# A Model for a Practical Evaluation of a DASH-based Rate Adaptive Algorithm over HTTP.

**Muhammad Usman Younus** ( usman1644@gmail.com )
UPS https://orcid.org/0000-0001-9033-1767
**Rabia Shafi**
NWPU: Northwestern Polytechnical University

**Research Article**

**Keywords:** Bitrate Adaptation, Bandwidth, DASH, Bitrate Switches, Buffer Level

**Posted Date:** July 1st, 2021

**DOI:** https://doi.org/10.21203/rs.3.rs-183558/v1

# A Model for a Practical Evaluation of a DASH-based Rate Adaptive Algorithm over HTTP

**Muhammad Usman Younus**[1*], **Rabia Shafi**[2]

1 Ecole Mathématiques, Informatique, Télécommunications de Toulouse, Université de Toulouse, France
2 School of Electronics and Information, Northwestern Polytechnical University, Xi'an 710129, China
[*]Corresponding Author: Muhammad Usman Younus. Email: usman1644@gmail.com

## Abstract

The proliferation of multimedia devices and user-generated content has driven massive growth in Internet traffic for video streaming. There is a dire need to propose solutions that can pose challenging multimedia streaming issues to achieve a user's quality. Dynamic adaptive streaming over HTTP (DASH) improves the user's quality through practical systems with limited bandwidth that enables the streaming media to run smoothly. This modern technology can easily improve user perception and defines the media presentation description (MPD) in terms of URL, content file, etc. The proposed adaptation algorithm attempts to determine the optimal solution to alleviate the conflict between maximizing the video quality and avoiding buffer stalls. We evaluate the proposed algorithm against alternative solutions such as ALDASH and FDASH for video content by taking into account the video bitrate, buffer level, and video bitrate switches for the single-user environment. A set of experiments have been conducted to investigate and analyze the benefits of our proposed algorithm. A network simulator NS-3 is used to conduct the performance evaluation by our proposed algorithm. Furthermore, simulation results show that our proposed algorithm enhances video quality performance compared to ALDASH and FDASH in terms of user satisfaction. Last but not least, the experimental results of our proposed algorithm can provide high viewer quality in adaptive streaming as compared to ALDASH and FDASH.

**Keywords:** Bitrate Adaptation; Bandwidth; DASH; Bitrate Switches; Buffer Level

## 1. Introduction

Nowadays, multimedia video content accounts for a dominant fraction of Internet traffic and is currently overgrowing. According to [1][2], the mobile data traffic will grow 46% from 2017 to 2022 as per the Compound Annual Growth Rate (CAGR), and it will account for 77.5 Exabytes per month. Internet activity is dominated by video traffic on both mobile and fixed access networks throughout the world. The video streaming services (e.g., Netflix, YouTube, Youku,etc.) [3] constitute the majority of real-time entertainment traffic. From the market perspective, there are some major streaming incumbents (such as Adobe HDS (HTTP Dynamic Streaming), Apple HLS [HTTP Live Streaming], Akamai HD, and MSS (Microsoft Smooth Streaming)) [4,5]. There are considerable challenges in encoding and delivering multimedia through such streaming technologies. Dynamic Adaptive Streaming over HTTP (DASH) [6] is a propitious adaptive video streaming standard released by the collaboration of MPEG and 3GPP in 2011. Table 1 shows a comparative study of the best-known adaptive streaming technologies.

**Table 1:** Comparison of Adaptive Streaming Technologies.

| Features | HLS | HDS | MSS | DASH |
|---|---|---|---|---|
| Official International Standards (e.g., ISO/ISC MPEG) | ✓ | | | ✓ |
| Deployment on HTTP Servers | ✓ | | | ✓ |
| UHD/4K | ✓ | | | ✓ |
| HTML5 Support | | | | ✓ |
| Multiple Video Views | | | | ✓ |
| Multiplex Audio and Video Content | ✓ | ✓ | | ✓ |
| Non-Multiplex Audio and Video Content | ✓ | | ✓ | ✓ |
| Efficient Ad Insertion | | | | ✓ |
| MPEG-2 TS Segments | ✓ | | | ✓ |
| Agnostic to Audio and Video Codecs | | | | ✓ |
| HbbTV (Version 1.5) Support | | | | ✓ |
| Parallel multiple support of CDNs protocols | | | | |

The growing popularity of DASH-based streaming is expected to continue because it enables the service providers to use existing network infrastructure for a seamless streaming experience. Furthermore, the use of HTTP over the top of TCP simplifies the traversal of firewalls to ensure the user access to multimedia services. Figure 1 shows a DASH system that contains a DASH client and a server. At the server-side, the encoding of multiple qualities levels is performed on an original video file, then divided into multiple segments of equal duration (2-10 seconds). For the storage of metadata (i.e., segment duration, set of available representation, unique URL for each segment, codec used, and playback duration) of each video, Media Presentation Description (MPD) file is used. The MPD file and video segments are hosted on the webserver. While the MPD file is first returned to the client that extracts the information to fetch the required segments adaptively during the streaming session. Typically, the segment size consists of a fixed duration in all existing systems, such as Apple recommends 10 seconds for segment size, Microsoft selects the segment size in the range of 2-4 seconds, YouTube, and Netflix suggest 5-10 seconds of segment size [6,7]. Then, the client measures the available throughput and dynamically adapts the quality levels to facilitate high-quality streaming. Depending on network conditions, the client selects the viewing frequency and starts streaming the content by using the HTTP GET request. The MPEG-DASH specification defines only MPD and segment formats.

