Signal, Image and Video Processing (2021) 15:1477–1485
https://doi.org/10.1007/s11760-021-01880-y

ORIGINAL PAPER

Time-efficient evaluation of adaptation algorithms for DASH with SVC: dataset, throughput generation and stream simulator

Mehmet Çalı[1] · Nükhet Özbek[2]

Received: 10 June 2020 / Revised: 6 January 2021 / Accepted: 1 March 2021 / Published online: 29 April 2021

Abstract
Bitrate adaptation algorithms have received considerable attention recently. In order to evaluate these algorithms objectively, multiple DASH datasets have been proposed. However, only few of them are compatible to SVC-based adaptation algorithms. Apart from the dataset, to fully implement and evaluate an adaptation algorithm, many time-consuming steps are required such as MPD parser design, adaptation logic design and network environment setup. In this paper, a dash simulator which assesses the performance of SVC-based adaptation algorithms without the requirement of any additional implementation steps is proposed. Also, an SVC dataset that includes both CBR and VBR encoded videos is designed. Demonstration is performed as evaluation of an SVC-based adaptation algorithm under several throughput scenarios using the designed dataset. Results show that the proposed system considerably reduces time requirement compared to real-time assessment. Dataset, throughput generation tool and simulator are all publicly available so that the researchers can test their implementation and compare with the results presented in this paper.

Keywords Dynamic adaptive streaming over HTTP (DASH) · Scalable video coding (SVC) · Streaming simulation · DASH dataset

1 Introduction

As the share of video network traffic is estimated to increase by 10 percent in 5 years [1], researchers pay more an more attention to optimization of adaptive video streaming. Many of the adaptive video streaming platforms based on Dynamic Adaptive Streaming over HTTP (DASH) [2] standard. However, the standard does not specify the adaptation part of the framework which requires optimizations in many aspects. Therefore, there are numerous implementations of DASH adaptation with multiple video encoding schemes under various network scenarios. With the development of large number of adaptation algorithms, accurate and extensive evaluation and comparison of these studies become more significant issue.

Recently, a number of researches covering evaluation of DASH implementations have been conducted. In [3], average bitrate and bitrate switching amount of some well-known open source and commercial applications are measured with a setup based on netem emulation. In [4], an environment is designed for assessment of popular adaptive HTML5 players using Mininet emulation. In [5], evaluation of authors' previous study fDASH [6] and some of the recent implementations are provided with the help of measured bitrate and interrupt duration in vehicular scenarios. These researches have a common approach of focusing on results which are achieved with low number of trials in mostly non-reproducible test conditions. On the other hand, proposed work focuses on detailed description and generation of publicly accessible, reproducible evaluation framework. Even if the studies with larger scope are generally more result oriented, studies handling the adaptive streaming evaluation in more specific categories presents favorable outcomes for future researches. These studies of DASH evaluation framework can be catego-

Supplementary Information The online version contains supplementary material available at https://doi.org/10.1007/s11760-021-01880-y.

✉ Nükhet Özbek
nukhet.ozbek@ege.edu.tr

Mehmet Çalı
mehmetcali@iyte.edu.tr

1 Department of Electrical and Electronics Engineering, Izmir Institute of Technology, Izmir, Turkey

2 Department of Electrical and Electronics Engineering, Ege University, Izmir, Turkey

rized into dataset design, throughput waveform creation and simulation parts.

In general, DASH datasets contain several video chunks each of which has representation and segment ids. Segments specify the time interval where the video chunk belongs to. A video chunk at a specific segment can have different representation ids. Representations specify the quality or bitrate of a video chunk for a given segment. For scalable encoded videos, video chunks are represented with layer and segment ids [7]. When the layer id increases the quality of the video chunk increases as well. Higher layers have dependencies to the lower layers; therefore, the adaptation algorithm can improve the quality of segments inside the buffer at any time. This flexibility of scalable coded videos [8] and lower storage size requirement [9] make SVC favorable for DASH implementations. Although there are a number of DASH datasets in the literature [10–14], few of them employ scalable encoding [14]. In [14], videos are encoded with SVC in three to five layers in four different variants. However, they have only variable bitrate encoded videos in their dataset. In the proposed dataset, the raw videos are encoded in both variable and constant bitrate modes so that DASH implementations can be compared for datasets with different characteristics. Additionally, SSIM [15] values of video chunks are included in the media presentation description (MPD) file of each stream.

Most of the rate adaptation studies adopt two main approaches for throughput determination in the experiment setup phase. One of them is manually specifying throughput waveform [16–18], and the other one is using network tracing [19–21]. Manual specification of throughput is time-consuming thereby limiting the number of tests applied to the adaptation algorithm. Using network tracing on the other hand requires getting measurements from various locations under different conditions. The papers that adopt network tracing either make their own measurements or benefit from available tracing studies [22–24]. For example, one of the recent network trace database study [24] provides throughput measurement of scenarios such as bus, static and car in 4G networks. However, to utilize such studies, measured throughput mean and variation should be compatible to those of DASH dataset. For instance, if the measurement has higher throughput than the average bitrate of maximum layer, the test will not be able to compare adaptation algorithms properly. In other words, network traces should be compatible to dataset while satisfying high coverage of various network scenarios which may not occur for many experiments. In the proposed work, to compare DASH applications with extensive and compatible network characteristics, throughput generation with the help of a stochastic model is determined. With this method 1000 waveforms with a duration of 180 s are produced with a large scale of network feature variability. This variability is defined in three main features which are mean, variance and stationarity of the throughput waveform.

