Technical Note

Not peer-reviewed version

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

Daisuke Sugisawa *

Posted Date: 14 October 2025

doi: 10.20944/preprints202510.1054.v1

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

 doi:10.20944/preprints202510.1054.v1

Technical Note

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

Daisuke Sugisawa

Xander, Independent Researcher, 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 presents the development of a smartphone-oriented application utilizing a QUIC protocol stack capable of estimating and measuring RTT in user space. 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 Figures 2 and 3).

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

 doi:10.20944/preprints202510.1054.v1

Table 1. Module Specifications.

Video Stream Origin	
Input	2K 30fps(FHD)
Encode	AV1(NVENC)
Output	CBR/ABR, multi-rate MPEG-TS Stream
Edge Termination, nginx or CloudFront(sec 8.1)	
HTTP3	HTTP3-Endpoints, Backend HTTP2 Proxy
Edge-Module, UDP-MPEGTS to HTTP3-Chunked-Data(sec 8.2)	
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.

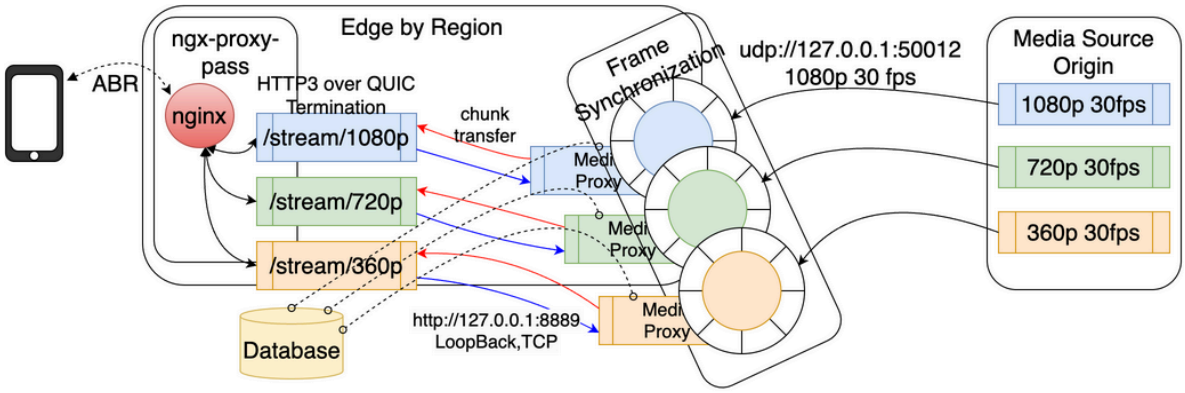

Figure 1. Video Stream System Architecture.

