# Development of User-Centric HTTP Adaptive Video Streaming Technique

Temitope A. Komolafe[1], Akinola A. Ibikunle[1], Sunday Adeola Ajagbe[2,3*], Ayodeji O. Oluwatope[4], Korede Israel Adeyanju[5], and Adedayo A. Olayiwola[6]

[1]Department of Computer Engineering, Faculty of Engineering, Redeemer's University, Ede, Nigeria
[2]Department of Computer Science, Faculty Science Agriculture and Engineering, University of Zululand, Kwadlangezwa, South Africa
[3]Department of Computer Engineering, Faculty Engineering and Technology, Abiola Ajimobi Technical University, Ibadan, Nigeria
[4]Department of Computer Engineering, Faculty of Technology, Obafemi Awolowo University, Ile Ife, Nigeria
[5]Department of Computing, College of Business, Technology and Engineering, Sheffield Hallam University, Sheffield, United Kingdom
[6]Department of Computer Engineering, Ladoke Akintola University of Technology, Ogbomoso, Nigeria
Email: komolafet@run.edu.ng (T.A.K.); ibikunleakin@run.edu.ng (A.A.I.); saajagbe@pgschool.lautech.edu.ng (S.A.A.); aoluwato@oauife.edu.ng (A.O.O.); deyanjukorede@gmail.com (K.I.A.); aaolayiwola42@lautech.edu.ng (A.A.O.)
*Corresponding author

Manuscript received January 5, 2026; revised January 22, 2026; accepted March 6, 2026; published April 15, 2026

***Abstract*—User-perceived Quality of Experience (QoE) is vital in Internet video applications, influencing revenue for content providers and delivery systems. Due to limited network support for QoE optimization, bottlenecks may arise anywhere in the delivery system. Consequently, a robust bitrate adaptation algorithm in client-side players is critical to ensure good user experience. This study proposes a client-side buffer control model for mobile video streaming to enhance QoE under dynamic network conditions, supporting Dynamic Adaptive Streaming over HTTP (DASH). The model employs a Double Leaky Bucket (DLB) traffic shaping scheme with a First-In-First-Out scheduler. A DLB-based algorithm was developed in C++ and simulated using Network Simulator 3 (NS-3) with a trace-based dataset of a 4420-second video comprising 221 segments across 8 representations. Simulation parameters included segment duration, buffer size, packet size, and number of packets. Performance was evaluated using Packet Loss, Jitter, and Throughput metrics, comparing the DLB model to the existing Single Leaky Bucket (SLB) approach. Results showed DLB reduced packet loss from 60.8% to 39.2%, jitter from 53.4% to 46.6%, and increased throughput from 33.7% to 66.3%. These improvements demonstrate that the DLB algorithm significantly enhances mobile video streaming QoE by ensuring smoother playback under fluctuating network conditions.**

***Keywords*—Dynamic Adaptive Streaming over HTTP (DASH), buffer control model, Double Leaky Bucket (DLB), Quality of Experience (QoE), traffic shaping**

## I. Introduction

Video streaming traffic has become increasingly dominant over past few years. Mansy [1] stated that video streaming traffic accounted for 57% of total Internet traffic in 2012, according to Cisco's Visual Networking Index (VNI), and it is predicted to exceed 69% in 2017. In another work, it was established that today's internet is bigger than ever and still growing fast [2]. A historical projection from Cisco's Visual Networking Index (2008) forecasted that global IP traffic would increase nearly 100-fold between 2005 and 2020. This growth trajectory has largely been realized and surpassed, with current trends being shaped by the dominance of video and cloud services. Industry analyses have consistently shown that global internet traffic has grown by orders of magnitude over the past 2 decades, a trend accelerated by the rise of high-bandwidth applications like video streaming and large-scale cloud computing. Globally, video traffic continues to dominate internet bandwidth. Recent forecasts, including those from Cisco's updated analyses, indicate that video now constitutes over 80% of all internet traffic, a trend accelerated by the growth of streaming, social media, and video conferencing. This expansion is due in part to the widespread availability of broadband access networks, which provide users with high-bandwidth Internet access, allowing users to stream content at higher quality levels. Furthermore, mobile devices (smartphones and tablets) are driving online video demand, which accounts for a large portion of this development.

Taiwo *et al.* [3] pointed video streaming dominates global bandwidth. While Netflix remains one of the top contributors, YouTube often matches or exceeds its traffic share in many regions, with both services collectively accounting for a major portion of internet traffic. Wu *et al.* [4] stated that the increase in online video content is accompanied by an increase in video quality, which in turn increases the network throughput demands.

The Internet was not developed with multimedia streaming in mind. Transmission Control Protocol (TCP) was even largely thought to be a poor transport protocol for streaming multimedia. As a result, the Real-time Transport Protocol (RTP) was developed as the primary transport protocol for transmitting multimedia over the Internet. For many years, RTP over multicast was thought to be the best means to provide multimedia over the Internet. Unfortunately, this turned out to be false for a number of reasons.

RTP is typically utilized on top of User Datagram Protocol (UDP), and it was not designed with dependable data transmission and congestion control in mind. RTP congestion control and reliability, on the other hand, are still outstanding issues. The widespread adoption of Content Distribution Networks (CDNs) as the primary platform for online content delivery has solidified HTTP-based streaming as the de facto industry standard. This approach leverages standard web protocols and infrastructure, significantly reducing the operational complexity and cost of deploying and managing CDNs for large-scale video distribution. Its proven efficacy in delivering content to a global audience of billions has made HTTP adaptive bitrate streaming the foundational technology for virtually all major on-demand and live-streaming services today.

Video streaming is often degraded by various forms of losses, such as packet loss over the Internet. When packets going across a network fail to reach at their intended destination, this is known as packet loss. Packet loss is a critical network performance measure for applications such as traditional server–client video streaming transmission, peer-to-peer systems, Vehicular Ad-hoc NETworks (VANET), and congestion control [5]. Despite the efforts of scholars to reduce the negative impact of packet loss through error concealment algorithms, packet prioritizing strategies, and other methods, packet loss will never be eliminated due to the dynamic nature of IP networks.

This work focused on traffic shaping to enhance user quality of experience during video streaming by adapting to the dynamically network conditions to support adaptive streaming over HTTP (HAS) and reduce the rate of packet loss. This research has contributed a client-side buffer control technique for mobile streaming using HAS to the existing body of knowledge in multimedia communication. Also, quality of experience perceived by the user during mobile video streaming using HAS has been improved.

Unlike traditional traffic shaping methods such as the Single Leaky Bucket (SLB), which discard packets immediately upon buffer saturation, this study introduces a Double Leaky Bucket (DLB) mechanism. The novelty lies in the hierarchical buffering strategy where a secondary bucket ($\beta 2$) captures overflow from the primary bucket ($\beta 1$). This allows the system to absorb transient traffic bursts and reduce packet loss without requiring immediate re-requests from the server, which is a significant departure from standard DASH buffer-based methods.

The structure of the paper in the subsequence section is such that Section II gives the review of related works, model description is in Section III, results and discussion is in Section IV, conclusion is in Section VI.

## II. Literature Review

Web-based systems are a popular way to offer streaming video content, which is now one of the most frequent Internet applications. The visual quality that viewers perceive is the key to a successful video streaming system. Packet loss is one of the most significant difficulties for video streaming services. Studies show that existing solutions have flaws in many tests, since they have failed to give users with a comfortable viewing experience. Few of these solutions are looked into briefly under this section.

Recent advancements in HTTP Adaptive Streaming (HAS) have focused on mitigating the impact of dynamic network conditions on User Experience. Rath *et al.* [6] and Kim *et al.* [7] established foundational approaches to congestion control using single-buffer traffic shaping. While these methods provided basic rate stability, they often suffered from high packet loss during rapid throughput fluctuations. Mansy [1] proposed multipath transmission framework for real-time video traffic to improve service by combining bandwidth and taking use of path diversity. When streaming over the internet, real-time video traffic has severe latency requirements and is bandwidth intensive. Due to packet loss, jitter, and inadequate bandwidth, single path transmission may provide customers with unsatisfactory visual quality. These issues led the authors to propose multipath transmission. The congestion management method and packet scheduling algorithm were built on the server side and work on the application layer in this research effort, and real-time video traffic was sent through UDP. Studies showed that TCP-based adaptive rate control can achieve higher performance than UDP due to TCP's adaptability to changing network conditions, which is fueled by improved network performance. Zhang *et al.* [8] pointed packet loss is a performance indicator for video applications, congestion control, and routing since it characterizes the network's performance. As a result, their contributions to the study are fourfold. First, they created a theoretical model that characterizes the influence of packet loss on video distortion for each frame type, as well as the degree of video quality degradation in the case of various packet losses. Second, they employed simulations and tests to investigate the effect of I, P, and B packet loss on video quality [9]. Finally, a packet loss estimation approach termed Packet Loss Measurement Based on User Data (PLBU) is proposed. The packet loss pattern of video cannot be determined using PLBU measurement results.

