An Adaptive Quality Switch-aware Framework for Optimal Bitrate Video Streaming Delivery

Wafa A. Alqhtani[1], Maazen S. Alsabaan[3]
Department of Computer Engineering
King Saud University
Riyadh, CO 11543 Saudi Arabia

Ashraf A. Taha[2]
Department of Computer Networks
City of Scientific Research and Technological Applications
SRTA-CITY, Alexandria CO 21934 Egypt

***Abstract*—Given a large number of online video viewers, video streaming, over various networks, is important communication technology. The multitude of viewers makes it challenging for service providers to provide a good viewing experience for subscribers. Video streaming capabilities are designed based on concepts including quality, viewing flexibility, changing network conditions, and specifications for different customer devices. Adjusting the quality levels, and controlling various relevant parameters to stream the video content with good quality and without interruption is vital. This paper proposes an adaptive framework to balance the average video bitrates with respect to appropriate quality switches, making the transition to higher switches more seamless. The quality adaptation scheme increases the bitrates to the maximum value at their current quality switch before shifting to a higher level. This reduced switching times between levels and guarantees the stability of viewing and avoids interruptions. The use of a dynamic system ensures optimal performance, by controlling system parameters and making the algorithm more tunable. We built the system using an open-source DASH library (Libdash) with QuickTime player, studied the video load changes on two performance parameters, Central Processing Unit and Memory usages that have a high impact on multimedia quality. Consequently, the values of parameters that affected the performance of video streaming could be decreased, permitting users to regulate the parameters according to their preferences. Further, reducing the switching levels will reduce the overloads that occur while transferring from one level to another.**

Keywords*—Adaptive video streaming; average bit rate; mobile devices; modeling; quality of experience; quality switches; wireless networks

I. Introduction

With the rapid growth of technology and wireless devices, the necessity for supporting various applications in the future has increased with the quality of service (QoS) requirements. A vital aspect of communication over various networks is video streaming, given the increase in the number of viewers of videos over the internet. There are several challenges that must be addressed to achieve seamless video streaming, such as avoiding interruption of playback, increasing video quality, reducing the initialization time, and reducing the number of video level switches. These can be addressed by adaptive video streaming, with control algorithms to deliver video with appropriate quality and with changes in network parameters. Applications of video streaming include live streaming, video on demand (VoD) services, and mobile applications. VoD delivers video over the internet by dividing the video into parts called fragments, transmitting these parts, and enabling the receiver to decode and playback the video. This service allows for smooth streaming without having to wait for the entire video to be delivered, and allows the user to view the video at any time. Live streaming (real-time) transmits the contents to all users simultaneously, so that the fragments are transmitted at the same time as they are viewed by the users. With mobile applications, users operate mobile devices to download videos from online sources, such as YouTube, Vimeo, LiveTV, and PPStream. Several mobile applications have been available to enable users to stream videos online. There are three techniques for streaming video data. Using the first technique, referred to as progressive download, the server sends the video through the hypertext transfer protocol (HTTP) and at the receiver, the device downloads the file and can run the file after the buffering. This method of the basic version uses HTTP over the Transmission Control Protocol (TCP) so that the file is downloaded sequentially, where the user can watch after downloading part of the whole file. The second technique uses specialized protocols for streaming, real-time messaging protocol (RTMP) and real-time streaming protocol (RTSP), by sending chunks of data continuously to the media, which is displayed without a buffering or local caching. This technique is known as RTMP/RTSP streaming. The most popular technique, adaptive video streaming, collects the segments, encodes then indexes them, and subsequently determines their location (references) by using a profile file from the HTTP server. Table I provides a comparison of these three streaming technologies.

The adaptive streaming technique allows for the optimal video streaming experience for a range of dissimilar devices over a wide range of connection speeds. Generally, in an adaptive video streaming system (Fig. 1), a client plays a video that is received from a server. The client uses control algorithms to dynamically select the optimal video switch level for each downloaded segment and the period of idle times introduced to shape the received rate. In addition, the client uses a buffer to perform the synchronization inside the contents of the video, because the bit rate and the bandwidth available to the network cannot be predicted. Adaptive video streaming technologies face a variety of issues that affect performance including bandwidth, error rate, delay and jitter, synchronization, heterogeneity, and the user interface [2].

TABLE I. COMPARISON OF STREAMING TECHNOLOGIES [1]

	Progressive Download	**Streaming**	**Adaptive Streaming**
Basic Principle	Client requests for file using HTTP GET method and server sends the entire file over HTTP.	Server sends fragments of data based on client request. Just in time transfer of data.	Content is encoded at multiple bit rates. A manifest file maintains the details of the fragments and their location. Client requests best suited fragments from the list.
Transport Protocol	HTTP over TCP	RTMP/RTSP over TCP/UDP	Simple HTTP server over TCP
Bandwidth usage	Less efficient and wastage of bandwidth as the entire file may not be played.	More efficient as only part of the file is downloaded being played.	Fragments can be cached and reused, thus saving bandwidth.
Content Security	Stored locally. Less secure.	No temporary storage. More secure.	Digital rights management (DRM) integration possible for specific adaptive streaming technology.
Advantages	Easy to setup. No special licenses required.	Can access any part of the video without waiting for an entire download.	High flexibility to change video quality.
Disadvantages	Bandwidth is wasted on data which is downloaded but not watched.	Adds significant cost and complexity to the setup and operations require special network configuration for port enabling.	Requirement to have multiple encoded version requiring additional content processing and storage.
Example of online video platforms	YouTube, Vimeo	Hulu	Network Television BBC, Netflix

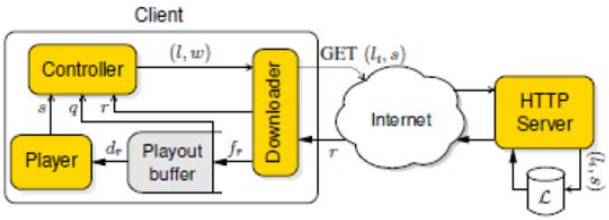


Fig. 1. Adaptive Video Streaming System [2].

Dynamic adaptive streaming over HTTP (DASH), also known as MPEG-DASH, provides high quality video streaming over the internet from an HTTP server. Fig. 2 illustrates a model of the MPEG-DASH setup. First, the multimedia content is captured and stored on an HTTP server and sent by HTTP. There are two types of content on the server. The Media Presentation Description (MPD) describes a manifest of the available content, its various alternatives, URL addresses, and other characteristics. Segments contain the real multimedia bitstreams in the form of fragments, in single or multiple files. The DASH client plays the content by parsing the MPD; therefore, the DASH client has information about the program timing, media content availability, media types, resolutions, minimum and maximum bandwidths, and the existence of various encoded alternatives of multimedia components, accessibility features and required digital rights management (DRM), media-component locations on the network, and other content characteristics. The DASH client uses this information to select the appropriate encoded alternative and to start streaming the content by fetching segments, using HTTP GET requests. After appropriate buffering to allow for network throughput variations, the client continues fetching the subsequent segments and monitors the network bandwidth fluctuations. Subsequently, the client decides how to adapt to the available bandwidth by fetching segments of different alternatives (with lower or higher bitrates) to maintain an adequate buffer, depending on its measurements. The MPEG-DASH specification only defines the MPD and segment formats [3].