 doi:10.20944/preprints202510.1054.v1

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

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
```

Figures 2 and 3 visualize and analyze the first 5000 received packets when requesting the same media origin over an HTTP3 over QUIC session.

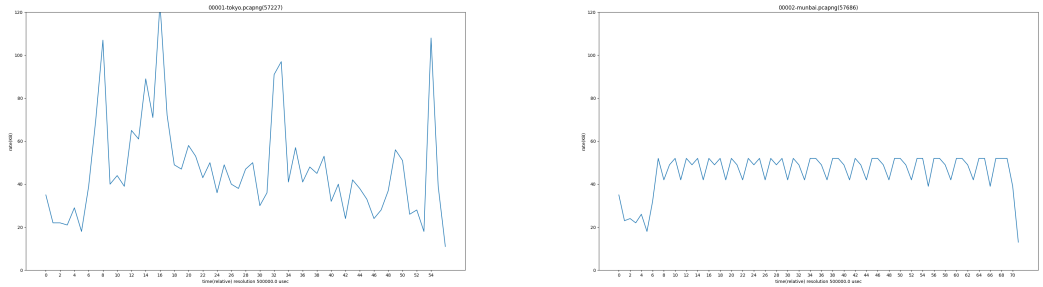

Figure 2. Packet Analysis Statistics: Neighbor (Tokyo, left), Far (Mumbai, right) CUBIC.

 doi:10.20944/preprints202510.1054.v1

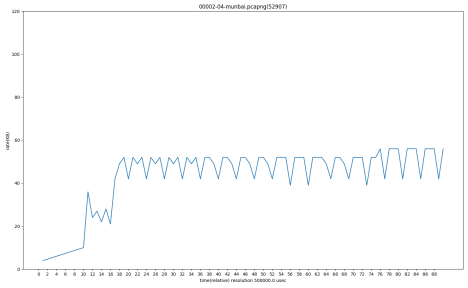

Figure 3. Packet Analysis Statistics: Far(Mumbai)BBR, proxy_buffering on;

3.2. Edge-Termination Proximity

Comparisons between neighbor (Tokyo) and distant (Mumbai) terminations revealed significant RTT effects. As shown in Figures 2, 3, 4, Smoothed RTT in Mumbai (133 ms) led to degraded quality (5 FPS), with nginx internal buffer analysis (Table 3) indicating ACK delay and silent packet drops as primary causes.

Figure 4. Packet Analysis Statistics: Far(Mumbai,up),Neighbor(Tokyo, down)Oritin to Docker