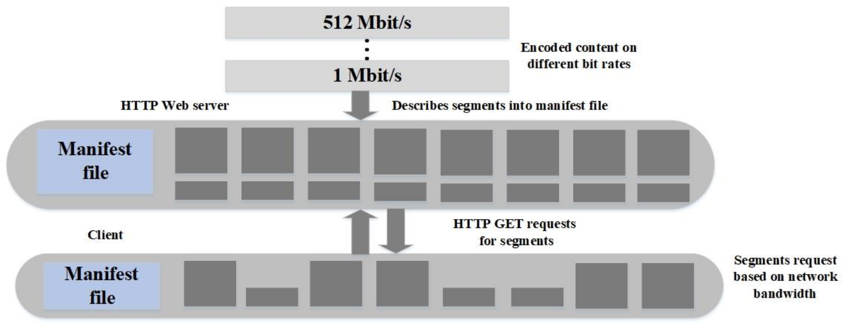
512 Mbit/s
1 Mbit/s
Encoded content on different bit rates
HTTP Web server
Describes segments into manifest file
Manifest file
Client
HTTP GET requests for segments
Manifest file
Segments request based on network bandwidth

**Figure 1:** MPEG-DASH Client-Server System.

Rate adaptation algorithms play an important role in choosing the video bitrates and optimizing the playback experience by meeting the conflicts of video quality objectives (e.g., increasing video quality, avoiding bitrate switches, preserving the buffer level, etc.). Deprived of an effective rate adaptation mechanism, the DASH clients may face lower video quality and increase quality fluctuations. In an adaptive streaming scenario, the client starts to download the video segments as soon as possible at the beginning of the streaming session to quickly fill the playback buffer. When the playback buffer is full, the client undergoes an ON-OFF scheduling pattern represented in Figure 2. The client requests the next segment during the ON state but waits for sufficient space in the buffer during the OFF state. The multiple DASH clients may face the unfair bandwidth distribution during the ON-OFF phase because of varying network conditions, resulting in a poor video streaming experience.

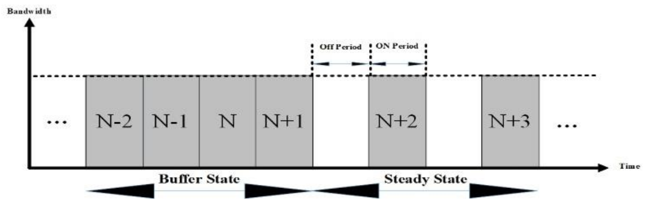

**Figure 2:** Playback buffer states during streaming session.

The effect of the segment duration and buffer space in relation to the DASH streaming plays a critical role. The configuration of the segment size is important to achieve an efficient adaptive playback experience. Small segment duration reduces the data required to initiate the video playback. Low start-up time and lower latency are, therefore, observed by shortening the video segments [9]. However, increasing the video segment duration could lead to increased video bitrate and fewer video quality switches and the number of HTTP requests over different bandwidth profiles [10]. Moreover, increasing length of the segment is highly susceptible to interruptions in playback.

Although many efforts have been made in video streaming but did not study the rate adaptation algorithms in detail to choose the appropriate video bitrate after estimating the network condition providing better performance. Therefore, a good rate adaptation algorithm is described by its responsiveness to network changes and obtaining the optimum possible throughput. The primary contributions of our paper are:

1. We proposed a rate adaptation algorithm by investigating some of the video challenges that affect the user's quality. We offered a client-side rate-adaptive algorithm that chooses the video bitrate by investigating the effect of buffer levels (e.g., 40s, 60s) and segment duration (e.g., 2s, 4s) over a wired network. Our proposed algorithm achieves better performance in terms of video bitrate, buffer level, and video bitrate switches as compared to ALDASH and FDASH.
2. Simulation performance of three different adaptation algorithms under a single-user scenario highlighting the impact of video quality metrics using NS-3. Our simulation experiments show that the proposed algorithm performs well by maintaining the resolution changes to lower levels to achieve user satisfaction.
3. We also describe our insights for future research.

The rest of the paper is organised as follows: Section 2 describes the related work in which the old video streaming techniques detail is given. Section 3 explains the system design, including adaptive goals, throughput estimation, chunk scheduling, and rate adaptation algorithm. However, Section 4 describes the performance evaluation. Finally, Section 5 concludes the paper and gives the direction of some future work.

## 2. Related Work

Multimedia service providers try to obtain the methods for enhanced playback of streaming content without entirely downloading the media content [11]. MPEG-DASH has become the de-facto framework for media streams adaptation that deals with trade-offs in several resources (e.g., buffer, bandwidth, Region-of-Interest (RoI), etc.) using different segment sizes. Lately, the researchers have focused on adjusting the segment size and buffer duration to meet the high quality of experience levels. This section entails information about related work to our field of interest. We discuss the previous research that was potentially discussing for better adaptation.

Many studies have been conducted for user-end satisfaction to adapt the high video bitrates through the DASH standard, while considering the trade-off in several resources (e.g., quality, bandwidth). As the stability in terms of quality levels is one of the significant issues to compete with HAS players. The authors in [12] discussed that a HAS player behaviour in adaptive video streaming may lead to the fairness issue when HAS clients are competing for a bottleneck link. In another study [13], the experimentation tools have been provided by introducing an evaluation framework for adaptive streaming. But, this framework does not facilitate any support for commercial stream player. The study of adaptation algorithm is our primary concern. In [14], the issue of the ON-OFF steady-state phase has been solved by an adaptation algorithm, which is similar to TCP congestion control. Hence, the buffer-based adaptation employs the playback buffer space to choose the next video segments' appropriate bitrate. However, most buffer-based adaptation schemes may encounter instability and poor QoE, especially under the challenging network conditions. The study [15] provides a buffer-based adaptation algorithm where the video rate is directly chosen by current buffer occupancy during the startup phase, thereby reducing the rebuffer rate while delivering a high video rate in a steady-state ensuring the stability of buffer occupancy for smooth video playback.

At present, the most common bitrate adaptive methods of panoramic video are mainly divided into two categories: rate-based adaptive method and buffer-based adaptive method. The main idea of the adaptive method based on bitrate is to select the maximum bitrate which is not larger than the predicted bandwidth. In [16], the client calculates the optimal bitrate according to the bandwidth situation, then compares the two bitrate threshold values to set the appropriate bitrate. However, bandwidth prediction is not accurate sometimes. When the client's chosen bitrate exceeds the available bandwidth, the use of bitrate-based adaptive method will lead to greatly reduce the user quality of experience. However, due to network fluctuations, the bandwidth prediction is sometimes inaccurate. The main idea of the adaptive method based on cache is to select the appropriate bitrate to make the size of the video cache in a stable state. The PID controller is the most common method for controlling the cache size. In the panoramic video playback system designed in [17], the video is divided into the base layer and the enhancement layer. The base layer is downloaded first to ensure that the cache size does not fall below a certain limit. For the enhancement layer, a PID controller is used, and the appropriate bitrate is selected through the feedback mechanism to control the cache size in a stable state. However, the adaptive method based on cache can only guarantee the smooth video playback, and cannot achieve the goal of optimizing the user's quality.

