Variable-Threshold Buffer Based Adaptation for DASH Mobile Video Streaming

Rabee Mustapha Abuteir, Anne Fladenmuller, Olivier Fourmaux, Mostafa Ammar

Variable-Threshold Buffer Based Adaptation for DASH Mobile Video Streaming

Rabee Mustapha Abuteir *, Anne Fladenmuller *, Olivier Fourmaux * and Mostafa Ammar †
* Sorbonne Universités, UPMC Univ Paris 06, CNRS, LIP6 UMR 7606, 4 place Jussieu 75005 Paris
{mustapha.abuteir, anne.fladenmuller, olivier.fourmaux}@lip6.fr
† Georgia Institute of Technology
ammar@cc.gatech.edu

***Abstract*—Dynamic Adaptive Streaming over HTTP (DASH) was introduced to enable high video quality streaming over HTTP. DASH depends on the adaptation logic at the client to choose which video bitrate to stream from the content server for each chunk. For clients receiving video over a cellular network, the cellular first hop tends to be the bandwidth bottleneck and can exhibit significant swings in available bandwidth. In this paper we develop and evaluate a Dynamic Adaptation for mobile Video Streaming (DAVS), a technique that can be used within DASH adaptation to handle the significant bandwidth variability experienced by cellular mobile clients. In our scheme the main innovation is that the client chooses a bitrate based on whether the playout buffer occupancy (BO) falls below or above a *dynamic* threshold. In addition, the scheme attempts to minimize bitrate switching by again delaying a change of video bitrate selection by a window of time – that is also dynamically determined. We evaluate the performance of DAVS over real traces collected from a mobile network operator. DAVS shows better performance over different video streaming metrics. Furthermore, It increases the QoE by a range 15% - 55% compared to benchmark algorithms.**

***Index Terms*—DASH, HTTP, LTE, Mobile Networks, TCP, Video Streaming.**

I. Introduction

There is significant growth in mobile data traffic in the last few years. Some predictions expect the mobile data traffic to account for 19% of total Internet traffic by 2020 [1]. It is also expected that video traffic will reach 79% of all Internet traffic by 2020. The bulk of video streaming on the Internet today (including streaming to mobile devices) uses some version of the Dynamic Adaptive Streaming over HTTP (DASH) protocol which aims to deliver video with high Quality of Experience (QoE) [2].

Figure 1 shows the architecture of a DASH video streaming system. The video is available at an Over-The-Top (OTT) video provider and pre-encoded at various bitrates. The resulting videos are segmented into chunks (typically with duration of 2-10 seconds each) and hosted in the OTT content servers (Content Distribution Network). Initially the DASH player receives information about the available bitrates among other video meta-information in a Media Presentation and Description (MPD) file which is downloaded from the content server before video streaming begins. The DASH player requests the video chunks using HTTP protocol which uses TCP as transport layer [3].

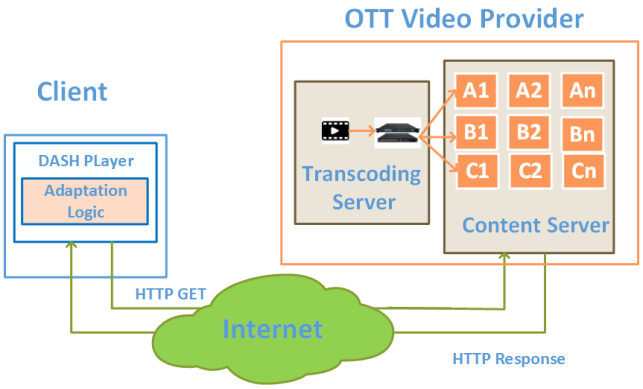

Fig. 1: The End-to-End architecture of DASH video streaming system. The video is pre-encoded at various bitrates and segmented into chunks. The DASH player uses the adaptation logic to select the streamed chunks bitrates.

Adaptive Bit Rate (ABR) adaptation logic is used by the DASH player to choose which bitrate to request from the content server for each chunk. A variety of adaptation schemes have been proposed in the literature and used in commercial video streaming systems. Some of these are described in the related work section. In typical adaptation schemes the decisions are based on measurements of download throughput and/or playout buffer occupancy. The main objective of the adaptation is to maximize user engagement by delivering video with highest QoE [4], [5]. Typically this involves some combination of maximizing the average video bitrate, minimizing video stall (or rebuffering) events and minimizing bitrate switching at the client.

ABR adaptation logic, however, performs best when the bandwidth from the client to the servers [1] is relatively stable [7] and its performance degrades in the presence of significant bandwidth instability [8], [9]. For clients receiving video over a cellular network, the cellular first hop tends to be the bandwidth bottleneck. It can also exhibit significant swings in available bandwidth. Illustrating this behavior is

Mostafa Ammars work is supported in part by the National Science Foundation under Grant Number NETS-1409589.