Hu and Zhang [10] pointed that Dynamic Adaptive Streaming over HTTP (DASH) is a standardized technology designed for Over-The-Top (OTT) adaptive video streaming. In DASH, a video source is segmented into a series of small, sequentially numbered chunks, each typically 2 s to 10 s long. Each segment is then encoded at multiple bitrates and quality levels. This allows a client player to dynamically select the next segment at the most appropriate bitrate based on real-time network conditions, ensuring seamless playback. The video server is essentially a Web server that hosts video files that represent these segments, as well as a manifest file that describes them and their bitrates. The video is streamed by a client receiving chunks from the video server through HTTP. The client assesses the available bandwidth to the server while downloading segments and switches between video bitrates accordingly.

Høiland-Jørgensen *et al.* [11] pointed that last several years has seen a renewed interest in smart queue management to curb excessive network queueing delay traffic congestions, as people have realized the prevalence of buffer bloat in real networks. Over buffering is becoming common in today's data networks. While adequate buffer management may potentially help in limiting packet drops, buffer overflow may result in high end-to-end latency [12].

Ajagbe *et al.* [13] conduct a comprehensive literature review to identify existing research gaps and recommend areas for future improvement, focusing on security demands and technological innovation. The methodology involves selecting relevant studies, conducting a descriptive analysis of the literature, and classifying the papers based on key phrases used. The main finding is that research has paid limited attention to logistics management within the broader context of cybersecurity and the Supply Chain (SC), highlighting an urgent need for improvement, particularly in human-dominated areas like Saudi Arabia during high-traffic seasons such as the Hajj and Umrah. The primary limitation is the identified lack of focus on logistics-specific security. Consequently, the authors suggest redirecting research to leverage Artificial Intelligence (AI) and Internet of Things (IoT) to enhance security for life and property in these densely populated seasonal environments.

Oyediran *et al.* [14] proposes and evaluates a novel machine learning approach, the White Shark Optimizer-Support Vector Machine (WSO-SVM), specifically designed for gender identification from video data. The objective is to leverage the strengths of both algorithms, the bio-inspired White Shark Optimizer for efficient parameter selection and the Support Vector Machine (SVM) for powerful classification, to significantly enhance gender identification accuracy and robustness. The method involved conducting extensive experiments on a diverse dataset and comparing the WSO-SVM's performance against conventional SVM methods and other state-of-the-art techniques. The work demonstrated that the proposed WSO-SVM achieved superior accuracy, specifically an overall accuracy of 93.00%, and exhibited robustness in handling complex real-world variables like variations in lighting, poses, and facial expressions. Furthermore, it reported strong classification metrics including an average Sensitivity of 93.06% and Specificity of 92.86%, with a quick recognition time of 45.83 s.

Vergados *et al.* [15] pointed a fuzzy control-based algorithm for rate adaption in Motion Picture Experts Group-Dynamic Adaptive Streaming over HTTP (MPEG-DASH) is proposed that employs fuzzy logic to control the buffering time and the resolution in order to avoid buffer overflows. The resolution of the next video segment that each client should get from the server is determined using this method. This prevents buffer overflows and unwanted bit rate fluctuations by keeping the buffering period at each client above a specified buffering time. The presented rate adaption technique has a positive effect on the quality each user sees, according to simulation results. To mitigate buffer underruns, the proposed technique seeks to keep the buffering duration at the client above the target buffering time designated as T, while also keeping the difference between the current and previous resolutions near to zero. Although MPEG-Dynamic Adaption Streaming over HTTP (MPEG-DASH) removes data packet loss, playback interruptions may still occur depending on the recipient's rate adaptation method and network circumstances.

Bassey *et al.* [16] addresses the challenge of successful live video streaming over the internet by mitigating packet loss and jitter. They proposed an Adaptive Media Play-out (AMP) algorithm to securely transfer packets and enhance visual quality. The core of their approach uses the Peak Signal-to-Noise Ratio (PSNR) as the video quality assessment metric to make streams more tolerant to packet loss. The analysis is based on 2 premises: that video quality deteriorates due to packet loss, and that users cannot tolerate frames below a specified PSNR threshold. Simulation experiments demonstrated that the AMP algorithm outperforms traditional buffering by establishing that a higher average frame rate corresponds to a higher PSNR, a lower loss rate, and better video quality.

Zhou [17] focus on cloud video service delay announcements. He pointed out that because video data is so much larger than regular data like text, image, or music, large-scale video services in the cloud generate substantial delays, lowering consumer QoE. He further stated that, in this scenario, delay announcement is beneficial because it instructs some users to leave the waiting line immediately or abandon it during the waiting period, reducing the average waiting time while having little effect on the QoE of other users. However, how to make a simple and efficient delay announcement in the cloud mobile media environment is challenging problem and the author analytically studied the characteristics of delay announcement by analyzing the components of the user response. By developing an objective user response function, the author created a QoE-driven delay announcement method.

Recent studies emphasize that the future of HAS lies in moving beyond purely reactive network-based models toward proactive, user-centric frameworks. Sculpting the future of adaptive streaming requires algorithms that integrate machine learning to optimize content delivery based on individual user behaviour and preferences [18]. Furthermore, a recent study outlines that as streaming scales toward immersive media and 5G/6G networks, minimizing latency and stabilizing buffer occupancy remain paramount challenges [19]. Our proposed Double Leaky Bucket (DLB) model aligns with these trends by introducing a client-side safety net that preserves Quality of Experience (QoE) through more granular packet-level flow control.

### *A. Progressive Video Streaming vs HTTP Adaptive Video Streaming*

A progressive video stream is a single video file that is transmitted over the internet in real time. The video file will always be the same, regardless of the device playing it. With progressive download, a server stores one or several versions of a media file and advertises their location through a manifest file (or index file). To play a video, users first request the manifest. Based on what is available, they choose the characteristics of the video (quality, bitrate, frame rate, resolution, etc.) and keep them for the entire session. They then send byte-range requests to progressively download the video from the server, relying on HTTP to take care of the entire delivery. The technique is unsuitable for mobile environments, where bandwidth varies considerably more than in static environments [20]. Progressive download does not support live streaming [21]. Progressive Download is the first method of providing video via HTTP/TCP; the client just downloads the complete (one) video file (at constant video quality) as quickly as TCP allows. Before the download is finished, the video player on the client begins playing the video. One significant disadvantage of this method is that different clients with different capacities and network connections receive the same video quality, which might result in unwelcome playback stalls.

HTTP Adaptive Streaming (HAS), also known as Dynamic Adaptive Streaming over HTTP (DASH) [22], transmits video to users in the most efficient and usable quality possible for each individual user. Even during a streaming session, clients can adjust video quality dynamically in response to changing network conditions. On today's Internet, HTTP Adaptive Streaming (HAS) is commonly used. It adapts the quality of a video streaming session to the needs of the end-user by constantly modifying it to network conditions. DASH, unlike other UDP-based methods, is based on TCP transport. Despite the fact that different video streaming technologies are still in use, the video streaming industry has landed on DASH as the primary component of video transmission

over the Internet.

### B. Quality of Experience (QoE) and Quality of Service (QoS)

HTTP Adaptive Streaming (HAS) has recently gotten a lot of attention from researchers [23], and it's now employed by the vast majority of video streaming services like Netflix and YouTube [24]. With the rise in video streaming demand, there appears to be a growing demand for Quality of Experience (QoE), which considers the end user's experience and satisfaction [25].

Recognizing the perceptions and requirements of end users is critical for the development of future services as well as the enhancement of current technologies and services. While QoS has typically been used to assess the performance of a service, it does not account for end-user-related aspects (user expectation, environmental factors, etc.). Furthermore, QoS is restricted to telecommunication services and is solely based on technical metrics. On the other side, Quality of Experience (QoE) extends beyond telecoms.