Media content has several components (audio, video, and text), with each component having multiple characteristics. In MPEG-DASH, these characteristics are described in the MPD, in an XML format. Fig. 3 illustrates the MPD hierarchical data model. The MPD consists of one or multiple periods; each period has a starting time and duration, and consists of one or multiple adaptation sets. An adaptation set provides information about one or multiple media components and its various encoded alternatives. Each adaptation set usually includes multiple representations. A representation is an encoded alternative of the same media component, varying from other representations by bitrate, resolution, number of channels, or other characteristics. Each representation consists of one or multiple segments; media stream fragments in temporal sequence. Each segment has a URL that is an addressable location on a server that can be downloaded using HTTP GET or HTTP GET with byte ranges. DASH client first parses the MPD XML document. The client selects the set of representations it will use based on descriptive elements in the MPD, the client's capabilities, and the user's choices. The client then builds a timeline and starts playing the multimedia content by requesting appropriate media segments. Each representation's description includes information on its segments, which enables requests for each segment to be formulated in terms of the HTTP URL and byte-range [3].

The main contributions of this paper are as follows:

- Demonstrating some of the challenges that video streaming faces and how it affects video quality and investigating video streaming with a control algorithm to deliver video inappropriate quality with respect to network parameters changes.
- Designing an adaptive framework to balance the average video bitrate with respect to the appropriate quality switches and making the transition to higher switches more seamless.
- Proposing a method to decrease the impact of the parameter values on the performance of video streaming.

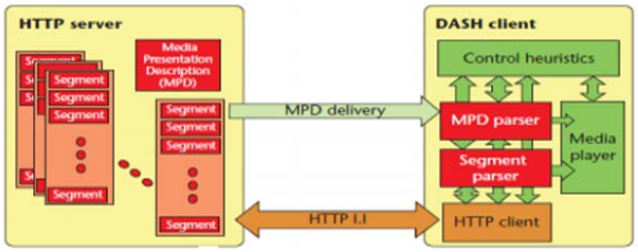


Fig. 2. MPEG-DASH System Model [3].

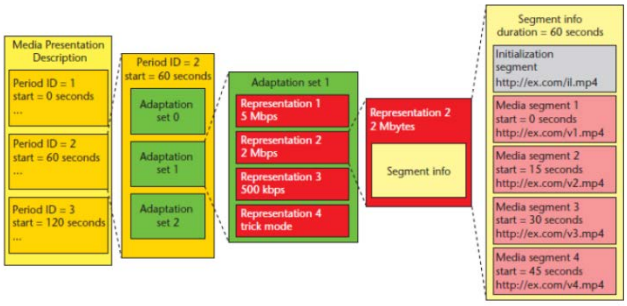


Fig. 3. MPD Hierarchical Data Model [3].

Our research can avoid missing live moments and adjust the video quality switch to the optimal bitrate level. In our proposal, we used two performance parameters: central processing unit (CPU) usage and memory usage. CPU usage limitation is about 25%, whereas memory usage limitation depends on the memory size and the registration of memory space during the streaming video, or when it exceeds the limit, which appears in red line. Imposing these conditions reduces switching between quality levels, and causes bitrates to balance with quality switching.

The remainder of this paper is organized as follows. Section II describes Literature Review. In Section III, the proposed quality adaptation framework is introduced and its functionality is explained. Section IV describes the datasets used and details of the results and discussion. Section V concludes the paper with a final review and presents future work.

II. Literature Review

The major challenge associated with video streaming is delivering seamless video with maximum quality of experience (QoE). To this end, changes are made adaptively by the design control algorithm. Stream switching is a common control algorithm. The server encodes video content in different bitrates, while the control algorithm at the client side selects the appropriate video level. This approach is used by two main standards: MPEG-DASH and HTTP live streaming (HLS). Riad et al. [4] proposed a quality scheme to obtain a balance between the number of quality switches and the average bitrates at certain cutoff points. They achieved this by measuring the variation of bandwidth values, calculating the throughput change between pairs of consecutive results, and then making quality selection decisions by matching the channel alteration to the threshold. They evaluated their proposal by testing on actual datasets, comparing them with Liu's and Adobe algorithms [4]. These results showed their scheme was successful in minimizing the number of qualities switching decisions, whilst keeping high average bitrates. In general, the proposed algorithm attains a good trade-off point between the number of quality switches and the average bitrates. Fig. 4 illustrates the influence of the cutoff on the average bitrate and over switches quality on a certain stream.

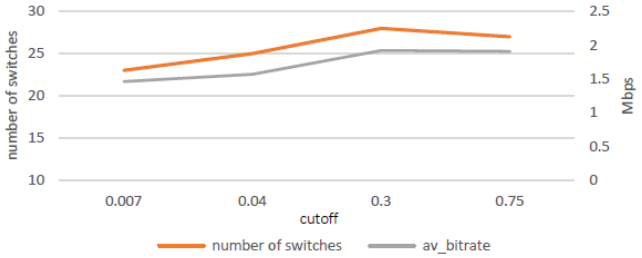


Fig. 4. Effect of a Cutoff on Average Bitrate and Quality Switches for Specific Stream [5].

In [5], Xiang et al. designed a rate adaptation algorithm to find an ideal streaming strategy in a user- aware QoS, playback breaks, average playback quality, and playback smoothness. They formulated the rate adaptation problem as a finite Markov Decision Process (MDP) using dynamic programming. The optimal strategy requires the offered bandwidth statistics and a large number of states; therefore, it is hard to obtain the optimal solution in real-time, making it difficult to create an optimum streaming policy. To counter this, they produced an online algorithm that accumulates bandwidth statistics. The online algorithm also makes streaming decisions in real-time, using a reward parameter to ensure a good balance between average playback quality and playback smoothness. The experimental results presented showed the proposed algorithm is possible; however, several issues were encountered, which required further investigation. Improving quality control and adaptation algorithms has a noticeable effect on video streaming, there are recent studies that improve these algorithms based on the video quality scale. In [6], authors depend their studies on the P.1203 series of standards proposed by ITU-T is one such example for bit stream-based models. This series consists of three main parts: Pv: short term video quality prediction,