As the next step, testing of the DASH application is required. Most of the related studies employ a client and server hardware setup with a network emulation tool embedded in the server [16–19]. In this setup, client requests and receives the video chunks from server with a throughput set by a network emulator such as Linux traffic control utility. The disadvantage of this setup is the requirement of too much time for an extensive test since it needs to wait for the playback time before the evaluation. Moreover, the network emulation tools cause multiple reproducibility issues. First one is being unable to produce the exact same waveform in different executions due to emulation error. The other one is the synchronization issue that stems from execution of network emulation and adaptation from different devices that are server and client. Another approach on adaptation test phase is designing a system which employs Mininet instead of client and server hardware setup. For example, a framework for both subjective and objective evaluation of adaptive streaming algorithms is proposed in [25]. Although they come up with a well-designed framework with a number of metrics, there is no information regarding generation of throughput waveforms. Besides, their system still suffers from excessive execution time while testing with an extensive throughput dataset as it still relies on real time emulation of network. To overcome these problems, a program which simulates the streaming of video chunks is designed in the proposed work. The simulator runs after adaptation logic in each streaming cycle to inform adaptation inputs such as instantaneous buffer length, bitrate and SSIM of the video chunks. Alongside adaptation inputs, QoE parameters such as average length of playback interrupts, SSIM mean and variance of streamed video chunks and low buffer detection information are calculated and illustrated according to user's selection. Designed dataset, throughput generation code and simulator are all provided in [26] for researchers to test their implementations.

The organization of remaining paper is structured as follows. In Sect. 2, the evaluation system details are described including designed SVC dataset, throughput generation process and stream simulator framework. In Sect. 3, performance of our previous work on SVC-based implementation [27] is tested against two adaptation algorithms with the help of designed evaluation system. Lastly, the summary of the paper is provided in Sect. 4.

2 Evaluation system

The majority of the system is designed in Python while throughput generation and graphic outputs are acquired using MATLAB. With all three sub-sections designed individually,

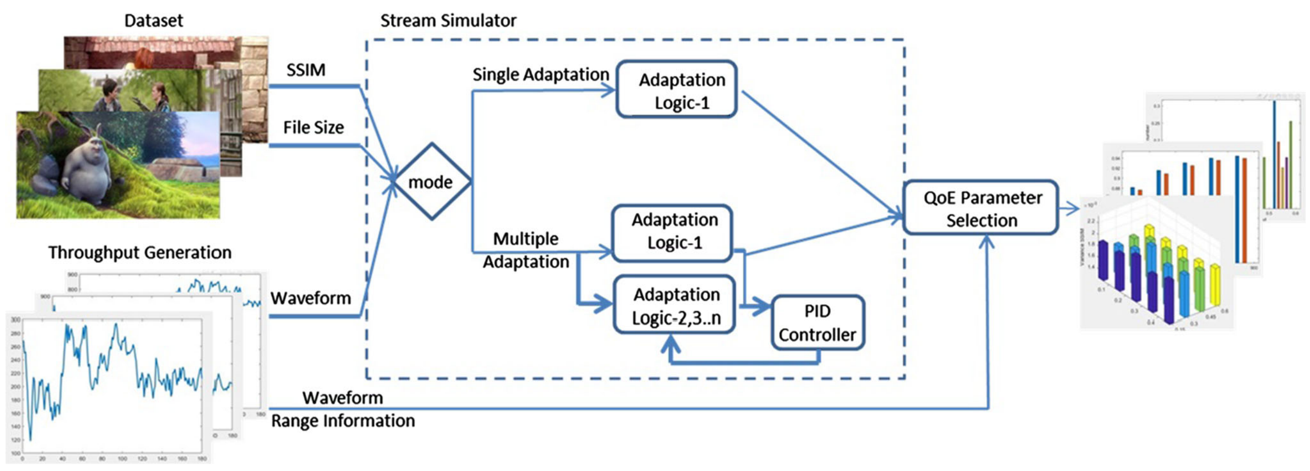

Fig. 1 Algorithm evaluation system

there are no external requirements. A brief overview of the evaluation system is represented in Fig. 1. The proposed system requires two main inputs, namely DASH dataset and TCP throughput waveform. The system is designed specifically for SVC datasets and SVC-based adaptation algorithms; however, with minor modifications, it can be adjusted to work with other standards. The user can directly take the results of the default dataset and throughput generation configuration as well as specifying these according to their test scenario. As the next step, streaming simulator computes the performance of the adaptation much more efficiently than the conventional approach which is the real-time streaming under emulated network. By means of a PID controller, the streaming simulator can also compare two adaptation algorithms while eliminating possible fairness-related issues which can stem from their target buffer length difference. Finally, selected output QoE parameters are displayed with respect to desired throughput characteristics. Hence, the users can observe deficient aspects of their algorithm and tune it easily with the help of fast computation.