Many novel strategies have been proposed for adaptive streaming on various varying network conditions. As it is challenging to provide high-quality video due to network conditions. Our aim is to avoid from the traditional streaming protocols and to use the concept of DASH that provides the high video quality and minimizes the buffering level in the current video. Therefore, the authors in [18] proposed an ensemble rate adaptation algorithm that leverages the benefits of multiple methods to improve the user's quality at various decision times to network fluctuations. An adaptive framework in [19] balances the average video rate to guarantee viewing stability by minimizing the switching time. The system parameters are controlled to ensure the dynamic system's optimal performance and make the algorithm more tunable. In [20], the authors proposed a fuzzy logic rate adaptation scheme, known as, FDASH that efficiently adjusts video rate to suit network conditions while avoiding buffer underflow and frequent quality changes. It controls the buffering time by delivering the video resolution based on high quality distributed video segments. While in [21], an adaptive logic for dynamic adaptive streaming over HTTP (ALDASH) has been proposed to improve the user's quality by dynamically selecting the video bitrate and also estimates the throughput based on network conditions in a single-user environment.

Our study considers both the buffer level and throughput measurements to select the next segment's video rate on a segment-by-segment basis. Initially, the client does not have any information about the network conditions. Then, we investigated the effect of the segment duration and buffer space on adaptive streaming by choosing different segment sizes (i.e., 2s, 4s) and buffer levels (i.e., 40, 60s) for a single-user environment. Our proposed scheme considerably surpasses the rate adaptation algorithms, namely FDASH and ALDASH, from the perspective of playing high-quality segments without interruptions to facilitate a promised viewing experience. Moreover, the proposed algorithm maintains the resolution changes to the minimum level under different streaming scenarios to attain higher user satisfaction.

## 3. System Design

This section discusses the proposed system to alleviate the problem caused by the conventional video streaming systems. Figure 3 shows that the adaptive video streaming of the client-side. It sends the HTTP GET request to the HTTP server to download the video segments by using the rate selection module that selects the video rate dynamically for each segment. An MPD file downloaded from the server to get all the information about the quality levels before downloading the first video segment. The HTTP server uses a bottleneck link to send the video to the HTTP client. The downloaded segments are stored by playback buffer and then feed to the player decoder. The playback buffer is linked with each independent video segment.

Furthermore, the bandwidth measurement module feeds the rate selection module with the measurements such as estimated throughput $Test(i)$and playback buffer level $B(i\text{-}1)$ to select the video rate for segment $(i)$. The downloaded segments are placed in a First-in First-out (FIFO) buffer that is handled by the media player decoder. After selecting the video rate, the interval is controlled by a scheduler module.

**Table 2:** Notations

| Notations | Meaning |
|---|---|
| $(i)$ | Index of segment number |
| $Test\ (i)$ | Estimated throughput for segment $(i)$ |
| $B(i\text{-}1)$ | Playback buffer level for segment $(i\text{-}1)$ |
| $r[i]$ | Bitrate of $(i)th$ segment |
| $\tau$ | Playback duration |
| $t_i^{start}$ | Start time of receiving the $(i)th$ segment. |
| $t_i^{end}$ | End time of receiving the $(i)th$ segment. |
| $T(i)$ | Segment throughput |

| | |
|---|---|
| $\alpha$ | Linear factor |
| $T_d(i)$ | Average throughput of downloaded segments during a specified period *d* |
| $B(target)$ | Target buffer level |
| $B(i)$ | Current buffer level in seconds |
| R | Set of *n* bitrates |
| $B(p)$ | Panic buffer level |
| $B(g)$ | Growing buffer level |
| $B(s)$ | Stable buffer level |
| $R_{next}$ | Next segment's video rate |
| $r_{min}$ | Minimum available video rate |
| $r^{\uparrow}$ | Increase the quality by one level |
| $r^{\uparrow\uparrow}$ | Increase the quality by two level |

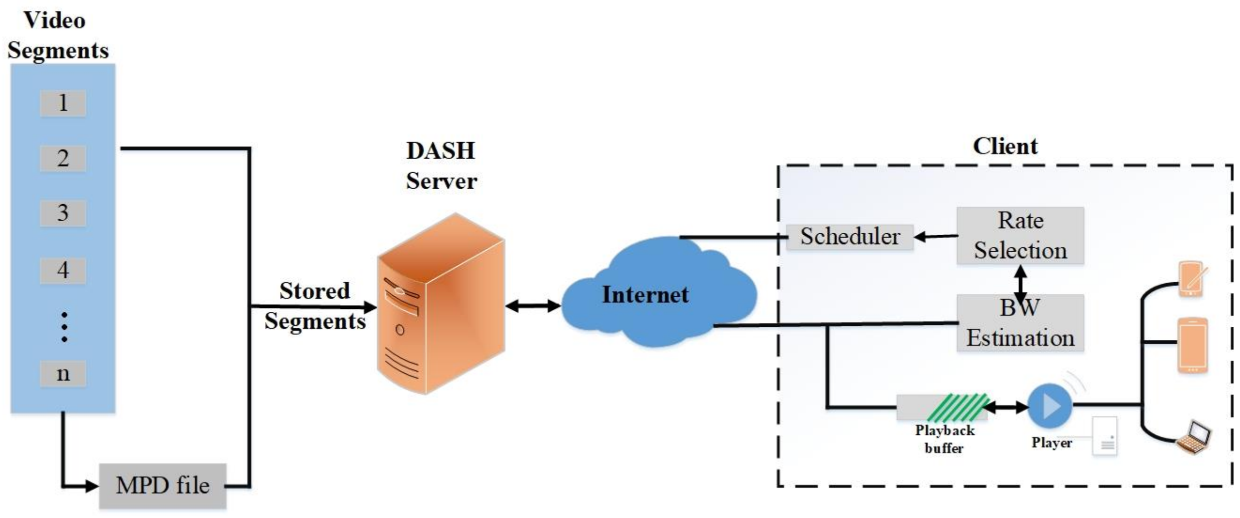

**Figure 3:** Block Diagram of Client-Side Rate Adaptation Streaming.

## *3.1 Adaptation Goals*

According to study performed by Jackson[1], there is a need to design an efficient ABR algorithm to provide the enhanced QoE during the middle and end of the streaming session with minimum quality fluctuations and buffer underflow. The lower video quality provides the lower end-users' satisfaction.