Pa: audio short term quality, Pq: overall integration of quality. This paper focus on extending the existing mode 0 model to support the aforementioned newer codecs and higher resolutions and frame rate. They propose correction mapping for new codecs, resolutions and frame rates and not retrain the existing model. As a result, we use the unmodified mode 0 predictions from the existing model and then do a correction on this prediction for the newer use cases. This approach of just using a correction mapping and not re-training ensures that we can rely on the well-developed P.1203 models. To ensure that the proposed correction reflects quality ratings by humans, two subjective tests were conducted. They first test considered, H.264, H.265 and VP9 with resolutions up to 4K, 60 fps as frame rate and realistic bitrate settings , a second test, included H.265 and AV1 as codec . These tests give a good example of using a simple correction chart. In thesis [7], Huang et al. designed a buffer-based algorithm to adaptation video rate by using the buffer to select a video bitrate, then request when

capacity estimation is required. This approach has two phases of the process. In the steady-state phase, when the buffer encodes appropriate information, the algorithm selects the video rate depending on the playback buffer. In the startup phase, when the buffer holds few information, we expand the buffer-based design with capacity estimation. Huang et al. revealed that this approach led to a reduction in the re-buffered rate by 10–20% compared to Netflix's default available bit rate (ABR) algorithm, while refining the steady-state video bitrate. The main reason is that DASH is solely a client-side standard. A DASH client is the only agent that manage the video streaming process despite (i) its limited information about the network and (ii) being unaware of actions taken by the other clients. in [8] the authors propose to maximize fairness and efficiency of end-users' QoE by achieving a level of cooperation between clients and servers without requiring any modification on the client-side, by using Dec-POMDP model and use RL to train two neural networks to find an optimal solution to the fairness problem. This optimal solution is then enforced, through client and server cooperation, to make their system fully compatible with the DASH standard. The experimental results proved that algorithm outperformed the state-of-the-art algorithm. In [9], they propose a novel algorithm for video rate adaptation in HTTP Adaptive Streaming (HAS), based on online learning, named Learn2Adapt (L2A), is to provide a robust rate adaptation strategy which, unlike most of the state-of-the-art techniques, does not require parameter tuning, channel model assumptions or application-specific adjustments. Simulations show that L2A improves on the overall Quality of Experience (QoE) and in particular the average streaming rate. The robustness property of L2A allows it to be classified in the small set of rate adaptation algorithms for video streaming, that mitigate the main limitation of existing mobile HAS approaches. Learning-based Adaptive Bit Rate (ABR) is approaches to learn outstanding strategies without any presumptions, has become one of the research hotspots for adaptive streaming. However, it typically suffers from several issues, i.e., low sample efficiency and lack of awareness of the video quality information. In [10], they propose Comyco, a learning-based ABR system which aim to thoroughly improve the performance of learning-based algorithm. To overcome the sample inefficiency problem, they leverage imitation learning method to guide the algorithm to explore and exploit the better policy rather than stochastic sampling, also including its NN architectures, datasets and QoE metrics. With trace-driven emulation and real-world deployment, the results of Comyco significantly improves the performance and effectively accelerates the training process. Joseph and De Veciana [11], established the online algorithm NOVA to optimize video delivery that supports DASH-based clients. NOVA is asynchronous, distributing the tasks of resource allocation to the network controller, and quality adaptation to respective video clients using minimal communication. In [12], Mao et al. proposed the Pensieve system. This system automatically learns algorithms without any predefined control guidelines or assumptions about the operating environment. This is achieved by using modern strengthening learning systems to learn the strategy of controlling adaptive bitrate through reinforcement. This enhancement is in the form of reflective traffics QoE for previous video resolutions. This system takes special information about the real performance of the previous decisions to improve the control policy in the form of a neural network, so that the observations are used to decide the bitrate of the next fragment. The authors proposed learning-based approaches to producing ABR algorithms that rely on an effort to learn ABR policy from observations, especially as this method depends on learning enhancement. The aim of reinforcement learning (RL) is to increase the predictable cumulative discounted reward. Fig. 5 shows how RL is able to achieve bitrate adaptation. The ABR agent collects the metric information (bandwidth, bitrate of previous fragment, buffer occupancy) and applies it in a neural network model in the form of actions. The result is a bitrate decision; this QoE result is returned to the ABR agent as a reward. The ABR agent uses the reward for the training and development of the neural network model to improve performance. After applying the Pensieve system, experiments revealed that the proposed system deviated from ABR algorithms by 12% to 25%.

Cofano and De Cicco [13] proposed rules to guide the controller design, by designing a model to control the level based on a hybrid dynamic system. Based on this model, they derived a relationship between minimum switching frequency and control system parameters. They also proposed a methodology to adjust the lowest playout buffer that must be guaranteed to prevent rebuffing events as shown in Fig. 6. The general goal of the proposed control algorithm ([13]) is maximizing QoE for users in available bandwidth. To achieve this goal, they designed an official model of the closed-loop system by using a level-based hysteresis controller. Fig. 7 presents a comparison between the numerical simulations and the experimental results. The model in [13] fits the performance of the real system with good precision, and is able to expect system performance in terms of video level switching frequency and no rebuffing probability.

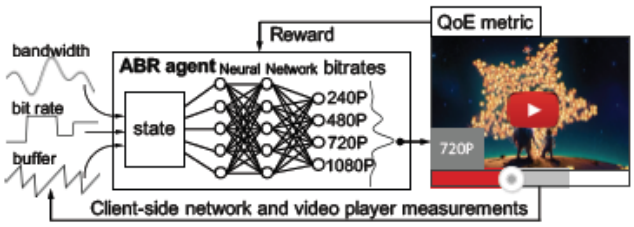


Fig. 5. Applying Reinforcement Learning to Bitrate Adaptation [10].

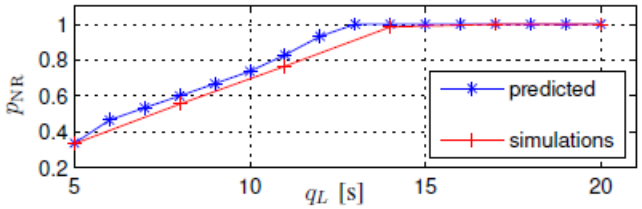


Fig. 6. No Rebuffering Probability PNR Function of q_L, the Lower Playout Buffering Threshold [13].

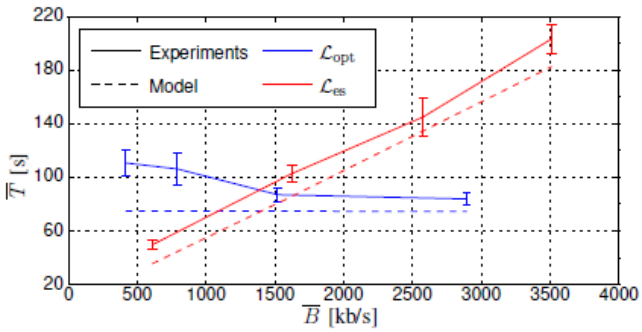


Fig. 7. Comparison between the Video Level Switching Period Obtained with Simulation Model and Experimental Result [13], between Two Sets of Data; the Optimal Level Set (Blue) and the Equally Spaced Level Set (Red).

Adaptive streaming is a technology that has to adapt video playback according to the network conditions. It is achieved by switching between representations of different bitrates and resolutions. These resolution changes affect the users' perceived quality. In [14], Asan et al. proposed a method to analyze resolution changes and their impact on QoE, as well as investigate adaptive patterns with respect to their mean opinion score (MOS). The results of this study are still inconclusive concerning single impairment factors that are typical for HTTP adaptive streaming (HAS) services. Their method attempted to predict the effect of a specific switch in terms of MOS degradation. This experiment is just one of the series of tests that they will conduct. In [15], Rodríguez et al. determined that changes in video quality level (VQL) had an effect on the user QoE. They proposed a DASH algorithm, including a decision parameter named the switching degradation factor (SDF) that captured a correlation between the QoE and VQL switching types, the frequency of VQL switching events and their temporal locations. The DASH algorithm was improved by performing VQL switching depending on SDF values. Parameter testing was done on the SDF model, alongside testing to assess the performance of the quality prediction in the MOS model. After analyzing and verifying the results, it was revealed that the MOS provided by the monitors had improved, through the incorporation of the SDF with the DASH algorithm.