### C. Performance Metrics

The following are the performance metrics used in this study.

(i) Throughput: The average data rate of a source delivering packets and received by the receiver is known as throughput (T). It is defined as the number of packets successfully transmitted in a particular amount of time, and is commonly measured in bits per second (bps), such as megabits per second (Mbps) or gigabits per second (Gbps).

(ii) Packet loss: Packet Loss (PL) refers to the probability of a packet to be lost while it transits from a source node to the final destination. Packet loss in this research work was as a result of buffer overflows at the client side while downloading packets from video files. Buffer overflows occur due to following reasons: The more packets arrive, the higher buffer utilization and lesser available buffer space. Therefore, when buffer space used (utilization) is equal to buffer capacity, buffer overflows set in which results in packets loss. PL, the amount of failed data, is measured relative to the total sent packets, that is number of packets not received (NPL) per time (T). Eq. (1) shows the general equation for the packet lost.

$$Generally,\ PL = \frac{NPL}{T} = packets/time \qquad (1)$$

(iii) Jitter: The variation in packets delivery affecting packets accuracy, timeliness and delivery. It is the delay that varies over time when the signal diminishes. Jitter is the term used to refer to variation in latency of packet. It's measured in milliseconds and can be devastating to real-time services such as video streaming. Jitter is when packets don't arrive in the same order they were sent, the longer data packets take to arrive, the more jitter can negatively impact the video.

## III. Model Description

The shaping mechanism system was utilized as a double-leaky bucket algorithm as shown in Fig. 1, which necessitated the usage of a high-speed network. A leaky bucket behaves similarly to a bucket with a hole in the bottom, as the name suggests. DLB was used to evaluate excess packets that are often discarded by the existing leaky bucket technique. DLB was used to formulate this research model. The formulated algorithm was developed using C++ programming language and incorporated to the existing TOBASCO adaptation algorithm. The double leaky bucket is a traffic control mechanism consisting of 2 buckets connected to the scheduler through its bottom holes. The flow was defined by the arrived packet size (P), bucket size in bytes (discuss later) for the first bucket (Q) and second bucket size € respectively. The main goal of the scheme lies in the evaluation of excess packets that were discarded by the existing technique (SLB). This excess packet is denoted by $e_1$ from buffer 1 ($\beta 1$) and $e_2$ from buffer 2 ($\beta 2$).

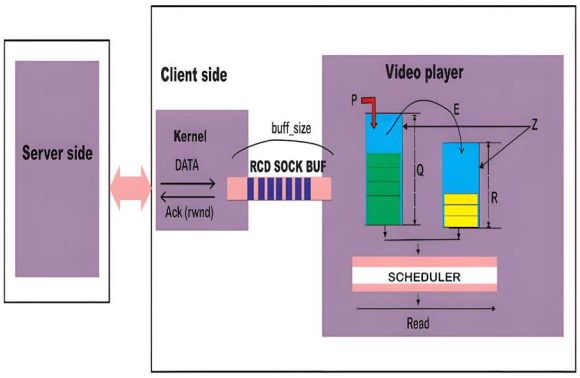


Fig. 1. Client-side buffer control technique. (P = arrived packet size; Q = size of the first bucket; R = size of the second bucket; E = excess packets from; Z = leaky buckets; Scheduler = FIFO (First in First Out).

New packets are added to the bucket at rate of *r* packets/s, the maximum packet that can be accumulated in a bucket shall be discussed later. The Double Leaky Bucket (DLB) operates as a dual-buffer traffic shaping mechanism where both $\beta 1$ and $\beta 2$ have direct, independent links to a centralized FIFO scheduler. When the primary buffer $\beta 1$ reaches capacity, incoming packets are diverted to the secondary buffer $\beta 2$ (which serves as a temporary overflow buffer) to prevent immediate dropping except in exceptional cases. The scheduler continuously monitors both buckets, processing packets in the order they were received to ensure timing accuracy. It serves packets in FIFO order across both buffers. The FIFO scheduler operates as a continuous background process that polls both $\beta_1$ and $\beta_2$ and transmits packets in order of arrival, regardless of their originating buffer. This configuration allows the system to evaluate and recover excess packets that would typically be discarded by a Single Leaky Bucket (SLB) approach, effectively reducing packet loss. The more packets arrived, the more buffer utilization and less available space. When buffer used equals buffer capacity, buffer overflow sets in.

### A. DLB Algorithm Pseudocode

The procedure defines in DLB Algorithm Pseudocode show how packets are handled between the 2 buckets and the scheduler as shown in the Algorithm 1

**Algorithm 1: DLB Algorithm**

START
Initialize:
  Q ← empty queue (capacity = *Qn*)
  R ← empty queue (capacity = *Rn*)
  PL ← 0
**WHILE** system is active *DO*
  // Packet arrival

```
    RECEIVE Pi
    IF Q is not full THEN
      ENQUEUE Pi into Q//Add packet to Buffer 1
    ELSE
      IF R is not full THEN
        ENQUEUE Pi into R//Add packet to Buffer 2
      ELSE
        DROP Pi
        PL ← PL + 1// Packet loss counter
      ENDIF
    ENDIF
    // Scheduler runs continuously (independent process)
    IF Q not empty THEN
      DEQUEUE packet from Q
      SEND to client
    ELSE IF R not empty THEN
      DEQUEUE packet from R
      SEND to client
    ENDIF
  ENDWHILE
  STOP
  Where:
  Pi = Arrived packet size
  Q = Buffer 1 current level
  Qn = Buffer 1 full capacity
  R = Buffer 2 current level
  Rn = Buffer 2 full capacity
  e = Excess packet from Buffer 1
  e2 = Excess packet from Buffer 2
  PL = Packet loss counter
```

*B. Workflow Description*

Step 1: The DASH client requests a video segment based on TOBASCO's bitrate selection.

Step 2: Arriving packets enter the DLB system at the application layer.

Step 3: Packets are enqueued in $\beta 1$ if space is available; otherwise, they overflow into $\beta 2$.

Step 4: The FIFO scheduler dequeues packets from the buckets at a constant leak rate, ensuring a smooth flow to the video player.

*C. Double Leaky Bucket Parameters*

To manage the flow of traffic, the double leaky bucket employs 2 parameters.

(i) Rate of data: The number of packets per second that leak from the leaky bucket, allowing data to access the network.

(ii) Burst size: The maximum number of packet groups that can be stored in the bucket.

DLB comprises 2 leaky buckets $\beta 1$ and $\beta 2$. The following is a summary of each bucket's capacity.

$\delta_{12}$: the coefficient of capacity of the 2 buckets, which is 1 (indicating the presence of packets in the bucket) and value 0 (indicating absence of packets in the bucket).

$P$: an output data rate.

$Q$: the bucket size of $\beta 1$.

$R$: the bucket size of $\beta 2$.

Max_Size: the number of packets.

$I$: a rate of data entry.

DLB_Max: the DLB's total capacity.

Eq. (2) and Eq. (3) establish the foundational relationship between the primary $\beta 1$ and secondary $\beta 2$ buckets, while Eq. (4) and Eq. (5) introduce the coefficient $\delta_{12}$ to represent the dynamic activation of the hierarchical overflow mechanism based on real-time traffic conditions.

$$DLB\_Max=\alpha \tag{2}$$

$$DLB\_Max=\alpha(Q+R) \tag{3}$$

$$DLB_{Max}=\delta_{12}(Q+R) \tag{4}$$

Either there is a very high-speed network or not, there will be packets in $\beta 1$. Hence, the coefficient of $Q$ will always be 1. Eq. 6 shows the DLB burst size.

Therefore,

$$DLB_{Max}=Q+\delta_{12}R \tag{5}$$

where,

$$\delta_{12}=\begin{cases}1, & if\ I>P \text{ and } Q<Max\_\text{Size}\\ 0, & Otherwise\end{cases} \tag{6}$$

*D. Model Simulation*

The simulation program chosen was Network Simulator-3 (NS-3). NS-3 is a C++-based discrete event network simulator. Using a non-blocking sockets interface, we were able to create networked programs using NS-3, which is open source, extendible, and actively maintained. These were then 'installed' on nodes that were joined together using C++ programs to construct the desired network. NS-3 is a suitable choice for this project because of its ease of use and active development, as well as the big open-source community that surrounds it. Because of its popularity among researchers, existing models in NS-3, such as TCP implementations and queueing algorithms, have matured and been well tested. We have included a module to NS-3 that contains generic client and server application models as well as multiple implementations of the basic traffic shaping system components in this effort. Clients and servers can be matched with the proper components plugged in to construct different traffic shaping systems. We also employed a scripting tool that lets us specify network and application settings in external text files, which are then interpreted by NS-3 to generate and run the simulation.