[1] A User Study of Netflix Streaming

Similarly, the video bitrate switches annoy the user. Furthermore, the user starts leaving the video if the video does not play within 2 seconds. Many efforts have been made to know the effects of QoE on user engagement. Thus, our proposal tries to maintain higher video quality, a lower number of quality switches, and a sufficient buffer level to avoid playback interruptions. We consider the following design goals for the proposed scheme.

### *3.1.1 Optimize the video rates*

One of the main goals of adapative algorithms is to adaptively choose a video rate from a video set to maximize the viewing experience. This is considered as an important factor because users normally desire high video quality.

### *3.1.2 Reduce the frequency of video rate switches*

Switching frequency is a rate of video quality switch in a specific time unit during streaming session. It is found that change in quality represents to a non-zero quality (e.g., video bitrate, etc.) of two conservative video segments. The high frequency of video rate switching exasperates to the user. The fewer bitrate changes can provide the high user's satisfaction as compared to the video of higher bitrates with a higher number of bitrate changes.

### *3.1.3 Minimize the number of interruptions*

In terruption is the freezing of video playback due to buffer underflows. A user can tolerate at most one interruption of just a few seconds; therefore, it should be avoided during the streaming session to achieve good quality. Although, it becomes very important for adaptation algorithms to avoid playback interruptions while adjusting the bitrate according to network conditions to achieve high video quality. By getting a better experience and a longer video playback time, the user will consume content more frequently. The video rates and rebuffering events are contradictory as maximization of the video rate would risk extensive rebuffering.

To achieve these goals, our solution consists of following aspects.

- Throughput estimation for each segment.
- A segment scheduling mechanism to restrict the request interval for each segment.
- Selection of the appropriate video quality for each segment through an efficient rate adaptation algorithm.

## *3.2 Throughput Estimation*

In HTTP streaming, a consecutive series of HTTP request-response messages is used to deliver the media segments. For a rate adaptation algorithm, the client estimates throughput for the next segments by observing the throughput changes during the downloading of the last segment to select the video bitrate. Throughput measured at the application layer is given as:

$$T\ (i) = \frac{r[i]*\tau}{t_i^{end} - t_i^{start}} \qquad (1)$$

where $r[i]$ is the bitrate of $(i)th$ segment, $\tau$ represents the playback duration, $t_i^{end}$ and $t_i^{start}$ indicate the end time and start time of receiving the $(i)th$ segment. The last segment's throughput decides the video rate for the next segment due to which short-term fluctuations may cause frequent fluctuations in the video rate. For example, due to an erroneous approximation, if the measured throughput is higher than the available throughput, the client selects a high video bitrate that will lead to network congestions. Contrary, if the measured throughput is lower than the available network throughput, the video quality would not be according to the maximum acceptable levels.

The average throughput measurement stabilizes the estimation over time to mitigate the frequency of fluctuations. The average throughput of *n* downloaded segments is estimated during a specified period *d* as follows:

$$T_d(i) = \frac{1}{n} * \sum_{i=1}^{n} T(i) \qquad (2)$$

where $T^{\ est}(i)$ is the estimated throughput, $T(i)$ is a segment throughput and $i$ is an index of segment number. The throughput is estimated for the next segment based on the current zone of the buffer level and is given as:

$$T^{est}(i) = \ \alpha * T_d(i) \qquad (3)$$

where $\alpha$ is a linear factor and its value defines the efficiency of adaptive algorithm during the streaming session. It should be set after the exact application scenario known. Once $\alpha$ value is set at the beginning of the session, it remains the same until the end. It is calculated as:

$$\alpha = \begin{cases} 0.5 & B(i) \leq B(p) \\ \frac{0.5(B(i)+B(g))-B(p)}{B(g)-B(i)} & B(i) < B(g) \\ 1 & Otherwise \end{cases} \qquad (4)$$

If $B(i) \leq B(p)$ then adaptation logic operates in the panic phase and $\alpha$ is set to 0.5. It continues to request segments of the base bitrate until it moves to growing phase. When the buffer size becomes more than 50% of the buffer capacity, the adaptation logic sets the next video bitrate depending on the buffer gain $B(g)$. It indicates that the network conditions are favorable and higher video bitrates are supported. If the $B(g)$ is not sufficient, then $B(g)$ gets accumulated over subsequent segment requests and client continues to requests the most suitable video bitrate to the network condition that does not exceed the estimated throughput to improve the video quality.

### *3.3 Chunk Scheduling*

The chunk scheduling strategy aims to define the interval to launch the request of the next segments. There are mainly three chunk scheduling strategies: (i) immediate downloading strategy that greedily improves the player's ability to avoid future buffering events. At the highest bitrate, when users leave prematurely, then it may increase the cost of bandwidth. While at a low bitrate, chunks may avoid switching to a higher quality in case the network conditions improve. (ii) The periodic strategy minimizes the rebuffering by maintaining a constant playback buffer. Thus, HTTP client gets stuck in suboptimal throughput allocations because of the biased view of the network state. (iii) A randomized segment fetching strategy mitigates this bias by selecting the target buffer randomly that tries to maintain the playback buffer level. In this paper, we are using a randomized segment scheduling strategy.

Let a client make a request for the segment (*i*+1) in time $t^{start}(\mathrm{i}+1)$ *and* $t^{end}(\mathrm{i})$ *is that time* when transmission of the segment (*i)* is completed. $B(target)$ *denotes the* target buffer level while the length of the current buffer level is denoted by $B(i)$ in seconds, $\tau$ represents the length of segment *playback duration*. Hence, the time to request segment (*i* + 1) can be written as:

$$t^{start}(\mathrm{i}+1) = \begin{cases} t^{end}(\mathrm{i}) & if\ B(i) < B(target) \\ t^{end}(\mathrm{i}) + (B(i) - B(target) + \tau)/\tau & if\ B(i) \geq B(target) \end{cases} \quad (5)$$

The above equation indicates that the requests for the next segment immediately after downloading the previous segment when $B(i)$ is less than the target buffer level, i.e., $B(target)$. While, the client has to wait for $(B(i) - B(target) + \tau)/\tau$ seconds to make the next segment request when $B(i)$ increases above the $B(target)$.