2.1 SVC dataset

Created dataset can be investigated under two main categories which are CBR and VBR modes. In CBR mode, representations correspond to different bitrate levels. Although the bitrate within a particular representation stays near constant, the subjective quality of that representation may fluctuate a lot between segments. In VBR mode on the other hand, bitrate fluctuations increase while quality stays steadier compared to CBR mode. Implementing both modes brings the ability to measure the performance of adaptation against opposite dataset features. Moreover, some adaptation algorithms assume that the representations have steady bitrate or steady subjective quality and they design their adaptation logic accordingly. Such assumptions make an adaptation sensitive to changes in encoding modes. Hence, usage of separate coding modes enables measuring the dependency of adaptation to the characteristics of dataset. The modes of the designed dataset have some common properties which are described as follows. Both dataset modes are encoded with JSVM [28] at 640 × 360 px resolution. They have the frame rate, GOP size and segment duration of 24 frame/s, 8 frames and 2 s respectively. IDR period is selected as 48 frames since the start of each segment must be encoded independently. Each mode is encoded from publicly available three raw videos which are Big Buck Bunny [29], Tears of Steel [31] and Sintel [30]. MPD files are created using the descriptions of SVC demultiplexer tool proposed by [14]. Additionally, SSIM of individual video chunks are provided in MPD files.

2.1.1 VBR mode

VBR mode of the dataset consists of five layers. Since JSVM permits two CGS enhancement layers, MGS is used to add remaining enhancement layers. Individual CGS layers have constant QPs which are specified considering their impact on average bitrate and average quality. For example, the average bitrate of maximum layer have two constraints. First one is that average bitrate of the highest layer should be lower than the maximum limit that can be set from Token Bucket Filter (tbf), which is a tool to arrange the network traffic using Linux tc command. With this constraint the dataset is made applicable to the test setups that utilize tbf. Another constraint is the maximum SSIM which is set according to experiment of [32] in low-resolution images. Their experiment measures SSIM value of the compressed images that corresponds to Just Noticeable Difference (JND). Hence, average SSIM of the maximum layer is limited by SSIM value of 0.96 considering the JND match. After setting maximum layer QPs of raw videos according these constraints, lower layer QPs

Table 1 VBR mode average dataset characteristics

Parameters	Base layer	CGS-1		CGS-2	
		MGS-1	MGS-2	MGS-3	MGS-4
QP	39.3	33.3		29.3	
MGS Weights	–	3	13	3	13
Bitrate (Kbit/s)	153	302	405	635	845
Mean SSIM	0.880	0.917	0.931	0.947	0.957
Variance SSIM	0.00368	0.00178	0.00111	0.00068	0.00041

Table 2 CBR mode average dataset characteristics

Parameters	Base layer	CGS-1	CGS-2
Bitrate (Kbit/s)	177	350	703
Mean SSIM	0.8732	0.9177	0.9552
Variance SSIM	0.00784	0.00395	0.00133

Table 3 Desired throughput features

	Minimum	Maximum	Interval length
Mean (μ) (Kbit/s)	150	900	150
Variance (σ^2)	0.1μ	0.6μ	0.1μ
Percent stationarity	15%	75%	15%

and MGS vector weights are set such that a smooth quality and bitrate transition can occur between layers. Then, compressed videos are segmented by grouping the frames with the identical layer id. In this process, layer id of a frame is identified with the help of NALU prefixes and NALU headers. Average VBR mode dataset characteristics of all three videos are given in Table 1. The individual characteristics of each input video is provided in supplementary material for both modes.

2.1.2 CBR mode

Unlike VBR encoding, input videos are segmented before the compression in this part. A base layer and two CGS enhancement layers are used. Two seconds long segments are encoded iteratively until the desired bitrate thresholds are satisfied for each layer. Layer QPs are updated in each iteration with the help of JSVM FixedQPEncoder tool. If the iteration number reaches a specified maximum limit, the last iteration is accepted. The desired bitrates of layers are determined as 180, 360 and 720 Kbit/s. The thresholds are set as $\pm 3\%$ of the desired bitrates, while the maximum iteration number is set to 10. Since the input files are segmented before the encoding phase, only layer grouping is applied to the output of the encoder. Features of video chunks are presented in Table 2. The configuration file of each mode is supplied in [26] as well as the output dataset.

2.2 Sample throughput generation

One of the most underestimated parts of performance evaluation may be throughput generation. In many of the researches, implementations are compared using insufficient number of manually generated waveforms. In this paper, designing sufficient number of throughput waveforms for many different network characteristics is aimed.