[1]The chunks of the same video could be distributed over multiple content servers [6].

Figure 2 which shows the throughput observed by a mobile client accessing content server over a 4G LTE network. This significant variability in bandwidth is due to the change in link capacity. In cellular networks, the link capacity depends on many factors, such as channel fading and the bottleneck in the radio access network. In previous work it has been observed that this variability requires changes to traditional Internet congestion control [10].

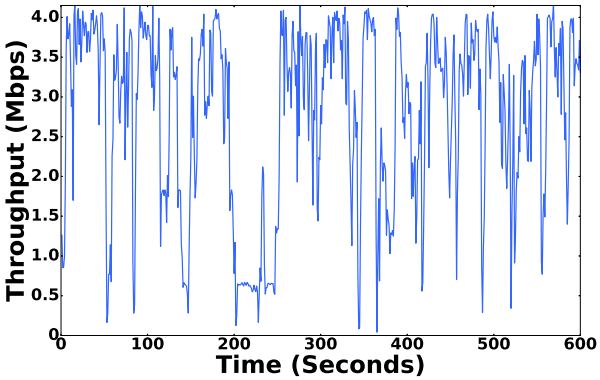

Fig. 2: The variation of throughput observed by a mobile client. Results from an experiment performed over 4G LTE cellular network of one of the main telecommunication provider in France.

In this paper we develop and evaluate a technique that can be used within DASH ABR adaptation logic to handle the significant bandwidth variability experienced by cellular mobile DASH player. In our scheme the adaptation logic chooses a bitrate based on whether the playout buffer occupancy (BO), measured in seconds of video in the buffer, falls below or above a *dynamic* threshold. The main innovation in the algorithm is the use of a dynamic threshold that enables the adaptation to deal with highly variable bandwidth profiles. In addition, the scheme attempts to minimize bitrate switching by again delaying a change of video bitrate selection by a window of time – that is also dynamically determined.

The rest of the paper is organized as follows. Related work is presented in Section II. In Section III we describe and motivate the proposed adaptation algorithm. The adaptation scheme performance is evaluated in section IV. The paper is concluded in Section V.

II. Related Work

The state of art ABR adaptation logic algorithms can be classified into three categories: (1) **Bandwidth estimation based:** the adaptation logic uses the bandwidth estimation to select the next chunk bitrate. (2) **Buffer occupancy based:** the adaptation logic models the next chunk bitrate as a function of the current buffer occupancy (3) **Hybrid:** the adaptation logic uses a mix of both bandwidth estimation and buffer occupancy techniques.

FESTIVE [11] uses an adaptation logic aims to improve fairness, stability and efficiency of the DASH player. FESTIVE uses a harmonic mean, chunk scheduler and delay update to select the next chunk bitrate. However, FESTIVE does not take into account buffer occupancy which can lead to buffer starvation (rebuffering) in mobile networks where the bandwidth is highly variable and unpredictable.

In [12], the authors proposed a Buffer Based Adaptation (BBA) scheme. In BBA, the next chunk bitrate is a function of the buffer occupancy. At the beginning of a streaming session BBA chooses the lowest bitrate to retrieve from the content server. As the buffer occupancy increases the player starts to retrieve chunks at higher bitrates. The authors developed four buffer-aware ABR algorithms namley BBA-0, BBA-1, BBA-2 and BBA-O. BBA-2 uses the buffer occupancy mapping in addition to stream at higher bitrates at the beginning of the streaming session.

Zahran et al. [13] evaluate the performance of different adaptation logics with different schedulers over cellular LTE networks. The results show that proportional fairness is the most suitable scheduler for video streaming in LTE networks. Furthermore, the results show that the BBA-2 algorithm is more suitable for cellular networks compared to FESTIVE. In their evaluation they found that FESTIVE suffers from high instability and rebuffering.

Zou et al. [14] asked the question: *"Can Accurate Predictions Improve Video Streaming in Cellular Networks?"*. The authors assumed the bandwidth for a few seconds in the future is well known. The results show the prediction for short time horizon only is insufficient. To the best of the authors knowledge, there is no available accurate long term prediction model for last mile mobile access network.

The Open Source Media Framework (OSMF) [15] is an HTTP video streaming platform developed by Adobe Systems. OSMF adaption logic uses the chunk duration and download time as metrics to select the next chunk bitrate. In mobile networks the chunk download time can vary considerably due to the variation in network bandwidth which will lead to high player instability.