### *3.4 Rate Adaptation Algorithm*

In the proposed rate adaptation algorithm, the streaming video is divided into a number of segments, each containing $\tau$ seconds of duration. Note that the feasible bitrate of video segments can be chosen through a set of *n* bitrates, i.e., R= *{r(1), r(2), …, r(n)}.* A continuous HTTP connection is assumed that sequentially requests the segments by using HTTP GET requests. The client-side buffer is associated with three different buffer levels named and represented as panic $B(p)$, growing $B(g)$, and stable $B(s)$. The next segment's video rate, $R_{next}$, is made differently that depends on area where the current buffer occupancy *B(i) lies*. Algorithm 1 provides the pseudo-code.

The algorithm considers both the buffer level and throughput feedback signals, which are used for the next segment's video rate selection on a segment-by-segment basis. The quality level of *(i)th* segment is decided after downloading $(i-1)th$ segment. Initially, the client does not know any kind of information about the network conditions. Therefore, a conservative strategy is implemented to fill the buffer as soon as possible. For $B(i-1) \leq B(p)$, the minimum available video rate, $r_{min}$, is selected by the client to limit the initial stall time as small as possible.

The algorithm attempts to fill the buffer level by smoothly increasing the quality level when the buffer occupancy goes beyond $B(p)$. The next high quality bitrate that is less than or equal to the estimated throughput is selected for $B(i-1) \leq B(g)$ provided that the buffer level is not lower than $\frac{r^{\uparrow}}{r_{min}}$. If the previously selected bitrate is higher than the available throughput and is not the minimum, the client decreases the quality by one level instead of an aggressive decrease.

For $B(i-1) \leq B(s)$, the buffer level is established and we can improve the quality by switching aggressively. The client prefers to switch two quality levels if the buffer level increases by $r^{\uparrow\uparrow}/r_{min}$ seconds and the next two higher quality levels are below the estimated throughput, i.e., $r^{\uparrow\uparrow} \leq T^{est}(i)$. If the network throughput is not feasible to increase the bitrate by two level the client will check to increase the quality by one level, denoted as $r^{\uparrow}$. Otherwise, it will maintain the previous bitrate.

If the current buffer level exceeds the stable level $B(s)$, the most suitable video bitrate (e.g., probably the highest bitrate from set *R*) to network condition is selected that does not exceed the estimated throughput to improve the video quality.

## 4. Performance Evaluation

For evaluating the proposed rate adaptive algorithm, the network simulator NS-3 has been used to perform the experiments. Table 3. defines the simulation parameters used for experimental work. Figure 4 shows the network topology implemented in this work for the single-user scenario. The used topology consists of three nodes, i.e., server, router, and client nodes. The server is an Apach HTTP server running on Ubuntu 14.04 LTS, while the client is implemented in python and runs on a Window 10 with Core i7 2.6 GHz CPU and 8GB RAM. DASH server contains the Big Buck Bunny video sequence with framerate of 30fps and a resolution of 1280x720. Each video is divided into small video sequence of 2s and 4s. The client node downloads video segments from the server over the bottleneck link, whose bandwidth is shown in Figure 5. For the analysis of rate adaptation schemes performance, the bottleneck link varies between HTTP client and server. The bandwidth link of 3Mbps remains constant till 100s and then it repeatedly switches between 2Mbps and 5Mbps after every 5s until the end of the simulation at 300s. The bandwidth has a rectangular shape with two bandwidth levels such as 2Mbps and 5Mbps. This is important in evaluating the proposed adaptation method because we want to know how it performs when bandwidth fluctuates.

The sequence length is encoded into 8 different bitrates, i.e., {131, 434, 791, 1500, 2500, 3500, 3800, and 4200} Kbps to obtain the adaptive streaming. The performance is evaluated against two benchmark rate adaptation algorithms named ALDASH and FDASH in the single-user scenario. Three metrics are used to evaluate the performance, including average video bitrate, average buffer level, and video bitrate switches.

**Algorithm**

**Input:**

$R = \{r_1, r2, \ldots\ldots r_n\}$ ← *n number of video bitrates*

$T^{est}(i-1)$ ← *throughput measured after the download of (i-1)th segment.*

$B(i-1)$← *Buffer occupancy after the download of (i-1)th segment*

**Result:**

$R_{next}$ ← *Next representation to fetch from the server for the (i)th segment*

```
1.  if $B(i-1) \le B(p)$ then
2.      $R_{next} \leftarrow \min\{r_i | r_i \varepsilon R, r \le T^{est}(i-1)\}$   // Assign the lowest quality to boost the buffer quickly
3.  else if $B(i-1) \le B(g)$ then
4.  if $B(i-1) \ge r^{\uparrow}/r_{min}$ && $r^{\uparrow} \le T^{est}(i-1)$
5.      $R_{next} = r^{\uparrow}$   //Increase the quality by one level
6.      else if $r_{prev} != r_{min}$ ^^ $r_{prev} > T^{est}(i-1)$ then
7.          $R_{next} = r^{\downarrow}$   //decrease the quality level
8.  else if $B(i-1) \le B(s)$ then
9.      if $B(i-1) \ge \frac{r^{\uparrow\uparrow}}{r_{min}}$ && $r^{\uparrow\uparrow} \le T^{est}(i-1)$
10.     $R_{next} = r^{\uparrow\uparrow}$   //Buffer level is established, increase by two levels
11.     else if $B(i-1) \ge \frac{r^{\uparrow}}{r_{min}}$ && $r_{prev} \ne r_{max}$ && $r^{\uparrow} \le T^{est}(i-1)$
12.         $R_{next} = r^{\uparrow}$   //increase by one level
13.     else
14.         $R_{next} = r_{prev}$   //Continue the previous bitrate to avoid fluctuations
15. else if $B(i-1) > B(s)$ then
16.     $R_{next} \leftarrow \max\{r_{prev}, r_i | r_i \varepsilon R, r_i \le T^{est}(i-1)\}$
```

As, the client can more quickly switch to a higher quality level at start, or also can recover from low buffer occupancy [22]. Therefore, the proposed client's buffer levels *B(s), B(g),* and *B(p)* are selected as 80%, 20%, and 10% of the playback buffer size, respectively. When client is in panic state *B(p)* chosen as

*10%* of the palyback buffer, It enables a minimum buffer fill state. Then, the client will download the segments with lowest available video rate until the buffer occupancy starts to increase above minimum threshold level. To set the value of *B(g)* to 20% of the playback buffer means that the adaptation algorithm tries to fill buffer level by smoothly increasing the quality level. Similarly, when we choose the *B(s)* as 80%, it means that buffer level aggressively select the video rate. Table 4 depicts the statistics of different adaptation algorithms for the single-user scenario. The four families of experiments were performed for a single-user scenario as follows:

1. Three different rate adaptation schemes performance is evaluated with a segment duration of 2s for buffer level 40s.
2. Three different rate adaptation schemes performance is evaluated with a segment duration of 2s for buffer level 60s.
3. Three different rate adaptation schemes performance is evaluated with a segment duration of 4s for buffer level 40s.
4. Three different rate adaptation schemes performance is evaluated with a segment duration of 4s for buffer level 60s.

