Kurdistan Journal of Applied Research (KJAR)
Print-ISSN: 2411-7684 | Electronic-ISSN: 2411-7706

Original Article

Check for updates

Improving Live Streaming QoE Through HLS Parameter Tuning and Load Balancing to Mitigate Packet Loss

Bzav Shorsh Sabir [a *], **Aree Ali Mohammad** [b]

[a] Department of Information Technology, Technical College of Informatics, Sulaimani Polytechnic University, Sulaymaniyah, Iraq.
[b] Computer Science Department, College of Science, University of Sulaimani, Sulaymaniyah, Iraq.

Submitted: 25 May 2025
Revised: 1 July 2025
Accepted: 21 August 2025

*** Corresponding Author**:
bzav.shorsh.tci@spu.edu.iq

Keywords: HTTP Adaptive streaming, Quality of service, Transmission control protocol, H.265.

How to cite this paper: B. S. Sabir, A. A. Mohammad," Improving Live Streaming QoE Through HLS Parameter Tuning and Load Balancing to Mitigate Packet Loss", KJAR, vol. 10, no. 2, pp: 77-92, Dec 2025, doi: 10.24017/science.2025.2.7

Abstract: Live video streaming denotes a video distribution service that concurrently captures and transmits media material to all consumers in real time. In recent years, most of the internet has been used for video streaming, as platforms have transformed content consumption, providing immediate access to films, television programs, live events, and user-generated materials worldwide. Platforms like Twitch, YouTube, and Amazon Prime are built on technologies that facilitate efficient content delivery, adaptive playback, and personalized recommendations. Hypertext Transfer Protocol live streaming is a popular protocol for adaptive video delivery that adjusts to network bandwidth but not to packet loss, which can severely impact viewer Quality of Experience (QoE). This study addresses the challenge of maintaining live video streaming quality in environments with varying packet loss. To improve QoE, this study proposes optimizing HLS configuration parameters and evaluating the effects of two load balancing algorithms, round robin and ring hash, in a simulated testbed. The study investigates how adjusting the segment length, list length, and the group of pictures size affects the resilience of the system to packet loss, as assessed by objective evaluation metrics including peak signal-to-noise ratio and data loss percentage. Results show that the ring hash algorithm consistently outperforms round robin in reducing data loss, and with the optimal parameter configuration, data loss remained below 1.4% even under 5% network packet loss.

1. Introduction

The widespread adoption of multimedia distribution platforms such as YouTube, Twitch, and Facebook live has driven rapid growth in new social networking paradigms [1]. Advances in user devices—featuring improved processing power, display resolution, and network connectivity—enable seamless video quality adaptation, enhancing viewer satisfaction and fueling market expansion. Live video streaming refers to the real-time recording and simultaneous broadcasting of media content to multiple users over the internet [1]. According to the Cisco Visual Networking Index, a majority of the internet has been used for video streaming over the past few years, from 73% in 2017 and increasing to 82% by 2022 [2].

It is important to provide a good Quality of Experience (QoE) for viewers, as even minor disruptions can significantly diminish user satisfaction. QoE refers to the overall level of satisfaction perceived by a user when interacting with a service or application. It transcends conventional Quality of Service (QoS) measures, which emphasize technical parameters such as bandwidth, latency, and error rates, to encompass the subjective evaluation of the user's experience. QoE considers elements such as content quality, user expectations, emotional state, and contextual usage, offering a holistic assessment of the

service's effectiveness in fulfilling user requirements. It is essential, as it directly influences customer retention and happiness, rendering it a primary concern for service providers seeking to optimize performance and improve the user experience [3].

The conventional IP-based streaming method is push-based, where the media is typically streamed over User Datagram Protocol (UDP) [4]. However, significant challenges are faced when delivering content to environments that involve various platforms [5]. One way to achieve a good QoE is by using Hypertext Transfer Protocol (HTTP) Adaptive Streaming (HAS). HAS is a media streaming technique that transmits video content in accordance with real-time network conditions and user capabilities. It functions by dividing the video into separate parts, each encoded at different bitrates. The video player adaptively alternates among different bitrates, guaranteeing seamless playback with minimal buffering [4]. HAS simplifies the delivery of content by using HTTP to transmit video fragments, facilitating navigation around network address translation and firewalls [6]. A client individually requests and retrieves each segment while preserving the playback session state, whereas the server is not obligated to retain any state. Consequently, the client can download segments from many servers without affecting system scalability [7]. Therefore, it takes advantage of standard web servers or caches found in networks of internet service providers and Content Distribution Networks (CDN). The server does not maintain responsibility for the client state; therefore, the client can download the segments from different servers, and it eliminates the need for a persistent connection between the client and the server [4].

There are two types of servers in a CDN; origin and replica. The origin server is where the original version of the content is, and the replica has a copy of the content. A replica server can act as a media server, cache server, or web server [8]. CDN is a useful method for raising the quality of the networks; it significantly increases the quality of internet services, with video streaming platforms greatly benefiting from it [9]. In a CDN, the content is dispersed from the origin server across multiple replica servers, and the client retrieves the content from the closest replica server, which will result in better QoE and QoS [10, 11]. Edge servers can be used in conjunction with a load balancer to distribute the network load; a load balancer's main job is to get requests from clients, check the load on available servers, and direct each request to the server most capable of processing it. This choice is often made using set strategies or algorithms [12].

HTTP Live Streaming (HLS) [13] is a video streaming protocol created by Apple in 2009 to transmit continuous video over the internet in a reliable manner. HLS is one of the HAS protocols, which use HTTP as the application and Transmission Control Protocol (TCP) as the transport layer protocol [4]. It is a pull-based streaming method where the video is segmented into chunks and encoded into multiple different bitrates. The server also generates a manifest file containing metadata for the video, audio, and subtitles, along with their corresponding retrieval locations. The client decides to pull the segments of the appropriate bitrate based on the available bandwidth to maximize QoE [4, 5, 14]. Other HAS protocols are Microsoft smooth streaming, Adobe's Dynamic HTTP Streaming, and Dynamic Adaptive Streaming over HTTP (DASH) [15].

HLS has two primary parameters that can be tuned to optimize both QoE and QoS. The first is HLS Time, which defines the length of each segment, expressed in seconds, and will be referred to as segment length [16]. The second is HLS List Size, which determines the maximum number of segments included in the playlist. During HLS streaming, metadata are generated containing the most recent available segments along with their corresponding retrieval locations; this parameter will be referred to as list length [16].

Various studies have explored techniques to improve QoS and QoE in live streaming. However, they have not examined the combined effect of segment length, list length, and Group of Pictures (GOP) size in HLS when used alongside CDN and a load balancer to reduce packet loss. This study has a threefold contribution: (1) improving QoE in live streaming using HLS by reducing the impact of packet loss on data transmission through the use of a CDN and load balancer; (2) investigating the effects of different segment lengths and list lengths on packet loss; and (3) examining the influence of various GOP sizes on packet loss.

The remainder of this paper is as follows: recent papers that studied video streaming are reviewed in section 2. Section 3 details the proposed method. Section 4, test results are shown in detail. Simulated testbed discussion is explained in section 5, and section 6 concludes and suggests potential future research directions.

2. Related Works

Providing a good QoE for viewers is the main focus when it comes to video streaming. Buffering, delay and quality loss can lead to the viewer being unsatisfied. In recent years, various research studies conducted on live streaming have been reviewed that contributed to improving QoE or QoS parameters through improving data transmission or reducing data loss. Among these efforts, Gutterman *et al.* [17] presented STAndard Low-LAtency vIdeo cONtrol (STALLION), which is an Adaptive Bitrate (ABR) algorithm that dynamically modifies bitrate selection in response to network throughput and latency variations in order to enable low-latency video streaming. STALLION uses the standard deviation of throughput and latency to make more stable and responsive bitrate decisions than traditional ABR systems, which only use buffer levels or average throughput. STALLION achieved a 1.8x boost in bitrate and lowered stall duration by 4.3x. Martinez-Caro *et al*. [18] presented a machine learning-based approach to detecting and predicting stalling events in video streaming over the DASH protocol to improve QoE. Two models are proposed: one for identifying stalling events in real-time based on transport-layer packet behavior and another using a recurrent neural network with long short-term memory to predict future stalling events before they occur. Using a long-term evolution emulation environment, their method effectively classified stalling events and forecast upcoming playback interruptions, achieving an F1 score of 0.923 with an error rate of 10.83%. The findings suggest that network-layer analysis can enhance video streaming performance without requiring direct access to the user's playback device.

