ARBITER: ADAPTIVE RATE-BASED INTELLIGENT HTTP STREAMING ALGORITHM

Ahmed H. Zahran[1,2] *and Cormac J. Sreenan*[1]

[1]Dept. of Computer Science, University College Cork, Ireland
[2]EECE Dept., Faculty of Engineering, Cairo University, Egypt
a.zahran, cjs@cs.ucc.ie

ABSTRACT

Dynamic Adaptive streaming over HTTP (DASH) is widely used by content providers for video delivery and dominates traffic on cellular networks. The inherent variability in both video bitrate and network bandwidth negatively impacts the user Quality of Experience (QoE), motivating the design of better DASH-compliant adaptation algorithms. In this paper we present ARBITER, a novel streaming adaptation algorithm that explicitly integrates the variations in both video and network dynamics in its adaptation decisions. Our simulation-based performance evaluation, using real video content and cellular bandwidth traces, shows that ARBITER achieves an excellent tradeoff among streaming metrics in terms of received video quality, stall count, stall duration, and switching dynamics, leading to a relative improvement of 17-45% in user QoE in comparison to state-of-the-art algorithms.

Introduction

The volume of mobile video traffic is growing rapidly. However, the shared and highly-variable nature of wireless negatively impacts the streaming performance [1]. Hence, there is a need to design intelligent streaming systems capable of accommodating the frequent variations in operating conditions.

In HTTP streaming systems, the video is split into multiple small duration chunks, known as *segments*, that are requested from the content distribution networks (CDN) using HTTP. Each segment is encoded into different representations that may vary in resolution, quality, and/or data rate. The DASH standard [2] defines the structure of a media presentation description (MPD) file that specifies segment encoding and related details. However, the DASH standard does not define the algorithm used by the client for deciding which segment quality to request or how to adapt so as to achieve good Quality of Experience (QoE).

The design of an adaptive streaming policy involves tradeoffs between several visual quality impairments, such as low video quality, and temporal impairments, such as stalls. The state-of-the-art in adaptive streaming algorithms can be generally classified as buffer-based [3], rate-based [4], and hybrid strategies [5, 6]. In wireless systems, the channel features temporal variations due to its multi-path nature, user mobility, and medium sharing. Additionally, the variable bitrate nature of video traffic further contributes to the inherent system variability. Such highly variable operating conditions may significantly affect the performance of different adaptation algorithms leading to degrading the user QoE.

In this paper, we propose ARBITER as a novel algorithm that integrates application state, network state, and video dynamics to intelligently adapt to operating conditions. Fundamentally, ARBITER is a rate-based algorithm that establishes an estimate for the available bandwidth based on the measured throughput of previously downloaded segments. However, we include a new algorithmic feature that scales the throughout estimate based on the variations in the measured throughput samples for segments and on a buffer level. Additionally, ARBITER considers a short-term average for quality representation bitrate in its adaptation decisions. In our evaluation, we compare the performance of ARBITER to the state-of-the-art algorithms BBA [3] and ELASTIC [6], showing a significant improvement in user QoE. For example, with a typical 60 second client buffer and 4 second segment duration, ARBITER reduces stalls by 33% in comparison to BBA while maintaining similar average video quality and switching dynamics, yielding 26% improvement in QoE. Over a wide variety of parameters, ARBITER improves QoE over the state-of-the-art algorithms by 17-45%.

In Section 2 we highlight the background and related work. ARBITER design is presented in Section 3 followed by its performance evaluation in Section 4. Conclusions are then presented in Section 5.

Background and Related Work

DASH is dominating video transmission due to its ability to traverse firewalls and the availability of HTTP infrastructure. A DASH client implements a media buffer as an interface between the media transmission and decoding process. As the buffer depletes, the media playout stalls and only resumes when sufficient media is replenished in the media buffer. HTTP streaming clients implement an adaptation algorithm so that the next segment quality is selected so as to maintain sufficient media in the buffer while avoiding visual impairments such as reduced video quality and irritating quality

978-1-4799-0000-0

changes.

