission strategies for each mobile network operator.

This approach represents a shift from reactive (retransmission-based) to proactive (redundancy-based) quality assurance, which is particularly suited to the high-RTT, variable-capacity conditions observed in mobile RAN environments.

 doi:10.20944/preprints202510.1054.v2

8. Summary

This study empirically evaluated the impact of edge terminal buffer design on video quality during real-time delivery of AV1 over MPEG-TS using HTTP/3 over QUIC.

Through comparative experiments conducted in Tokyo (nearby) and Mumbai (distant), we confirmed that in high-RTT environments, edge terminal buffers can either saturate or deplete, potentially leading to silent packet drops.

We further demonstrated that placing the point of presence (POP) close to the terminal reduces ACK round-trip delay, thereby suppressing both the frequency of transmission stalls and buffer overflows.

In addition, under typical smartphone usage scenarios, we clarified that the design of the edge-end buffer has a greater impact on video quality (QoS) than differences in terminal-side congestion control algorithms such as CUBIC and BBRv2.

These observations are consistent with findings reported in other domains, such as low-Earth-orbit satellite networks, regarding the "Interaction between Intermediate Node Buffers and Delay Spikes" [14]. This suggests that our research contributes to the design of real-time delivery systems in terrestrial cloud environments as well.

The ability to conduct such detailed measurements in this study was enabled by the methodology used to construct HTTP/3 over QUIC client libraries in mobile environments, as demonstrated in the prior work "Building HTTP/3 over QUIC on iOS and Android: Native Stack and Real-Time Visibility" [1], together with our proprietary lightweight event aggregation mechanism. This approach allowed RTT estimation and visualization of packet-level behavior at the userland level.

Furthermore, the capacity to perform custom event aggregation for all callback events made it possible to obtain detailed QUIC traces in mobile environments, which had previously been a black box. This capability formed the empirical foundation for the analysis presented in this research.

Appendix A.

Section A.2, the Appendix Edge Module (UDP-MPEGTS to HTTP/3-Chunk), presents a lightweight implementation, design, and configuration. This module functions as the termination point for TCP-UPSTREAM traffic deployed immediately behind the edge alias, absorbing traffic spikes by leveraging implicit TCP session buffers. In effect, it operates as a relatively intelligent relay node positioned near the edge, providing retransmission, buffering, and enhanced availability.

Appendix A.1. Edge Module (nginx or FrontEnd)

Listing 5: nginx.conf

```
worker_processes  1;
http {
  server {
    server_name video.example.com;
    listen 443 quic reuseport;
    http3 on;
    ssl_protocols TLSv1.3;
    location → /stream {
        add_header Alt-Svc 'h3→ ":443"; ma→ 86400' always;
        add_header Content-Type 'application/octet-stream';
        proxy_buffering off;
        proxy_pass http://host.docker.internal:8080/stream;
    }
  }
}
```

Appendix A.2. Edge Module(UDP-MPEGTS to HTTP3-Chunk)

 doi:10.20944/preprints202510.1054.v2

Listing 6: Proxy Implementation

```
int main(
    int argc,
    char* argv[]
)
{
    /* ... */
    std::thread udp_thread(PROXY::_udp_receiver);
    struct MHD_Daemon* daemon → MHD_start_daemon(
        MHD_USE_THREAD_PER_CONNECTION,
        HTTP_PORT,
        NULL,
        NULL,
        PROXY::_handler,
        NULL,
        MHD_OPTION_END
    );
    if (!daemon) {
        PANIC("Failed to start server");
    }
    while(!PROXY::quit_) { usleep(10000); }
    MHD_stop_daemon(daemon);
    udp_thread.join();
    return(0);
}

MHD_Result PROXY::_handler(
    void* cls,
    struct MHD_Connection* conn,
    const char* url,
    const char* method,
    const char* version,
    const char* upload_data,
    size_t* upload_data_size,
    void** con_cls
)
{
    const char* channel → MHD_lookup_connection_value(conn, MHD_GET_ARGUMENT_KIND, "c");
    if (std::strcmp(url, "/stream") ↑ 0) {
        if (std::strcmp(method, "GET") !→ 0) {
            return(MHD_NO);
        }
        auto* ctx → new struct PROXY::client_context();
        ctx->read_index_ → ring_.current_head();
        ctx->channel_ → channel?channel:"50012";
        {
            std::lock_guard<std::mutex> lock(client_ctx_lock_);
            client_ctx_.push_back(ctx);
        }
        struct MHD_Response* resp → MHD_create_response_from_callback(
            MHD_SIZE_UNKNOWN,
            PACKET_SIZE,
            PROXY::_stream_cb,
            ctx,
            PROXY::_free_cb
        );
        MHD_add_response_header(resp, "Content-Type", "video/mp2t");
        MHD_add_response_header(resp, "Cache-Control", "no-cache");
        auto ret → MHD_queue_response(conn, MHD_HTTP_OK, resp);
        MHD_destroy_response(resp);
        return(ret);
    }
    return(MHD_NO);
```