The ns3 building module includes a class (Building) that mimics the presence of a building in a simulation scenario, as well as the ability to specify the position, size, and attributes of buildings in the simulated area, as well as the placement of nodes inside those buildings.

The module is designed to function with other NS-3 wireless technologies and is not dependent on any specific Long-Term Evolution (LTE) network type (e.g. wifi, wimax). Different types of structures (e.g., residential), wall materials (e.g., wood), and the number of floors and rooms are all supported by the building module.

Table 1. The technical table

| Parameter | Value/Model |
|---|---|
| Simulator | NS-3 |
| Wi-Fi Standard | IEEE 802.11n |
| PHY Rate | MCS 7 (65 Mbps) |
| MAC Layer | AdhocWifiMac |
| Propagation Loss Model | Log-Distance Path Loss Model |
| Fading Model | Nakagami-m Fading |
| Mobility Model | RandomWalk2dMobilityModel |
| MTU Size | 1446 bytes |
| Frequency/Bandwidth | 2.4 GHz/20 MHz |
| Transport Protocol | TCP |

To ensure the reproducibility of the simulation results and characterize the radio environment, the key network and Physical (PHY) and Medium Access Control (MAC) parameters configured in the NS-3 environment are summarized in Table 1.

The integrated adaptation algorithm (TOBASCO), 20 s segments, 3 clients, buffer size, and segment size are all included in the model's input. It also has a simulation ID, which was used to identify the output files that were generated. The segment sizes provide the simulation's duration; the simulation ends when all clients have finished watching the entire video.

The network topology used is explained as follows. The Buildings Model included with NS-3 was used to construct a structure of 60 × 36 × 36 meters, with 6 stories and 8 × 3 rooms on each floor. The Access Point (AP) was set to (1, 1, 1), which corresponds to the first floor's left bottom room. The access point was installed near the base of the wall in the first room on the first floor, and this location was chosen at random because clients are scattered throughout the building. Using the NS-3 framework's random position allocator, clients were deployed at random across the building. 1446 bytes was chosen as the Maximum Transmission Unit (MTU) [26, 27].

The metrics of the network were logged into files which were used to analyze the algorithm and finally obtain a graphical plot. The client module was responsible for all logging. Download, buffer level, playback, packet loss, delay, and adaptation are the 6 categories that it falls within. The logging was implemented in the "tcp-stream-client.cc" file. Realistic segment sizes, ideally from an actual DASH-encoded video, are required to obtain valid findings. The video utilized in this study is a 4420 s clip with a 1920 × 1080 resolution. The video is divided into 221 segments of various lengths, such as 2 s, 4 s, 10 s, 16 s, and 20 s (20 s was employed in this study), and encoded in 8 distinct representations with average bit rates ranging from 0.088 Mbps to 50.5 Mbps.

To evaluate the simulated model, an NS-3 trace-based dataset of video clips was obtained. The datasets were utilized in the evaluation of Dynamic Adaptive Streaming over HTTP (DASH) using High Efficiency Video Coding (HEVC).

### *E. Buffer Size*

The 1.5× ratio between the Double Bucket (DB) and Single Bucket (SB) capacities was selected as an optimized design parameter. While $\beta 1$ (equivalent to SB) handles the baseline traffic flow, the additional 0.5× capacity provided by $\beta 2$ serves as a dedicated buffer for evaluating and recovering excess packe € ($e$) from $\beta 1$ during dynamic network fluctuations. This specific ratio ensures that the system provides sufficient overhead to reduce packet loss

The buffer size used for simulation was categorised into four groups. Each group is a pair consisting Single Buffer (SB) and Double leaky Bucket (DB). SB (which can also be referred to as Single Leaky Bucket (SLD)) has only one buffer while DB (also known as double leaky bucket) has 2, but not equal in size, buffers. The buffer 1 in DB has the same size with SB and the buffer 2 is an added or extended one, lesser in size to buffer 1. The size of the buffer 2 in DB was obtained as follow:

For every size (say X) used for SB was also used for buffer 1 in DB and the size of buffer 2 was taken to be 50% of the size of buffer 1. That is, SB size = X

DB: Buffer 1 = X; Buffer 2 = 50% of X = 0.5X (this is then added to the original size). The double leaky bucket size used was gotten according to Eq. (7).

$$Buffer\ 2 = 0.5X + 50 \tag{7}$$

Table 2. Category of buffer size for simulation

| Type | Category (kB) | SB | DB | | | No of Simulation |
|---|---|---|---|---|---|---|
| | | | B1 | B2 (50% of B1) | DB Total | |
| 1 | 50 vs 75 | 50 | 50 | 25 | 75 | 2 |
| 2 | 100 vs 150 | 100 | 100 | 50 | 150 | 2 |
| 3 | 250 vs 375 | 250 | 250 | 125 | 375 | 2 |
| 4 | 500 vs 750 | 500 | 500 | 250 | 750 | 2 |

4 different sizes were used for the simulation and the simulation was carried out 8 times, 2 times for each category of buffer size. The 4 categories size used are summarized in Table 2.

## IV. Results and Discussion

### *A. Packet Loss Performance Metric*

Fig. 2, Fig. 4 and Fig. 5 show the simulation results for total packet loss when SB is set to: 50 kB, 100 kB, 250 kB and DB set to: 75 kB, 150 kB, 375 kB respectively. The graph in Fig. 3 presents average packet loss when SB is set to 50 kB and DB set to 75 kB. The average packet loss is obtained by dividing total packet loss over a space of 25 points. Fig. 3 is plotted for clarity as against result presented in Fig. 2. Also, Tables 3–5 present packet loss comparison when SB is set to: 50 kB, 100 kB, 250 kB and DB set to: 75 kB, 150 kB, 375 kB respectively.

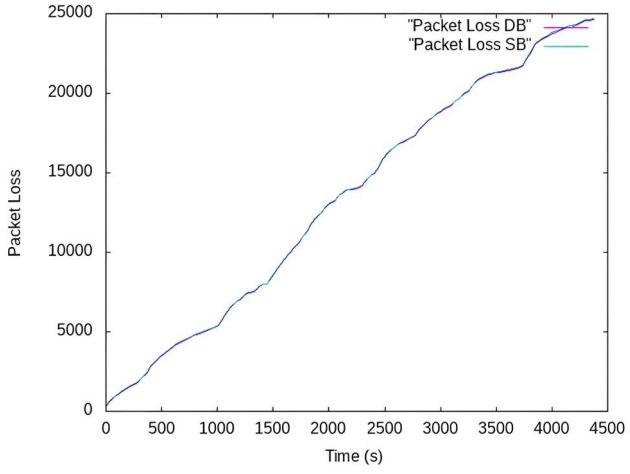


Fig. 2. Total packet loss when SB is set to 50 kB and DB set to 75 kB.

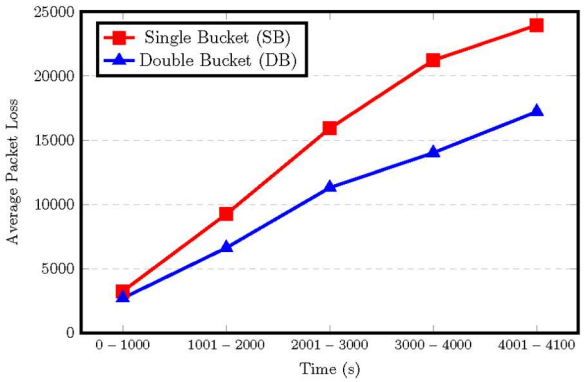


Fig. 3. Avg packet loss when SB is set to 50 kB and DB set to 75 kB.