Firstly, the produced waveform should cover a wide range of network characteristics. To this end, we specify three features to represent the characteristics of throughput, which are mean, variance and stationarity ratio. Mean is the most straightforward feature to be defined since it enables observation of the adaptation algorithm under various average connection speeds. Variance allows assessment of the algorithm behavior against the alteration in connection conditions. Lastly, stationarity ratio provides performance measurement against both predictable and non-predictable changes in the waveform. Before throughput generation, desired ranges of these three features are determined. These ranges are divided into intervals with certain length as described in Table 3. With the combination of mean, variance and stationarity intervals, 100 desired feature slots are specified as default. In the next phase, throughput generation will be performed such that ten waveforms are produced for each feature slot.

Using a successful throughput estimation model enables generation of more realistic waveforms. In this manner, throughput prediction studies can be investigated in two categories which are formula-based approaches and history-based approaches [33]. Formula-based approaches make prediction according to their calculations using network parameters such as measured packet loss, round trip time and bandwidth. On the other hand, history-based approaches treat TCP throughput as time series and make prediction with stochastic modeling. Since stochastic modeling is more applicable in the design procedure of realistic throughput waveform, we apply one of the history-based method which is proposed by [34].

$$y_t = c + \phi y_{t-1} + \epsilon_t \tag{1}$$

In [34], AR(1) process is used for throughput modeling after converting TCP throughput to time series as shown in Eq. (1). This process is stationary for $\phi < 0$ and non-stationary for $\phi = 0$. Firstly, stationarity of the process is measured with unit root test or Dickey-Fuller Test [35] specifically. Unit root test is calculated on a window with the length of 30 samples[1] in the waveform. The sampling window is shifted to measure the fluctuations in stationarity at each time instance. Therefore, the result of stationarity analysis forms another time series whose elements can take only the values 0 and 1. Afterward, stationarity analysis results are smoothed using exponential moving average so that they can be used as a random variable to specify probability of being stationary. Additionally, the probability of next time instance in an AR(1) process can be described as Gaussian process whose parameters can be calculated using estimations of c, ϕ and ϵ. Equations (2), (3), (4) and (5) state mean and variance of these Gaussian processes for stationary and non-stationary cases, respectively.

$$E_s = c + \phi y_{t-1} \tag{2}$$
$$V_s = \sigma^2_{\epsilon_t}(1 + \phi) \tag{3}$$
$$E_{ns} = y_{t-1} \tag{4}$$
$$V_{ns} = \sigma^2_{\epsilon_t} \tag{5}$$

Finally, [34] models the probability density function of y_t in Eq. (6) as a mixture of two normal distributions that belong to stationary and non-stationary cases. The model in [34] is in fact more general and includes estimation of all of the future values starting from the latest measurement(y_{t_0}). In this paper, on the other hand, we only require the pdf of y_{t_0+1}. Therefore, Eqs. (2), (3), (4) and (5) are provided for particular case of $t = t_0 + 1$ where t_0 is the time instance of latest available sample.

$$f^{mix}_{y_{t_0+1}}(y) = r_{t_0+1} f^{non-st}_{y_{t_0+1}}(y) + (1 - r_{t_0+1}) f^{st}_{y_{t_0+1}}(y) \tag{6}$$

To generate TCP throughput samples using MATLAB from the explained model, 30 normal random numbers are generated within desired mean and variance intervals. These random numbers work as initiator of the time series. Unit root test result, estimations of c, ϕ and ϵ, and pdf of y_{t_0+1} are calculated. The sample of the next time instance is generated randomly from the conditional pdf of y_{t_0+1} given $0 < y_{t_0+1} < 3\mu_d$ in Eq. (7) where μ_d stands for desired throughput mean. The remaining samples of the time series are generated similarly after updating latest 30 available samples.

$$f^{final}_{y_{t_0+1}}(y) = \frac{f^{mix}_{y_{t_0+1}}(y)}{P(0 < y < 3\mu_d)}, \quad \text{where} \quad 0 < y < 3\mu_d \tag{7}$$

After eliminating the first 30 initiator samples, throughput waveforms with the length of 180 samples are generated. The generation is performed iteratively until all the desired feature slots given in Table 3 are filled with ten waveforms. If the features of a generated waveform is not inside the desired threshold or the corresponding slot is filled, the waveform is discarded. As a result, 1000 time series are acquired with homogeneous distribution within desired network feature ranges to represent TCP throughput.

2.3 Stream simulator

The stream simulator takes each generated sample throughput waveform as input and simulates the behavior of streaming process. Simulator is called after the layer and segment decision processes to calculate the video buffer time and playback position. It informs the adaptation logic about the streaming process so that the adaptation logic can determine which video chunk to stream in the next cycle. It also records the SSIM waveform and total playback interrupt time to evaluate the performance of adaptation algorithm. The details of the stream simulator are described in Algorithm 1.

```
$B \longleftarrow cumulativeSum(b)$;
for each adaptation cycle do
    Determine $l, s$ in adaptation logic
    $S \longleftarrow S + filesize_{l,s}$;
    if $l = 0$ then
        $t_{buf} \longleftarrow t_{buf} + t_{seg}$;
    end
    // Updating $t_{play}$, $t_{act}$ and $t_{buf}$
    for $i \leftarrow i_{last}$ to $N_B$ do
        if $B_i > S$ then
            $i = i - 1$;
            break;
        end
    end
    // Interpolation to get finer details
    $inc \longleftarrow i + (S - B_i)/(B_{i+1} - B_i) - t_{act}$;
    $t_{act} \longleftarrow t_{act} + inc$;
    $i_{last} \longleftarrow roundDown(t_{act})$;
    if playing then
        $t_{play} = t_{play} + inc$;
        $t_{buf} = t_{buf} - inc$;
        if $t_{buf} < 0$ then
            $t_{play} \longleftarrow t_{play} + t_{buf}$;
            $t_{buf} \longleftarrow 0$;
            // Notify Playback Interrupt
        end
    end
end
```

Algorithm 1: Stream Simulator

[1] In the rest of the paper, we describe the process with a window length of 30 samples which is the same value used in [34].