In [3], Huang et al. propose multiple versions of buffer-based strategies that select the video representation based on the buffer level. A couple of the proposed variants adopt rate-based strategies for the initial buffering to improve the video quality while initially filling the buffer. In rate-based strategies, the selected quality rate is bounded by the client estimate for the average network throughput. In [4], Jiang et al. propose FESTIVE as a streaming heuristic that employs a harmonic mean estimate for the network throughput, a randomized request scheduler, stateful rate switching, and delayed rate update. These different components target improving the streaming performance when multiple users share a network bottleneck. Li et al. [7] propose PANDA as a streaming heuristic that selects the segment quality that ensures an accurate estimate for the measured segment throughput by using the network persistently. This is smoothed using a harmonic mean as a rate bound for the selected quality. All of the aforementioned rate-based algorithms abstract the measured throughput by its mean. ARBITER goes beyond this by using both the first and second moments of the throughput samples to better accommodate the high variability in wireless network throughput. In [8], Xie et al. exploit physical layer information to improve the available network throughput estimation and hence, enhance the streaming performance in cellular networks. In contrast, ARBITER only depends on application level information and can be used with any access technology.

Hybrid adaptation algorithms consider both application and network states in their decision. In [6], De Cicco et al. propose ELASTIC that employs a proportional integral controller combined with a harmonic network throughput estimator to determine the next representation quality. In [9], Bokani et al. present an adaptation engine using a Markov Decision Process (MDP) framework. The authors propose three heuristics that use online or offline estimates for bandwidth statistics to accommodate MDP complexity. In [5], Yin et al. propose model predictive control (MPC) as a QoE optimised adaptation framework that combines both network and application status. Additionally, they propose a heuristic by quantising their decision parameter space. In [10], OSCAR optimises visual quality subject to a probabilistic constraint on the stalls using a detailed optimisation framework. In contrast, ARBITER is a generic light-weight decision heuristic that intelligently integrates application and network states without relying on any optimisation tools or stored guiding data.

ARBITER Design

We consider a client streaming a video split into N segments each of which contains T seconds of video. We assume a persistent HTTP connection over which segments are sequentially requested using HTTP GET requests. Each segment is encoded into Q quality representations with each representation having an average encoding rate R_q, where $q \in \{1, ..., Q\}$. For every segment n, the streaming client chooses the streaming rate $r_n \in \{R_1, ..., R_Q\}$ and the corresponding segment size of this segment is denoted as S_{r_n}.

The main objective of ARBITER is to improve the user streaming experience by realising a balanced profile of relevant performance metrics, such as number of stalls, duration of stalls, average quality rate, average number of quality switches, and average level of quality switching. ARBITER includes two components in its segment quality selection: an adaptive throughput estimator and a quality-aware adaptation policy.

Adaptive Throughput Estimation

Our throughput mean estimate μ_s is based on an exponentially weighted moving estimation window that accommodates W samples. Let k be the index of the last downloaded segment. Segment k throughput would be assigned a weight, denoted as ω. The throughput of previous segments in the throughput estimation window would be assigned a weight $\omega(1-\omega)^{k-i}$, where i is the segment index. These weights are then normalised to their geometric sum and the final weights are expressed as

$$w_i = \frac{\omega(1-\omega)^{k-i}}{1-(1-\omega)^W} \; \forall i \in \{1, .., W\}. \tag{1}$$

The weighted throughput mean $\mu_s$can be expressed as

$$\mu_s \quad = \quad \sum_{i=1}^{W} w_i b_{k-i}, \tag{2}$$

where b_n represents the measured throughput for segment n. Typically, ω would be a fraction, i.e., $\omega \in (0, 1)$. Large values of ω would improve the responsiveness of the estimator to changes in the bandwidth but also could lead to frequent irritating switches in video quality level. ARBITER adjusts this average throughput estimate based on network and application state information. A scale parameter is estimated based on the variations in the observed segment throughput samples.

In highly variable link conditions, such as wireless, the streaming client should be more cautious about its rate estimate. Many of the algorithms [6, 7, 4, 5] employ a harmonic mean estimator, which tends to be conservative and could lead to streaming low video quality and under-utilising the system resources. ARBITER adopts a different approach by employing adaptively scaling its estimate based on the second moment of the throughput samples. We define our throughput variance scaling factor, denoted as ρ_v, as

$$\rho_v = \underline{\rho_v} + (1 - \underline{\rho_v}) \left(1 - min(\theta, 1)\right)^2, \tag{3}$$

where $\underline{\rho_v}$ represents a minimum bound on ρ_v and θ represents the coefficient of variation of segment throughput samples and is expressed as

$$\theta = \sqrt{\frac{W}{W-1} \sum_{i=0}^{W-1} w_i (b_{k-i} - \mu_s)^2 / \mu_s}. \tag{4}$$

Hence, if the throughput measurements feature a small variation, θ tends to 0 and; hence, ρ_v would tend to 1 and no reduction is applied to the estimated mean rate μ_s. On the contrary, as the throughput variance increases, ρ_v would tend to the minimum bound $\underline{\rho_v}$. This design implies that the average throughput is a considered a good estimate at stable network condition. Alternatively, ρ_v would kick in to provide a more network-cautious estimate.