**Table 3:** Simulation Parameter Settings.

| Parameters | Values |
|---|---|
| Representation Set | {131, 434, 791, 1500, 2500, 3500, 3800 & 4200} Kbps |
| Video Sequence | Big Buck Bunny[1] |
| Segment Duration in Seconds | 2s & 4s |
| Buffer Size in Seconds | 40s & 60s |
| Bandwidth | 6Mbps |
| Simulation Period | 300s |

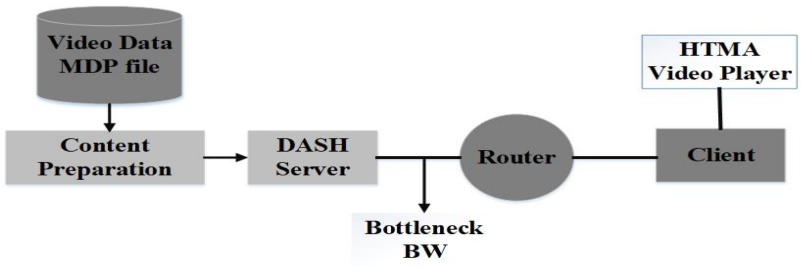

**Figure 4:** Network Topology for performance evaluation.

[1] http://w ww-itec.uni-klu.ac.at/ftp/datasets/DASHDatas et2014/BigBuckBunny/2sec/

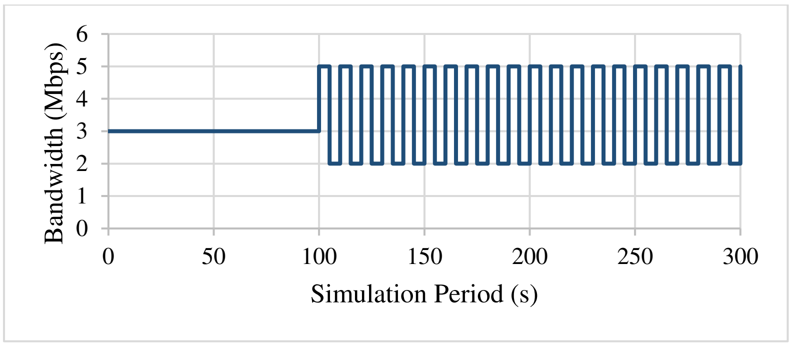

**Figure 5:** Bandwidth's Behaviour of the network.

### *4.1 Average Video Bitrate*

The average video bitrate represents the average per chunk quality during the streaming session. Figure 6 shows the client's adaptive behaviour in terms of average video bitrates achieved by three clients in two different scenarios. In this section, we demonstrate how the proposed algorithm performs as segment duration and buffer size vary. In the following experiements, we set the segment duration to 2s and 4s to evaluate the performance of proposed rate adaptation algorithm. Also, the buffer size is set to 40s and 60s for all the experiemts.

In the first scenario, the segment duration and buffer size are set to 2s and 40s, respectively. We observe that the proposed algorithm gains a video rate of approximately 2.91Mbps compared to the 2.78 Mbps of ALDASH and FDASH. In the second scenario, the buffer level is kept the same, and the segment duration is increased to 4s. Our proposed client achieves 10 Kbps more average video bitrate as compared to ALDASH and FDASH. In the second scenario, the buffer size is set to 60s. Figure 6 shows the average video rates achieved by the proposed algorithm. The client attains the highest video rate with our proposed algorithm. On average, our proposed client gain nearly 25 Kbps higher bitrate than ALDASH and FDASH for both 2s and 4s segment durations. Thus, the proposed client achieves a higher average video bitrate to give high quality streaming to the end-user, as shown in Figure 6. For the 1st scenario, all three clients achieve higher video bitrates compared to the other scenarios. This is because the small buffer space quickly increases to the buffer operating thresholds instead of remaining long in the initial threshold. Furthermore, all three clients achieve a little lower bitrate for the second scenario than the first scenario because the longer segment duration takes a longer time to download the segments for the given bottleneck link.

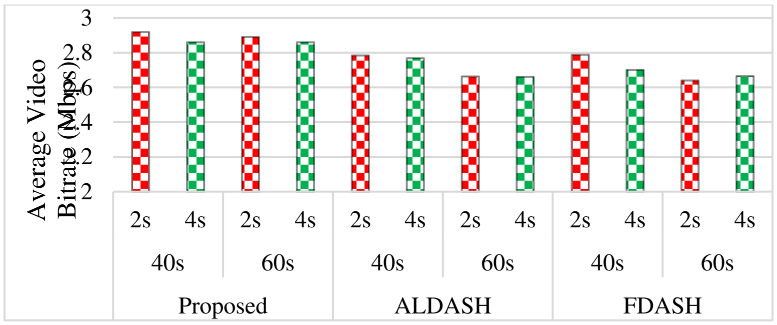

**Figure 6:** Performance comparison of Average Video Bitrate.

### *4.2 Average Buffer Level*

This section provides the average buffer level for three different bitrate adaptation schemes over two different scenarios. Figure 7 shows the results of our proposed algorithm. The rate adaptation algorithm should be able to guarantee the best quality under different client settings. In the following experiments, we set the segment duration to 2s and 4s to evaluate the performance of proposed rate adaptation algorithm. The buffer size is set to 40s and 60s for the evaluation of rate adaptive algorithms performance.