The internet's architecture fundamentally adheres to a best-effort principle, lacking guarantees for the reliable transmission of packets. Consequently, applications must address challenges such as packet loss [19]. This issue is particularly critical for live streaming applications, as it leads to stalls, lowering the QoE and increasing the possibility of increasing the streaming delay in TCP-based streaming protocols. To address this issue, Bienik *et al*. [20] examined how Packet Loss Rate (PLR) affects the quality of compressed high-resolution videos that are transmitted across IP networks using the H.264 and H.265 codecs. 11,200 full high definition and ultra-high-definition video sequences encoded at various bitrates (1–15 Mbps) and employed simulated packet loss rates ranging from 0% to 1%. Peak Signal-to-Noise Ratio (PSNR), Structural Similarity Index Measure (SSIM), and absolute category rating were used to assess video quality objectively and subjectively. The findings support the notion that video quality declines with increasing PLR, with higher bitrates being more susceptible to this phenomenon. Furthermore, the results imply that in lossy contexts, smaller bitrates can occasionally preserve higher perceived quality. In a more network-focused approach, Clayman *et al*. [21] introduced a novel approach to streaming using Big Packer Protocol (BPP) with its in-network packet wash mechanism, which eliminates specific chunks rather than dropping the packets as it occurs in UDP or retransmitting them like TCP. BPP is tailored for high-bandwidth and low-latency applications, and they mapped H.264 Scalable Video Coding (SVC) into BPP packets, allowing dynamic adaptation while transmitting. Performance evaluation of BPP compared to TCP and UDP shows BPP significantly reduces latency and packet loss while maintaining better QoE than UDP or TCP. To improve packet handling in wireless environments, Taha *et al*. [22] proposed a smart algorithm that uses adaptive Quantization Parameters (QP) to improve QoE for video streaming in 5G wireless networks. The approach ensures smoother streaming by dynamically modifying QP to lessen the incidence of packet loss. According to experimental findings using Mean Opinion Score (MOS), QoE is enhanced by an ideal QP of 35 for low-motion videos and a QP of 30 for high-motion videos.

Building on resilience to burst losses, Rudow *et al*. [23] introduced Tambur, a novel loss recovery technique based on streaming codes that effectively handles bursty packet losses to enhance the quality of video conferences. By leveraging sequential packet decoding to optimize loss recovery, Tambur is able to reconstruct lost frames while consuming 35.1% less bandwidth than current forward error

correction techniques. Simulations on real-world Microsoft Teams traces revealed that Tambur reduces decoding errors by 26.5% and results in a 29% reduction in video freeze length. Tüker *et al*. [24] also employed BPP and its Packet Wash mechanism to investigate packet trimming at the edge as an in-network video quality adaptation strategy to selectively remove less critical video data rather than losing complete packets when bandwidth is constrained. By trimming packets at edge nodes, this technique maximizes bandwidth utilization while preserving low latency and high QoE. According to experimental findings, edge packet cutting offers higher QoE than HAS, guaranteeing smoother playback and improved scalability. Focusing on codec performance under loss, Abdullah *et al*. [25] carried out a comparative study comparing the effects of wireless network packet loss on real-time video streaming with H.265 and H.266 codecs. Through their work, an experimental testbed using FFmpeg and NetEm was introduced to assess codec performance in a variety of packet loss scenarios. The study discovered that, although both codecs showed quality degradation with increased packet loss, H.266 consistently provided superior compression efficiency and resilience by concentrating on objective QoE metrics like PSNR. Most recently, Meng *et al*. [26] introduced a packet loss recovery technique designed for edge-based interactive video streaming, such as Stadia, called Hairpin. It dynamically differentiates between initial transmissions and retransmissions to strike a balance between low latency and effective bandwidth usage. Hairpin lowers bandwidth costs by 40% and deadline miss rates by 32%, according to experiments conducted on real-world deployments.

CDN was initially developed to overcome the limitations of web caching. In video streaming, it seeks to improve QoE in terms of providing consumers with content while making better use of network resources [27, 28]. To examine this influence on content delivery and network efficiency, Shabrina *et al*. [29] looked into how CDNs affected QoS during live video streaming with HLS. Using Amazon Web Services (AWS) CloudFront as the CDN infrastructure, the researchers ran an experiment in which live video was broadcast from Bandung, Indonesia, to Tokyo, Japan. They measured throughput and packet loss ratio to evaluate streaming performance with and without a CDN, reporting an average throughput of 4452.6 kbps (compared to 3990.4 kbps without CDN) and a packet loss reduction to 0.08% (from 0.33% without CDN). Patel *et al*. [30] examined the performance of video streaming using cloud-based services and investigated the effects of different cloud-based CDNs on video streaming, including AWS CloudFront, Google Cloud, Azure, and Akamai, managing encoding, storage, and adaptive streaming to optimize video transmission by utilizing Spark for distributed analysis and Kafka for real-time video processing. According to the study, Google Cloud performs better in South America and Europe, while AWS CloudFront offers superior QoE in places such as Asia and North America. Further investigating CDN integration, Sangeetha *et al*. [31] examined the improvement of QoS for HLS with the incorporation of CDN alongside H.265 encoding. The findings indicate that CDN integration enhances average throughput, concurrently reducing the packet loss ratio to 0.072, in contrast to a 0.297% loss in the absence of CDN.

Table 1 highlights the key aspects of each study, emphasizing their research focus, streaming protocol employed, use of CDN, video coder utilized, experimental environment, and outcomes.

Table 1: Comparison of related works.

#	Research Focus	Streaming Protocol	CDN	Video Coder	Environment	Result
[18]	Predict stalling events	DASH	Not used	H.264	Simulation	Achieved F1 Score of 0.923.
[20]	Decrease impact of packet loss	RTP	Not used	H.264 and H.265	Simulation	Quality of sequences affected by packet loss ratio declines with increasing bit rate.
[21]	Decrease latency and packet loss	BPP	Not used	H.264 SVC	Simulation	BPP gives better QoE than UDP or TCP.
[22]	Decrease impact of packet loss	UDP	Not used	H.264 and H.265	Real world	Optimal QP is 35 for low motion and 30 for high motion.
[23]	Decrease impact of packet loss	UDP	Not used	VP9	Simulation	Tambur reduces the frequency of decoding failures for video frames by 26% and the bandwidth used for redundancy by 35%.

Table 1: Continue

[24]	Decrease packet loss	BPP	Not used	H.264 SVC	Simulation	Shows trimming packets at the edge is better than using an ONOS controller.
[25]	Decrease impact of packet loss	UDP	Not used	H.265 and H.266	Simulation	Shows H.266 is more robust against network packet loss
[26]	Decrease packet loss	Customized RTP	Not used	N/A	Real world	Hairpin lowers bandwidth costs by 40% and deadline miss rates by 32%
[29]	Decrease packet loss	HLS	Used	H.264	Real world	Shows CDN reduces packet loss from 0.33% to 0.08%
[31]	Decrease packet loss	HLS	Used	H.265	Real world	Shows CDN reduces packet loss from 0.297% to 0.072%

3. Materials and Methods

A virtual environment is created using VMware Workstation to simulate two scenarios for many cases on a host machine with a Ryzen 9 7900 CPU and 32GB of DDR5 RAM. Figure 1 shows the general diagram of the proposed system.