We analyzed the downstream TCP packets (port 8080) destined for nginx-quic within the Docker container (Figure 4) 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.v1

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 2, Figure 3, and Figure 4 Based on detailed packet analysis at the TCP layer, we suspect that temporary ACK delays and spike traffic-induced buffer overflows, and the resulting server-side (ngx_http3_module) silent packet drops as the primary causes of 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 2) and BBRv2 (Figure 3) [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 confirmed that traffic between the origin and the edge remains highly 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.v1

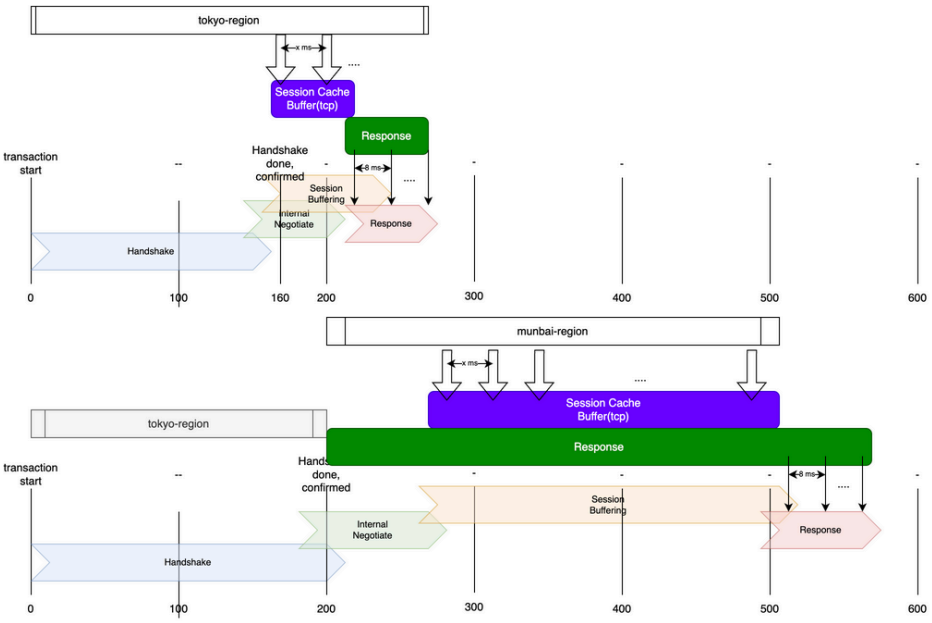

Figure 5. Neighbor, Distant Origin

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 6, Figure 7, Figure 8, Figure 9, Table 5, Table 4), the results showed convergence, indicating that aggregate behavior was largely consistent despite carrier-level differences.