For the first scenario, the buffer level is set to 40s, and segment duration is set to 2s and 4s. Figure 7 shows the average video rates obtained by the ALDASH, FDASH, and the proposed algorithms over the streaming session. It is observed that our proposed algorithm achieves 17s and 21.8s of the average buffer level for 2s and 4s, respectively. While, ALDASH obtains 28.7s and 29.8s and FDASH achieves 26.4s and 29s of average buffer level for 2s and 4s, respectively. We can observe that FDASH performs well as compared to the ALDASH. Likewise, for the second scenario, the buffer level is set to the 60s for the segment duration of 2s and 4s, respectively. Again, we observe that the proposed algorithm achieves a better average buffer level compared to ALDASH and FDASH algorithms, as shown in Figure 7. Consequently, the video segments will lead to a higher number of segments with a small duration than the large segment size.

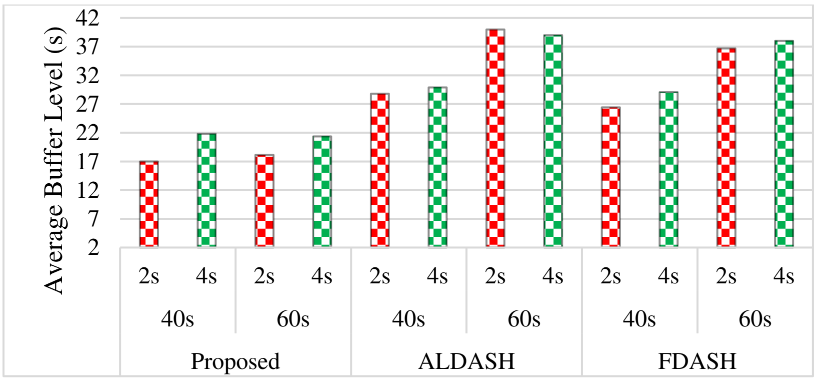

**Figure 7:** Performance Comparison of Average Buffer Level.

### *4.3 Video Bitrate Switching*

Figure 8 shows the performance of adaptive algorithms to demonstrate the effect of the number of bitrate switches with different segment duration by the client in two different scenarios. In the following experiements, we set the segment duration to 2s and 4s to evaluate the performance of proposed rate adaptation algorithm. Also, the buffer size is set to 40s and 60s for all the experiemts.

In the first scenario, the segment duration and buffer size are set to 2s and 40s, respectively. We observe that our proposed algorithm attains the low bitrate changes, approximately 13 compared to 36 of FDASH. While, ALDASH performs well as compared to FDASH. In the second scenario, the buffer level is kept the same and segment duration is increased to 4s. It can be seen from the figure that the client employing the proposed adaptation algorithm experiences the minimum number of quality switches than other competing algorithms while maintaining the playback buffer within the limits. Likewise, for the second scenario, the buffer level is set to the 60s for the segment duration of 2s and 4s, respectively. We observe that our proposed algorithm attains the low bitrate changes, approximately 15 compared to 38 of FDASH. While, ALDASH performs well than FDASH. In the second scenario, the buffer level is kept the same, and the segment duration is increased to 4s. Again, we observe in the following Figure 8; the proposed algorithm achieves a lower number of video rate changes than ALDASH and FDASH algorithms. Furthermore, the higher value of bitrate switching by proposed algorithm shows that it will aggressively improve the video rate for better utilization of available throughput.

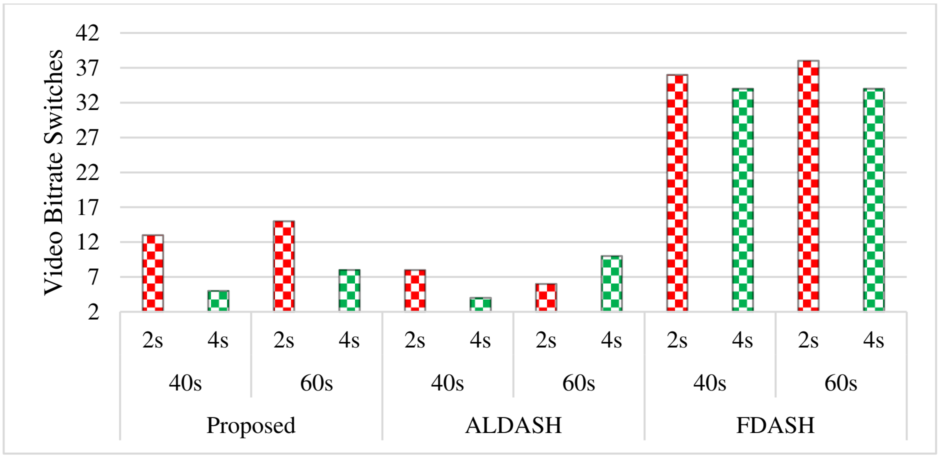

**Figure 8:** Performance Comparison of Video Bitrate Switches

**Table 4:** Performance comparison of ALDASH, FDASH and proposed algorithm.

| | Proposed | | | | ALDASH | | | | FDASH | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 40s | | 60s | | 40s | | 60s | | 40s | | 60s | |
| | 2s | 4s | 2s | 4s | 2s | 4s | 2s | 4s | 2s | 4s | 2s | 4s |
| Average Video Bitrate | 2.92 | 2.86 | 2.89 | 2.86 | 2.78 | 2.77 | 2.66 | 2.66 | 2.79 | 2.7 | 2.64 | 2.67 |
| Average Buffer Lev-el | 17 | 21.81 | 18.12 | 21.36 | 28.78 | 29.90 | 40 | 39 | 26.43 | 29.07 | 36.69 | 38 |
| Video Bitrate Switches | 13 | 5 | 15 | 8 | 8 | 4 | 6 | 10 | 36 | 34 | 38 | 34 |

## 5. Conclusion and Future Work

DASH has been extensively used as an important adaptive streaming technology over the Internet. It has the potential to provide the video quality to the user even with different network conditions. We presented a description of each strategy by investigating the issues they are trying to resolve, their goals, and findings. This paper proposed a DASH-based rate adaptation algorithm that provides a smooth video bitrate in a single-client environment. Our proposed algorithm improves the viewing experience through playback buffer occupancy and available network throughput. The proposed algorithm guarantees better video quality from the simulation results compared to other comparative algorithms ALDASH and FDASH in terms of the video bitrate, a minimum number of interruptions, and video quality switches. Our comparison may help the researchers in adaptive streaming by facilitating a general consistent framework for comparing different rate adaptation strategies. Since our experimental work environment is limited to the wired network rather than the wireless network. Our focus is on investigating and evaluating the effect within the network environment of different metrics.