In Algorithm 1, storage variables b, B, S, correspond to throughput, cumulative throughput and downloaded cumulative video chunk size, respectively. S and B are comparable since b is generated in Sect. 2.2 with sampling period of 1 s. Hence, b and B may have units of both kbit/s and kbit. Adaptation variables l and s stand for layer and segment. Time variables t_{act}, t_{play} and t_{buf} correspond to current time in throughput graph, playback position and current buffer length, respectively. When the adaptation algorithm make the decision of video chunk to be streamed, the S is updated as the first step. If the video chunk is from the base layer, t_{buf} is increased by segment duration (t_{seg}). Then, the algorithm searches the point where downloaded cumulative video size just exceeds the cumulative throughput. The search is performed starting from the last search output (i_{last}) to the length of B (N_B). Since the cumulative waveform is monotonically increasing, and the search is usually resulted in the first a few trials, linear search is adopted. The index of B that is returned by the search roughly gives the time passed since the start of the video stream (t_{act}). In order to get more detailed time information, interpolation is applied. Afterward, t_{play} and t_{buf} are updated if the initial buffering phase ends and video is not paused by the user. In case of any stalling, the t_{play} and t_{buf} is corrected and playback interrupt-related information is recorded in the corresponding variables. In a broader sense, the designed simulation algorithm consists of two modes as described in Fig. 1. In single adaptation mode, a selected algorithm is assessed with respect to various datasets or throughput properties directly as described in Algorithm 1. The researchers may use this mode to observe the weak aspects of their algorithm and adjust its streaming parameters accordingly. Additionally, multiple adaptation mode can be used to sequentially simulate selected number of algorithms. However, additional fairness-related adjustment should be considered as a crucial step which is widely disregarded in the previous researches.

Regarding the fairness of the simulation, setting the final evaluated segment for compared DASH implementations should be one of the major concerns. When the test concluded as the throughput waveform reaches to final time sample, compared algorithms may naturally have different buffer length and playback positions. Since SVC-based adaptations have opportunity to upgrade the segments inside buffer, this part should not be included in the evaluation. However, difference in playback position still poses a problem. In this case, some of the segments may be included in one algorithm, while the same segments may be omitted or not streamed in the other one. Therefore, the minimum of playback segments among the compared algorithms can be chosen for latest segment to include in the evaluation. Final fairness problem in this case is the size of the omitted segments. They may as well differ between the tested algorithms. To solve this problem, a constant of one of the algorithms should be adjusted such that average[2] omitted video chunk size should be equal among the tested algorithms. The constants to adjust can be determined from the desired buffer size-related parameters, since video chunks to omit are selected from the buffer. In this sense, a PID controller is added at the output of the simulation. If the difference between the omitted chunk sizes of the adaptations is above a specified threshold, the PID controller minimizes it by adjusting the selected parameter.

Compared to more general network emulators such as [36], the proposed simulator is specifically designed for SVC DASH implementations, and it is very easy to implement new adaptations algorithms. Briefly, the new adaptation loop can be added as a function that takes single object as parameter. This object informs the adaptation loop about the parameters regarding DASH dataset and the streaming process such as current buffer level, playback position, video chunk sizes, segment and layer numbers.

3 Test results

Evaluation is performed using comparison of DASH implementations with a variety of QoE parameters. Also, the comparison of VBR and CBR datasets with the same QoE parameters are provided in the supplementary material. These parameters are average SSIM of the streamed video chunks, average variance, average interrupt time and low buffer experience time. Low buffer experience time consists of five counters. The low buffer term is defined as five time regions ([0, 1], [1, 2]..[4, 5] seconds), and whenever the buffer stays in one of these regions during the streaming, the corresponding counter is incremented. The user selects evaluation parameter as well as the throughput parameters which the output are illustrated for.

In this part, our previous study on SVC DASH implementation[3] [27], Video Quality Adaptation Framework (VQAF) [37] and a straightforward threshold-based algorithm are simulated in multiple adaptation mode. To summarize the approaches in these adaptations, sDASH selects the desired buffer length as a function of instantaneous buffered video quality and has a novel segment prioritization method. In VQAF, enhancement level to download is adjusted considering both measured throughput and current buffer length. They try to smooth the quality of previously requested segments if their buffer reaches a certain threshold. The threshold-based algorithm streams from the base layer if the buffer threshold is not met and it starts streaming enhancement layers in the opposite condition. According to the QoE and through-

[2] The average omitted video chunk size of corresponding adaptation algorithm for all throughput waveforms.