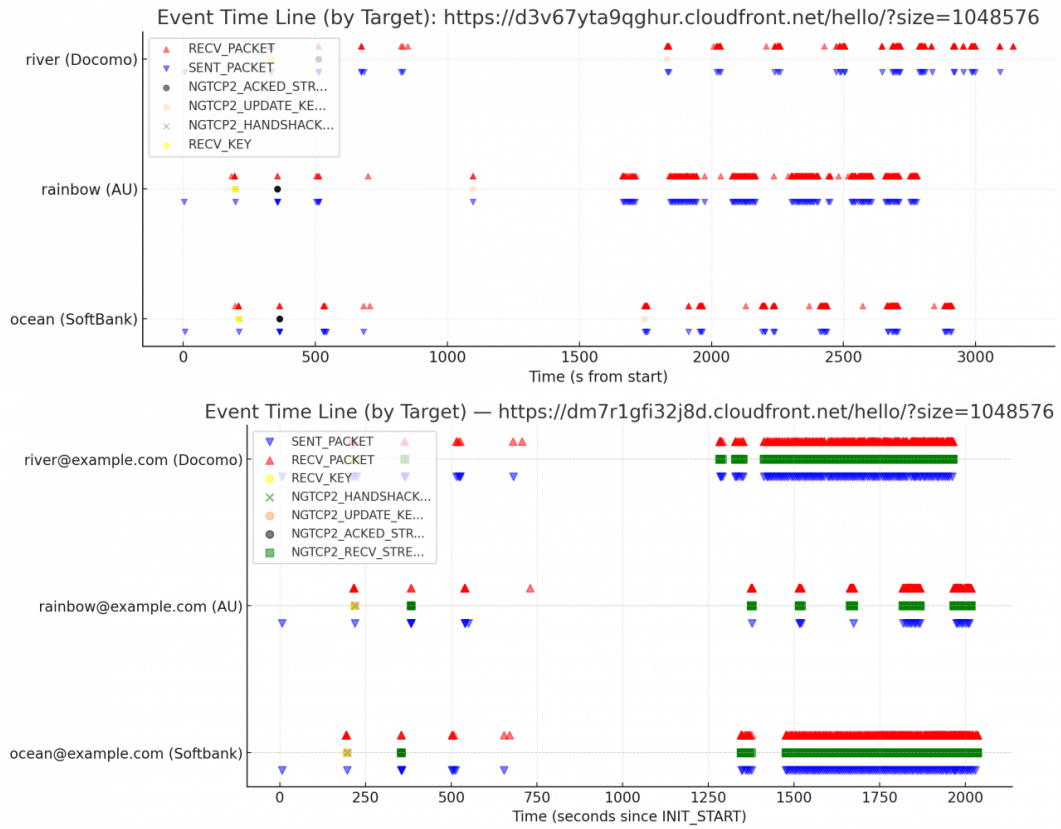

Figure 6. Event Analysis: Far(Mumbai,up),Neighbor(Tokyo, down), 1MB-DL HighBandwidth office-RAN

 doi:10.20944/preprints202510.1054.v1

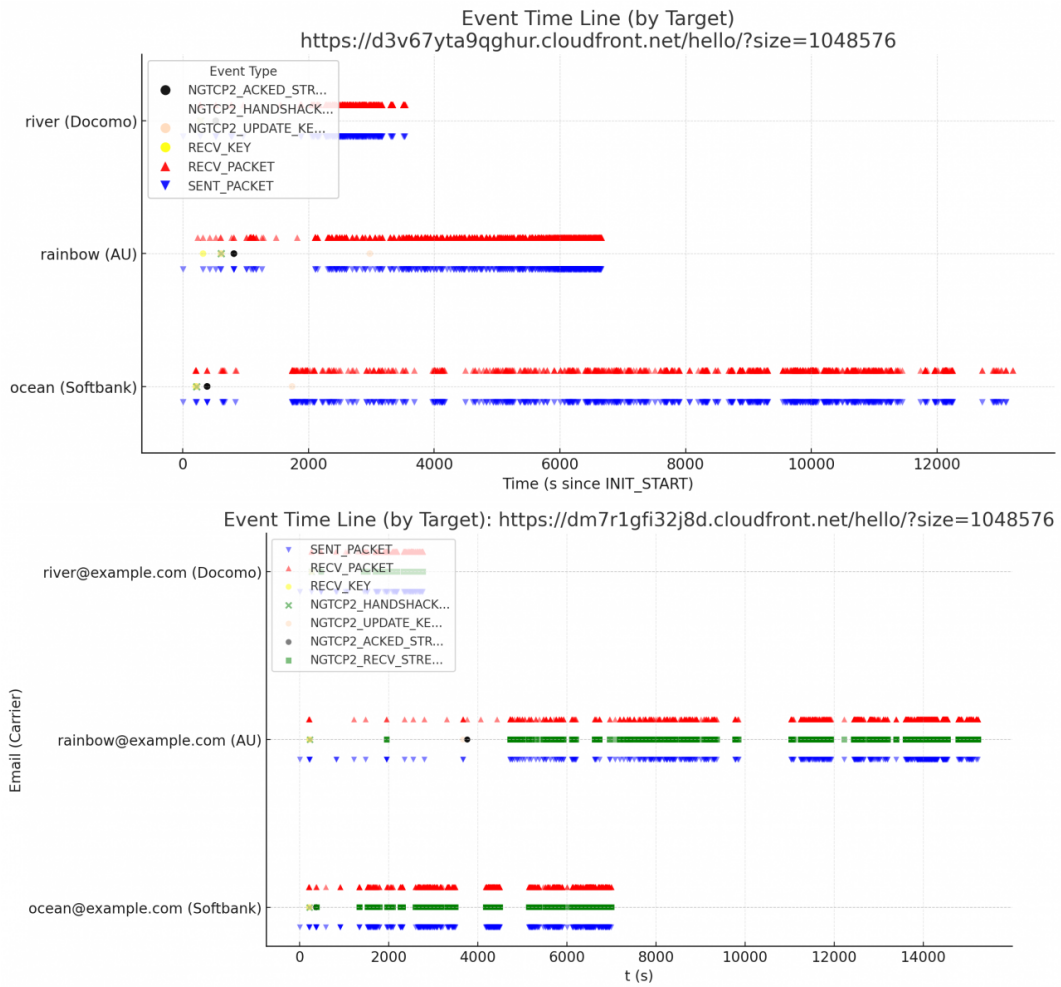

Figure 7. Event Analysis: Far(Mumbai,up),Neighbor(Tokyo, down), 1MB-DL Public-RAN

 doi:10.20944/preprints202510.1054.v1

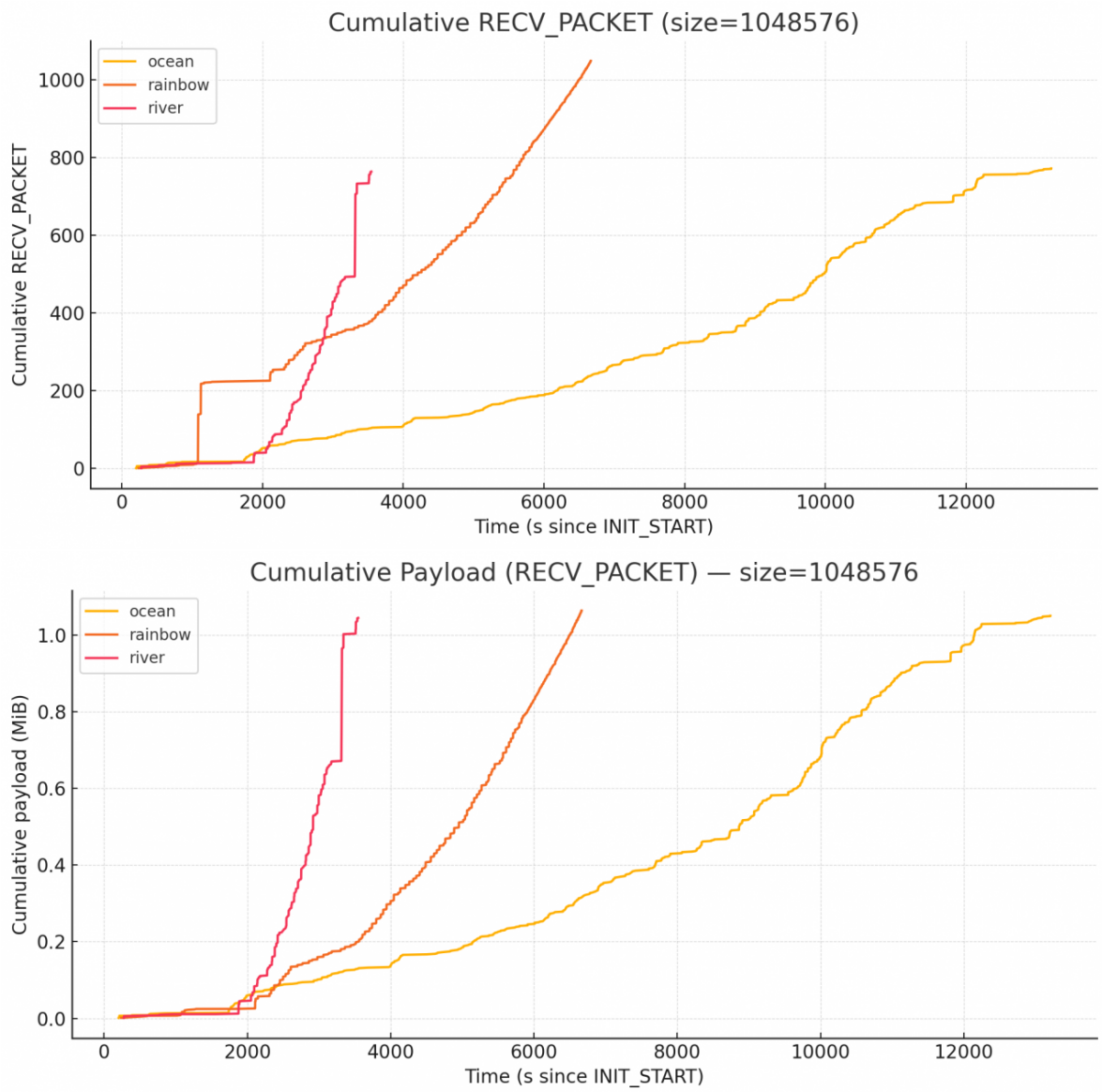

Figure 8. event analyze 1MB-DL, Recieve:packet-count, payload-size, cumulative

 doi:10.20944/preprints202510.1054.v1

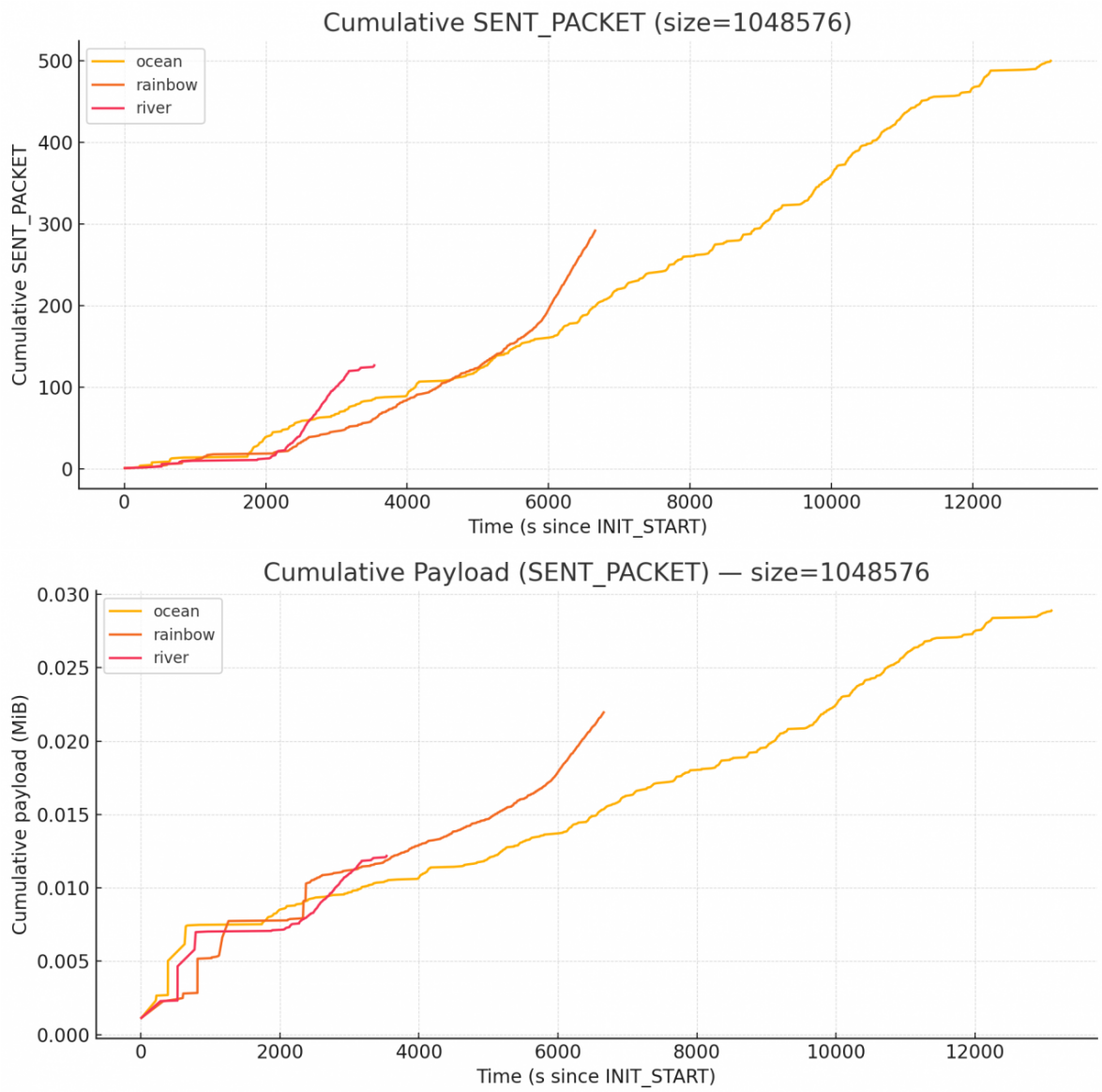

Figure 9. event analyze 1MB-DL, Send:packet-count, payload-size, cumulative

 doi:10.20944/preprints202510.1054.v1

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		

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.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 14 October 2025 doi:10.20944/preprints202510.1054.v1

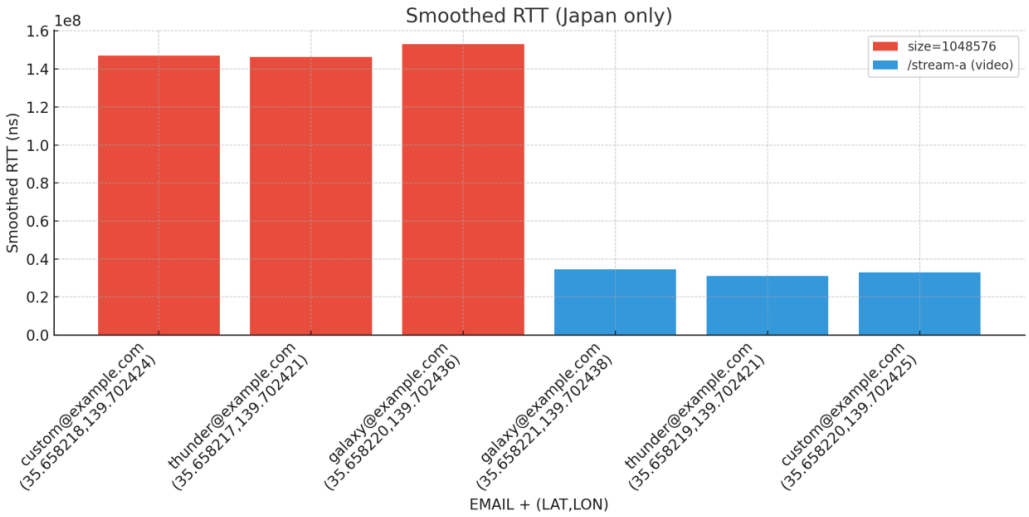

Figure 10. Smoothed RTT(japan)

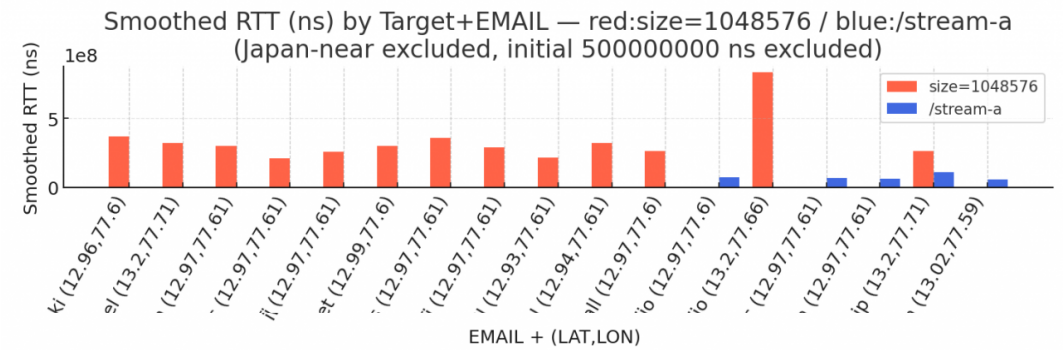

Figure 11. Smoothed RTT(Mumbai)