Our throughput estimate so far is unaware of the application state. Typically, a low application buffer level is a good indicator of imminent stalls that noticeably affect the user-perceived QoE. A high buffer-level could be an indicator for good network conditions. However, as the buffer saturates, the streaming client stops requesting new segments until there exists enough buffer space for one or more segments. This documented ON-OFF [11] behaviour may significantly affect the performance of rate-based algorithms as the client would not have an accurate estimate for the available network throughput during OFF periods. Hence, ARBITER employs a buffer-based rate scaling factor, denoted as ρ_b, to reduce the throughput estimate at low buffer levels to avoid stalls and build up the buffer. At high buffer levels, ρ_b scales the estimated rate up to avoid OFF periods and improve the video quality. The buffer-based scaling factor ρ_b is expressed as

$$\rho_b = \underline{\rho_b} + (\overline{\rho_b} - \underline{\rho_b})\frac{B_k}{B_m}, \tag{5}$$

where B_k represents the buffer level on receiving segment k, B_m represents a buffer size, $\underline{\rho_b}$ and $\overline{\rho_b}$ respectively represent lower and upper bounds for ρ_b. Finally, our adaptive throughput estimate, denoted as r_t, is calculated as

$$r_t = \mu_s * \rho_v * \rho_b. \tag{6}$$

Video-Aware Adaptation Policy

Encoded video features significant variations in the actual segment bitrate from the average representation rate advertised in the MPD file. Such variations may mislead an adaptation policy that establishes its decision based on the advertised average representation rate, which represents a long-term average rate. Hence, ARBITER bases its adaptation decision based on a short-term average rate estimated for the next few segments, denoted as r_a, for a window of future video segments for each candidate quality rate. The actual rate is estimated as

$$r_a(q) = \frac{\sum_{i=1}^{W_v} S_{k+i}(q)}{W_v * T}, \tag{7}$$

where W_v represents the size of the look ahead window in segments and $S_i(q)$ represents the size of segment i at quality q. ARBITER then identifies a set of candidate qualities whose actual rates are upper bounded by r_t and selects the highest quality representation among the candidate set provided that quality up-switches are limited to n_s levels. The latter constraint is added to ensure progressive quality improvement and avoiding abrupt quality changes. The implementation of ARBITER is presented in Algorithm.

Algorithm 1 ARBITER

INPUT: b_n, ω, W, $\underline{\rho_v}$, $\overline{\rho_b}$, $\underline{\rho_b}$, B_m, r_k, W_v
OUTPUT: selected rate r_s
estimate μ_s, θ, ρ_v, ρ_b, r_t [i.e., 1-6]
$r_s = \max\, r_i \in R | r_i < r_t$
$s = Index(r_s)$ // index of last selected representation
$l = Index(r_k)$; // index of last received segment
if $(s - l \geq n_s)$
  $s = l + n_s$
while $(Not(r_a(s) \leq r_t))$
  s- -; // reduce representation rate if short-term rate is larger than target rate
request r_s

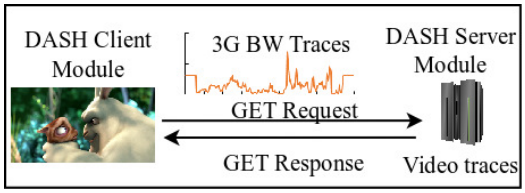


Fig. 1: Evaluation Setup

Performance Evaluation

We first present our evaluation setup and performance metrics followed by our performance evaluation results.