[3] sDASH term will be used to refer this implementation in the rest of the paper.

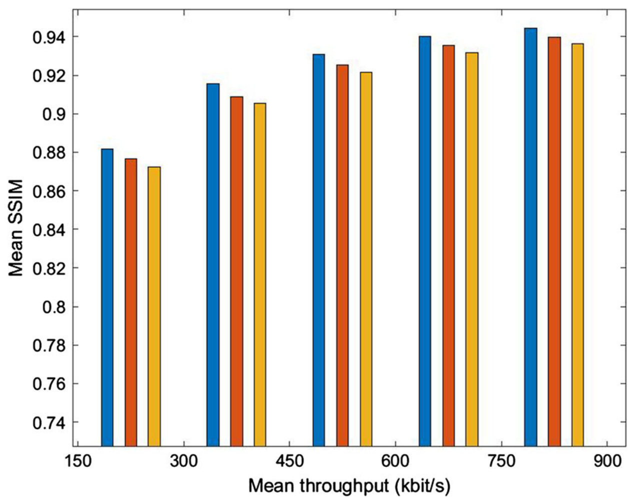

Fig. 2 Mean SSIM with respect to mean throughput where blue represents sDASH orange represent VQAF and red represents threshold-based implementation

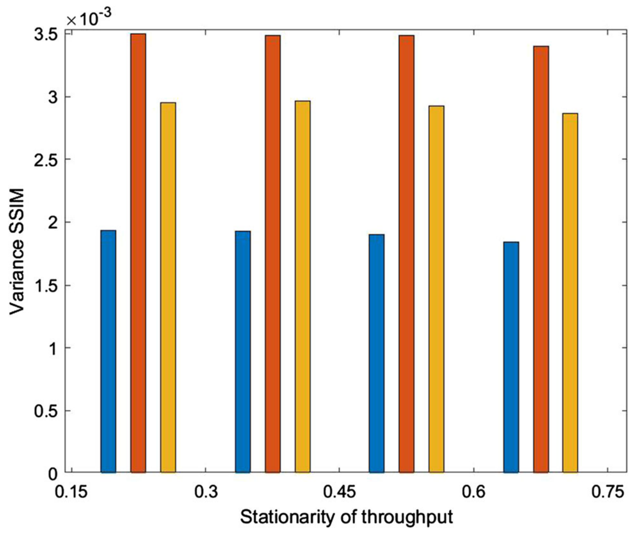

Fig. 3 Variance SSIM with respect to stationarity of throughput with previous color indication

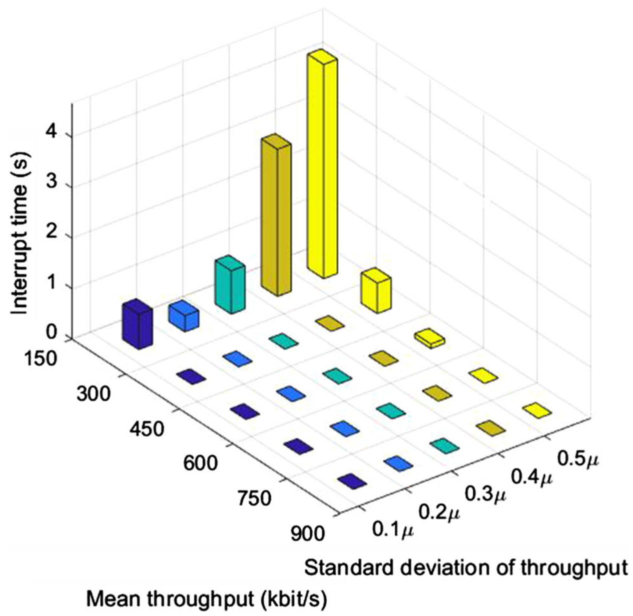

Fig. 4 Average stalling time versus mean and standard deviation of throughput for sDASH

put parameter selection of the user, nine output graphs can be acquired in multiple adaptation modes for single stream. Additionally, the graphs can be created with respect to one or two different throughput parameters in single adaptation mode which results in 12 additional graphs including low buffer detection. Due to space constraints, two results from multiple and one result from single adaptation modes for Big Buck Bunny in VBR mode are given in Figs. 2, 3 and 4, respectively. Figures 2 and 3 indicate higher average quality and lower quality switching of sDASH compared to other two implementations. As threshold-based algorithm only focuses on maximizing quality, it outperforms VQAF in terms of average SSIM whereas VQAF has smoother quality variations compared to threshold-based method. Figure 4 illustrates the range of standard deviation and mean throughput values which results in stalling.

Simulation graphs can also be acquired by averaging the simulation results of entire designed dataset which correspond to 6000 streaming measurement (1000 waveforms, 3 videos and 2 modes) for each implementation. Total run time depends on numerous factors such as initial input of the PID controller and selected acceptable omitted chunk size difference. However, computation time of 30 min is achieved with the default settings in a system with 2.8 GHz, 4 core processor and 8 GB of RAM. If the real-time evaluation were applied instead, this comprehensive test would last around 900 hours (6000 streaming measurements for each of the three adaptation algorithms with a streaming duration of 180 s) without the PID controller. The time efficiency of the proposed method is favorable for tuning adaptation in its design process as well as comparing with other implementations in the evaluation phase.

