A model for a practical evaluation of a DASH-based rate adaptive algorithm over HTTP

Muhammad Usman Younus1,2 · Rabia Shafi3 · Sultan S. Alshamrani4

Accepted: 31 May 2022 / Published online: 12 August 2022
© The Author(s), under exclusive licence to Springer Science+Business Media, LLC, part of Springer Nature 2022

Abstract
The proliferation of multimedia devices and user-generated content has driven a massive growth in Internet traffic for video streaming. There is a dire need to propose such solutions that can address 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 makes it easy to improve user perception and define the media presentation description (MPD) in terms of Uniform Resource Locator, 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. Our proposed algorithm is tested using the NS-3 network simulator, which shows that it improves video quality performance when compared to ALDASH and FDASH in terms of user satisfaction. Last but not least, the experimental results of our proposed algorithm provide high viewer quality in adaptive streaming as compared to its competitors.

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]. Dynamic Adaptive Streaming over HTTP (DASH) [6] is a propitious adaptive video streaming standard released by the collaboration of Moving Picture Experts Group (MPEG) and Third Generation Partnership Project (3GPP) in 2011.

Although rate adaptation algorithms have become increasingly important in video streaming, they are used to select the optimal video bitrate after analyzing the network condition for a better user experience. Therefore, a good rate adaptation algorithm is described by its ability to respond to network changes while achieving the optimum feasible throughput. The following are the main contributions of our paper:

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., 40 s, 60 s) and segment duration (e.g., 2 s, 4 s) 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 highlights the impact of video quality metrics using NS-3. Our simulation results show that the proposed algorithm performs well for achieving user satisfaction by keeping resolution changes to lower levels.

3. We also describe our insights for future research.

The rest of the paper is organized as follows: Sect. 2 discusses MPEG-DASH framework and some related work, which includes a description of the old video streaming techniques. The system design is explained in Sect. 3, which incorporates adaptive goals, throughput estimation, chunk scheduling, and the rate adaptation algorithm. The performance evaluation, on the other hand, is discussed in Sect. 4. Finally, Sect. 5 concludes the whole paper and suggests some areas for future research.

2 Related work

Multimedia service providers are trying to obtain the methods for enhanced playback of streaming content without downloading the complete file [7]. 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 (QoE) 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.

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 Transmission Control Protocol (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 s). 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. On the webserver, the MPD file and video parts are hosted. During the streaming session, the MPD file is initially returned to the client, which extracts the metadata in order to adaptively fetch the appropriate segments. Typically, the segment size consists of a fixed duration in all existing systems, such as Apple recommends 10 s for segment size, Microsoft selects the segment size in the range of 2–4 s, YouTube, and Netflix suggest 5–10 s of segment size [6, 9]. Then, the client measures the available throughput and dynamically adapts the quality levels to facilitate high-quality streaming. The client chooses the viewing frequency based on network conditions and starts the content streaming using an HTTP GET request. The MPEG-DASH specification defines only MPD and segment formats.

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 filled, the client goes through an ON–OFF scheduling pattern, as shown in Fig. 2. During the ON state, the client requests the next segment, while during the OFF state, it waits for enough buffer space. Due to varying network conditions, multiple DASH clients may have unequal bandwidth distribution during the ON–OFF phase, resulting in a poor video streaming experience.

The effect of the segment duration and buffer space associated with DASH streaming plays a critical role. The segment size configuration is important for an efficient adaptive playback experience. The small segment duration reduces the amount of data required to initiate the video playback. Shortening the video segments results in low start-up time and lower latency [10]. However, increasing the video segment duration may result in high video bitrates and fewer video quality switches, and the number of HTTP requests over different bandwidth profiles [11]. Furthermore, as the length of the segment gets longer, it becomes more vulnerable to interruptions in playback.

Many studies have been conducted to ensure the user satisfaction to adapt the high video bitrates through the DASH standard, while considering the trade-off in several resources (e.g., quality, bandwidth). One of the major problems in competing with HAS players is stability in terms of quality levels. When HAS clients compete for a bottleneck link, the authors of [12] explored how HAS player behavior in adaptive video streaming can lead to a fairness issue. In another study [13], the experimentation tools have been provided by introducing an evaluation framework for adaptive streaming. However, this framework does not facilitate any support for commercial stream player. Our primary concern is the study...
of adaptation algorithms. 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 appropriate bitrate for the next video segment. 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, reducing the rebuffering 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 based on bandwidth situation, then compares the two bitrate threshold values to set the appropriate bitrate. When the client’s chosen bitrate exceeds the available bandwidth, the use of the bitrate-based adaptive method will lead to greatly reduce the user’s QoE.

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 Proportional Integral Derivative (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, 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 QoE.