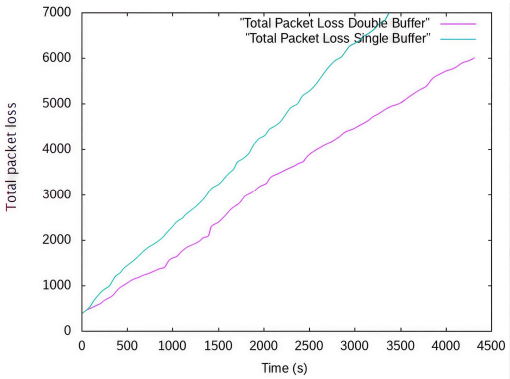


Fig. 4. Total packet loss when SB is set to 100 kB and DB set to 150 kB.

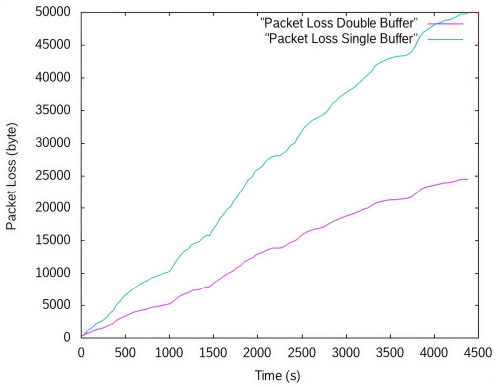


Fig. 5. Total packet loss when SB is set to 250 kB and DB set to 375 kB.

With the client's continuous success of packet receptions, the size of receiver socket buffer increases which may result in buffer overflows as seen in Figs. 2–5. Comparing the existing technique and the proposed model, there are buffer overflows more in SB than DB which results in packet loss. Packet loss occurs due to buffer overflow and this affects the QoE of the user. Base on the results obtained, QoE is more affected with SB than DB. DB, as a proposed model minimizes packet loss compared to the existing one. For example, in Fig. 3 when SB is set to 50 kB and DB is set to 75 kB, DB has an average packet loss of 2703 packets within the time 0s to 1000s and SB, in order hand, has an average of packet loss of 3244 packet as displayed in Table 3. In the same way, Fig. 4 shows that at time 1005s, SB has 5445 packet loss while DB has, at the same time, 5391 packet loss. Also, Fig. 5 shows that the proposed model, DB, minimizes packet loss. Therefore, the need to minimize packet loss due to buffer overflows to enhance user Quality of Experience (QoE) during video streaming is justified by the proposed model. As the buffer size increases, the number of packet loss by DB reduces compared to SB.

Table 3. Packet loss comparison when SB is set to 50 kB and DB set to 75 kB

| Time Range (S) | SB | | DB | |
|---|---|---|---|---|
| | Number of Packet Loss | | Number of Packet Loss | |
| | Total | Average | Total | Average |
| 0–1000 | 81100 | 3244 | 67575 | 2703 |
| 1001–2000 | 231200 | 9248 | 165575 | 6623 |
| 2001–3000 | 398125 | 15925 | 282650 | 11306 |
| 3000–4000 | 530675 | 21227 | 350625 | 14025 |
| 4001–4100 | 239610 | 23961 | 172240 | 17224 |
| **%Packet Loss** | **58.8** | | **41.2** | |

Table 4. Packet loss comparison when SB is set to 100 kB and DB set to 150 kB

| Time Range (S) | SB | DB |
|---|---|---|
| | Total Number of Packet Loss | Total Number of Packet Loss |
| 0–1000 | 78825 | 53950 |
| 1001–2000 | 223444 | 218575 |
| 2001–3000 | 356650 | 210575 |
| 3000–4000 | 530400 | 338525 |
| 4001–4500 | 239990 | 145250 |
| **% Packet Loss** | **59.6** | **40.4** |

Table 5. Packet loss comparison when SB is set to 250 kB and DB set to 375 kB

| Time Range (S) | SB | DB |
|---|---|---|
| | Total Number of Packet Loss | Total Number of Packet Loss |
| 0–1000 | 75154 | 51154 |
| 1001–2000 | 219771 | 107506 |
| 2001–3000 | 391522 | 191516 |
| 3000–4000 | 505647 | 296637 |
| 4001–4500 | 237930 | 154590 |
| **%Packet Loss** | **64.1** | **35.9** |

### *B. Throughput Performance Metric*

Fig. 6 and Fig. 8 illustrate the number of packets delivered successfully in a given time when SB/DB is set to 50 kB/75 kB and 100 kB/150 kB respectively. Fig. 7, Fig. 9 and Fig. 10 show the throughput when SB/DB is set to 50 kB/75 kB, 100 kB/150 kB and 250 kB/375 kB respectively with their respective Table as shown in Tables 6–8.

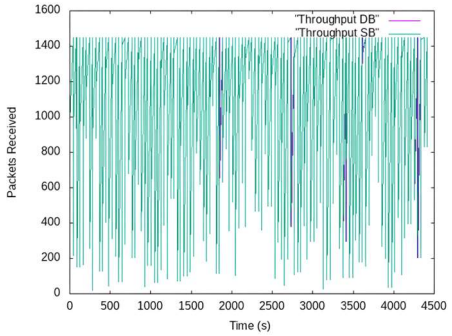


Fig. 6. Total packet received when SB is set to 50 kB and DB set to 75 kB.

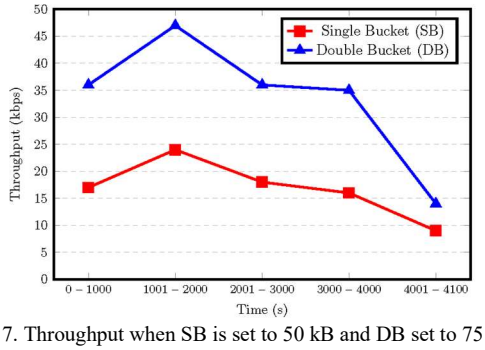


Fig. 7. Throughput when SB is set to 50 kB and DB set to 75 kB.

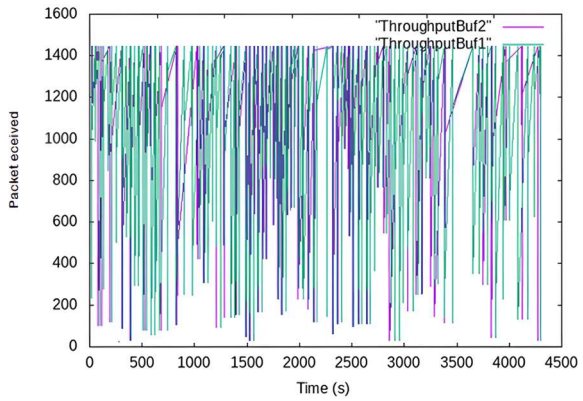


Fig. 8. Total packet received when SB is set to 100 kB and DB set to 150 kB.

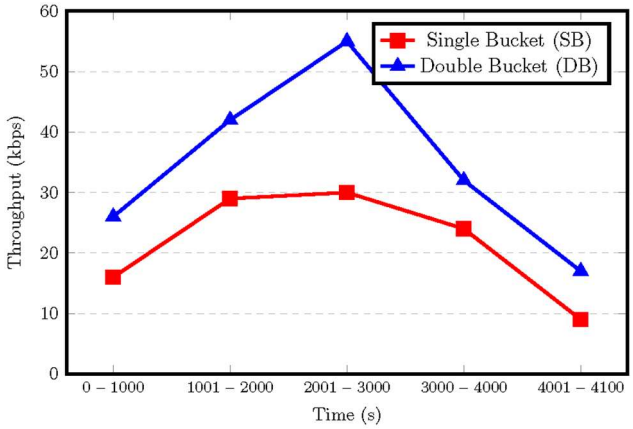


Fig. 9. Throughput when SB is set to 100 kB and DB set to 150 kB.

Another thing observed in this simulation is that at the second 40 points (1001 s to 2000 s), SB records drop in throughput by 2 kBps (from 28 kBps to 26 kBps) while DB, at the same time range, increases by 12 kBps (44 kBps to 56 kBps). The observed throughput drops correlate directly with periods of high buffer utilization where $\beta_1$ and $\beta$ reached maximum capacity. This suggests that while DLB mitigates loss, the physical layer throughput is strictly governed by the available bandwidth and signal conditions.