As the final part of this section, the accuracy of simulator is compared to that of real-time assessment. To this end, a netem emulated client–server setup is constructed. A reference adaptation algorithm (sDASH) is tested for both systems with identical throughput traces. Firstly, the performance of sDASH is computed for both systems in terms of average SSIM and variance of SSIM. Then, mean absolute percentage error (MAPE) for both QoE parameters are calculated by taking the real-time assessment results as reference. MAPE of average SSIM is obtained as 0.17%, while MAPE of variance of SSIM is obtained as 3.95%. As described in Sect. 1, real-time assessment itself does not produce ideal results due to imperfect control of emulator and synchronization issues. To measure the reliability of real-time assessment, the same waveforms are tested multiple times for the same adaptation algorithm. In this case, relative standard deviation (RSD) of average SSIM and variance of SSIM are calculated as 0.128% and 3.97%, respectively. As a result, the proposed system achieves much faster computation accuracy with a decent reliability compared to real-time assessment.

4 Conclusion

The proposed work contains an SVC dataset, throughput waveform generator and adaptive streaming simulator, which can be used both individually and integrally. Researchers may adjust throughput characteristics and choose datasets to test implementations according to a desired scenario. The simulation results of three implementations are provided so that the researchers can also compare their work directly. The proposed methods can easily be adapted to non-scalable encoding styles.

Funding The online version supplementary material available at https://doi.org/10.1007/s11760-021-01880-y.

References

1. Cisco, Cisco Visual Networking Index (VNI) Complete Forecast Update, 2017–2022 (2018)
2. Stockhammer, T.: Dynamic adaptive streaming over HTTP: standards and design principles. In: Proceedings of the Second Annual ACM Conference on Multimedia Systems, pp. 133–144 (2016)
3. Ayad, I., Im, Y., Keller, E., Ha, S.: A practical evaluation of rate adaptation algorithms in http-based adaptive streaming. Comput. Netw. **133**, 90–103 (2018)
4. Zabrovskiy, A., Petrov, E., Kuzmin, E., and Timmerer, C.: Evaluation of the performance of adaptive HTTP streaming systems. Preprint at arXiv:1710.02459 (2017)
5. Vergados, D.J., Kralevska, K., Michalas, A., Vergados, D.D.: Evaluation of HTTP/DASH adaptation algorithms on vehicular networks. In: 2018 Global Information Infrastructure and Networking Symposium (GIIS), pp. 1–5. IEEE (2018)
6. Vergados, D.J., Michalas, A., Sgora, A., Vergados, D.D., Chatzimisios, P.: FDASH: a fuzzy-based MPEG/DASH adaptation algorithm. IEEE Syst. J. **10**(2), 859–868 (2015)
7. Schwarz, H., Marpe, D., Wiegand, T.: Overview of the scalable video coding extension of the H. 264/AVC standard. IEEE Trans. Circuits Syst. Video Technol. **17**(9), 1103–1120 (2007)
8. Zhao, M., Gong, X., Liang, J., Wang, W., Que, X., Cheng, S.: QoE-driven cross-layer optimization for wireless dynamic adaptive streaming of scalable videos over HTTP. IEEE Trans. Circuits Syst. Video Technol. **25**(3), 451–465 (2015)
9. Kalva, H., Adzic, V., Furht, B.: Comparing MPEG AVC and SVC for adaptive HTTP streaming. In: 2012 IEEE International Conference on Consumer Electronics (ICCE), pp. 158–159. IEEE (2012)
10. Lederer, S., Müller, C., Timmerer, C.: Dynamic adaptive streaming over HTTP dataset. In: Proceedings of the 3rd Multimedia Systems Conference, pp. 89–94 (2012)
11. Le Feuvre, J., Thiesse, J. M., Parmentier, M., Raulet, M., Daguet, C.: Ultra high definition HEVC DASH data set. In: Proceedings of the 5th ACM Multimedia Systems Conference, pp. 7–12 (2014)
12. Quinlan, J.J., Zahran, A.H., Sreenan, C.J.: Datasets for AVC (H. 264) and HEVC (H. 265) evaluation of dynamic adaptive streaming over HTTP (DASH). In: Proceedings of the 7th International Conference on Multimedia System, pp. 1–6 (2016)
13. Zabrovskiy, A., Feldmann, C., Timmerer, C.: Multi-codec DASH dataset. In: Proceedings of the 9th ACM Multimedia Systems Conference, pp. 438–443 (2018)
14. Kreuzberger, C., Posch, D., Hellwagner, H.: A scalable video coding dataset and toolchain for dynamic adaptive streaming over HTTP. In: Proceedings of the 6th ACM Multimedia Systems Conference, pp. 213–218. ACM (2015)
15. Wang, Z., Bovik, A.C., Sheikh, H.R., Simoncelli, E.P.: Image quality assessment: from error visibility to structural similarity. IEEE Trans. Image Process. **13**(4), 600–612 (2004)
16. Xiong, P., Shen, J., Wang, Q., Jayasinghe, D., Li, J., Pu, C.: NBS: a network-bandwidth-aware streaming version switcher for mobile streaming applications under fuzzy logic control. In: 2012 IEEE First International Conference on Mobile Services, pp. 48–55. IEEE (2012)
17. Zhou, C., Lin, C.-W., Guo, Z.: mDASH: a Markov decision-based rate adaptation approach for dynamic HTTP streaming. IEEE Trans. Multimedia **18**(4), 738–751 (2016)
18. Mori, S., Bandai, M.: QoE-aware quality selection method for adaptive video streaming with scalable video coding. In: 2018 IEEE International Conference on Consumer Electronics (ICCE), pp. 1–4. IEEE (2018)
19. Sieber, C., Hoßfeld, T., Zinner, T., Tran-Gia, P., Timmerer, C.: Implementation and user-centric comparison of a novel adaptation logic for DASH with SVC. In: 2013 IFIP/IEEE International Symposium on Integrated Network Management (IM 2013), pp. 1318–1323. IEEE (2013)
20. Spiteri, K., Urgaonkar, R., Sitaraman, R.K.: BOLA: near-optimal bitrate adaptation for online videos. In IEEE INFOCOM 2016-The 35th Annual IEEE International Conference on Computer Communications, pp. 1–9. IEEE (2016)
21. Van der Hooft, J., Petrangeli, S., Bouten, N., Wauters, T., Huysegems, R., Bostoen, T., De Turck, F.: An HTTP/2 push-based approach for SVC adaptive streaming. In: NOMS 2016-2016 IEEE/IFIP Network Operations and Management Symposium, pp. 104–111. IEEE (2016)
22. Riiser, H., Endestad, T., Vigmostad, P., Griwodz, C., Halvorsen, P.: Video streaming using a location-based bandwidth-lookup service for bitrate planning. ACM Trans. Multimed. Comput. Commun. Appl. (TOMM) **8**(3), 1–19 (2012)
23. Riiser, H., Vigmostad, P., Griwodz, C., Halvorsen, P.: Commute path bandwidth traces from 3G networks: analysis and applications. In: Proceedings of the 4th ACM Multimedia Systems Conference, pp. 114–118 (2013)
24. Raca, D., Quinlan, J.J., Zahran, A.H., Sreenan, C.J.: Beyond throughput: a 4G LTE dataset with channel and context metrics. In: Proceedings of the 9th ACM Multimedia Systems Conference, pp. 460–465 (2018)
25. Timmerer, C., Zabrovskiy, A., Begen, A.: Automated objective and subjective evaluation of HTTP adaptive streaming systems. In: 2018 IEEE Conference on Multimedia Information Processing and Retrieval (MIPR), pp. 362–367. IEEE (2018)
26. Link to be supplied
27. Çalı, M., and Ozbek, N.: Dash-Simulator. GitHub repository, https://github.com/mehmet-cali/dash-simulator (2021)