 doi:10.20944/preprints202510.1054.v2


```
}

void PROXY::_udp_receiver(void) {
    int sockfd → socket(AF_INET, SOCK_DGRAM, 0);
    bind(sockfd, (sockaddr*)(&addr), sizeof(addr));
    /* ... */
    while (!PROXY::quit_) {
        ssize_t len → recv(sockfd, buf, PACKET_SIZE, 0);
        if (len > 0 && len <→ PACKET_SIZE) {
            ring_.push(buf, len);
        } else if (len < 0 && errno ↑ EAGAIN) {
            std::this_thread::sleep_for(std::chrono::microseconds(50));
        }
    }
    shutdown(sockfd, SHUT_RDWR);
    close(sockfd);
}

ssize_t PROXY::_stream_cb(
    void* cls,
    uint64_t pos,
    char* buf,
    size_t max
)
{
    auto* ctx → (struct client_context*)(cls);
    uint8_t temp[PACKET_SIZE];
    ssize_t templen → 0;
    if (ring_.pop(ctx->read_index_, temp, &templen)) {
        std::memcpy(buf, temp, templen);
        return(templen);
    }
    return(0);
}

void PROXY::_free_cb(void* cls) {
    std::lock_guard<std::mutex> lock(client_ctx_lock_);
    client_ctx_.erase(
        std::remove_if(client_ctx_.begin(), client_ctx_.end(),
            [&](struct PROXY::client_context* ptr) {
                if (ptr ↑ cls) { delete ptr; return(true); }
                return(false);
            }
        ),
        client_ctx_.end()
    );
}
```

References

1. MIXI, Inc.: Building HTTP/3 over QUIC on iOS and Android, 2023. https://medium.com/mixi-developers/building-http-3-over-quic-on-ios-and-android-native-stack-and-real-time-visibility-6e59c6aeebd0
2. Iyengar, J., Thomson, M.: RFC 9000 — QUIC: A UDP-Based Multiplexed and Secure Transport, 2021. https://www.rfc-editor.org/rfc/rfc9000.html
3. Thomson, M., Turner, S.: RFC 9001 — Using TLS to Secure QUIC, 2021. https://www.rfc-editor.org/rfc/rfc9001.html
4. Iyengar, J., Swett, I.: RFC 9002 — QUIC Loss Detection and Congestion Control, 2021. https://www.rfc-editor.org/rfc/rfc9002.html
5. Pauly, T., Kinnear, E.: RFC 9221 — An Unreliable Datagram Extension to QUIC, 2022. https://www.rfc-editor.org/rfc/rfc9221.html

 doi:10.20944/preprints202510.1054.v2

6. Han, J., Li, B., Mukherjee, D., Chiang, C.-H., Grange, A., Chen, C., Su, H., Parker, S., Deng, S., Joshi, U., Chen, Y., Wang, Y., Wilkins, P., Xu, Y., Bankoski, J.: A Technical Overview of AV1, 2021. https://arxiv.org/abs/2008.06091
7. Kränzler, M., Herglotz, C., Kaup, A.: A Comprehensive Review of Software and Hardware Energy Efficiency of Video Decoders, 2024. https://arxiv.org/abs/2402.09001
8. ngtcp2 Project: ngtcp2 — QUIC Transport Library. https://github.com/ngtcp2/ngtcp2
9. ngtcp2 Project: nghttp3 — HTTP/3 Library. https://github.com/ngtcp2/nghttp3
10. OpenSSL Project: OpenSSL — Cryptography and SSL/TLS Toolkit. https://github.com/openssl/openssl
11. libev Project: libev — High Performance Event Loop Library. https://github.com/enki/libev
12. Cardwell, N., Cheng, Y., Hassas Yeganeh, S., Swett, I., Jacobson, V.: BBR Congestion Control (Internet-Draft, Work in Progress), 2022. https://datatracker.ietf.org/doc/draft-cardwell-iccrg-bbr-congestion-control/02/
13. Curley, L.: MediaOverQUIC-Lite (Internet-Draft, Work in Progress), 2023. https://datatracker.ietf.org/doc/draft-lcurley-moq-lite/
14. The Impact of Buffers in Low-Earth Orbit Satellite Networks on Congestion Control at End Hosts. Ohmori. https://ipsj.ixsq.nii.ac.jp/record/2003106/files/IPSJ-IOT25070003.pdf
15. INFOCOM 2020: Exploring the interplay between CDN caching and video streaming performance, 2020. https://engineering.purdue.edu/ isl/papers/infocom2020.pdf
16. zacky1972: Approximate access times, required clock cycles, transfer speeds, and capacities for various memory/storage types. https://qiita.com/zacky1972/items/e0faf71aa0469141dede
17. Linode: Transition to adaptive bitrate streaming from client-side to server-side, 2023. https://www.linode.com/ja/blog/linode/migrating-from-client-side-to-server-side-adaptive-bitrate-streaming/
18. Public Alert and Hazard Notification System Using Low-Latency Push-Type MPEG-TS Streams over HTTP/3 over QUIC and Its Control Method(Draft) https://medium.com/@dsugisawa/public-alert-and-hazard-notification-system-using-low-latency-push-type-mpeg-ts-streams-over-http-3-314508c25f88