In previous studies, researchers relied on various proposed methods and analyzed the results based on some of the parameters and algorithms, like measuring the variation of bandwidth pairs in consecutive results or using the MDP. Other studies depending on measuring buffer occupancy, throughput, average playback quality, and decision parameter to analyze their results. Another study testing their proposed algorithm with other algorithms by using H.264 codec. Our proposed method depends on some of the above previous studies by using H.264 codec, makes decision parameters based on preferences to improve the quality, and views stability and reduces switching levels, using average playback quality with evaluating network parameter values.

III. Proposed Framework

The proposed method is to design an adaptive framework to balance the average video bit rates with respect to appropriate quality switches and make the transition to higher switches seamless. The quality adaptation scheme increases the bitrates to their maximum value corresponding to the current quality switch, before shifting to the higher switch. This will help in reducing the number of switching levels and reducing switching times between levels to guarantee viewing stability and avoiding interruptions. A dynamic system is required to achieve optimal performance by controlling system parameters and making the algorithm more tunable, allowing each user to regulate the parameters with respect to their own preferences. Further, reducing the switching levels will reduce the overloads that occur because of transferring from one level to another. Our research can avoid missing live moments and reduce the video interruptions by adjusting the video quality switch to the optimal bitrate level. The system model is designed as proposed in Fig. 8. The green parts (server) are standardized and contain the MPD and segment formats. The delivery of the MPD, DASH streaming control, media player, and segment parser, are depicted in blue. These parts are not standardized, allowing developers to modify or add features to improve their performance. The open-source (Libdash) is depicted at the client, containing the MPD parsing and HTTP module that is responsible for HTTP download. Therefore, the library provides interfaces for these modules to access the MPD and the downloadable media segments. The DASH streaming control is responsible for downloading the order of media segments. The DASH server provides segments in several bitrates and resolutions (MPD files). The client initially receives the MPD through Libdash; the MPD contains the temporal relationships for the various qualities and segments. Based on that information, the client can download individual media segments through Libdash at any point in time. Therefore, varying bandwidth conditions can be handled by switching to the corresponding quality level at segment boundaries, in order to provide a smooth streaming experience. This adaptation is not part of the Libdash and MPEG-DASH standard and we will implement it in our system to obtain our goals.

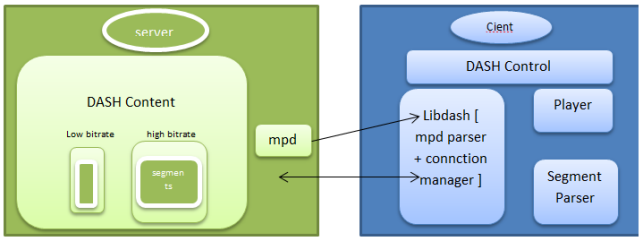


Fig. 8. System Model.

A. MPEG-DASH

Providing video content over the internet faces many challenges. Initially, the Real-Time Transport Protocol (RTP) was designed to define packet formats for audio and video content. However, the protocol performance is poor because it is used in internet protocol (IP) networks rather than content delivery networks (CDNs), and in the firewall most RTP packages are not supported. Therefore, the HTTP appeared to deliver the media content through it which is good with the firewall and uses streaming and smooth and dynamic streaming. But each of the streaming protocol has to deliver its own manifest and segment formats, so the content received

from the device must support the client protocol. MPEG-DASH is a technology that can provide interoperability between various servers and clients' devices. MPEG-DASH delivers a multimedia file to the client by using HTTP protocol and an MPEG coder. MPEG is the standard digital content format for transmission and storage of audio and video. The segment information called (MPD) Media Presentation Description.

After describing the details of the MPEG-DASH system, a description of the building of a system including client-side and server-side is provided. The server-side obtains MPD files and the client-side uses the Libdash open-source tool and DASH control. We installed the Libdash Library and opened it in visual studio 2015, with MPEG coder and QuickTime (QT) 5.11.2 tool to support the QT sample player. Libdash is the official reference software of the ISO/IEC MPEG-DASH standard, and is an open-source library that provides an object-oriented (OO) interface to the MPEG-DASH standard, developed by Bitmovin [16].

B. Performance Parameters

The dynamic systems have multiple parameters to measure the performance; in our system, we have used the common performance parameters, namely its CPU usage and memory usage. The CPU time is the amount of time for which the CPU was used for processing the instructions of a computer program or operating system. The CPU time is measured in clock ticks or seconds; it is useful to measure CPU time as a percentage of the CPU's capacity, called CPU usage. Now will explain how using Libdash Player and the details adaptation sets to balance between selecting the best level of quality and CPU usage. Starting performance profiling and selecting CPU usage, using first representation 320 x 240 (47 kbps), showed the CPU usage at less than 20%. By selecting another representation 1280 x 720, better quality and an increase in CPU usage to 39% were attained. Therefore, a high-resolution selection increases the amount of work for the processor. Referring back to the many studies of the maximum limitation of CPU usage in DASH streaming, it is expected than the upper limit of CPU usage on DASH streaming did not exceed 25% [17]. The diagnostic report is shown in Fig. 9, sorted by total CPU from highest to lowest in the selected time range. For the total CPU, the milliseconds and CPU percentage used by calls to the function in the selected time range include functions called by the function. This is different from the CPU utilization timeline graph, which compares the total CPU activity in a time range to the total available CPU [18].

$$TotalCPU\% = \frac{TotalMethodActivity}{AppaActivity} \times 100 \quad (1)$$

Self CPU unit refers to the time (in milliseconds) and CPU percentage used by calls to the function in the selected time range exclude functions called by the function [18].

$$SelfCPU\% = \frac{SelfMethodActivity}{AppActivity} \times 100 \quad (2)$$

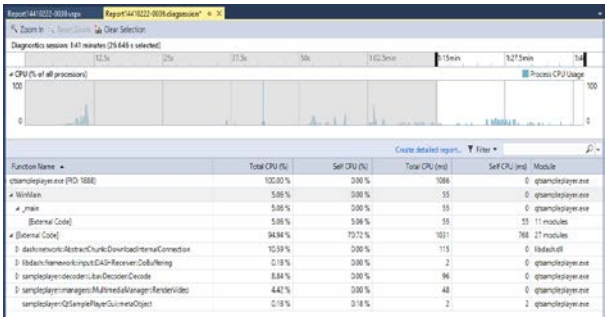


Fig. 9. CPU usage Report.

Here, (1) and (2) demonstrate the calculations of the CPU usage during the video flow and the average results.

Memory usage is the amount of memory currently in use by all applications. In the memory usage result, we obtained some snapshots from the memory by using a diagnostic tool to read the quantity of the data on the memory, including one or more snapshots of the managed and native memory heaps. A single snapshot can be analyzed to understand the relative impact of the object types on memory use, and to determine the code in the app that uses memory inefficiently. Two snapshots of an app can be compared to determine the areas in the code that cause the memory used to increase over time. A live graph of memory application is shown Fig. 10. The byte counter increases with time. The red dotted bar displayed over the graph is the memory threshold (Memory Limitation) [18]; the maximum limit does not exceed 1400 MB. To analyze memory usage, detailed report of memory usages can be obtained. The details of the difference between the current snapshot and the previous snapshot, such as the increase or decrease in memory usage, can also be obtained. The numbers in the snapshot panes indicate the bytes and objects in memory when each snapshot was taken, as well as the difference between the current snapshot and the previous one. We analyzed the numbers of Fig. 10 in depth as.