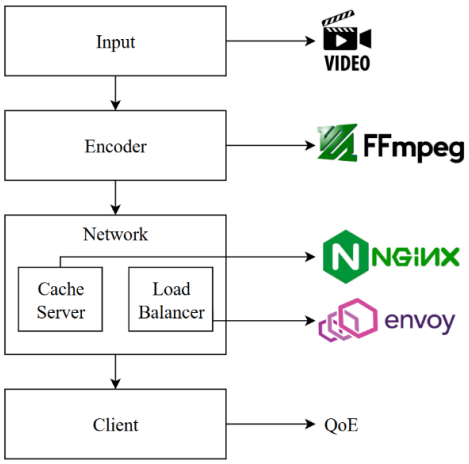

Figure 1: General block diagram of the proposed system.

3.1. Video Characteristics

The input video is 1920x1080, 4 minutes and 32 seconds, 24 frames per second. The video is encoded in real-time using FFmpeg into three different bitrates using H.265 using the ultrafast preset:

- High: 4.8 Mbps bitrate, ~160 MB total file size.
- Medium: 3 Mbps bitrate, ~100 MB total file size.
- Low: 1 Mbps, ~35 MB total file size.

When encoding a video for streaming with HLS, a suitable GOP size is important; the GOP size designates the quantity and arrangement of I-frames, P-frames, and B-frames [32]. I-frames refer to intra-coded frames that can function as independent images and are frequently utilized as a point of reference for a brand-new scene or a significant alteration to the previously sent frame sequence. Only predictive data are contained in P-frames, which are produced by examining the deltas between the current and prior frames. B-frames are produced by comparing the differences between the preceding and subsequent reference frames. Both P-frames and B-frames have the benefit of using far fewer resources when stored or transferred; however, they lack the information necessary to watch the associated video frame [33].

The segment lengths and list lengths used in this study introduce noticeable latency; they were chosen because their delay is significantly lower than that of larger values, while maintaining sufficient resilience to packet loss. The GOP size should align with the segment length, meaning it should be

equal to, or a divisor of, the number of frames in a segment (e.g., the same, half, one-third, one-fourth, etc.) [16]. However, a very small GOP size can lead to high resource utilization.

3.2. Network Design

3.2.1. Origin Server

The origin server is a Debian 12 Virtual Machine (VM), it uses FFmpeg to encode and segment the video in real time, and it uses Nginx server to stream the video via HLS.

3.2.2. Cache Servers

50 Docker containers of Nginx are created and configured to act as cache servers to fetch, cache, and send content to the receivers upon request. The Docker containers are created inside Graphical Network Simulator 3 (GNS3) VM, which is a network experimentation framework that supports multivendor models and emulation of actual devices [34].

3.2.3. Load Balancer

The load balancer is a Debian 12 VM that runs EnvoyProxy to distribute requests from the clients based on the algorithm used (round robin or ring hash) to one of the cache servers.

- Round robin is one of the most straightforward load balancing algorithms; it makes use of a circular list and a pointer to the most recent server that was chosen [35].
- Ring hash implements consistent hashing, where every request is routed to a host by hashing a property of the request and locating the closest corresponding host clockwise around the ring [36].

Ring hash is proposed, as it has minimal disruption when servers are added/removed to provide the least data loss to improve QoE, and round robin is tested for comparison to ring hash.

3.2.4. Clients

The client is a Debian 12 VM. For the first scenario, FFmpeg is used to download the content. For the second scenario, Java is used with FFmpeg to simulate 60 clients requesting the load balancer to fetch the contents.

3.3. Simulation Scenarios

Two distinct scenarios have been designed to analyze the differences in segment lengths, list lengths and GOP sizes. Table 2 outlines the key differences between the two scenarios for a clear and structured comparison.

Table 2: Details of scenario 1 and 2.

Test Parameters	Scenario 1	Scenario 2
Number of cache servers	1	50
Usage of load balancer	Not used	Used
Number of clients	1	60
Tested load balancing algorithms	N/A	Round Robin and Ring Hash
Tested segment length	1, 1.5 and 2 seconds	1 and 1.5 seconds
Tested list length	5 and 10 segments	5 and 10 segments
Tested GOP sizes	12, 24, 36 and 48 frames depending on the segment length.	12, 24 and 36 depending on the segment length.
Total tested cases	14 cases	16 cases

3.3.1. Scenario 1: One Client and One Cache Server

In the first case, the main server streams the video in real time, and the client requests a cache server to download the high-quality video. The downloaded video is then compared to the original video to observe the QoE. Figure 2 shows the network design of this scenario; the packet loss simulations are put on the link in GNS3 between the main server/cache server and the cache server/client. The diagram in figure 3 shows all the tested cases performed in this scenario, for example:

- Case 1: the segment length was 1 second, the GOP size was 12 frames, and the list length was 5 segments.
- Case 2: the segment length was 1 second, the GOP size was 12 frames, and the list length was 10 segments.
- Case 3: the segment length was 1 second, the GOP size was 24 frames, and the list length was 5 segments.
- etc.

For every case, packet loss was simulated from 1% to 6%.

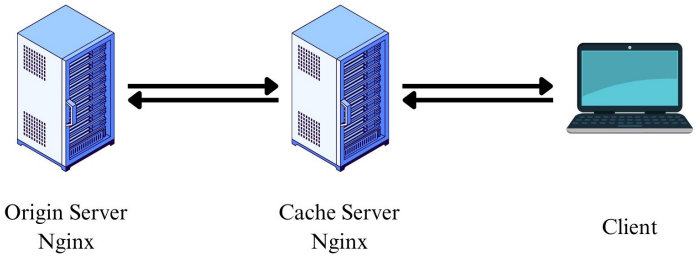

Figure 2: Network design of scenario 1.

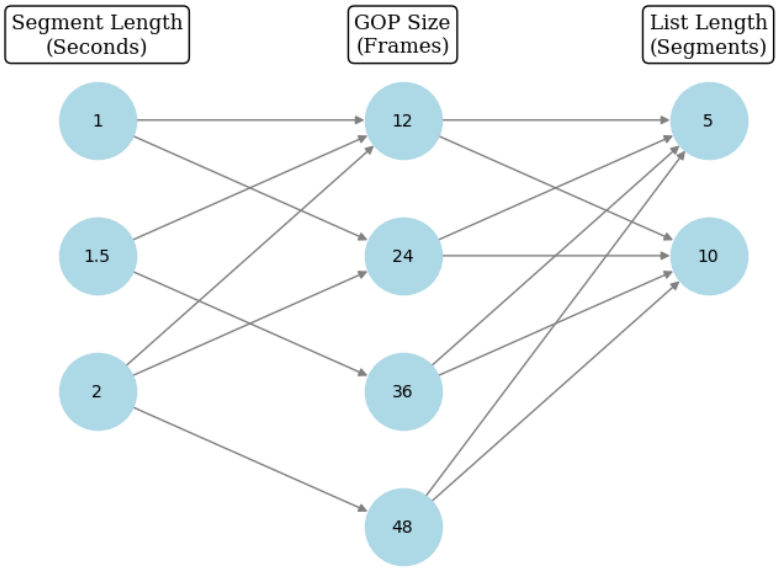

Figure 3: Tested cases of scenario 1.

Figure 4 shows the process diagram of this scenario. The client requests the cache server for the contents to pull the stream, the cache server checks if the content is available, and if it already has the content, it will reply with the content. If the content is not available in the cache server, the cache server requests the origin server, stores a copy of the content and sends the content to the client. This process occurs while the origin server continuously encodes and segments the live video into multiple representations.