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 purpose is 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 (FDASH) scheme 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. 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. It also estimates the throughput.
depending on network conditions in a single-user environment.

In this paper, we consider the both 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., 2 s, 4 s) and buffer levels (i.e., 40, 60 s) 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. A MPD file is 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 $B(i-1)$ 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. A scheduler module controls the interval after the video rate is chosen.

### 3.1 Adaptation goals

According to a study conducted by Jackson, an effective ABR algorithm is required to provide better QoE at the middle and end of the streaming session with the least amount of quality fluctuations and buffer underflow. The video bitrate switching annoys users because end-users suffer from lower video quality. In addition, the user starts leaving the video if the video does not play within 2 s. An optimal solution of rate adaptive streaming is to facilitate the highest possible QoE under given network conditions. Here, we consider the following three components of QoE:

#### 3.1.1 Optimize the video rates

One of the main goals of an adaptive algorithm is to adaptively choose a video rate from a video set to maximize the user’s QoE. 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 the rate at which video quality is switched in a specific unit of time during streaming session. It is found that a non-zero quality (e.g., video bitrate, etc.) of two consecutive video segments corresponds to a change in quality. The fewer bitrate changes can provide the high user’s satisfaction than video of higher bitrates with a higher number of bitrate changes.

#### 3.1.3 Minimize the number of interruptions

Interruption is the freezing of video playback due to buffer underflows. To achieve good video quality, the adaptation algorithms should minimize playback interruptions while adjusting the bitrate according to network conditions. The user will consume content more frequently if they have a

---

1 A User Study of Netflix Streaming.
better experience and a longer video playback time. The video rates and rebuffering events are incompatible, because increasing the video rate would result in extensive rebuffering.

In this work, the above aspects are achieved by providing the solution consists of following factors:

- 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:

\[ T(i) = \frac{r[i] \cdot \tau}{t_{end}^{i} - t_{start}^{i}} \]  

where \( r[i] \) is the bitrate of \((i)th\) segment, \( \tau \) represents the playback duration, \( t_{end}^{i} \) and \( t_{start}^{i} \) 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.

To reduce the frequency of variations, the average throughput measurement stabilizes the estimation over time. The average throughput of \( n \) downloaded segments is estimated during a specified period \( d \) as follows:

\[ T_d(i) = \frac{1}{n} \cdot \sum_{i=1}^{n} T(i) \]  

where \( T_d(i) \) is the estimated throughput, \( T(i) \) is a segment throughput and \( i \) is an index of segment number. The throughput for the next segment is estimated using the current buffer level zone and is presented as:

\[ T_d^e(i) = \alpha \cdot T_d(i) \]  

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:

\[
\alpha = \begin{cases} 
0.5B(i) \leq B(p) \\
0.5(B(i)+B(g)) - B(p) & \text{if } B(i) < B(g) \\
1 & \text{otherwise}
\end{cases}
\]  

(4)

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

Suppose a client makes a request for the segment \((i + 1)\) in time \( t_{start}(i + 1) \) and \( t_{end}(i) \), it is that time when transmission of the segment \((i)\) is completed. \( B(\text{target}) \) 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:

\[
t_{start}(i + 1) = \begin{cases} 
t_{end}(i) + (B(i) - B(\text{target}) + \tau)/\tau & \text{if } B(i) \geq B(\text{target}) \\
\end{cases}
\]  

(5)
The above equation indicates that it requests for the
next segment immediately after downloading the previous
segment when \( B(i) \) is less than the target buffer level,
\( \text{i.e., } B(\text{target}) \). While, the client has to wait for \((B(i) -
B(\text{target}) + \tau)/\tau \) seconds to make the next segment request
when \( B(i) \) increases above the \( B(\text{target}) \).

### 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,
\( \text{i.e., } R = \{r(1), r(2), ..., r(n)\} \). 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, 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 \( \frac{\tau}{\tau} \).

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 \( r_{↑↑}/r_{min} \) seconds and the next two higher quality
levels are below the estimated throughput, \( \text{i.e., } r_{↑↑} \leq
T^{est}(i) \). 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 \( r_{↑} \). Otherwise, it will maintain
the previous bitrate.

If the current buffer level exceeds the stable level \( B(s) \),
the most suitable video bitrate (\( \text{e.g., probably the highest bitrate}
from set } R \) to network condition is selected that does not exceed the estimated throughput to improve the video
quality (Table 1).