1) The total number of bytes in memory when the snapshot was acquired; a snapshot details report sorted by the total size of the type instances is displayed.

2) The total number of objects in memory when the snapshot was acquired; a snapshot details report sorted by the count of instances of the types is displayed.

3) The difference between the total size of memory objects in this snapshot and the previous snapshot (a positive number means the memory size of this snapshot is larger than the previous one, a negative number means the size is smaller); a snapshot difference report sorted by the difference in the total size of instances of the types is displayed.

4) The difference between the total number of memory objects in this snapshot and the previous snapshot; a snapshot difference report sorted by the difference in the total count of instances of the types is displayed.

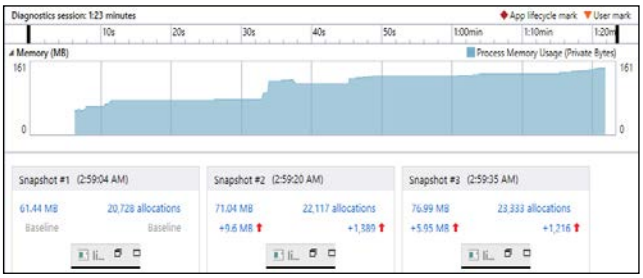


Fig. 10. Memory usage.

IV. Results and Discussion

After the text edit has been completed, the paper is ready for the template. Duplicate the template file by using the Save As command, and use the naming convention prescribed by your conference for the name of your paper. In this newly created file, highlight all of the contents and import your prepared text file. You are now ready to style your paper; use the scroll down window on the left of the MS Word Formatting toolbar.

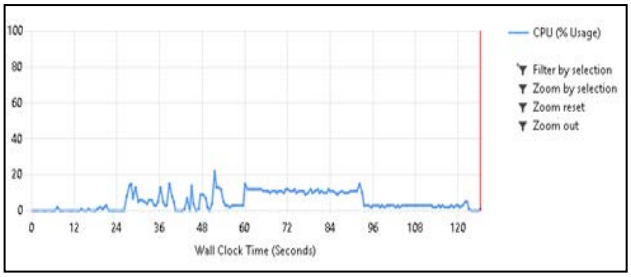
A. Simulation and Test Scenarios

Our objectives were to balance the average video bit rates with the appropriate quality switches and make the transition to higher switches more seamless, by reducing the number of switching levels to reduce the overloads on the network while transferring from one level to another. The goal were achieved by studying the video load changes on the parameters of performance, such as CPU and memory usages, that have a high impact on multimedia quality. Maximizing the average bit rates for the current quality switch before shifting the switching level to the higher one enabled control over system parameters, thereby making the algorithm more tunable. We have divided the adaptation set into three groups of levels; every level has a set of converged qualities and data rates of CPU and memory usages. The transmission among levels is based on high or low quality and its effect on CPU and memory usages.

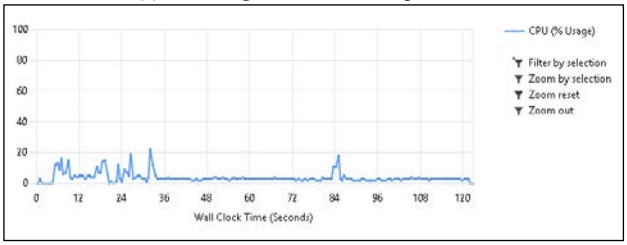
Table II shows the results of CPU and memory usage averages of all adaptation sets (according to the bitrates in the MPD file) excluding the parameters that have the biggest result from the limitation of CPU and memory usages. The average results were recorded because the results changed over time; memory usage value increases with time. The resolution value has multiple data rates; every data rate has value that effects CPU and memory usage. 320 x 240 resolution takes up approximately 9% of CPU usage and consumes between 100 to 120 MB of memory; however, at a higher resolution of 480 x 360, we observed that less than 10% of CPU usage and up to 137 MB of memory were consumed. Therefore, a switch to higher resolution consumes a larger amount of CPU and memory use, thus affecting the performance of video streaming. Further, as shown in Table II, we can observe that an 854 x 480 resolution has a CPU limitation and memory limitation, which means that other higher resolutions such as 1280 x 720 and 1920 x 1080 violate the conditions of CPU and memory limits. Therefore, we applied the reduction of switching quality levels only on resolutions from 320 x 240 to 854 x 480. Fig. 11 to 17 illustrate the results with curves of CPU usage and memory usage associated with the streaming data, and the effect of multiple instances of switching quality on these performance parameters.

TABLE II. Adaptation Set

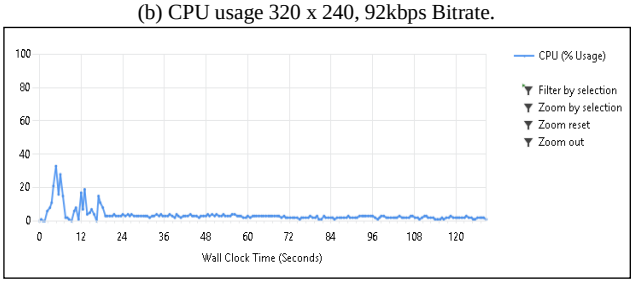
Resolutions	Data rates	CPU Usage	Memory Usage
320 x 240	47 kbps 92 kbps 135 kbps	8 % 8 % 9 %	120 MB 126 MB 1126 MB
480 x 360	182 kbps 226 kbps 270 kbps 353 kbps 425 kbps	12% 12% 13% 13% 13%	117 MB 117 MB 125 MB 127 MB 127 MB
854 x 480	538 kbps 621 kbps	19% 20%	132 MB 143 MB
1280 x 720	808 kbps 101, 103, 17 Mbps	39% up 42%	161 MB Up 210 MB
1920 x 1080	202, 206, 303, 308, 402, 407 Mbps	Up 60%	Up 230 MB



(a) CPU usage 320 x 240, 47kbps Bitrate.

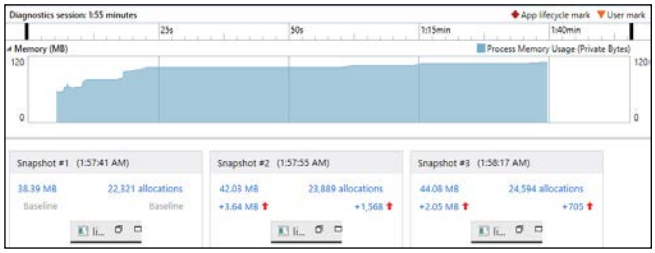


(b) CPU usage 320 x 240, 92kbps Bitrate.