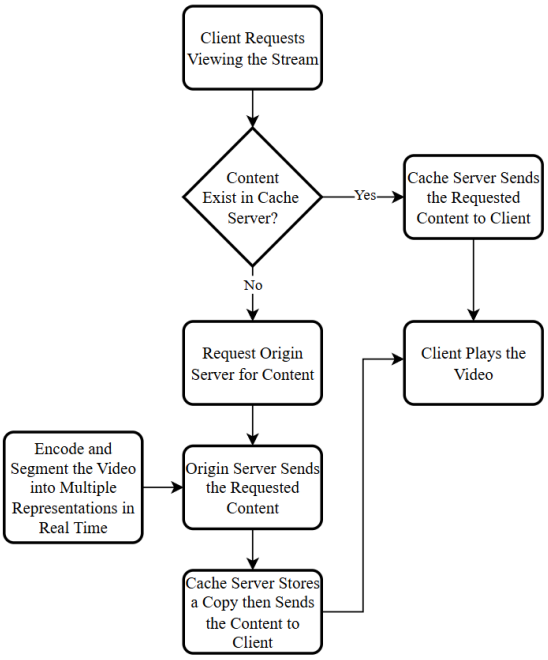

Figure 4: Process diagram of scenario 1.

3.3.2. Scenario 2: 60 Clients and 50 Cache Servers

Sixteen cases are tested with different parameters of segment length, list length, GOP size, and load balancer algorithm, as seen in figure 5. Similar to scenario 1, for each case, packet loss is simulated from 1% to 6%.

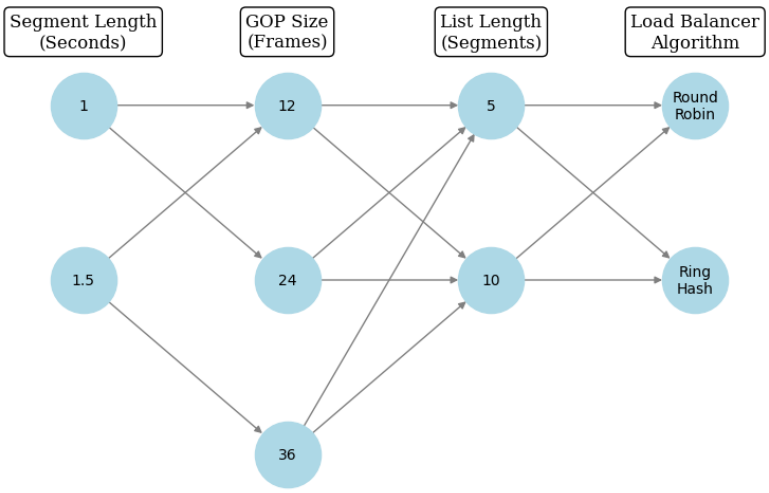

Figure 5: Tested cases of scenario 2.

For every case, packet loss from 1% to 6% is simulated. Sixty clients request the streamed video and EnvoyProxy is used to distribute the requests of the clients to 50 cache servers. Java is used to simulate the clients at the same time: 20 clients request the high-quality stream, 20 clients request the medium-quality stream, and 20 clients request the low-quality stream. The clients request the load balancer to download the streamed video; round robin and ring hash algorithms are tested in EnvoyProxy, and their differences are compared with the different parameters shown in Figure 5 above to see the impact of packet loss on data loss. Figure 6 shows the network design of scenario 2, the packet loss simulations are put on the link in GNS3 between the main server-cache server and the cache server-load balancer.

Figure 7 shows the process diagram of scenario 2. The client requests the load balancer for the contents to pull the stream, the load balancer requests the cache server based on the load balancing algorithm chosen, the cache server checks if the content is available, if it already has the content, it will reply with the content. If the content is not available in the cache server, the cache server requests the origin server, stores a copy of the content, and sends the content to the load balancer, and the load balancer replies to the client. This workflow is carried out while the origin server continuously encodes and segments the live video into multiple representations.

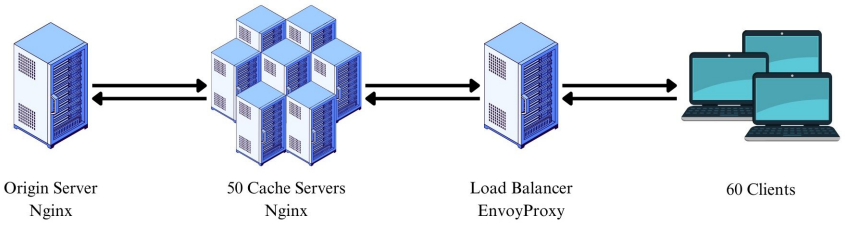

Figure 6: Network design of scenario 2.

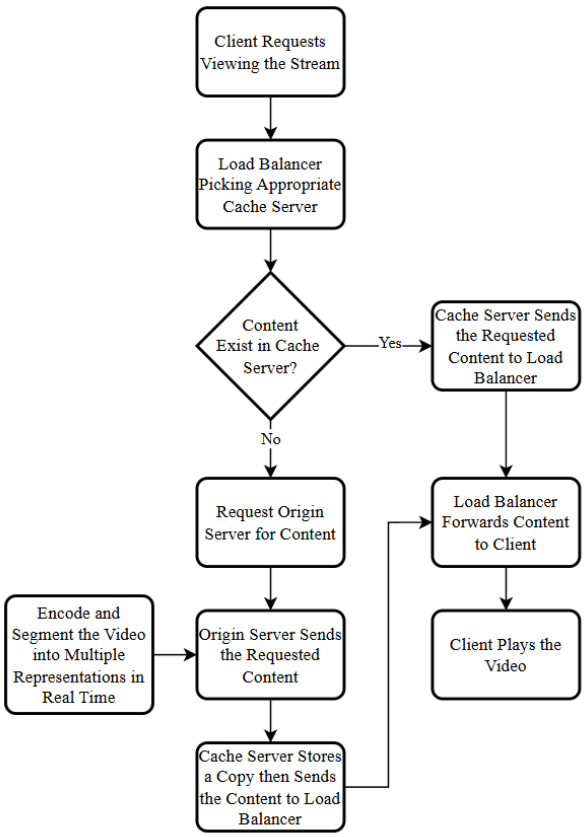

Figure 7: Process diagram of scenario 2.

3.4. Performance Analysis Metrics

For the first scenario, for evaluation of the tests conducted, PSNR is used. PSNR is frequently employed as a metric to assess image quality, its primary component is the Mean Square Error (MSE), from which it is formed [37]. Calculated over the image's dimensions, MSE is the sum of the squared differences between the original and processed image's pixel values. The MSE value is transformed logarithmically to produce PSNR. It shows the inaccuracy in an image; higher PSNR values, which imply higher image quality, are produced by fewer disparities in pixel values. In theory, PSNR can become close to infinity if the pixel values of the original and processed images remain unchanged. On

the other hand, PSNR values decrease with more pixel value discrepancies, suggesting a drop in image quality [37]. For the second scenario, the total size of the downloaded files is calculated and compared with the expected size to quantify the impact of packet loss on data loss during transmission.

4. Results

Scenario 1

As seen in figure 8 and figure 9, videos sent with a higher segment length are less affected by network packet loss, leading to a higher PSNR. A bigger list length (10, in this case) also decreased the impact of packet loss, especially when the segment length was 1 second, due to the client having more time to recover the packets before segment expiration happened. GOP size did not have a noticeable impact. In most cases, GOP size only had a slight impact when packet loss became 6%; higher GOP size was better when segment length was 1 second, and lower GOP size was better when segment length was 1.5 or 2 seconds. Table S5 presents the numerical results of this scenario.

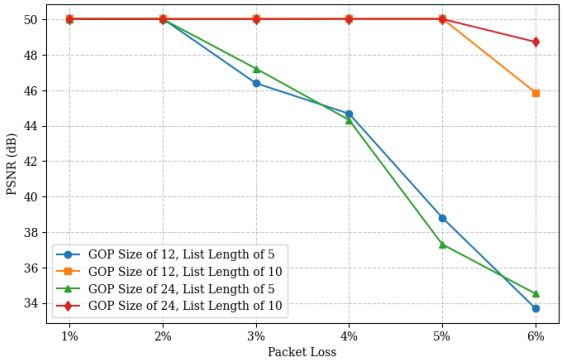

Figure 8: Scenario 1, PSNR when segment length is 1 second.