stream-a has a CBR of 1Mbps, so queueing delay is unlikely to accumulate. size=1048576 is 8Mbps best effort. On the POP→last-mile path, the buffer (FIFO) temporarily saturates, resulting in a higher RTT: packet round-trip time recognizable at the receiver. near the POP, temporarily saturating the buffer (FIFO). Consequently, the RTT (round-trip time) recognizable at the receiver increases. Assuming this scenario, the optimal CBR value for each target session can be calculated from the following formula based on the SMOOTHED-RTT measurement values. This represents the "true limit" calculable only at the terminal side:

$$\approx \mathrm{CBR}_{\mathrm{opt}} \quad \text{and} \quad SEQID_{\mathrm{bitmap}}$$

can be communicated to the server per session at minimal cost, enabling real-time ABR.

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.

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.

 doi:10.20944/preprints202510.1054.v1

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

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

4. Discussion: Non-HLS Architecture Specialized for Real-Time Delivery in This Study's Context

Unlike file-segment caching employed in HLS and related systems, the architecture in this study is specialized for real-time delivery through short-lived caches at the packet granularity. Whereas file caching is storage-oriented and introduces redundant delays from a real-time perspective, packet-level caching maximizes fanout efficiency to viewer terminals by using ring buffers with only a few seconds of capacity.

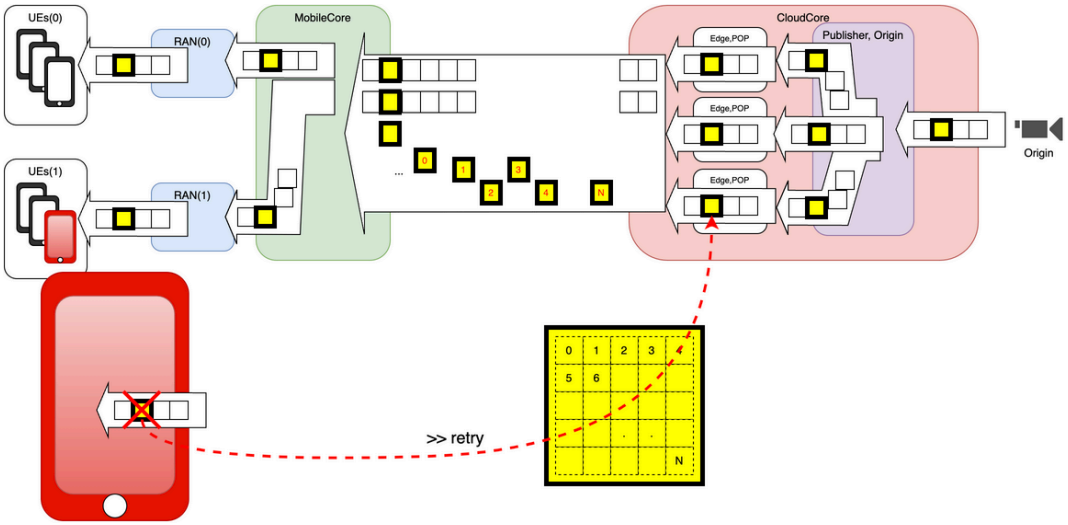

Figure 12. System Overview

As illustrated in Figure 1, Figure 13, 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, highly distributed live streaming system. Such a design is expected to contribute to scalable real-time delivery in edge environments and mobile networks [18].

 doi:10.20944/preprints202510.1054.v1

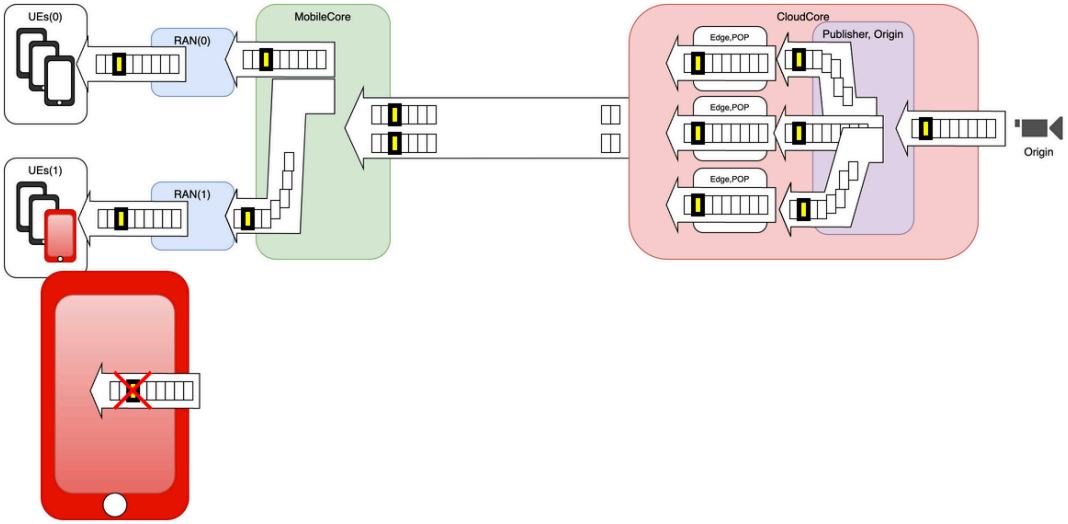

Figure 13. System Overview

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
Not compatible with HLS	Purpose differs from storage-oriented file caching