#### Table 1 Notations

| Notations | Meaning |
|-----------|---------|
| \((i)\)   | Index of segment number |
| Test \((i)\) | Estimated throughput for segment \((i)\) |
| \(B(i)\)  | Playback buffer level for segment \((i-1)\) |
| \(r[i]\)  | Bitrate of \((i)th \) segment |
| \(\tau\)  | Playback duration |
| \(t_{start}\)  | Start time of receiving the \((i)th \) segment |
| \(t_{end}\)  | 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 \(d\) |
| \(B(\text{target})\) | 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 |
| \(r_{↑}\)  | Increase the quality by one level |
| \(r_{↑↑}\) | Increase the quality by two level |

#### 4 Performance evaluation

For evaluating the proposed rate adaptive algorithm, the
network simulator NS-3 has been used to perform the experiments. Table 2, 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, \( \text{i.e., server, router, and}
\)client nodes. The server is an Apache 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 8 GB RAM. DASH server contains the Big Buck Bunny

#### Table 2 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 | 2 s & 4 s |
| Buffer size in seconds | 40 s & 60 s |
| Bandwidth | 6Mbps |
| Simulation period | 300 s |
A model for a practical evaluation of a DASH-based rate adaptive algorithm…

![Network Topology for performance evaluation](http://witec.uni-klu.ac.at/ftp/datasets/DASHDatasets2014/BigBuckBunny/2s)

**Fig. 4** Network Topology for performance evaluation.

![Bandwidth's Behaviour of the network](http://www-itec.uni-klu.ac.at/ftp/datasets/DASHDatasets2014/BigBuckBunny/2s)

**Fig. 5** Bandwidth’s Behaviour of the network

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.

As the client can more quickly switch to a higher quality level at start, or also can recover from low buffer occupancy [2, 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 playback 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 3 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 2 s for buffer level 40 s.
2. Three different rate adaptation schemes performance is evaluated with a segment duration of 2 s for buffer level 60 s.
3. Three different rate adaptation schemes performance is evaluated with a segment duration of 4 s for buffer level 40 s.
4. Three different rate adaptation schemes performance is evaluated with a segment duration of 4 s for buffer level 60 s.

| Table 3 Performance comparison of ALDASH, FDASH and proposed algorithm |
|------------------|------------------|------------------|------------------|
|                  | Proposed         | ALDASH           | FDASH            |
| 40 s             | 60 s             | 40 s             | 60 s             | 40 s             | 60 s             |
| 2 s              | 4 s              | 2 s              | 4 s              | 2 s              | 4 s              |
| 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 Level | 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 |
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 experiments, we set the segment duration to 2 s and 4 s to evaluate the performance of proposed rate adaptation algorithm. Also, the buffer size is set to 40 s and 60 s for all the experiments.

In the first scenario, the segment duration and buffer size are set to 2 s and 40 s, respectively. We observe that the proposed algorithm gains a video rate of approximately 2.91 Mbps compared to the 2.78 Mbps of ALDASH and FDASH. When the buffer level is kept the same and the segment duration is increased to 4 s then 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 60 s. Figure 6 shows the average video rates achieved by the proposed algorithm. The client attains the highest video rate with our proposed algorithm. FDASH has approximately equal video bitrate to our proposed algorithm, but our proposed approach has a more stable response to bandwidth fluctuations with lower quality switches. Whereas, ALDASH shows the aggressive response of video bitrate in terms of bandwidth drops.

On average, our proposed client gains nearly 25 Kbps higher bitrate than ALDASH and FDASH for both 2 s and 4 s segment durations. Thus, the proposed client achieves a
higher average video bitrate to give high quality streaming to the end-user, as shown in Fig. 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.

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 2 s and 4 s to evaluate the performance of proposed rate adaptation algorithm. The buffer size is set to 40 s and 60 s for the evaluation of rate adaptive algorithms performance.

For the first scenario, the buffer level is set to 40 s, and segment duration is set to 2 s and 4 s. 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 17 s and 21.8 s of the average buffer level for 2 s and 4 s, respectively. While, ALDASH obtains 28.7 s and 29.8 s and FDASH achieves 26.4 s and 29 s of average buffer level for 2 s and 4 s, respectively. We can observe that FDASH performs well as compared to the ALDASH.

For the second scenario, the buffer level is set to the 60 s for the segment duration of 2 s and 4 s, respectively. Again, we observe that the proposed algorithm achieves a better average buffer level compared to ALDASH and FDASH algorithms, as shown in Fig. 7. Consequently, the video segments will lead to a higher number of segments with a small duration than the large segment size.

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 experiments, we set the segment duration to 2 s and 4 s to evaluate the performance of proposed rate adaptation algorithm. Also, the buffer size is set to 40 s and 60 s for all the experiments.

In the first scenario, the segment duration and buffer size are set to 2 s and 40 s, 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 4 s. 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 kept the same, and the segment duration is increased to 4 s. Again, we observe in the following Fig. 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.

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 to investigate and evaluate 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.

Author contributions The contributions of the authors are listed below: M.U. Younus: Conception, coding, experimentations, writeup and proof read. R. Shafi: Conception, experimentations and writeup. S.S Alshamrani: Proof read and reviewed the manuscript.