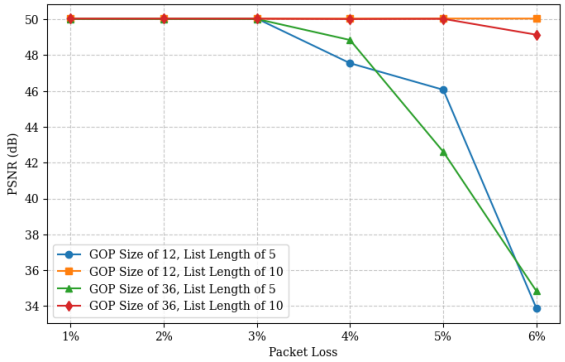

Figure 9: Scenario 1, PSNR when segment length is 1.5 seconds.

HLS retrieves three segments before initiating playback; therefore, segment length and list length have a substantial effect on stream delay, as seen in figure 10. The minimum delay is three times the segment length, and the maximum delay is the segment length multiplied by the list length; for instance, with a segment length of 1.5 seconds and a list length of five segments, the stream delay ranges from 4.5 to 7.5 seconds.

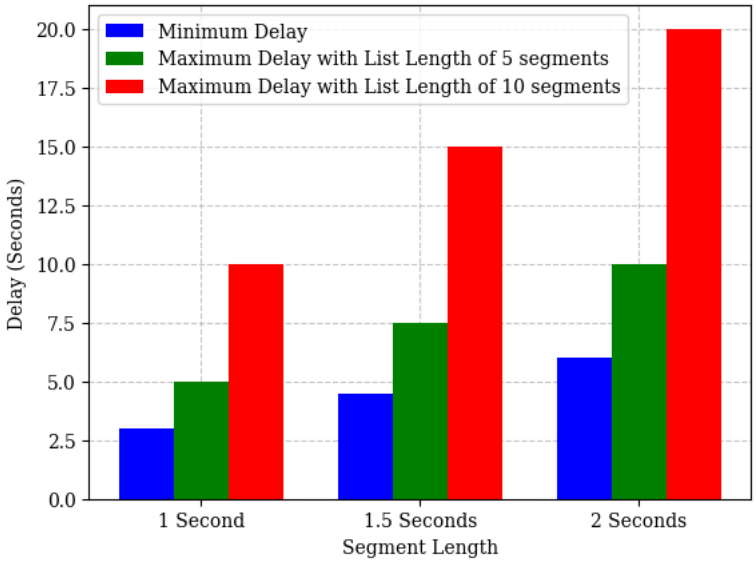

Figure 10: Expected delay depending on segment and list length.

Scenario 2

When there are more clients and a load balancer is used, the results are not as straightforward as in scenario 1; for the segment length of 1 second and list length of five segments, as seen in figure 11, different GOP sizes did not have a markable difference with round robin. However, while using ring hash, a GOP size of 24 frames consistently had better results than 12 frames.

When the list length increased to 10 segments, the differences became more noticeable. As shown in figure 12, a GOP size of 12 frames performed better with round robin, while a higher GOP size was more effective with ring hash. For a segment length of 1 second with ring hash, from a packet loss of 4% to 6%, on average, the amount of data sent with the higher GOP size was 1.93% more than the lower size.

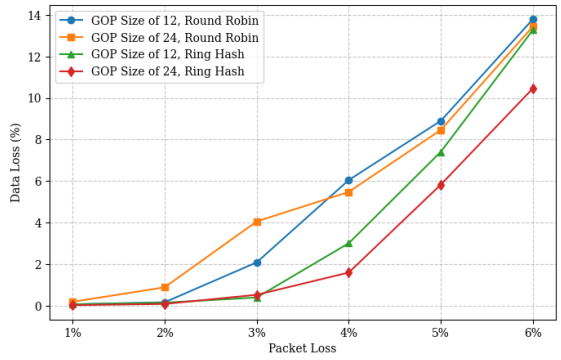

Figure 11: Scenario 2, data loss ratio when segment length is 1 second and list length is five segments.

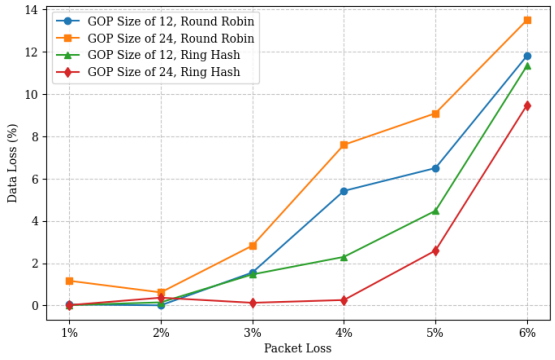

Figure 12: Scenario 2, data loss ratio when segment length is 1 second and list length is 10 segments.

Similar to the segment length of 1 second, differences were less noticeable for the segment length of 1.5 seconds when the list length was five segments compared to 10 segments. However, with ring hash, a higher GOP size resulted in more data being lost, as seen in figure 13. Less data was lost when the GOP size was 12 frames. When the list length increased to 10 segments, the GOP size had little effect when using round robin; however, with ring hash, a GOP size of 12 frames consistently resulted in less data loss, as shown in figure 14.

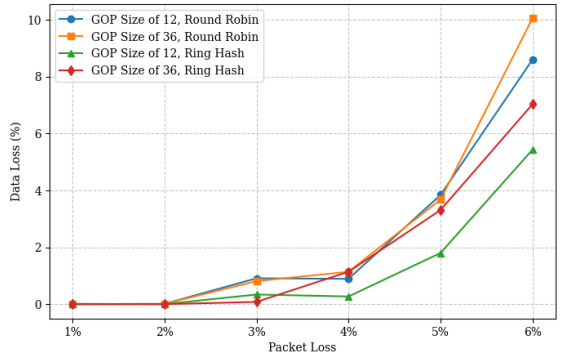

Figure 13: Scenario 2, data loss ratio when segment length is 1.5 seconds and list length is five segments.

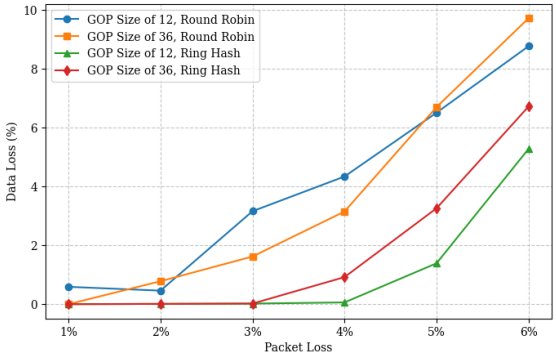

Figure 14: Scenario 2, data loss ratio when segment length is 1.5 seconds and list length is 10 segments.

With ring hash, when comparing the segment length of 1 and 1.5 seconds with a list length of five segments, a longer segment length can make a bigger difference than GOP size, as seen in figure 15. However, with a list length of 10 segments, GOP size can also have a substantial influence, as seen in figure 16, where a segment length of 1 second with a GOP size of 24 frames had better results in certain cases than a segment length of 1.5 seconds with a GOP size of 36 frames.

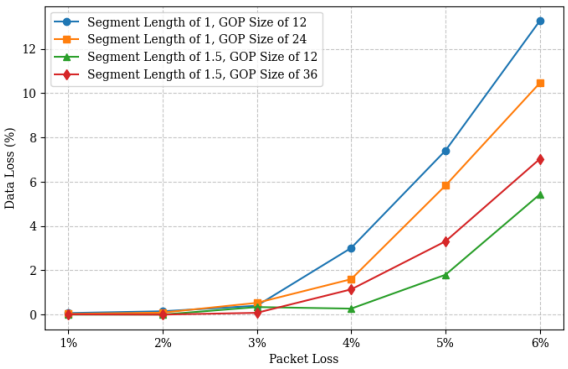

Figure 15: Scenario 2, data loss ratio when list length is five segments and load balancer algorithm is ring hash.