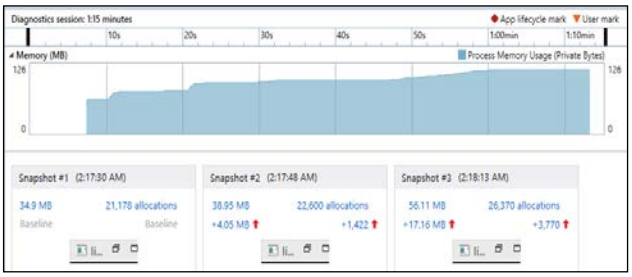


(c) CPU usage 320 x 240, 135kbps Bitrate.

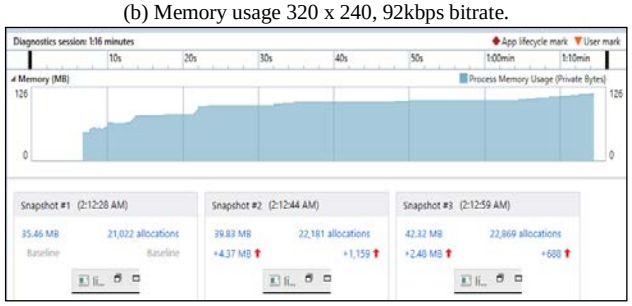
Fig. 11. CPU usage of 320 x 240 for Bitrates 47, 92, 135kbps.



(a) Memory usage 320 x 240, 47kbps bitrate.

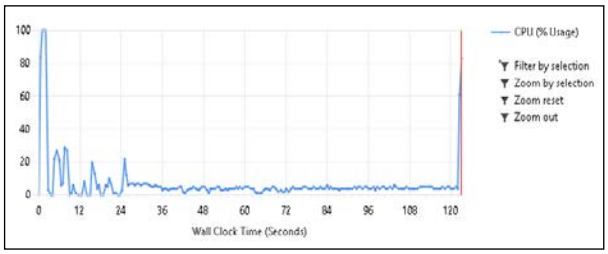


(b) Memory usage 320 x 240, 92kbps bitrate.

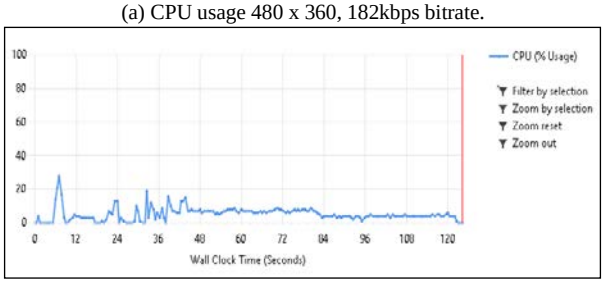


(c) Memory usage 320 x 240, 135kbps bitrate.

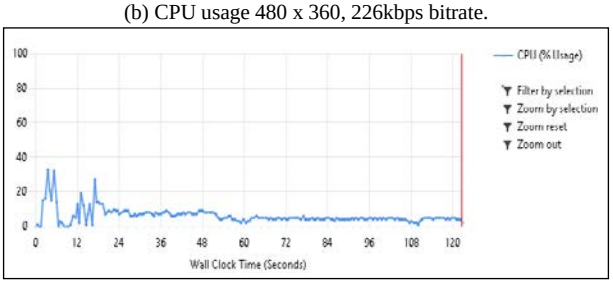
Fig. 12. Memory usage of 320 x 240 for bitrates 47, 92, 135kbps.



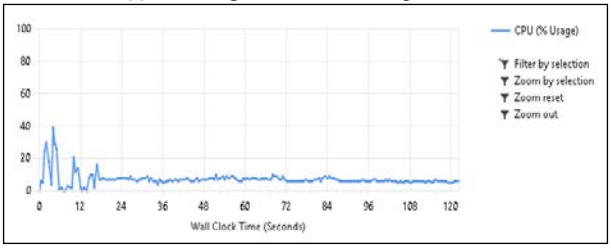
(a) CPU usage 480 x 360, 182kbps bitrate.



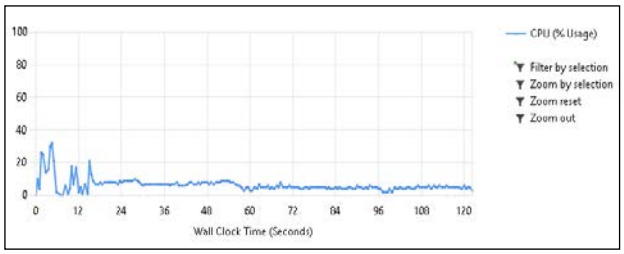
(b) CPU usage 480 x 360, 226kbps bitrate.



(c) CPU usage 480 x 360, 270kbps bitrate.

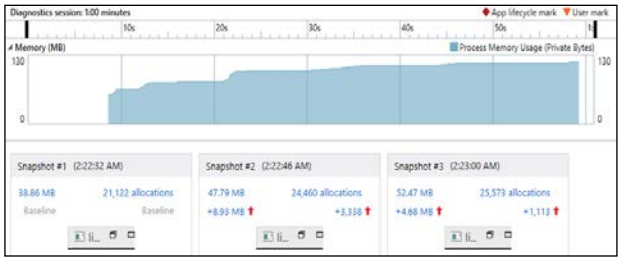


(d) CPU usage 480 x 360, 353kbps bitrate.

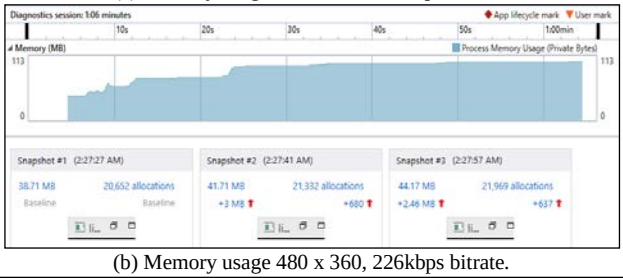


(e) CPU usage 480 x 360, 425kbps bitrate.

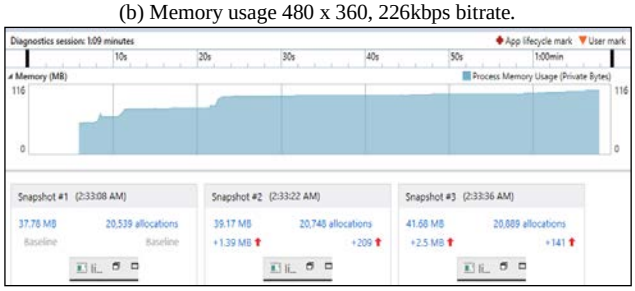
Fig. 13. CPU usage 480 x 360 for Bitrates 182, 226, 353, 425kbps.



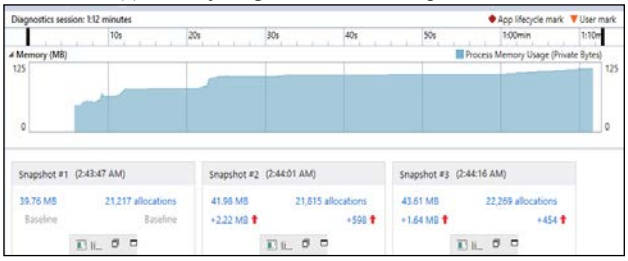
(a) Memory usage 480 x 360, 182kbps bitrate.