Parikshit et al. [16], [17] propose Segment Aware Rate Adaptation (SARA) algorithm. The authors notice the variation in chunks size with the same bitrate encoding due to the change in the content complexity and motion. They propose to modify the Media Presentation and Description (MPD) file to include the chunks sizes. This technique was proposed also by the authors of buffer aware adaptation [12]. SARA uses the weighted harmonic mean to estimate the throughput. Further, it selects the next chunk bitrate based on the buffer occupancy and a set of static thresholds. SARA divides the buffer into four regions and based on the current buffer occupancy and the bandwidth weighted harmonic mean it selects the next chunk bitrate. The threshold values depend on the buffer sizes and that leads to different behavior between players with different buffer sizes. Further, SARA shows high player instability due to the high number of switches which impacts the user QoE[18].

Post Streaming Quality Analysis (PSQA) is a unified framework proposed to improve the QoE of DASH video streaming

over mobile networks [19]. PSQA uses machine learning to generate the best adaptation logic parameters from past throughput traces training. In cellular network, users have different bandwidth fingerprints with high variability. A unified framework for video streaming with the same parameters for different users can potentially lead to poor performance and impacts the users QoE.

A flow level framework for adaptive video delivery over mobile networks is proposed in [20]. The framework is designed to work at the gateway level and shapes the traffic between different mobile clients. The framework was designed to maximize both (resource utilization and fairness) and minimize the DASH player instability. We propose in this work a new adaptation logic that takes into account the variation in link quality independently of the DASH player buffer size. In addition, our proposed adaptation logic can work with any other framework such as that proposed in [20] to maximize the overall performance.

III. Proposed Adaptation Logic

In this work we propose a new adaptation logic called Dynamic Adaptation for mobile Video Streaming (DAVS). DAVS specifically designed to handle the variability and unpredictability of bandwidth in the cellular networks. As mentioned earlier, DAVS is based on the use of two components (1) dynamic buffer threshold that enables the adaptation to deal with highly variable bandwidth profiles (2) dynamic delay updates that help minimize bitrate switching. We describe these two components next.

A. Dynamic Buffer-Threshold Adaptation

State of the art adaptation logic came with set of predefined parameters such as thresholds, which divided the buffer into areas and based on the buffer occupancy it guides the adaptation logic to choose the next chunk bitrate. DAVS uses buffer occupancy model like the proposed in [12], where the buffer is divide into two areas (risky and safe) as shown in Figure3. Risky area represents the region if the buffer occupancy is below then rebuffering event is highly possible. On other hand, if the buffer occupancy is enough to enter the save area then rebuffering event is lowly possible because there is enough time to download more chunks. The boundary between the two areas are determined through threshold. In contrast to traditional adaptation logic, where the threshold is predefined parameter, DAVS dynamically determines the threshold based on the variation in bandwidth available to the mobile client. In DAVS the threshold is determined dynamically based on the variation in chunk download time. When the chunk downloads time increases that lead to decrease the risky area and vice versa. Next, DAVS selects the next chunk bitrate based on the new threshold value and the Buffer Occupancy (BO). As a reminder BO and Th are measured in seconds of video. Specifically, BO is the number of chunks in the buffer multiplied by the chunks duration [2].

[2]OTT service provider use fixed chunk duration, for example Netflix uses 4 seconds chunk duration [21].

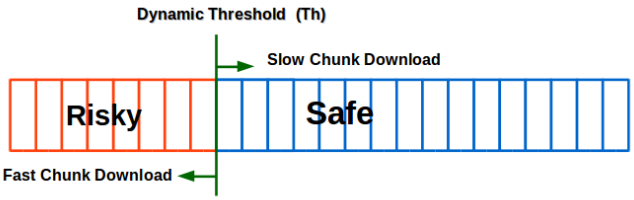

Fig. 3: Dynamic threshold for mobile video streaming . The threshold separates between risky and safe area. In DAVS the threshold is dynamically determined based on the variation in chunk download time.

When a video chunk is streamed over mobile network, it takes time to be delivered to the client. The Download Duration of Last streamed Chunk (DDLC) and the Chunk Size (CS) are used to calculate the last streamed chunk Estimated BandWidth (EBW) as shown below:

$$EBW = \frac{CS}{DDLC} \tag{1}$$

As shown in Figure 2, in mobile networks the bandwidth changes over time which causes variation in DDLC. Consequently, the use of single bandwidth estimation technique will increase the player instability and could lead to buffer starvation. To overcome this problem we use another bandwidth estimation namely, Long term average BandWidth (LBW) and defined as :

$$LBW = \sum_{n=1}^{N} \frac{EBW_n}{N} \tag{2}$$

Where N is the number of downloaded chunks and EBW_n is the estimated bandwidth for chunk n and calculated using equation 1.

DAVS adaptation logic takes as inputs the video Available Bitrates (ABitrates)[3], Buffer Occupancy (BO), DDLC, EBW, Th, Last streamed chunk bitRate LR, smoothing factor α and W the delay update Window. The DAVS adaptation logic pseudo code is shown in algorithm 1.