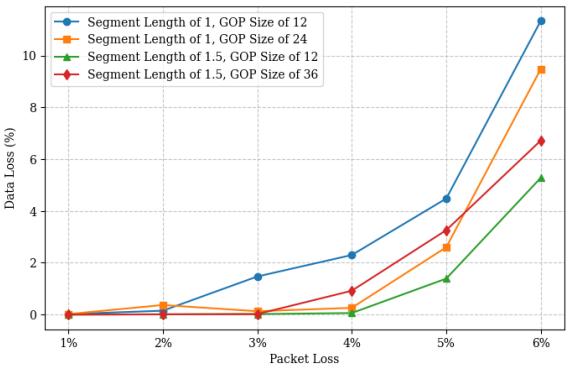

Figure 16: Scenario 2, data loss ratio when list length is 10 segments and load balancer algorithm is ring hash.

Figure 17 and figure 18 show the impact of list length on segment length of 1 and 1.5 seconds, respectively. List length had a much smaller impact on data received at the clients when segment length was 1.5 seconds compared to 1 second, GOP size had the majority of the impact. For a segment length of 1 second, both list length and GOP size had a remarkable impact on decreasing data loss. The detailed numerical results for this scenario are provided in Table S5.

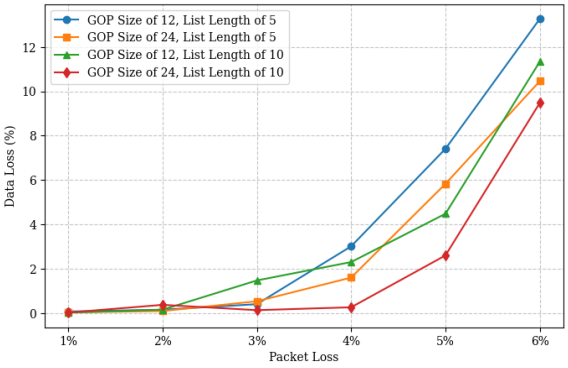

Figure 17: Scenario 2, data loss ratio when segment length is 1 second and load balancer algorithm is ring hash.

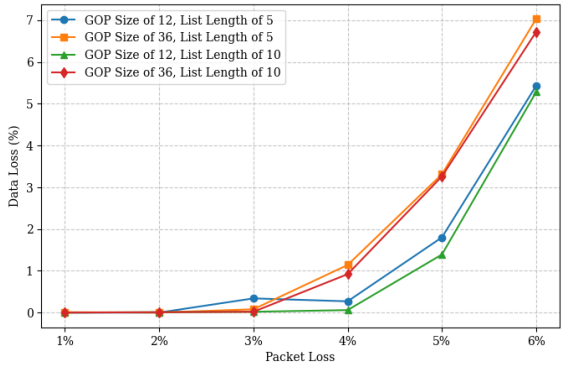

Figure 18: Scenario 2, data loss ratio when segment length is 1.5 seconds and load balancer algorithm is ring hash.

5. Discussion

According to the observations, segment length and list length can have a huge impact on the reduction of data loss due to packet loss. When using a load balancer, the ring hash algorithm consistently reduced the impact of packet loss on data arriving at the clients compared to round robin, as seen in figure 19. GOP size does not have any impact on stream delay, and the different GOP sizes tested also had little to no impact on the quality of the output video when their PSNR was calculated.

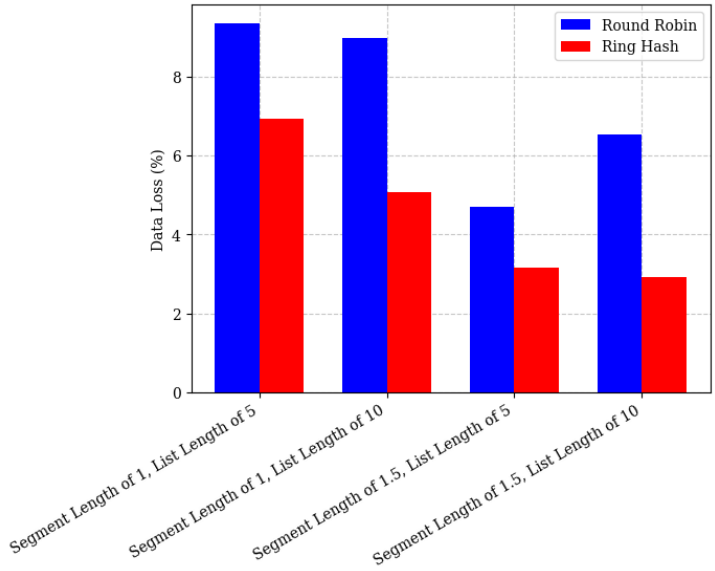

Figure 19: Average data loss comparison between round robin and ring hash.

The findings show that longer segment length and/or bigger list length will reduce the impact of packet loss due to the player having more time to recover the lost packet at the cost of longer stream delays as seen in figures 15, 16, 17 and 18. Optimal GOP size (24 frames for a segment length of 1 second, 12 frames for a segment length of 1.5 seconds) can make a significant difference in reducing the impact of packet loss on data arriving at the clients.

The optimal parameters for HLS depend on the application; if stream delay is not important, longer segment duration and/or list length will yield a better result against packet loss. As seen in Figure 18, a segment length of 1.5 seconds with a GOP size of 12 frames and a list length of 10 segments yields the best result to combat packet loss but with the consequence of a stream delay between 4.5 and 15 seconds. But with the list length of five segments, the maximum delay is cut by half while only being more susceptible to data loss by an average of 0.25% between 4% and 6% packet loss compared to when the list length is 10 segments.

Table 3 compares this study with relevant prior works. While Taha and Ali [22] examines the effect of QP and Abdullah *et al.* [25] explores the difference between H.265 and H.266 in a low latency live streaming environment with packet loss, this study focuses on the impact of segment length, list length, GOP size, and load balancing algorithms in a higher latency live streaming context.

Table 3: Comparison of this study to relevant related works.

Parameters	[22]	[25]	Scenario 1 of this study	Scenario 2 of this study
Protocol	UDP	UDP	HLS	HLS
Codec	H.264 and H.265	H.265 and H.266	H.265	H.265
CDN	Not used	Not used	Not used	Used
Evaluation Metric	PSNR, SSIM and MOS	PSNR and SSIM	PSNR	Data Loss
Environment	Real-world	Simulation	Simulation	Simulation
Stream latency	Low latency	Low latency	High latency	High latency
Range of packet loss	0% to 1%	0% to 10%	0% to 6%	0% to 6%
Contribution	Shows the effect of QP in low and high motion videos in a network with packet loss	Shows the degradation of H.265 and H.266 in a network with packet loss	Shows the effect of segment length, list length and GOP size in a network with packet loss	Shows the effect of segment length, list length, GOP size, and load balancing algorithm in a network with packet loss

Segment length of 1.5 seconds with a list length of five or 10 segments would not be viable for video chats or conferences due to the high stream delay, but it is viable for streams where delay does not affect

QoE, such as sport events, as 15 seconds of delay is acceptable while also being highly resistant to data loss. The limitations of this paper are the absence of subjective and other objective evaluation metrics, as PSNR and data loss may fail to reflect the full extent of the result, because in some cases no amount of data is lost but buffering is still a possibility. In addition, he implementation of the testbed in a virtual environment is another limitation, as no other real-world variables are taken into consideration other than packet loss.

6. Conclusions

The adaptive bitrate algorithm of HLS only takes bandwidth into consideration when picking the appropriate stream; network packet loss leads to data loss, stalls, and delays in the stream, resulting in a lower QoE for the viewers. In this study, data loss due to packet loss is investigated in a controlled environment, and various cases are observed to reduce the impact of packet loss. The findings highlight that higher segment and list lengths can improve live video streams against data loss, and GOP size can have a significant impact on it as well. Using a load balancer with ring hash consistently outperformed round robin for HLS.