28. Reichel, J., Schwarz, H., Wien, M.: Joint scalable video model 11 (JSVM 11). Joint Video Team, pp. 23 (2007)
29. Blender Foundation: Big Buck Bunny. https://peach.blender.org/. Accessed 29 April 2020
30. Blender Foundation: Sintel, the Durian Open Movie Project. https://durian.blender.org/. Accessed 29 April 2020
31. Blender Foundation: Tears of Steel. https://mango.blender.org/. Accessed 29 April 2020
32. Flynn, J. R., Ward, S., Abich, J., Poole, D.: Image quality assessment using the ssim and the just noticeable difference paradigm. In: International Conference on Engineering Psychology and Cognitive Ergonomics, pp. 23–30. Springer, Berlin (2013)
33. Mirza, M., Sommers, J., Barford, P., Zhu, X.: A machine learning approach to TCP throughput prediction. ACM SIGMETRICS Perform. Eval. Rev. **35**(1), 97–108 (2007)
34. Yoshida, H., Satoda, K., Murase, T.: Constructing stochastic model of TCP throughput on basis of stationarity analysis. In: 2013 IEEE Global Communications Conference (GLOBECOM), pp. 1544–1550. IEEE (2013)
35. Dickey, D.A., Fuller, W.A.: Distribution of the estimators for autoregressive time series with a unit root. J. Am. Stat. Assoc. **74**(366a), 427–431 (1979)
36. Netravali, R., Sivaraman, A., Das, S., Goyal, A., Winstein, K., Mickens, J., Balakrishnan, H.: Mahimahi: accurate record-and-replay for HTTP. In: 2015 USENIX Annual Technical Conference (USENIXATC 15), pp. 417–429 (2015)
37. Lekharu, A., Kumar, S., Sur, A., Sarkar, A.: A QoE aware SVC based client-side video adaptation algorithm for cellular networks. In: Proceedings of the 19th International Conference on Distributed Computing and Networking, pp. 1–4 (2018)