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

 doi:10.20944/preprints202510.1054.v1

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

4.1. Typical Access Cost [16]

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

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

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

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

 doi:10.20944/preprints202510.1054.v1

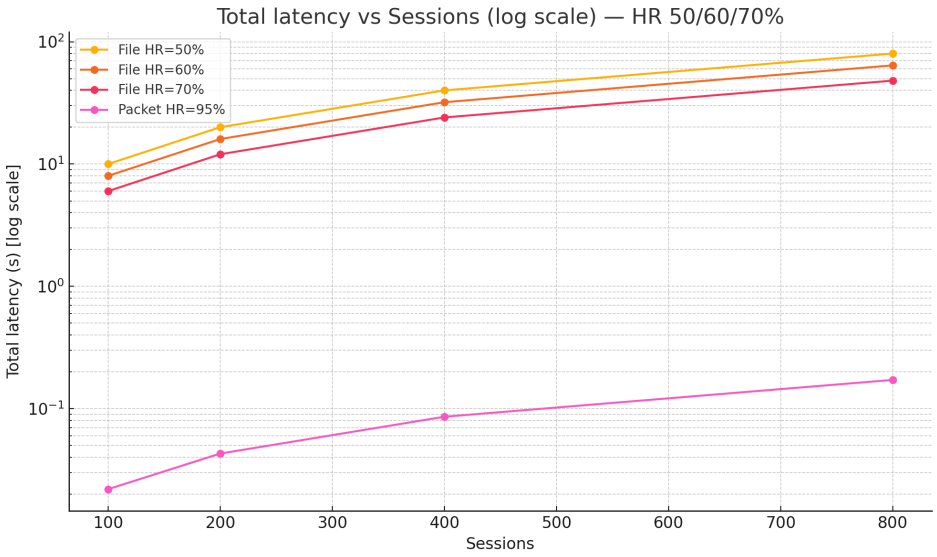

Figure 14. Total latency(Logical)

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.

Figure 15. AV1 over MPEG-TS Player iOS(PoC)

 doi:10.20944/preprints202510.1054.v1

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. 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.

8. Appendix

Section 8.2, the Appendix Edge Module (UDP-MPEGTS to HTTP/3-Chunk), presents an extremely 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.

8.1. 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;
    }
  }
}
```