Evaluation Setup

We evaluate the performance of ARBITER using ns-3[1]. Our evaluation setup is shown in Figure 1. We implemented a new module to emulate DASH operation in ns-3. The DASH client communicates with a DASH server over a single TCP connection over which HTTP requests and responses are communicated. In our implementation, we consider a fixed HTTP header size of 100 bytes for both requests and responses. To emulate wireless links, we connect the client and server using a point-to-point link whose bandwidth is varied according to cellular throughput traces [12]. These traces are collected with the users riding different transportation means including car, ferry, tram, metro, and bus. Since these traces do not include information about the link delay, we tested different RTT delays including 20ms, 40ms, and 80ms. Due to space limitations, we show the results for 40ms only; we observed that varying RTT does not have a significant impact on the performance trends and conclusions. Hence, we believe that bandwidth variations in the cellular systems has the dominant impact on DASH performance.

In our simulations, we used seven different five-minute clips including A New Hope, An Idiot Abroad, Avatar, Big Hero 6, Casino Royal, Game Plan, and Harry Potter from

[1]https://www.nsnam.org/

Table 1: The parameters of streaming algorithms

BBA [3]	$\underline{r} = 2T$	$\bar{r} = 0.6B_m$	$\Delta B = 0.875T$
	$\tau_h = 0.9B_m$		
ELASTIC [6]	$W = 5$	$k_p = 0.01$	$k_I = 0.001$
ARBITER	$\omega = 0.4$	$W = 10$	$W_v = 5$
	$\underline{\rho_v} = 0.3$	$\underline{\rho_b} = 0.5$	$\overline{\rho_b} = 1.5$

Dash dataset [13][2], where videos are encoded at {235, 375, 560, 750, 1050, 1750, 2350, 3000, 3850, 4300}. We used multiple videos in our evaluation as we were interested in investigating the interaction between bandwidth changes and temporal variation in rate demands of different videos. Interestingly, we found there exist significant differences in the performance of different videos with the same network trace.

We compare the performance of different streaming algorithms including ARBITER, BBA [3][3], and ELASTIC [6] in different configurations of segment and buffer sizes. ELASTIC represents the class of hybrid streaming clients while and BBA represents the class of buffer-based streaming clients. Table 1 shows the main parameters of different algorithms. The parameters of ARBITER identified based on scanning for its parameter space. In our experiments, we consider different application buffer sizes including 60 Sec and 120 seconds. Additionally, we consider different segment sizes including 4 sec and 10 sec.

Our performance metrics per-session include the average received quality rate (r_{av}) in kbps, the average number of stalls (n_{st}), the average total stall duration (t_{st}) in seconds, the average number of switches (n_{sw}), the average switching level (l_{sw}), and DASH-user experience (DASH-UE) QoE metric (χ) [14], which is highly correlated with subjective results. DASH-UE models temporal (initial startup delay, stall count, and stall delay) and visual impairments (persistence on low quality video and performing down switching). The quality and switching penalties are estimated based on the objective video quality metric that is known for its high correlation to subjective evaluation. The highest DASH-UE value is 100 and reduces as the level of impairment increases. Our results represent the average of 448 evaluation scenarios - all combinations of 64 mobility traces and 7 videos.

Simulation Results

Table 2-a shows the averaged results using a 60 sec buffer and 4 second segment. With 4 sec segment, we consider two segments for initial buffering and one segment rebuffering before resuming playout after stalls. ARBITER strikes a balance between different key performance metrics leading to the highest user QoE. On comparing ARBITER to BBA, we find that it closely attains the same average quality rate r_{av} with only $103kbps$ drop. This drop may not be perceived by users considering that the average difference between quality rates is $445kbps$. However, this insignificant quality reduction is accompanied by a noticeable 33% drop in the number of stalls n_{st} *and* 35% in the average stall duration per session. ARBITER has a similar relative number of switches n_{sw} as BBA, performed with lower amplitudes as suggested by l_{sw} and as shown in Figure 2 , which plots the selected quality rates of the compared algorithms in one of simulated scenarios. Hence, such quality transitions are probably less noticeable by the end-user. ARBITER scores a QoE metric χ that 1.26x BBA's QoE.