At the beginning, the algorithm evaluates the value of R1, which is the highest bitrate from the available bitrates lower than the EBW. R2 is the highest bitrate from the available bitrates lower than the LBW.

We use two different bandwidths (short term bandwidth EBW and long term average bandwidth LBW) because in mobile networks the bandwidth is unstable and the use of EBW only can lead to excessive bitrate switching and impacts the user QoE.

In Line 3, R0 represents the minimum bitrate in ABitrates. We use a dynamic buffer model with dynamic threshold as shown in Figure 3. The threshold (Th) changes dynamically based on the download duration of last chunk which depends

[3]This is the list of video bitrates available to stream from the server and is embedded in the MPD file.

Algorithm 1: Dynamic Adaptation for mobile Video Streaming (DAVS)

Input : ABitrates, BO, DDLC, EBW, LBW, Th, LR and α,W
Output: Next chunk bitrate

1 R1 = $\max(\{bitrate \in ABitrates \mid bitrate \leq EBW\})$;
/* R1 is the maximum bitrate in ABitrates and less than EBW */
2 R2 = $\max(\{bitrate \in ABitrates \mid bitrate \leq LBW\})$;
/* R2 is the maximum bitrate in ABitrates and less than LBW */
3 R0 = the minimum available bitrate;
/* R0 is the minimum bitrate in ABitrates */
4 LastTh = Th;
5 Th = $\alpha \times LastTh + (1-\alpha) \times DDLC$;
/* Dynamic threshold based on last threshold and and download duration of the last chunk */
6 **if** $BO < Th$ **then**
// DAVS in Risky Area
7 **if** *the cause of enter in the risky area is due to last increase in the streamed bitrate* **then**
/* Double the window size to reduce the risk in future */
8 $W.setSize(W.getSize() \times 2)$;
9 **if** $DDLC > Th$ **then**
/* Choose the minimum bitrate in ABitrates for next chunk */
10 **return** $R0$;
11 **else**
/* Choose the minimum bitrate between R01,R2 and LR */
12 **return** $min(LR, min(R1, R2))$;
13 **else**
// DAVS in Safe Area
14 R= $max(LR, max(R1, R2))$;
/* DelayUpdate will use to minimize the player instability */
15 **return** $DelayUpdate(LR, R)$;

on the available bandwidth. Lines 4-5 shows the smoothing equation that is used to dynamically update the threshold based on last threshold and DDLC. α is a smooth factor and could be included in the MPD file or pre-configured in the player.

DAVS uses the BO and Th to decide if it's in the risky area when the current BO is less than Th, otherwise the player buffer is in the safe area. Lines 6-13 represent the DAVS behavior when the BO is in the risky area. Lines 7-8 show the player behavior when it enters the risky area due to the last selected bitrate. In this case, DAVS increases the stability by doubling the DelayUpdate window length to make sure the decision next time will not cause the same problem (We will discuss the stability function in more details in the next subsection). Lines 9-10 show DAVS behavior when there is a high possibility for rebuffering. To overcome this it streams the next chunk at the lowest bitrate R0. Lines 11-12 show DAVS behaviour when the download duration of last chunk is greater than the threshold, in which case, it will request the next chunk with the minimum bitrate from R1,R2 and LR to avoid rebuffering. Lines 13-16 show the behaviour of DAVS when the BO is larger than Th, i.e., when the buffer occupancy is in the safe area. In this case the DASH player should stream at higher bitrates. Because the bandwidth can vary significantly, DAVS uses a DelayUpdate method to increase the player stability. We will discuss DelayUpdate function in details in the next subsection.

B. Improving Bitrate Stability

Algorithm 2: DelayUpdate

Input : LR, R
Output: bitrate

1 W.append(R);
/* Add the bitrate to the DelayUpdate window */
2 **if** *W.isFull* **then**
// DelayUpdate window is full
3 R3 = minimumValueIn(W) ;
// Return the minimum value in DelayUpdate window
4 W.clear();
// Clear the content of DelayUpdate window
5 **return** $R3$;
6 **else**
/* DelayUpdate window is not full, delay the bitrate update and return the last streamed chunk bitrate */
7 **return** LR;

We developed a new instability function namely, DelayUpdate inspired from two techniques (1) window function which used in signal processing [22] and (2) exponential growth backoff which widely used in wireless networks [23]. The objective from the stability function is to prevent excessive video bitrate switching and to avoid the buffer starvation which significantly influence user QoE. The DelayUpdate function will delay the peak increase in bitrate until it make sure its decision to increase is the right decision. Algorithm 2 shows the DelayUpdate function, where W is a list used to store the bitrates that are chosen by DAVS. Each new update will be appended to W and delayed until W is full. The DASH player will choose the lowest value in W as the bitrate for the next chunk to be streamed. On the other hand, if W is not full the next bitrate will be the same as the last streamed chunk bitrate (LR). Consequently, this technique delays the bitrate change

to avoid player instability and increases the user QoE. When the bitrate increasing leads to enter into the risky area , DAVS will double the W size as shown in algorithm 1 - lines (7-8).