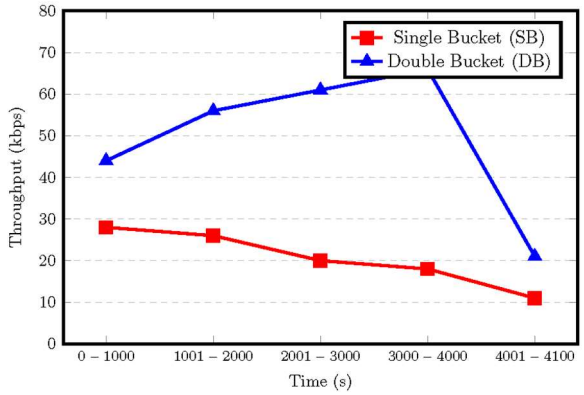


Fig. 10. Throughput when SB is set to 250 kB and DB set to 375 kB.

The total conforming packets received at the client side for both DB and SB are shown in the stated Tables 6–8. These total packets received are used to obtained throughput for each buffer size. By carrying out the first experiment via simulation with SB and DB set to 50 kB and 75 kB respectively, both techniques achieve significant throughput of 17 kBps and 36 kBps at time 0 s to 1000 s. By this result, DB achieves better throughput than SB by 19 kBps. Throughput for both techniques keeps increasing as shown in Fig. 7 and Table 6 until at time 2001 s to 3000 s when there is throughput drop from 24 kBps to 18 kBps for SB and from 47 kBps to 36 kBps for DB. This might occur due to latency or buffer overflow. Averagely, DB achieves better throughput during video downloading due to its minimized packets loss.

Fig. 9 shows the graph of throughput when SB is set to 100 kB and DB is set to 150 kBps as represented in Table 7. Just as its stated for Fig. 7, the throughput for both SB and DB experience increases from 0 s to 3000 s as shown in Table 7 and Fig. 9. At time 3001 s to 4000 s, there is a drop from 30 kBps to 24 kBps for SB and 55 kBps to 32 kBps for DB. With 100 kB/150 kB buffer, DB achieve better throughput when compare to SB. Same reason as for 50 kB/75 kB goes for the drop when simulate with 100 kB/ 150 kB buffers which might be buffer overflow or/and latency.

Table 6. Throughput comparison when SB is set to 50 kB and DB set to 75 kB

| Time (s) | SB | | DB | |
|---|---|---|---|---|
| | Total Pkt Received | Throughput (kbps) | Total Pkt Received | Throughput (kbps) |
| 0–1000 | 15920510 | 17 | 34575000 | 36 |
| 1001–2000 | 22685776 | 24 | 44817057 | 47 |
| 2001–3000 | 17758232 | 18 | 34207147 | 36 |
| 3001–4000 | 15199570 | 16 | 33703088 | 35 |
| 4001–4500 | 3614122 | 9 | 5724140 | 14 |
| **% Throughput** | | **33.3** | | **66.7** |

Table 7. Throughput comparison when SB is set to 100 kB and DB set to 150 kB

| Time (s) | SB | | DB | |
|---|---|---|---|---|
| | Total Pkt Received | Throughput (kbps) | Total Pkt Received | Throughput (kbps) |
| 0–1000 | 15535895 | 16 | 25259112 | 26 |
| 1001–2000 | 28228419 | 29 | 40484520 | 42 |
| 2001–3000 | 30115065 | 30 | 53283342 | 55 |
| 3001–4000 | 22612936 | 24 | 31128400 | 32 |
| 4001–4500 | 3614055 | 9 | 6780440 | 17 |
| **%Throughput** | | **38.6** | | **61.4** |

Table 8. Throughput Comparison when SB is set to 250 kB and DB set to 375 kB

| Time (s) | SB | | DB | |
|---|---|---|---|---|
| | Total Pkt Received | Throughput (kbps) | Total Pkt Received | Throughput (kbps) |
| 0–1000 | 27402174 | 28 | 42672495 | 44 |
| 1001–2000 | 24578321 | 26 | 54261410 | 56 |
| 2001–3000 | 19013712 | 20 | 58969420 | 61 |
| 3001–4000 | 17120526 | 18 | 63516050 | 66 |

| | | | | |
|---|---|---|---|---|
| 4001–4500 | 4415916 | 11 | 8559437 | 21 |
| **% Throughput** | | **29.3** | | **70.7** |

In the third simulation, buffer size was increased to 250 kB and 375 kB for SB and DB respectively as shown in Fig. 10 and Table 8. This is done to see if there is an improvement in the throughput as seen in Table 8 and graphically represented in Fig. 10, the throughput is now around 28 kBps/44 kBp for both SB and DB respectively at time 0s to 1000s against 17 kBps/36 kBps at the same time for the first simulation (50 kB/75 kB). Consequently, a higher number of received packets results in improved throughput. Another thing observed in this simulation is that at the second 40 points (1001 s to 2000 s), SB records drop in throughput by 2 kBps (from 28 kBps to 26 kBps) while DB, at the same time range, increases by 12 kBps (44 kBps to 56 kBps).

The simulation data confirms that throughput experienced a decline drops as seen in Fig. 7 and Fig. 9. Detailed log analysis confirms that during these intervals, the arrival rate exceeded the leak rate, leading to a 100% utilization of the primary buffer ($\beta 1$). The DLB's ability to maintain a 32 kBps throughput where the SB dropped further is evidence of $\beta 2$ successfully absorbing bursts that would otherwise result in total packet loss.

Therefore, more packets received result in higher throughput. DB technique for the 3 simulations achieves better throughput. The better performance of DB over SB obtained in the 3 simulations is as a result of total packet received and less packets loss compared to SB.

### *C. Jitter Performance Matric*

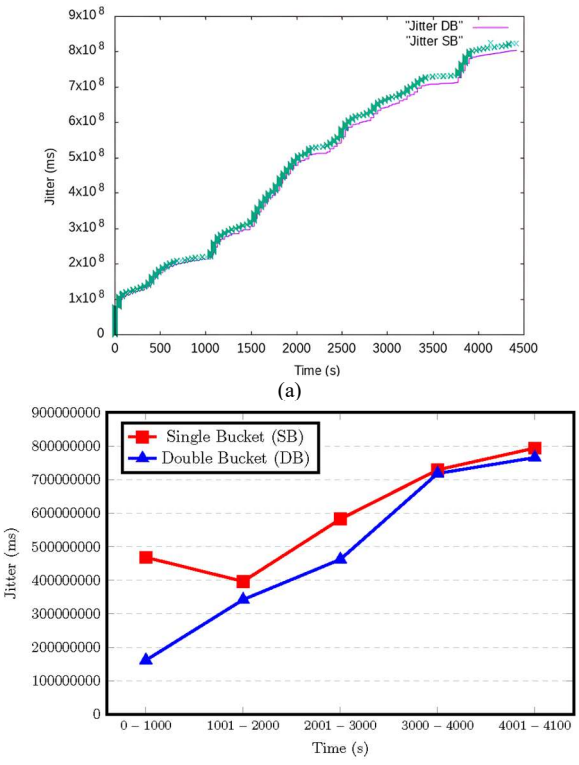


Fig. 11. Jitter when SB is set to 50 kB and DB set to 75 kB. (a) Snapshot from NS3 platform. (b) excel representation of (a).

In order to further evaluate the performance of the developed model, jitter was also measured. Figs. 11–13 illustrate the variation in the delay of the received packets in a given time when SB/DB is set to 50 kB/75 kB, 100 kB/150 kB and 250 kB/375 kB respectively with their respective Tables as shown in Tables 9–11 respectively. Although packets are usually sent at a consistent rate, yet delay between packets may be forced to vary due to congestions and limitations of queuing mechanisms.

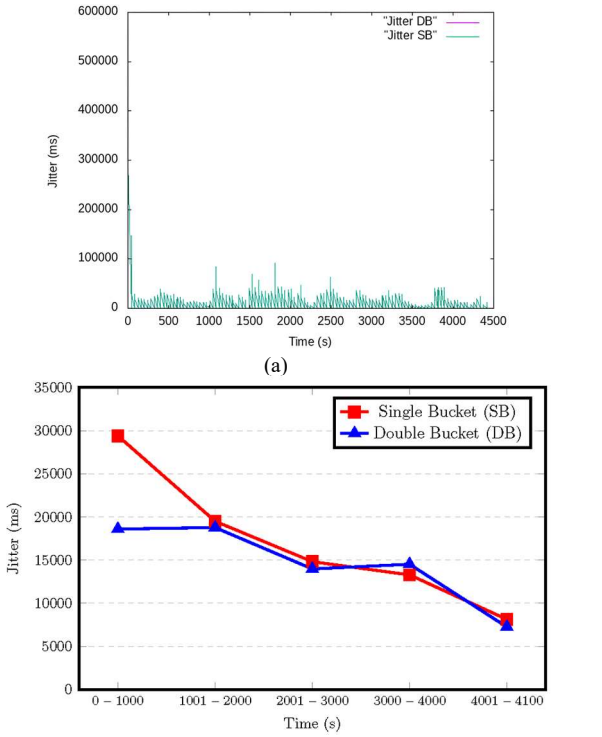


Fig. 12. Jitter when SB is set to 100 kB and DB set to 150 kB. (a) Snapshot from NS3 platform. (b) excel representation of (a).