Funding Not applicable.

Data availability Not applicable.

Code availability At this stage, we can not provide the code because we are extending our project.

Declarations

Conflict of interest The authors declare that there is no conflict of interest.

References

1. Forecast, G. M. D. T. (2019). Cisco visual networking index: Global mobile data traffic forecast update, 2017–2022. Update, 2017, 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, I. (2014). Global internet phenomena report: 1H. Report/ Index, C. V. N. (2017). Forecast and methodology, 2016–2021. White paper.
4. Pantos, R. (2010). Iphone HTTP live streaming: https://datatracker.ietf.org/doc/html/draft-pantos-http-live-streaming-05. Apple Inc., vol. 19, p. 22.
5. Stockhammer, T. (2011). Dynamic adaptive streaming over HTTP- standards and design principles. In Proceedings of the second annual ACM conference on Multimedia systems (pp. 133-144).
6. Timmerer, C., & Müller, C. (2010). HTTP streaming of MPEG media. Streaming Day.
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). IEEE.
8. Lederer, S. (2015). Optimal adaptive streaming formats MPEG-DASH & HLS segment length. Bitmovin.
9. Van Der Hooft, J., Petrangeli, S., Wauters, T., Huysseens, R., Bostoen, T., & De Turck, F. (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. Nguyen, T. T., Vu, D. K., Phuong, V. T. L., Le, C., & Nguyen, Q. D. (2019). Performance of DASH over Multipath TCP. REV Journal on Electronics and Communications, 9(1-2).
11. Ayad, I., Im, Y., Keller, E., & Ha, S. (2018). A practical evaluation of rate adaptation algorithms in http-based adaptive streaming. Computer Networks, 133(3), 90–103.
12. Spiteri, K., Urgaonkar, R., & Sitaraman, R. K. (2020). BOLA: Near-optimal bitrate adaptation for online videos. IEEE/ACM Transactions on Networking, 28(4), 1698–1711.
13. Li, Z., Zhu, X., Gahm, J., Pan, R., Hu, H., Begen, A. C., & Oran, D. (2014). Probe and adapt: Rate adaptation for HTTP video streaming at scale. IEEE Journal on Selected Areas in Communications, 32(4), 719–733.
14. Huang, T. Y., Johari, R., McKeown, N., Trunnell, M., & Watson, M. (2014, August). A buffer-based approach to rate adaptation: Evidence from a large video streaming service. In Proceedings of the 2014 ACM conference on SIGCOMM (pp. 187-198).
15. Nguyen, D. V., Tran, H. T., Pham, A. T., & Thang, T. C. (2017). A new adaptation approach for viewport-adaptive 360-degree video streaming. In 2017 IEEE International Symposium on Multimedia (ISM) (pp. 38-44). IEEE.
16. He, D., Westphal, C., & Garcia-Luna-Aceves, J. J. (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).
17. Yuan, H., Hu, X., Hou, J., Wei, X., & Kwong, S. (2019). An ensemble rate adaptation framework for dynamic adaptive streaming over HTTP. IEEE Transactions on Broadcasting, 66, 251–263.
18. Alqhtani, W. A., Taha, A. A., & Alsabaan, M. S. (2020). An adaptive quality switch-aware framework for optimal bitrate video streaming delivery. International Journal of Advanced Computer Science and Applications, 11(8), 1–10.
19. Vergados, D. J., Michalas, A., Sgora, A., Vergados, D. D., & Chatzimisios, P. (2015). FDASH: A fuzzy-based MPEG/DASH adaptation algorithm. IEEE Systems Journal, 10(2), 859–868.
21. Rahman, W. U., & Chung, K. (2017). A novel adaptive logic for dynamic adaptive streaming over HTTP. *Journal of Visual Communication and Image Representation*, 49, 433–446.
22. Chung, K. (2018). SABA: Segment and buffer aware rate adaptation algorithm for streaming over HTTP. *Multimedia Systems*, 24(5), 509–529.

**Publisher’s Note** Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

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 25 international journal and conference publications. He is a member of PEC, IEEE, etc. and reviewer of some journals and conferences.

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.

Sultan S. Alshamrani is currently working as an associate professor at Taif University in Saudi Arabia and he is the head of the department of Information Technology and the Chairman of the committee of faculty members affairs. DR. Sultan got his PhD from the University of Liverpool in UK and a master’s degree in Information Technology (Computer Networks) from the University of Sydney in Sydney, Australia. DR. Sultan finished his bachelor’s degree in computer science from Taif University in 2007 with General Grade “Excellent” With first honor and an accumulative GPA of (4.85) out of (5.00) where considered the highest GPA in the collage.