IV. Performance Evaluation

In this section, first we will discuss the evaluation framework. Then, we will discuss the performance metrics that used to evaluate DAVS. Next, we will discuss the results. Finaly, we will discuss the QoE utility function.

A. Evaluation Framework

We collected a set contains real traffic traces (30 traces) reported bandwidth every second over mobile networks. The traces were collected in the city of Paris over one of the main 4G LTE provider. The traces were collected in different locations and at different times to make sure they represent the heterogeneity in traffic patterns experienced over a cellular network. In addition, we encoded an open-source film (Big Buck Bunny [24]) which widely used in streaming algorithms with similar bitrates and resolutions used by Netflix [25] using FFmpeg H.264/MPEG-4 AVC codec [26]. We segmented the videos into chunks using GPAC open-source multimedia framework [27]. We choose the chunk duration to be 4 seconds because it is the same as the used by Netflix [21]. We built a trace-driven simulation to evaluate the performance of DAVS. We streamed the encoded chunks over the collected traces and compare the performance of DAVS and four benchmark algorithms (OSMF [15], BBA2 [12], SARA [16] and PSQA [19]).

B. Performance Metrics

We consider three video streaming performance metrics widely used in the literature to evaluate the streaming performance of DASH streaming and reflect the user QoE.

1) Average BitRate (BR): There is a non-linear relation between the video quality and its bitrate [28]. In general, the video quality metrics such as the Structural SIMilarity (SSIM) increases as the bitrate increases [29], [30]. The Average Bitrate (BR) is defined as:

$$\text{BR} = \frac{1}{N}\sum_{n=1}^{N} R_n \tag{3}$$

Where N is the number of bitrates used during streaming session, R_n is the video bitrate streamed by the client at n. High BR value means better video quality and vice versa.

2) Instability: The QoE of DASH video streaming is impacted by the bitrate switches [31]. The instability of the player is defined as the number of switches between bitrates over the streaming session. Less bitrates switches reflect better QoE and vice versa.

3) Rebuffering Duration (RD): Rebuffering Duration (RD) is defined as the duration of buffer starvation during a video streaming session. RD is one of the most important metrics for video streaming that affects the user QoE [5]. Less rebuffering duration means better QoE and vice versa.

C. Results

We use two different buffer sizes 240 and 120 seconds. We evaluate the performance of the algorithms using our evaluation framework over the collected traces. We compare the performance of the algorithms using the metrics discussed in subsection IV-B. We calculate the average value for each metric over all collected traces and different buffer sizes.

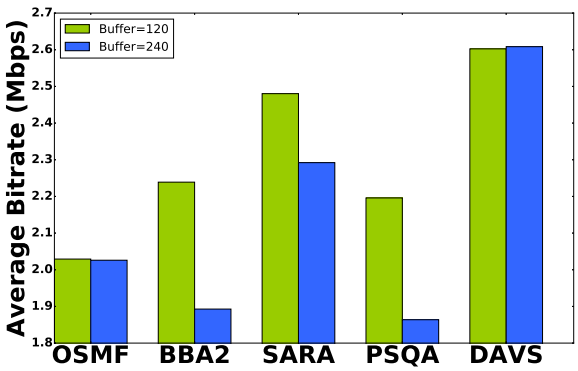

Fig. 4: Average Bitrate for different algorithms and different buffer sizes. DAVS shows consistently high average bitrate for different buffer sizes compared to the other benchmark algorithms.

Figure 4 shows the average bitrate for different buffer sizes. Both algorithms OSMF and DAVS are buffer-size independent algorithms. As shown in the Figure OSMF streams at lower bitrates compared to other algorithms for small buffer size. Contrarily, SARA streams at higher bitrates but the behavior of SARA is buffer size dependent. In addition, SARA is more suitable for small buffer size but for larger buffer it streams at lower bitrates. BBA2 adaptation logic selects the bitrate based on the buffer occupancy. Therefore, has different average bitrates. Likewise, PSQA uses the buffer occupancy as one of the metrics to select the next chunk bitrate. Consequently, BBA2 and PSQA stream at higher bitrate for small buffer size over mobile networks. Finally, our proposed algorithm DAVS streams at higher bitrate compared to the benchmark algorithms. It shows consistent behavior for small and large buffer sizes because it depends on the traffic behavior rather the buffer sizes.