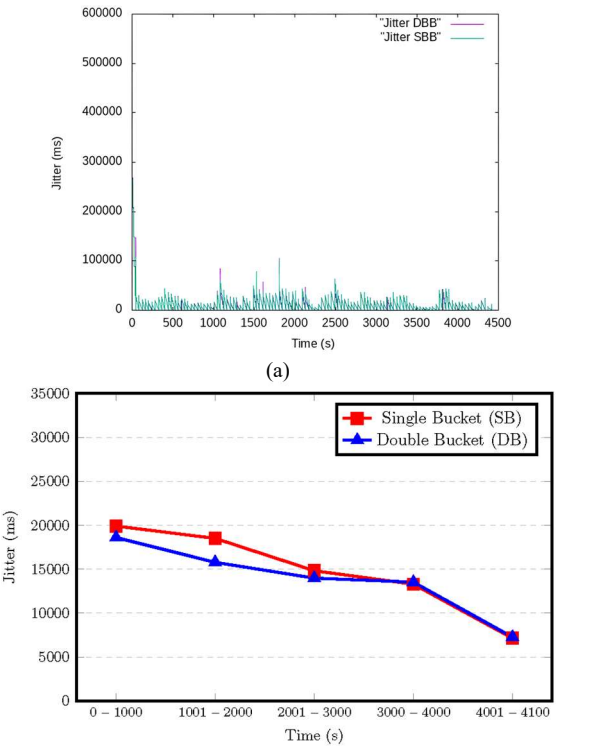


Fig. 13. Jitter when SB is set to 250 kB and DB set to 375 kB. (a) Snapshot from NS3 platform. (b) Excel representation of (a).

Table 9. Jitter Comparison when SB is set to 50 kB and DB set to 75 kB

| Time (s) | SB Jitter (ms) | DB Jitter (ms) |
|---|---|---|
| 0–1000 | 468114885.7 | 161184405.2 |
| 1001–2000 | 396350363.0 | 341966948.0 |
| 2001–3000 | 582248762.5 | 462117325.2 |
| 3001–4000 | 729103582.3 | 718227762.4 |
| 4001–4500 | 794060056.2 | 765739450.2 |
| **% Jitter** | **54.8** | **45.2** |

Table 10. Jitter comparison when SB is set to 100 kB and DB set to 150 kB

| Time (s) | SB | DB |
|---|---|---|
| | Jitter (ms) | Jitter (ms) |
| 0–1000 | 29405.056 | 18598.935 |
| 1001–2000 | 19502.676 | 18774.645 |
| 2001–3000 | 14819.810 | 13976.280 |
| 3001–4000 | 13285.596 | 14521.819 |
| 4001–4500 | 8133.617 | 7258.335 |
| **% Jitter** | **53.8** | **46.2** |

Table 11. Jitter comparison when SB is set to 250 kB and DB set to 375 kB

| Time (s) | SB | DB |
|---|---|---|
| | Jitter (ms) | Jitter (ms) |
| 0–1000 | 19905.056 | 18598.935 |
| 1001–2000 | 18502.676 | 15774.645 |
| 2001–3000 | 14819.810 | 13976.280 |
| 3001–4000 | 13285.596 | 13521.819 |
| 4001–4500 | 7133.617 | 7258.335 |
| **% Jitter** | **51.6** | **48.4** |

All the 3 simulations conducted for jitter show slight difference between SB and DB. Tables 9–11 confirm this fact except for buffer size 50 kB/100 kB as shown in Table 9 at time range 0 s–1000 s where the jitter for SB is 468114885.7 ms and jitter for DB is 161184405.2 ms. This is caused by initial delay or start up delay. Still on Fig. 11, the jitter obtained for SB as shown in Table 9 indicates that jitter dropped from 468 s to 396 s at time range 0 s to 1000 s and 1001 to 2000 respectively. It later increases to 582 s. SB records increment all through this simulation after the sec$^{on}$d 40th points (80 points). That is, SB has more jitter as time increases. Right from 2 s to 4400 s, DB records increase in jitter from 161184405.2 ms to 765739450.2 ms. Comparing SB and DB for this simulation (50 kB/100 kB), DB still has lesser jitter.

In Fig. 12 and Table 10, at time range 3001 s to 4000 s, DB has significantly incline from 13976.28 ms to 14521.819 ms and thereafter drops to 7258.335 ms. this is as a result of delay in packet delivery.

Fig. 13 and Table 11 show the simulation result for techniques when SB is to 250 kB and DB is set to 375 kB. At the beginning of the streaming, there is a high jitter, which could result from initial delay when initiating the download, for SD and DB: 19905.056 ms and 18598.935 ms respectively. The jitter has significantly declined from a maximum peak of 19905.056 ms and 18598.935 ms to 7133.617 and 7258.335 for SD and DB respectively as shown in Fig. 13(b). For these 3 simulations, both SB and DB have jitter. In comparing the 2 techniques, the proposed model achieved lesser jitter. Through the results, graphs, and Tables of throughput, jitter, and packet, a decline in performance has been seen for the existing technique as buffer size increase. In addition, improvement has been obtained when DB is used to measure packet loss, jitter and throughput. DB has been efficiently used than SB to control client-side buffer during play back video streaming to enhance user Quality of Experience (QoE).

### *D. Summary of Performance Metrics*

In summary, 8 simulations were carried out due to the categories of buffer size used. The average percentage of the 8 simulations carried out for each performance metric was obtained and the resulting values are as follows. For packet loss: SLB had 60.8% and DB had 39.2%; for Throughput: SLB achieved 33.7% and DB achieved 66.3% and for Jitter: SLB achieved 53.4% and DB achieved 46.6%. Therefore, the results obtained demonstrate that the developed algorithm employed considerably reduced the packet loss by 21.6% and achieved lower jitter by 6.8% thereby giving a higher throughput of 32.6%. Details of the summary is shown in Table 12. For each metric, simulation was done twice (for SB and DB). The values under columns SB and DB in Table 12 are the percentage obtained as recorded in Tables 3–11.

Table 12. Percentage of performance metrics

| Category | Buffer size | Packet Loss | | Throughput | | Jitter | |
|---|---|---|---|---|---|---|---|
| | | SB | DB | SB | DB | SB | DB |
| **Category1** | 50 | 58.8 | na | 33.3 | na | 54.8 | na |
| | 75 | na | 41.2 | na | 66.7 | na | 45.2 |
| **Category2** | 100 | 59.6 | na | 38.6 | na | 53.8 | na |
| | 150 | na | 40.4 | na | 61.4 | na | 46.2 |
| **Category3** | 250 | 64.1 | na | 29.3 | na | 51.6 | na |
| | 375 | na | 35.9 | na | 70.7 | na | 48.4 |
| Average | | 182.5 | 117.5 | 101.2 | 198.7 | 160.2 | 139.8 |
| **% of Average** | | **60.8** | **39.2** | **33.7** | **66.3** | **53.4** | **46.6** |

na* means not applicable.

As a result of this, Quality of Experience of mobile users during video streaming has been improved, there is a significant reduction in packet loss, jitter and increase in throughput. Therefore, client-side buffer control model gave an improved performance which is effective in enhancing user quality of experience during video streaming especially with the client's continuous success of packet reception, while a high QoE is needed. Table 12 shows the performance comparison of the system.

## V. Comparative Analysis of DLB vs. Existing Baselines

While standard DASH implementations rely on Application-level Bitrate Adaptation (ABR) algorithms such as Buffer-Based Adaptation (BBA) or MPC-style ABR, these methods primarily focus on 'what' bitrate to select based on high-level buffer occupancy. In contrast, the Double Leaky Bucket (DLB) operates at a lower architectural level,

managing how data packets are admitted from the receive socket into the application.