 doi:10.20944/preprints202510.1054.v1

8.2. Edge Module(UDP-MPEGTS to HTTP3-Chunk)

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);
```

 doi:10.20944/preprints202510.1054.v1


```
    }
    return(MHD_NO);
}

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. Building HTTP/3 over QUIC on iOS and Android. https://medium.com/mixi-developers/building-http-3-over-quic-on-ios-and-android-native-stack-and-real-time-visibility-6e59c6aeebd0
2. RFC 9000. QUIC: A UDP-Based Multiplexed and Secure Transport. https://www.rfc-editor.org/rfc/rfc9000.html
3. RFC 9001. Using TLS to Secure QUIC. https://www.rfc-editor.org/rfc/rfc9001.html
4. RFC 9002. QUIC Loss Detection and Congestion Control. https://www.rfc-editor.org/rfc/rfc9002.html
5. RFC 9221. An Unreliable Datagram Extension to QUIC. https://www.rfc-editor.org/rfc/rfc9221.html
6. A Technical Overview of AV1. https://arxiv.org/abs/2008.06091

 doi:10.20944/preprints202510.1054.v1

7. A Comprehensive Review of Software and Hardware Energy Efficiency of Video Decoders. https://arxiv.org/abs/2402.09001
8. ngtcp2 Project. https://github.com/ngtcp2/ngtcp2.git
9. nghttp3 Project. https://github.com/ngtcp2/nghttp3.git
10. OpenSSL Project. https://github.com/openssl/openssl.git
11. libev Project. https://github.com/enki/libev.git
12. BBR Congestion Control(expired). https://datatracker.ietf.org/doc/html/draft-cardwell-iccrg-bbr-congestion-control-02
13. MediaOverQUIC-Lite. 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. Exploring the interplay between CDN caching and video streaming performance. https://engineering.purdue.edu/isl/papers/infocom2020.pdf
16. Approximate access times, required clock cycles, transfer speeds, and capacities for various memory/storage types. https://qiita.com/zacky1972/items/e0faf71aa0469141dede
17. Transition to adaptive bitrate streaming from client-side to server-side. 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