Figure 5 shows the average instability for different algorithms and different buffer sizes. As it clear from the Figure, OSMF has the highest instability (on average 97.43 switches) because it chooses the next bitrate based on chunk download duration. In mobile networks, the chunks download durations varies due the high variation in bandwidth. In addition, PSQA and BBA2 show better average instability compare to SARA. BBA2 shows higher stability for larger buffer size due to its long delay before switching to a new bitrate. Further, PSQA shows a low number of switches (on average 8.56 switches) due to its attempt to fix the number of switches over a time period to stabilize the DASH player. Finaly, DAVS

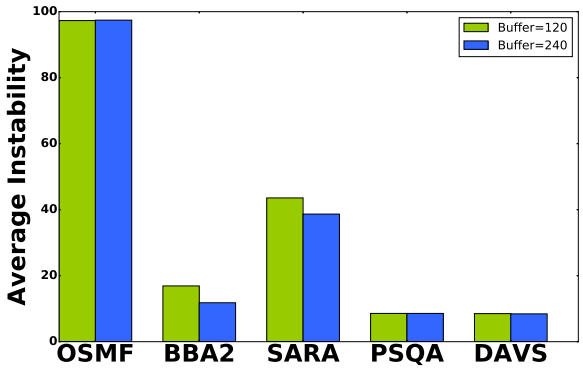

Fig. 5: Average Instability for different algorithms and different buffer sizes. DAVS shows low average instability for different buffer sizes compared to the other benchmark algorithms.

uses the DelayUpdate function (discussed in subsection III-B) to enhance the player stability. Consequently, DAVS shows higher stability (on average 8.43 switches) compared to the benchmark algorithms.

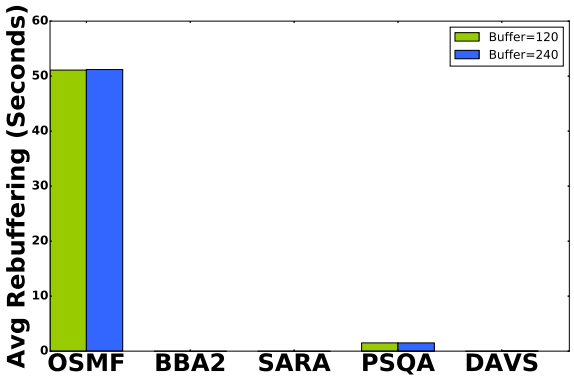

Fig. 6: Average rebuffering duration for different algorithms and different buffer sizes. Both OSMF and PSQA suffer from rebuffering.

Figure 6 shows the average rebuffering duration for different algorithms and different buffer sizes. Both OSMF and PSQA suffer from rebuffering. On the other hand, BBA2 and SARA reduce rebuffering by taking into account the buffer occupancy. As a result, when the buffer is consumed quickly they switch to lower bitrates. DAVS uses dynamic threshold to separate between the risky and safe areas. For slow chunk download duration it increases the risky area and for high chunk download duration it decreases the risky area. This dynamic adaptation of threshold-based on traffic behavior makes DAVS suitable for mobile video streaming.

D. QoE Utility Function (QUF)

There are many QoE utility functions proposed in the literature to reflect the user QoE [32], [33], [34]. In this paper we used the QoE utility function proposed in [35] because it is widely used in literature. Further, it considers different parameters (startup delay, average bitrate, instability and rebuffering duration) and defined as:

$$\text{QoE} = \sum_{k=1}^{K} R_k - \lambda \times \sum_{k=1}^{K-1} |R_{K+1} - R_K| - (\mu \times T_p) - (\mu_s \times T_s) \tag{4}$$

Where K is the number of streamed chunks, R_K is the bitrate that is used to stream chunk k, T_p is the rebuffering duration and T_s is the startup delay. The constants λ, μ and μ_s are weighting factors and the values proposed by the authors are (1, 3000 and 3000) [35]. We evaluate the QoE for DAVS and the benchmark algorithms for different buffer sizes over the evaluation framework introduced in subsection IV-A. First we evaluate the average QoE for every algorithm and for different buffer sizes. Then, we normalize the average QoE using the following equation:

$$\text{NAQoE} = \frac{AAQoE}{MAAQoE} \tag{5}$$

Where NAQoE refers to the Normalized Average QoE, AAQoE refers to the Actual Average QoE and MAAQoE refers to the Maximum Actual Average QoE. Based on Equation 5 we divide the AAQoE for each algorithm over the maximum AAQoE between all algorithms. The algorithm with highest value of NAQoE is the best among the evaluated algorithms.

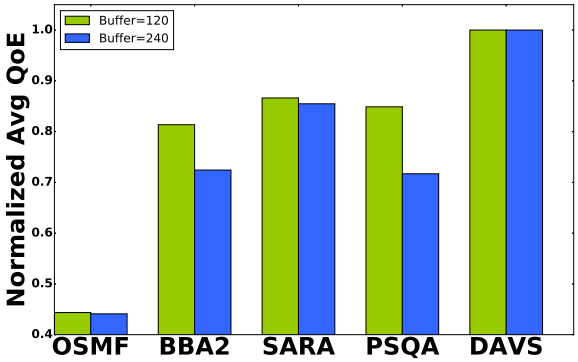

Fig. 7: Normalized Average QoE for different algorithms and different buffer sizes. The highest value means the best between the evaluated algorithm.