This research can be extended in several directions. One possibility is to implement load balancing logic within the client player to reduce data loss, eliminate the need for a dedicated load balancing server, and minimize stream delay. Another is to evaluate the proposed methods and parameters in real-world environments, as factors such as cost and varying network conditions were not considered in this study. Further work could involve comparing the ring hash algorithm with the maglev algorithm, which offers faster host lookup times but less stability when upstream hosts change. Additional directions include comparing and testing the optimal QP proposed in previous studies with different codecs in the same testbed, incorporating other objective evaluation metrics such as buffering, conducting subjective user evaluations, and assessing the resource requirements for the various configurations.

Author contribution: Bzav Shorsh Sabir: Investigation: Writing – original draft, Writing – review & editing. **Aree Ali Mohammad:** Supervision, Writing – review & editing.

Data availability: Data will be available upon reasonable request by the authors.

Conflicts of interest: The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Funding: The authors did not receive support from any organization for the conducting of the study.

References

[1] N. N. Dao, A. T. Tran, N. H. Tu, T. T. Thanh, V. N. Q. Bao, and S. Cho, "A contemporary survey on live video streaming from a computation-driven perspective," *ACM Computing Surveys*, vol. 54, no. 10s, pp. 1–38, 2022, doi: 10.1145/3519552.

[2] V. Cisco, "Cisco visual networking index: Forecast and trends, 2017–2022," *White paper*, vol. 1, no. 1, pp. 1–38, 2018.

[3] K. Bouraqia, E. Sabir, M. Sadik, and L. Ladid, "Quality of experience for streaming services: measurements, challenges and insights," *IEEE Access*, vol. 8, pp. 13341–13361, 2020, doi: 10.1109/ACCESS.2020.2965099.

[4] A. Bentaleb, B. Taani, A. C. Begen, C. Timmerer, and R. Zimmermann, "A survey on bitrate adaptation schemes for streaming media over HTTP," *IEEE Communications Surveys and Tutorials*, vol. 21, no. 1, pp. 562–585, Jan. 2019, doi: 10.1109/COMST.2018.2862938.

[5] S. Kesavan, E. Saravana Kumar, A. Kumar, and K. Vengatesan, "An investigation on adaptive HTTP media streaming Quality-of-Experience (QoE) and agility using cloud media services," *International Journal of Computers and Applications*, vol. 43, no. 5, pp. 431–444, 2021, doi: 10.1080/1206212X.2019.1575034.

[6] L. Popa, A. Ghodsi, and I. Stoica, "HTTP as the narrow waist of the future internet," in *Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks*, 2010, pp. 1–6. doi: 10.1145/1868447.1868453.

[7] X. Liu et al., "A case for a coordinated internet video control plane," in Proceedings of the ACM SIGCOMM 2012 conference on Applications, technologies, architectures, and protocols for computer communication, 2012, pp. 359–370. doi: 10.1145/2342356.2342431.

[8] M. Pathan and R. Buyya, "A taxonomy of CDNs," in *Content delivery networks*, Springer, 2008, pp. 33–77. doi: 0.1007/978-3-540-77887-5_2.

[9] D. A. S. George and A. S. H. George, "The evolution of content delivery network: how it enhances video services, streaming, games, ecommerce, and advertising," *International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering (IJAREEIE)*, vol. 10, no. 07, pp. 10435–10442, 2021, doi: 10.5281/zenodo.6788660.

[10] Z. Zeng and H. Zhang, "A study on cache strategy of CDN stream media," in *2020 IEEE 9th Joint International Information Technology and Artificial Intelligence Conference (ITAIC)*, IEEE, 2020, pp. 1424–1429. doi: 10.1109/ITAIC49862.2020.9338805.

[11] G. Peng, "CDN: Content distribution network," *arXiv preprint cs/0411069*, 2004, doi: 10.48550/arXiv.cs/0411069.

[12] M. Rahman, S. Iqbal, and J. Gao, "Load balancer as a service in cloud computing," in *2014 IEEE 8th international symposium on service oriented system engineering*, IEEE, 2014, pp. 204–211. doi: 10.1109/SOSE.2014.31.

[13] R. Pantos and W. May, "HTTP live streaming." Accessed: Mar. 01, 2025. [Online]. Available: https://www.rfc-editor.org/rfc/rfc8216

[14] T. Lyko, M. Broadbent, N. Race, M. Nilsson, P. Farrow, and S. Appleby, "Improving quality of experience in adaptive low latency live streaming," *Multimedia Tools and Applications*, vol. 83, no. 6, pp. 15957–15983, 2024, doi: 10.1007/s11042-023-15895-9.

[15] O. Oyman and S. Singh, "Quality of experience for HTTP adaptive streaming services," *IEEE Communications Magazine*, vol. 50, no. 4, pp. 20–27, 2012, doi: 10.1109/MCOM.2012.6178830.

[16] "FFMPEG HLS Parameters." Accessed: Mar. 01, 2025. [Online]. Available: https://ffmpeg.org/ffmpeg-all.html#hls-2.

[17] C. Gutterman, B. Fridman, T. Gilliland, Y. Hu, and G. Zussman, "Stallion: Video adaptation algorithm for low-latency video streaming," in *Proceedings of the 11th ACM Multimedia Systems Conference*, Association for Computing Machinery, 2020, pp. 327–332. doi: 10.1145/3339825.3397044.

[18] J. M. Martinez-Caro and M. D. Cano, "On the identification and prediction of stalling events to improve qoe in video streaming," *Electronics (Basel)*, vol. 10, no. 6, p. 753, 2021, doi: 10.3390/electronics10060753.

[19] D. Ray, V. Bobadilla Riquelme, and S. Seshan, "Prism: Handling packet loss for ultra-low latency video," in *Proceedings of the 30th ACM International Conference on Multimedia*, 2022, pp. 3104–3114. doi: 10.1145/3503161.3547856.

[20] J. Bienik, M. Uhrina, L. Sevcik, and A. Holesova, "Impact of packet loss rate on quality of compressed high resolution videos," *Sensors*, vol. 23, no. 5, p. 2744, 2023, doi: 10.3390/s23052744.

[21] S. Clayman and M. Sayıt, "Low latency low loss media delivery utilizing in-network packet wash," *Journal of Network and Systems Management*, vol. 31, no. 1, p. 29, 2023, doi: 10.1007/s10922-022-09712-1.

[22] M. Taha and A. Ali, "Smart algorithm in wireless networks for video streaming based on adaptive quantization," *Concurrency and Computing: Practice and Experience*, vol. 35, no. 9, p. e7633, 2023, doi: 10.1002/cpe.7633.

[23] M. Rudow, F. Y. Yan, A. Kumar, G. Ananthanarayanan, M. Ellis, and K. V Rashmi, "Tambur: Efficient loss recovery for videoconferencing via streaming codes," in *20th USENIX Symposium on Networked Systems Design and Implementation (NSDI 23)*, 2023, pp. 953–971. Available: https://www.usenix.org/system/files/nsdi23-rudow.pdf.

[24] M. Tüker, E. Karakış, M. Sayıt, and S. Clayman, "Using packet trimming at the edge for in-network video quality adaption," *Annals of Telecommunications*, vol. 79, no. 3, pp. 197–210, 2024, doi: 10.1007/s12243-023-00981-8.

[25] M. T. Abdullah, N. W. Abdulrahman, A. A. Mohammed, and D. N. Hama, "Impact of Wireless Network Packet Loss on Real-Time Video Streaming Application: A Comparative Study of H. 265 and H. 266 Codecs," *Kurdistan Journal of Applied Research*, vol. 9, no. 2, pp. 23–41, 2024, doi: 10.24017/science.2024.2.3.