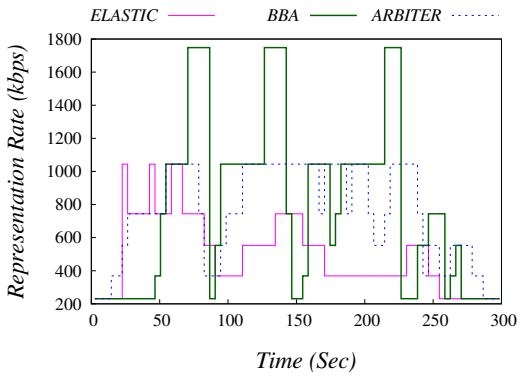


Fig. 2: Quality rates as determined by the algorithms in one of the tram traces

ELASTIC shows the best stall performance with 30% less stalls and 60% average session stall duration in comparison to ARBITER. Additionally, ELASTIC reduces the average number of switches relative to ARBITER by 50%. These gains are due to the conservative throughput estimate, as shown in Figure 2, that enables ELASTIC to fill in its buffer without making many switches. Hence, ELASTIC becomes more immune to stalls when the network conditions deteriorate for long periods. However, these gains are accompanied by a huge reduction in the average video quality rate that is approximately 63% of the rate achieved by ARBITER. This huge quality gap enables ARBITER to achieve a QoE metric that is 1.29x ELASTIC's QoE.

Large Buffer Size

Table 2-b shows the key performance metrics for the adaptation algorithms with 120 Sec client buffer and 4 Sec segment. Clearly, the larger buffer size enables all algorithms to improve their stall performance by reducing both the number of stalls and stall duration. ARBITER and ELASTIC shows the best stall performance followed by BBA. ELASTIC performs the least number of representation switches followed by BBA and then ARBITER. This is interpreted by the conservative throughput estimate of ELASTIC the leads to persisting on low representation qualities for longer periods. Hence, both ARBITER and BBA achieves an average quality rate that is 1.9x that achieved by ELASTIC. Interestingly, BBA shows a great improvement in its switching dynamics as the buffer

[2]http://www.cs.ucc.ie/~jq5/www_dataset/

[3]We implemented BBA-2 variant with linear mapping between the reservoir and the upper buffer threshold.

Table 2: Key Performance metrics

(a) 60 sec buffer - 4 Sec segment (b) 120 sec buffer - 4 Sec segment (c) 120 sec buffer - 10 Sec segment

	n_{st}	t_{st}	r_{av}	n_{sw}	l_{sw}	χ	n_{st}	t_{st}	r_{av}	n_{sw}	l_{sw}	χ	n_{st}	t_{st}	r_{av}	n_{sw}	l_{sw}	χ
ARBITER	0.34	2.8	1055	26.3	1.1	48.7	0.08	0.8	935	22.8	1.1	47.8	0.06	1.0	928	11.6	1.2	40.3
BBA2	0.44	4.3	1158	25.3	1.7	38.5	0.10	0.9	983	15.2	1.3	40.8	0.09	1.5	987	10.8	1.7	30.3
ELASTIC	0.24	1.7	662	13.3	1.2	37.3	0.08	0.7	495	10.4	1.2	30.5	0.03	0.8	474	5.5	1.5	27.9

was increased from 60 sec to 120 sec. We believe this is due to the larger cushion region that mandates larger buffer sub-regions for each representation quality. Hence, larger buffer size would improve both switching and stall performance of BBA but reduces its average quality rate. However, use of larger client buffers leads to wasted link use (and possibly increased user cost) in cases where a user abandons the session [15]. The improved switching performance and higher average quality enabled BBA to achieve a QoE that is 1.56x ELASTIC's QoE. ARBITER scores the highest QoE metric that is 1.17x BBA's QoE score.

Figures 3 plots the cumulative distribution function (CDF) of different performance metrics for different algorithms and buffer sizes. Fig. 3a confirms that ELASTIC's low average rate is due to picking lower quality rates in comparison to both BBA and ARBITER. It is also clear that with smaller buffer, all algorithms tends to pick higher rates for different reasons. BBA streams higher rates because the buffer gets full faster and hence the client starts to request higher quality representations. On the other hand, ARBITER tends to pick higher rates because the upscaling buffer factor is activated earlier as the buffer occupancy increases.

Figure 3b plots the CDF of number of stalls encountered by different algorithms. The figure shows that doubling the buffer size from 60 sec to 120 sec helps increasing the number of stall free sessions from 85% to 95% for BBA, from 88% to 96.5% for ARBITER and from 90% to 97% for ELASTIC. Increasing the buffer size also reduces the maximum number of stalls for all algorithms to 5 stalls from 8, 13, and 17 stalls for ELASTIC, ARBITER, and BBA, respectively.

Figure 3c shows that increasing the application buffer reduces the number of switches for all algorithms. Clearly, BBA features a larger reduction in the number of the switches for the same reasons explained above. While ARBITER performs more switches, its switches have smaller amplitude in comparison to ELASTIC and BBA.