We leave our experiments with different settings by considering the wireless environments for our future work. The different factors (e.g., video content, codec, etc.) can impact the user's video quality. Some issues have already been resolved, but we will make it our future work to analyze these issues with the latest and popular streaming services. Also, it facilitates additional research possibilities in areas where improvements are needed by paving the ways for future streaming applications.

**Funding:** Not applicable.
**Conflicts of Interest:** The authors declare that there is no conflict of interest.
**Availability of data and material:** Not applicable.
**Code availability:** At this stage, we can not provide the code because we are extending our project.
**Authors' contributions:** All authors contributed equally.

## References

1. G. M. D. T. Forecast,(2019) ”Cisco visual networking index: global mobile data traffic forecast” update, 2017–2022., vol. 2017, pp. 2022.
2. Shafi, R., Shuai, W., & Younus, M. U. (2020). 360-Degree Video Streaming: A Survey of the State of the Art. *Symmetry*, *12*(9), 1491.
3. Sandvine (2014), "The global internet phenomena report: 1H," Report.
4. C. V. N. Index (2016), "Forecast and methodology, 2016–2021," *White paper*.
5. PANTOS, R. (2010), Iphone HTTP Live Streaming: http://tools. ietf. org/html/draft-pantos-http-live-streaming-05. *Apple Inc.*, vol. 19, pp. 22.
6. T. Stockhammer,(2011). “Dynamic adaptive streaming over HTTP-- standards and design principles,” In : *Proc. of the second annual ACM conference on Multimedia systems*. (pp. 133-144).
7. Shafi, R., Shuai, W., & Younus, M. U. (2020). MTC360: A Multi-Tiles Configuration for Viewport-Dependent 360-Degree Video Streaming. In *2020 IEEE 6th International Conference on Computer and Communications (ICCC)* (pp. 1868-1873).
8. S. lederer (2015),. “Optimal Adaptive Streaming Formats MPEG-DASH & HLS Segment Length,” *Bitmovin*.
9. J. Van Der Hooft, S. Petrangeli, T. Wauters, R. Huysegems, T. Bostoen, and F. De Turck (2018), “An HTTP/2 push-based approach for low-latency live streaming with super-short segments,” *Journal of Network and Systems Management*, 26(1), 51-78).
10. T. nguyen, VU Thuan,, D. K. phuong, Vo Thi Luu (2019), “Performance of DASH over Multipath TCP,” *REV Journal on Electronics and Communications*, 9, (1-2).
11. C. Timmerer and C. Müller (2010), “HTTP streaming of MPEG media,” *Streaming Day*.
12. I. Ayad,, Y. Im, E. Keller, and S. Ha (2018), “A practical evaluation of rate adaptation algorithms in http-based adaptive streaming,” *Computer Networks*, 133(3), 90-103.
13. K. Spiteri, R. Urgaonkar, and R.K. Sitaraman (2020), “BOLA: Near-optimal bitrate adaptation for online videos,” *IEEE/ACM Transactions on Networking*, 28(4), 1698-1711.
14. Z. Li, X. Zhu, J. Gahm, R. Pan, H. Hu, A.C. Begen,, and D. Oran,(2014). “Probe and adapt: Rate adaptation for HTTP video streamzng at scale,” *IEEE Journal on Selected Areas in Communications*, 32(4), 719-733.
15. T.Y. Huang, R. Johari, N. McKeown, M. Trunnell, and M. Watson (2014), “A buffer-based approach to rate adaptation: Evidence from a large video streaming service,” In : *Proc. of the 2014 ACM conference on SIGCOMM*. (pp. 187-198).
16. Nguyen, Duc V., Huyen TT Tran, Anh T. Pham, and Truong Cong Thang (2017). "A new adaptation approach for viewport-adaptive 360-degree video streaming." In *2017 IEEE International Symposium on Multimedia (ISM)*, (pp. 38-44). IEEE.

17. He, Dongbiao, Cedric Westphal, and J. J. Garcia-Luna-Aceves. (2018)"Joint rate and fov adaptation in immersive video streaming." In *Proceedings of the 2018 Morning Workshop on Virtual Reality and Augmented Reality Network*, (pp. 27-32).
18. H. Yuan, X. Hu,, J. Hou, X. Wei, and S. Kwong (2019), "An Ensemble Rate Adaptation Framework for Dynamic Adaptive Streaming Over HTTP," *IEEE Transactions on Broadcasting*.
19. A. Wafa, A. Ashraf, T. and S. A. Maazen (2020), "An Adaptive Quality Switch-aware Framework for Optimal Bitrate Video Streaming Delivery," *International Journal of Advanced Computer Science and Applications(IJACSA)*,11(8).
20. D.J. Vergados, A. Michalas, A. Sgora, D. D. Vergados, and P. Chatzimisios (2015), "FDASH: A fuzzy-based MPEG/DASH adaptation algorithm," *IEEE Systems Journal*, 10(2), (pp. 859-868).
21. W. U. Rahman, and K. Chung (2017), "A novel adaptive logic for dynamic adaptive streaming over HTTP," *Journal of Visual Communication and Image Representation*, (vol. 49, pp. 433-446).
22. Ur Rahman, Waqas, and Kwangsue Chung (2018). "SABA: Segment and buffer aware rate adaptation algorithm for streaming over HTTP." *Multimedia Systems* 24(5), 509-529.

**Biography:**

**Dr. Engr. Muhammad Usman Younus** received his Doctorate and master's degree in engineering from the University of Toulouse (III) Paul Sabatier France and University of Engineering and Technology Lahore Pakistan 2020 & 2014, respectively. His research interests include Wireless Sensor Network, Software Defined Networking, Energy Optimization, Wireless Communication, and Machine Learning. He has more than twenty international journal and conference publications. He is a member of PEC, IEEE, etc. and reviewer of some journals and conferences.

**Engr. Rabia Shafi** received the B.S. degree in Electrical Engineering from Govt. College University Faisalabad, Pakistan in 2014. She is currently pursuing the Ph.D. degree in information and communication engineering from Northwestern Polytechnical University, Xi'an China. Her main research interests include image and video processing.