Figure 7 shows the Normalized Average QoE for the different algorithms. As is shown in the Figure the $NAQoE$ for BBA2, SARA and PSQA varies for different buffer sizes because the algorithms behavior depend on the static thresholds. On the other hand, OSMF and DAVS behavior is approximately the same for different buffer sizes. DAVS has the highest QoE for different buffer sizes because (1) DAVS behavior depends on the measured bandwidth, (2) DAVS streams at higher bitrates and it takes into account

the time-varying bandwidth behavior in mobile networks, (3) DAVS uses a dynamic threshold to avoid rebuffering, and (4) DAVS uses DelayUpdate stability to reduce the player switches between bitrates. DAVS increases the QoE between 15% to 55% compared to the benchmark algorithms.

V. Conclusion and Future Work

We proposed DAVS a new adaptation logic for DASH mobile video streaming. DAVS uses dynamic buffer occupancy threshold to deal with highly variable bandwidth profiles as well as dynamic delay updates to minimize bitrates switching. We evaluated DAVS using trace driven simulation and with widely used video streaming performance metrics. DAVS shows higher performance compared to other benchmark algorithms (OSMF, BBA2, SARA and PSQA). In addition, we used a QoE utility function widely used in literature to evaluate the performance of DAVS. DAVS increases the QoE between 15% and 55% compared to the benchmark algorithms.

Our work will continue in the following directions. First, we will enhance DAVS by considering the mobile device capabilities (such as screen resolution and processing speed) to select the most suitable bitrate. Second, we will address the use of MultiPath TCP (MPTCP) [36] to increase the TCP throughput in mobile networks.

References

[1] C. Systems, "Visual Networking Index: Global IP traffic forecast 2015-2020," https://goo.gl/rGQCL8, Last accessed Jan 20, 2017.

[2] T. Stockhammer, "Dynamic adaptive streaming over HTTP: standards and design principles," in *Proceedings of the second annual ACM conference on Multimedia systems*, 2011, pp. 133–144.

[3] B. Bing, *Next-generation video coding and streaming*. Wiley, 2015.

[4] F. Dobrian, V. Sekar, A. Awan, I. Stoica, D. Joseph, A. Ganjam, J. Zhan, and H. Zhang, "Understanding the impact of video quality on user engagement," in *ACM SIGCOMM CCR*, vol. 41, no. 4, 2011, pp. 362–373.

[5] H. Nam, K.-H. Kim, and H. Schulzrinne, "QoE matters more than QoS: Why people stop watching cat videos," in *IEEE INFOCOM, 2016*, 2016.

[6] W. Pu, Z. Zou, and C. W. Chen, "Dynamic adaptive streaming over http from multiple content distribution servers," in *Global Telecommunications Conference (GLOBECOM 2011), 2011 IEEE*. IEEE, 2011, pp. 1–5.

[7] T. Mangla, N. Theera-Ampornpunt, M. Ammar, E. Zegura, and S. Bagchi, "Video through a crystal ball: effect of bandwidth prediction quality on adaptive streaming in mobile environments," in *Proceedings of the 8th International Workshop on Mobile Video*, 2016, p. 1.

[8] R. M. Abuteir, A. Fladenmuller, and O. Fourmaux, "An SDN approach to adaptive video streaming in wireless home networks," in *IEEE International Wireless Communications and Mobile Computing Conference*, 2016.

[9] ——, "Sdn based architecture to improve video streaming in home networks," in *IEEE International Conference on Advanced Information Networking and Applications (AINA)*. IEEE, 2016, pp. 220–226.

[10] Y. Zaki, T. Pötsch, J. Chen, L. Subramanian, and C. Görg, "Adaptive congestion control for unpredictable cellular networks," in *ACM SIGCOMM CCR*, vol. 45, no. 4, 2015, pp. 509–522.

[11] J. Jiang, V. Sekar, and H. Zhang, "Improving fairness, efficiency, and stability in HTTP-based adaptive video streaming with festive," in *Proceedings of the 8th International Conference on Emerging networking experiments and technologies*, 2012, pp. 97–108.

[12] T.-Y. Huang, R. Johari, N. McKeown, M. Trunnell, and M. Watson, "A buffer-based approach to rate adaptation: Evidence from a large video streaming service," *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 4, pp. 187–198, 2015.

[13] A. H. Zahran and J. J. Quinlan, "Impact of the LTE scheduler on achieving good QoE for DASH video streaming," in *Proceedings of the 22nd IEEE International Symposium on Local and Metropolitan Area Networks (LANMAN 2016)*, 2016.