Unlike a traditional Token Bucket or Single Leaky Bucket (SLB), which often result in high packet loss (60.8%) when bursts exceed capacity, the DLB utilizes a dual-buffer FIFO scheduler to absorb micro-bursts that ABR logic is too slow to react to. This provides a stable physical foundation for algorithms like BBA; while BBA prevents buffer underruns at the video player level, the DLB prevents buffer overflows at the network socket level. The 32.6% increase in throughput and 21.6% reduction in packet loss achieved by the DLB demonstrates its effectiveness as a complementary traffic-shaping layer that enhances the overall stability of the DASH delivery chain.

## VI. Conclusion

Through the results and graphs shown for throughput, jitter, and packet, a decline in performance has been seen for the existing technique as buffer size increase. In addition, improvement has been obtained when DB is used to measure packet loss, jitter and throughput. DB has been efficiently used than SB to control client-side buffer during play back video streaming to enhance user Quality of Experience (QoE). As a result of this, Quality of Experience of mobile users during video streaming has been improved, there is a significant reduction in packet loss, jitter and increase in throughput. Therefore, client-side buffer control model gave an improved performance which is effective in enhancing user quality of experience during video streaming especially with the client's continuous success of packet reception, while a high QoE is needed.

## Conflict of Interest

The authors declare no conflict of interest.

## Author Contributions

TAK, AAI and AOO conceptualized the study and designed the work. TAK, KIA, and SAA drafted the initial manuscript, conducted the data preprocessing, implemented, and performed experimental analysis and contributed critical insights into model integration and validation. KIA, TAK, SAA contributed to result interpretation and visualizations. AOO and AAO editing and correction of manuscript, and supervised the research. All authors contributed to the review and editing process and approved the final version of the manuscript.

## Acknowledgment

The authors wish to thank the Department of Computer Engineering, Faculty of Engineering, Redeemer's University, Ede, Nigeria, Department of Computer Engineering, Faculty Engineering and Technology, Abiola Ajimobi Technical University, Ibadan, Nigeria and Department of Computer Engineering, Faculty of Technology, Obafemi Awolowo University, Ile Ife, Nigeria.

## References

[1] A. Mansy, "Network and end-host support for HTTP adaptive video streaming," Ph.D. dissertation, School of Computer Science, Georgia Institute of Technology, Georgia, GA, USA, 2014.

[2] L. V. Peschke, "Combining DASH with MPTCP for video streaming," M.S. thesis, Dept. Computer and Engineering, Université catholique de Louvain, Ottignies-Louvain-la-Neuve, Belgium, 2017.

[3] G. A. Taiwo, T. O. Akinwole, and O. B. Ogundepo, "Statistical analysis of stakeholders perception on adoption of AI/ML in sustainable agricultural practices in rural development," in *Proc. Ninth International Congress on Information and Communication Technology*, 2024, pp. 123–131.

[4] J. Wu, C. Yuen, B. Cheng *et al.*, "Streaming high-quality mobile video with multipath TCP in heterogeneous wireless networks," *IEEE Transactions on Mobile Computing*, vol. 15, no. 9, pp. 2345–2361, 2016.

[5] O. Akinlade, E. Vakaj, A. Dridi *et al.*, "Semantic segmentation of the lung to examine the effect of COVID-19 using UNET model," in *Proc. Applied Machine Learning and Data Analytics*, 2023, pp. 52–63.

[6] M. Rath, U. P. Rout, N. Pujari *et al.*, "Congestion control mechanism for real time traffic in mobile adhoc networks," in *Proc. Computer Communication, Networking and Inter-net Security*, 2017, pp. 149–156.

[7] S. Kim and C. Kim, "MAS: An efficient mobile adaptive streaming scheme based on traffic shaping," *IEEE Transactions on Multimedia*, vol. 21, no. 2, pp. 442–456, 2018.

[8] S. Zhang, W. Lei, W. Zhang *et al.*, "Congestion control and packet scheduling for multipath real-time video streaming," *IEEE Access*, vol. 7, pp. 59758–59770, 2019.

[9] M. Nwaiku, M. Diyan, S. Almakdi *et al.*, "Enhancing cloud security through anomaly detection: An artificial intelligence driven approach to secure authentication and authorization in SAML and OAuth 2.0 Protocols," in *Proc. International Conf. on Smart Systems and Emerging Technologies*, 2025, pp. 432–443.

[10] Z. Hu and Q. Zhang, "A new approach for packet loss measurement of video streaming and its application. multimedia tools and applications," *Multimedia Tools and Applications*, vol. 77, pp. 11589–11608, 2018.

[11] T. Høiland-Jørgensen, D. Täht, and J. Morton, "Piece of CAKE: A comprehensive queue management solution for home gateways," in *Proc. 2018 IEEE International Symposium on Local and Metropolitan Area Networks (LANMAN)*, 2018, pp. 37–42.

[12] A. Showail and B. Shihada, "Battling latency in modern wireless networks," *IEEE Access*, vol. 6, pp. 26131–26143, 2018.

[13] S. A. Ajagbe, J. B. Awotunde, A. T. Opadotun *et al.*, "Cybersecurity in the supply chain and logistics industry: A concept-centric review," in *Proc. International Conf. on Advances in IoT and Security with AI*, 2023, pp. 39–50.

[14] M. O. Oyediran, S. A. Ajagbe, O. S. Ojo *et al.*, "White shark optimizer via support vector machine for video-based gender classification system," *Multimedia Tools and Applications*, vol. 84, pp. 34645–34661, 2025.

[15] D. J. Vergados, A. Michalas, A. Sgora *et al.*, "A control-based algorithm for rate adaption in MPEG-DASH," in *Proc. IISA 2014, 5th International Conf. on Information, Intelligence, Systems and Applications*, 2014, pp. 438–442.

[16] A. Bassey, K. M. Udofia, and M. C. Uko, "Mitigating the effect of packet losses on real-time video streaming using PSNR as video quality assessment metric," *European Journal of Engineering and Technology*, vol. 4, no. 3, pp. 46–57, 2016.

[17] L. Zhou, "QoE-driven delay announcement for cloud mobile media," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 27, no. 1, pp. 84–94, 2017.

[18] K. Khan, "User-centric algorithms: Sculpting the future of adaptive video streaming," *International Transactions on Electrical Engineering and Computer Science*, vol. 2, no. 4, pp. 155–162, 2023.

[19] C. Timmerer, H. Amirpour, F. Tashtarian *et al.*, "HTTP adaptive streaming: A review on current advances and future challenges," *ACM Transactions on Multimedia Computing, Communications and Applications*, vol. 21, no. 7, Art. no. 198, 2025.

[20] S. Lederer, C. Müller, and C. Timmerer, "Dynamic adaptive streaming over HTTP dataset," in *Proc. 3rd Multimedia Systems Conf.*, 2012, pp. 89–94.

[21] T. Stockhammer, "Dynamic adaptive streaming over HTTP-design principles and standards," in *Proc. Second Annual ACM Conf. on Multimedia Systems*, 2011, pp. 2–4.

[22] H. Ott, K. Miller, and A. Wolisz, "Simulation framework for HTTP-based adaptive streaming applications," in *Proc. 2017 Workshop on NS-3*, 2017, pp. 95–102.

[23] Y. Go, H. Noh, G. Park *et al.*, "Hybrid TCP/UDP-based enhanced HTTP adaptive streaming system with multi-homed mobile terminal," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 5, pp. 5114–5128, 2019.

[24] N. Barman and M. G. Martini, "QoE modeling for HTTP adaptive video streaming–a survey and open challenges," *IEEE Access*, vol. 7, pp. 30831–30859, 2019.

[25] R. Dubin, A. Dvir, O. Pele *et al.*, "Adaptation logic for HTTP dynamic adaptive streaming using geo-predictive crowdsourcing for mobile users," *Multimedia Systems*, vol. 24, pp. 19–31, 2018.

[26] W. Chenchen, O. Akinlade, and S. A. Ajagbe, "Dynamic Resilience Assessment of Urban Traffic Systems Based on Integrated Deep Learning," in *Intelligent Transportation and Smart Cities*, vol. 70, Amsterdam: IOS Press, 2025, pp. 33–42.

[27] T. A. Kumar, R. Rajmohan, S. A. Ajagbe *et al.*, "POTMEC: A novel power optimization technique for mobile edge computing networks," *Computation*, vol. 13, no. 7, Art. no. 161, 2025