[26] Z. Meng *et al.*, "Hairpin: Rethinking packet loss recovery in edge-based interactive video streaming," in *21st USENIX Symposium on Networked Systems Design and Implementation (NSDI 24)*, 2024, pp. 907–926. Available: https://www.usenix.org/system/files/nsdi24spring_prepub_meng.pdf.

[27] G. Carofiglio, G. Morabito, L. Muscariello, I. Solis, and M. Varvello, "From content delivery today to information centric networking," *Computer networks*, vol. 57, no. 16, pp. 3116–3127, 2013, doi: 10.1016/j.comnet.2013.07.002.

[28] M. Ghaznavi, E. Jalalpour, M. A. Salahuddin, R. Boutaba, D. Migault, and S. Preda, "Content delivery network security: A survey," *IEEE Communications Surveys & Tutorials*, vol. 23, no. 4, pp. 2166–2190, 2021, doi: 10.1109/COMST.2021.3093492.

[29] W. E. Shabrina, D. W. Sudiharto, E. Ariyanto, and M. Al Makky, "The QoS improvement using CDN for live video streaming with HLS," in *2020 International Conference on Smart Technology and Applications (ICoSTA)*, IEEE, 2020, pp. 1–5. doi: 10.1109/ICoSTA48221.2020.1570613984.

[30] U. Patel, S. Tanwar, and A. Nair, "Performance analysis of video on-demand and live video streaming using cloud based services," *Scalable Computing: Practice and Experience*, vol. 21, no. 3, pp. 479–496, 2020, doi: 10.12694/scpe.v21i3.1764.

[31] K. B. Sangeetha and V. S. K. Reddy, "An Effective Investigation for Quality of Service Enhancement of Content Delivery Network for HTTP Live Streaming Using H. 265," *Scalable Computing: Practice and Experience*, vol. 25, no. 4, pp. 2703–2710, 2024, doi: 10.12694/scpe.v25i4.2830.

[32] H. Wu, M. Claypool, and R. E. Kinicki, "Guidelines for Selecting Practical MPEG Group of Pictures.," in *IASTED International Conference on Internet and Multimedia Systems and Applications (EuroIMSA)*, Innsbruck, Austria: Citeseer, 2006, pp. 61–66. doi: 10.5555/1169167.1169178.

[33] K. Panagidi, C. Anagnostopoulos, and S. Hadjiefthymiades, "Optimal grouping-of-pictures in iot video streams," *Computer Communications*, vol. 118, pp. 185–194, 2018, doi: 10.1016/j.comcom.2017.11.012.

[34] J. Gomez, E. F. Kfoury, J. Crichigno, and G. Srivastava, "A survey on network simulators, emulators, and testbeds used for research and education," *Computer Networks*, vol. 237, p. 110054, 2023, doi: 10.1016/j.comnet.2023.110054.

[35] T. Hidayat, Y. Azzery, and R. Mahardiko, "Load balancing network by using round Robin algorithm: a systematic literature review," *Jurnal Online Informatika*, vol. 4, no. 2, pp. 85–89, 2019, doi: 10.15575/join.v4i2.446.

[36] "EnvoyProxy Supported Load Balancers." Accessed: Mar. 01, 2025. [Online]. Available: https://www.envoyproxy.io/docs/envoy/latest/intro/arch_overview/upstream/load_balancing/load_balancers.

[37] D. R. I. M. Setiadi, "PSNR vs SSIM: imperceptibility quality assessment for image steganography," *Multimedia Tools and Applications*, vol. 80, no. 6, pp. 8423–8444, 2021, doi: 10.1007/s11042-020-10035-z.

Appendices

Table 4S: Raw data from the tested cases in scenario 1.

Case	Segment length	GOP size	List length	1% PLR	2% PLR	3% PLR	4% PLR	5% PLR	6% PLR
1	1 second	12 frames	5 segments	50.0411	50.0378	46.3973	44.6780	38.8172	33.6964
2	1 second	12 frames	10 segments	50.0416	50.0410*	50.0410	50.0377	50.0438	45.8729
3	1 second	24 frames	5 segments	50.0125	50.0126	47.2241	44.3270	37.3120	34.5264
4	1 second	24 frames	10 segments	50.0121*	50.0121*	50.0121	50.0192	50.0189	48.7280
5	1.5 seconds	12 frames	5 segments	50.0427*	50.0427*	50.0427	47.5523	46.0695	33.8686
6	1.5 seconds	12 frames	10 segments	50.0378*	50.0378*	50.0378	50.0427	50.0449	50.0443
7	1.5 seconds	36 frames	5 segments	50.0195*	50.0195*	50.0195	48.8468	42.6126	34.8299
8	1.5 seconds	36 frames	10 segments	50.0368*	50.0368*	50.0368	50.0188	50.0298	49.1417
9	2 seconds	12 frames	5 segments	50.0045*	50.0045*	50.0045	50.0459	50.0429	50.0452
10	2 seconds	12 frames	10 segments	50.0430*	50.0430*	50.0430	50.0430	50.0425	50.0442
11	2 seconds	24 frames	5 segments	50.0106*	50.0106*	50.0106	50.0027	50.0068	46.0885
12	2 seconds	24 frames	10 segments	50.0152*	50.0152*	50.0152	50.0025	50.0075	50.0110
13	2 seconds	48 frames	5 segments	50.0396*	50.0396*	50.0396*	50.0396	50.0431	45.4919
14	2 seconds	48 frames	10 segments	50.0403*	50.0403*	50.0403	50.0393	50.0436	50.0426

*Value not explicitly tested; it is derived from the result at higher packet loss, where no observable degradation was detected.

Table 5S: Raw data from the tested cases in scenario 2.

Case	Load balancing algorithm	Segment length	GOP size	List length	1% PLR	2% PLR	3% PLR	4% PLR	5% PLR	6% PLR
1	Round Robin	1 second	12 frames	5 segments	0.07%	0.17%	2.09%	6.05%	8.90%	13.79%
2	Round Robin	1 second	12 frames	10 segments	0.05%	0.01%	1.56%	5.42%	6.50%	11.81%
3	Round Robin	1 second	24 frames	5 segments	0.19%	0.89%	4.06%	5.48%	8.46%	13.44%
4	Round Robin	1 second	24 frames	10 segments	1.17%	0.62%	2.83%	7.60%	9.09%	13.50%
5	Round Robin	1.5 second	12 frames	5 segments	0.00%	0.01%	0.91%	0.89%	3.84%	8.60%
6	Round Robin	1.5 second	12 frames	10 segments	0.59%	0.46%	3.16%	4.34%	6.51%	8.77%
7	Round Robin	1.5 second	36 frames	5 segments	0.00%	0.01%	0.81%	1.14%	3.68%	10.05%
8	Round Robin	1.5 second	36 frames	10 segments	0.00%	0.78%	1.62%	3.15%	6.70%	9.72%
9	Ring Hash	1 second	12 frames	5 segments	0.07%	0.15%	0.40%	3.01%	7.41%	13.27%
10	Ring Hash	1 second	12 frames	10 segments	0.02%	0.15%	1.47%	2.30%	4.48%	11.34%
11	Ring Hash	1 second	24 frames	5 segments	0.03%	0.09%	0.53%	1.60%	5.83%	10.46%
12	Ring Hash	1 second	24 frames	10 segments	0.02%	0.37%	0.13%	0.26%	2.60%	9.48%
13	Ring Hash	1.5 second	12 frames	5 segments	0.00%	0.00%	0.34%	0.27%	1.80%	5.43%
14	Ring Hash	1.5 second	12 frames	10 segments	0.00%	0.01%	0.02%	0.06%	1.39%	5.28%
15	Ring Hash	1.5 second	36 frames	5 segments	0.01%	0.00%	0.08%	1.14%	3.31%	7.03%
16	Ring Hash	1.5 second	36 frames	10 segments	0.00%	0.01%	0.02%	0.92%	3.26%	6.72%