[14] X. K. Zou, J. Erman, V. Gopalakrishnan, E. Halepovic, R. Jana, X. Jin, J. Rexford, and R. K. Sinha, "Can accurate predictions improve video streaming in cellular networks?" in *Proceedings of the 16th International Workshop on Mobile Computing Systems and Applications*, 2015, pp. 57–62.

[15] A. Systems, "Open source media framework," http://goo.gl/89Re4W, Last accessed Jan 20, 2017.

[16] P. Juluri, V. Tamarapalli, and D. Medhi, "SARA: Segment aware rate adaptation algorithm for dynamic adaptive streaming over HTTP," in *2015 IEEE International Conference on Communication Workshop (ICCW)*, 2015, pp. 1765–1770.

[17] ——, "QoE management in DASH systems using the segment aware rate adaptation algorithm," in *NOMS 2016 IEEE/IFIP Network Operations and Management Symposium*, 2016, pp. 129–136.

[18] H. K. Yarnagula, S. Luhadia, S. Datta, and V. Tamarapalli, "Quality of experience assessment of rate adaptation algorithms in DASH: An experimental study," in *2016 8th International Conference on Communication Systems and Networks (COMSNETS)*, 2016, pp. 1–8.

[19] Y. Liu and J. Y. Lee, "A unified framework for automatic quality-of-experience optimization in mobile video streaming," in *IEEE INFOCOM 2016-The 35th Annual IEEE International Conference on Computer Communications*, 2016, pp. 1–9.

[20] J. Chen, R. Mahindra, M. A. Khojastepour, S. Rangarajan, and M. Chiang, "A scheduling framework for adaptive video delivery over cellular networks," in *Proceedings of the 19th annual international conference on Mobile computing & networking*, 2013, pp. 389–400.

[21] M. Riad, H. Abu-Zeid, H. S. Hassanein, M. Tayel, and A. A. Taha, "A channel variation-aware algorithm for enhanced video streaming quality," in *Local Computer Networks Conference Workshops (LCN Workshops), 2015 IEEE 40th*. IEEE, 2015, pp. 893–898.

[22] K. Prabhu, *Window functions and their applications in signal processing*. CRC Press, 2013.

[23] H. Wu and Y. Pan, *Medium access control in wireless networks*. Nova Publishers, 2008, vol. 8.

[24] "Big buck bunny homepage," https://peach.blender.org/, Last accessed Jan 20, 2017.

[25] "Netflix encoding bitrate," https://goo.gl/M32pFh, Last accessed Jan 20, 2017.

[26] "FFmpeg homepage," https://www.ffmpeg.org/, Last accessed Jan 20, 2017.

[27] "GPAC homepage," https://gpac.wp.mines-telecom.fr, Last accessed Jan 20, 2017.

[28] C. Sieber, P. Heegaard, T. Hoßfeld, and W. Kellerer, "Sacrificing efficiency for quality of experience: Youtube's redundant traffic behavior," in *IFIP Networking Conference (IFIP Networking) and Workshops, 2016*. IEEE, 2016, pp. 503–511.

[29] D. C. Mocanu, A. Liotta, A. Ricci, M. T. Vega, and G. Exarchakos, "When does lower bitrate give higher quality in modern video services?" in *Network Operations and Management Symposium (NOMS), 2014 IEEE*. IEEE, 2014, pp. 1–5.

[30] "The SSIMplus," https://goo.gl/3BFB7b, Last accessed Jan 20, 2017.

[31] A. Balachandran, V. Sekar, A. Akella, S. Seshan, I. Stoica, and H. Zhang, "Developing a predictive model of quality of experience for internet video," in *ACM SIGCOMM CCR*, vol. 43, no. 4. ACM, 2013, pp. 339–350.

[32] R. K. Mok, E. W. Chan, and R. K. Chang, "Measuring the quality of experience of http video streaming," in *Integrated Network Management (IM), 2011 IFIP/IEEE International Symposium on*. IEEE, 2011, pp. 485–492.

[33] X. Liu, F. Dobrian, H. Milner, J. Jiang, V. Sekar, I. Stoica, and H. Zhang, "A case for a coordinated internet video control plane," in *Proceedings of the ACM SIGCOMM 2012 conference on Applications, technologies, architectures, and protocols for computer communication*. ACM, 2012, pp. 359–370.

[34] J. Joskowicz and J. C. L. Ardao, "A parametric model for perceptual video quality estimation," *Telecommunication Systems*, pp. 49–62, 2012.

[35] X. Yin, A. Jindal, V. Sekar, and B. Sinopoli, "A control-theoretic approach for dynamic adaptive video streaming over HTTP," *ACM SIGCOMM CCR*, vol. 45, no. 4, pp. 325–338, 2015.

[36] "MPTCP," https://goo.gl/jL85kW, Last accessed Jan 20, 2017.