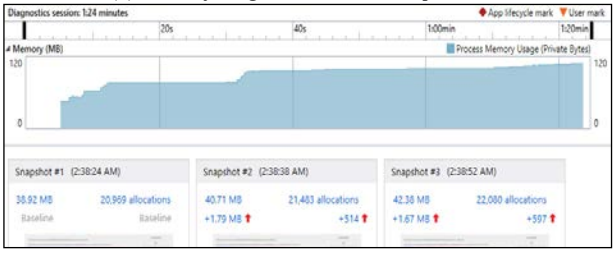
(b) Memory usage 480 x 360, 226kbps bitrate.



(c) Memory usage 480 x 360, 270kbps bitrate.

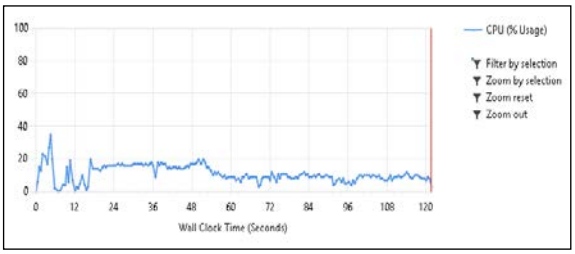


(d) Memory usage 480 x 360, 353kbps bitrate.

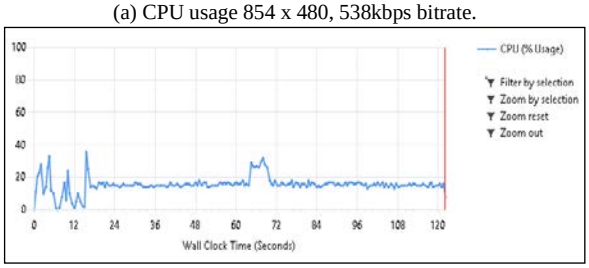


(e) Memory usage 480 x 360, 425kbps bitrate.

Fig. 14. Memory usage of 480 x 360 for bitrates 182, 226, 270, 353, 425kbps.

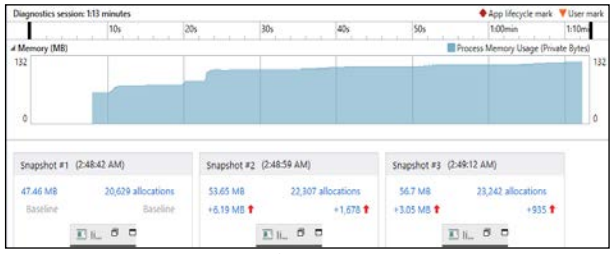


(a) CPU usage 854 x 480, 538kbps bitrate.

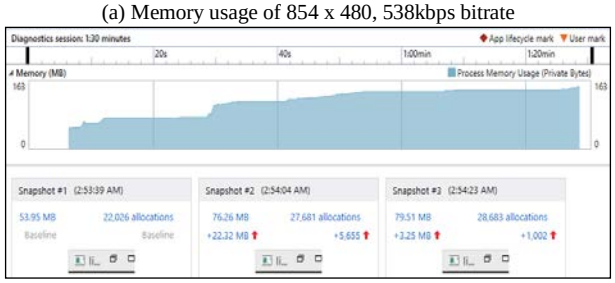


(b) CPU usage 854 x 480, 621kbps bitrate.

Fig. 15. CPU usage of 854 x 480 for bitrates 538, 621 kbps.

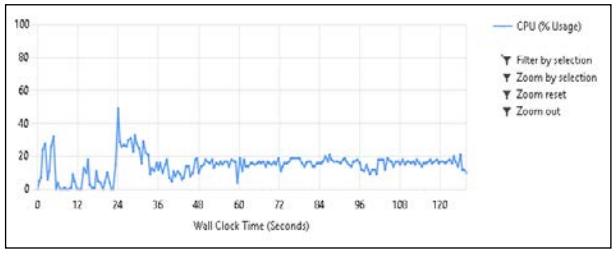


(a) Memory usage of 854 x 480, 538kbps bitrate

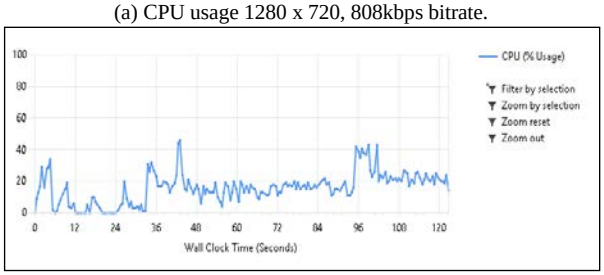


(b) Memory usage 854 x 480, 621kbps bitrate.

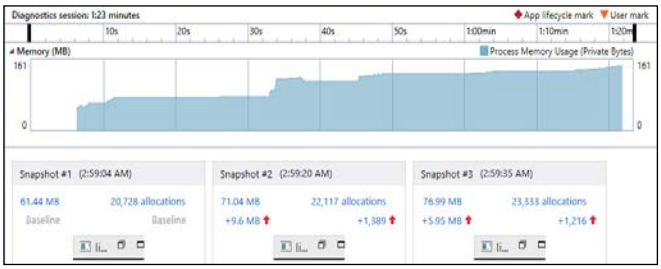
Fig. 16. Memory usage of 854 x 480 for bitrates 538, 621 kbps.



(a) CPU usage 1280 x 720, 808kbps bitrate.



(b): CPU usage 1280 x 720, 1.7Mbps bitrate.



(c): Memory usage 1280 x 720, 808kbps bitrate.

Fig. 17. CPU usages of 1280 x 720 for Bitrates 808kbps and 1.7Mbps and Memory usages for Bitrates 808kbps.

B. Results and Discussion

Our contribution herein is the design of an adaptive framework to balance the average video bit rates with respect to appropriate quality switches and make the transition to higher switches more seamless. The quality adaptation scheme increased the bitrates to the maximum value corresponding to the current quality switch, before shifting to the higher level. This helped reduce the number of switching levels, and hence reduce switching times between levels to guarantee stable viewing and avoid interruptions. A dynamic system was required to achieve optimal performance by controlling system parameters (CPU and memory). This dynamic system was also required to make the algorithm more tunable, permitting each user to regulate the parameters with respect to their own personal preferences. Further, reducing the switching levels reduced the overloads that occurred because of transferring from one level to another. In this study, we analyzed the results of highest and lowest value of each data rate in one level of resolution, and reduced these levels by combining the closest results to closest level. Trading off between data rates and quality to make the stream video more seamless, see Fig. 18. which illustrates the processes of adaptive framework flowchart. Table III illustrates the difference between bitrates before and after the study of minimizing levels of quality to reduce switching between levels. We divided the adaptation set onto three levels only; every group of quality and data rates has close results on parameters of performance such as (CPU and memory usages). After searching and studying the results of CPU usage and memory usage, Fig. 11 to 17, it is revealed we can have only three levels on the adaptation set, namely high level (HL), middle level (ML), and low level (LL), as organized in Table IV.

TABLE III. ADAPTATION SET BEFORE PROPOSED METHOD

Resolution	Bit Rates (Before)	Bit Rates (After)
320 x 240	47 kbps	47 to 135 kbps 480x360, 182 kbps
	92 kbps	
	135 kbps	
480 x 360	182 kbps	480x360, 226 to 425 kbps
	226 kbps	
	270 kbps	
	353 kbps	
	425 kbps	
854 x 480	538 kbps	854x480, 538 to 621 kbps
	621 kbps	

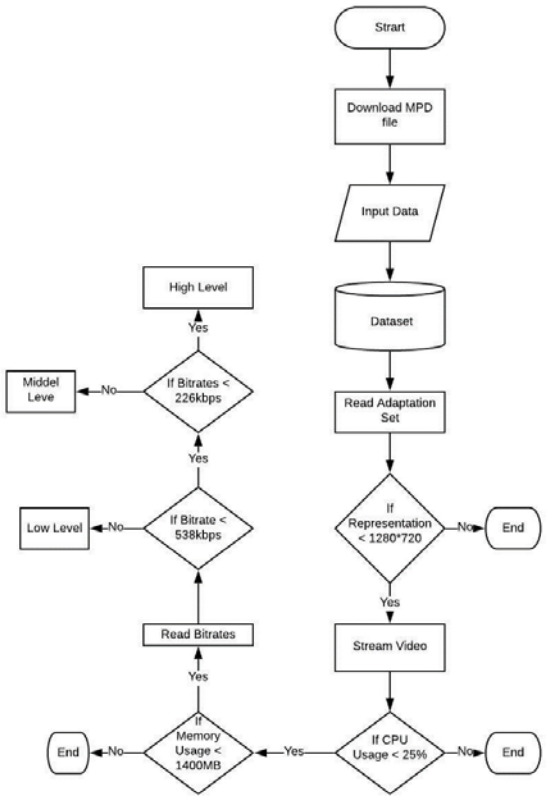


Fig. 18. Adaptive Framework Flowchart.

TABLE IV. ADAPTATION SET AFTER PROPOSED METHOD

Levels	Adaptation Set
High Level (HL)	320 x 240, 47 to 135 kbps 480 x 360, 182 kbps
Middle Level (ML)	480 x 360, 226 to 425 kbps
Low Level (LL)	854 x 480, 538 to 621 kbps

V. CONCLUSION

In this study, we have investigated video streaming with a control algorithm to deliver video in appropriate quality with respect to network parameters changes. We designed an adaptive framework to balance the average video bitrate with respect to the appropriate quality switches and made the transition to higher switches more seamless. We used a dynamic system (Libdash) to achieve optimal performance by controlling system parameters (CPU usage and memory usage) to make the algorithm more tunable. After analyzing the results, we minimized the level of quality switches to have only three levels in the adaptation set. So, this study was able to decrease the values of parameters that affected the performance of video streaming.

In future research, we will expand this study to include other influences that affect video streaming performance and quality.

ACKNOWLEDGMENT

The authors collectively thank all of those who supported the completion of this research, especially the Deanship of Scientific Research at King Saud University for supporting this research through the initiative of DSR Graduate Students Research Support (GSR). We thank Researchers Support and Services Unit (RSSU) at the Deanship for their technical support.

REFERENCES

[1] Suzen Saju Kallungal., “A Survey on Adaptive Video Streaming Technologies,” IJARCET, Advanced Research in Computer Engineering & Technology, vol. 6, issue 3, pp. 350–352, March 2017.

[2] G. Cofano, L. De Cicco and S. Mascolo, “A Hybrid Model of Adaptive Video Streaming Control Systems,” in Proc. 55th IEEE CDC, Las Vegas, NV, USA, Dec. 12–14, 2016.

[3] R. G. Asir, K. Kumar, and P. K. Reddy, “MPEG-DASH Enhanced Multimedia Streaming,” IJARCSSE, vol. 4, issue 3, pp 848–851, March 2014.

[4] M. Riad, H. Abu-Zeid, H. S. Hassanein, M. Tayel, and A. A. Taha, “A Channel Variation-aware Algorithm for Enhanced Video Streaming Quality,” in Proc. 4th IEEE LCN Workshops, Clearwater Beach, FL, USA, Oct. 26-29, 2015, pp. 893–898.

[5] S. Xiang, M. Xing, L. Cai, J. Pan, “Dynamic rate adaptation for adaptive video streaming in wireless networks,” Signal Process. Image Commun., vol. 39, pp. 305–315, November 2015.

[6] R.R.Rao, S. Goring, P. Vogel, N. Pachatz, J. J Villamar, W. Robitza, P. List, B. Feitin, A. Raake, " Adaptive video streaming with current codecs and formats: Extensions to parametric video quality model ITU-T P.1203" in Proc. of Conference: Electronic Imaging, At Burlingame, California , January 2019.

[7] T-Y. Huang, R. Johari, N. McKeown, M. Trunnell, and M. Watson, "A Buffer-Based Approach to Video Rate Adaptation," Stanford University, 2014.

[8] S. Altamimi, S. Shirmohammadi, " Client-Server Cooperative and Fair DASH Video Streaming" In Proc. Of Conference: the 29th ACM Workshop, June 21, 2019, Amherst, MA, USA.

[9] T. Karagkioules, G. S. Paschos, N. Liakopoulos, A. Fiandrotti, D. Tsilimantos, M. Cagnazzo, " Optmizing Adaptive Video Streaming in Mobile Networks via Online Learning " IEEE Transactions on Multimedia, 28 May 2019.

[10] T. Huang, C. Zhou, R. Zhang, C. Wu, X. Yao, L. Sun, " Comyco: Quality-Aware Adaptive Video Streaming via Imitation Learning" Proceedings of the 27th ACM International Conference on Multimedia, October 21–25, 2019, Nice, France.

[11] V. Joseph and G. de Veciana, “NOVA: QoE-driven optimization of DASH-based video delivery in networks”, in Proc. IEEE INFOCOM’14 Conference of Computer Communication, April 2014, pp. 82–90.

[12] H. Mao, R. Netravali and M. Alizadeh, "Neural Adaptive Video Streaming with Pensieve,” in Proc. Conference of the ACM Special Interest Group, Los Angeles, CA, USA, August 2017, pp. 4503–4653.

[13] G. Cofano and L. De Cicco, "Modeling and Design of Adaptive Video Streaming Control Systems,” IEEE Transactions on Control of Network Systems, vol. 99, pp. 1-1, 22 November 2016.

[14] A. Asan et al., "Impact of Video Resolution Changes on QoE For Adaptive Video Streaming,” in Proc. 18th IEEE International Conference on Multimedia and Expo., July 2017.

[15] D. Z. Rodríguez, Z. Wang, R. L. Rosa, G. Bressan, "The impact of video-quality-level switching on user quality of experience in dynamic adaptive streaming over HTTP," EURASIP Journal on Wireless Communications and Networking, vol. 216, Dec. 2014.

[16] Bitmovin, July 2013. [Online]. Available: https://github.com/bitmovin/libdash.

[17] A. Wiersma, "Determining meaningful metrics for Adaptive Bit-rate Streaming HTTP video delivery,” UVA University van Amsterdam, 15th June 2016.

[18] Visual studio 2019. [Online]. Available: https://docs.microsoft.com/en-us/visualstudio/profiling/cpu-usage?view=vs-2019. Microsoft.