Figure 3d shows the CDF of the QoE metric for the simulated algorithms and different buffer sizes. The figure shows that ARBITER sessions have higher QoE metric values in comparison to BBA and ELASTIC. This superiority is interpreted by ARBITER's ability to maintain an excellent profile of stall, switching and quality metrics. The lowest QoE metric values are attained by BBA with small buffer (BBA-60) and ELASTIC with large buffer (ELASTIC-120). BBA-60 low QoE is due to bad switching and stall performance while ELASTIC-60 low QoE is due to persisting on low video quality due to conservative throughput estimates.

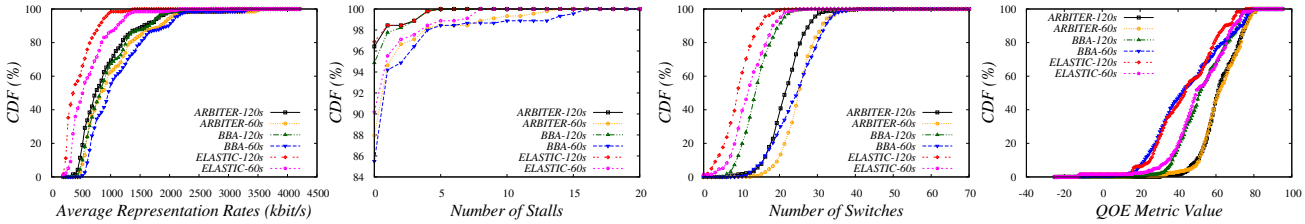
Clearly, the buffer size significantly affects the performance of different streaming algorithms in different ways. Note that, with the smaller buffer, ELASTIC scores higher QoE even though it features worse stall performance. We also see how BBA's performance metrics are affected by the buffer size. Hence, adjusting the buffer size plays a significant role in achieving a balanced profile of streaming performance metrics such that the user QoE is improved. Identifying the optimal buffer size of different streaming strategies based on streaming context, operating condition, and other relevant factors represents an interesting future research point.

Larger Segment

Some content providers prefer using large segment sizes for wireless networks. Table 2-c shows our key performance metrics for the tested algorithms when 10-sec video segments are used. For this set of results, video client initial buffering and rebuffering are set to 10 seconds (one segment). Note that the client need to download fewer but larger segments. In comparison to the 4-seg segment and 120 sec buffer case, using 10 sec segments helps all algorithms to reduce their average number of stalls by 25%, 9%, and 62% for ARBITER, BBA2, and ELASTIC, respectively. However, the average stall duration increased by approximately 37%, 77%, and 10% for ARBITER, BBA and ELASTIC, respectively. The reduced number of stalls is steered by two factors. The first is the reduction in stall opportunities as the client downloads fewer segments while the second is the larger media duration available in the buffer after every stall recovery. Yet, the second factor also leads to increasing the stall duration as longer download time is observed.

The impact of the segment size on the average received quality rate is insignificant as it remains less than 5% for all algorithms. The number of the switches is approximately halved as the segment size increases because the number of downloaded segments decreases. However, by estimating the ratio of number of switches to the switching opportunities (74 for 4-sec segment and 29 for 10-sec segment in a five minute video), all algorithms would have a higher relative switching frequency. The average quality level switching increases for all algorithms because the client would have to deal with larger variations that may take place over the longer segment download time. With smaller segments, the client would be able to promptly adapt to network changes leading to frequent but smoother quality changes.

The results also show that ARBITER achieves the highest



(a) Average received data rate (b) Number of stalls per session (c) Number of quality switches per session (d) User QoE

Fig. 3: CDF of Performance metrics (4 sec segment)

QoE score with an approximate improvement of 34% and 46% in comparison to BBA2 and ELASTIC, respectively. Our analysis indicated that the drop in QoE is mainly due to increasing the penalty of persisting on low qualities decreases and a slight increase in switching penalty of ELASTIC and ARBITER. These increases take over the drop in stall QoE penalty that took place for all algorithms. Hence, one can conclude that using a medium segment size (e.g., 4 sec) would help intelligent clients to improve the user QoE.

Conclusions

The design of adaptive algorithms for HTTP-based streaming systems is challenged by the inherent variability in both network conditions and video dynamics. In this paper, ARBITER is presented as a novel streaming algorithm that integrates application and network states in its adaptation decision. ARBITER captures the network condition by integrating both the average and variance of the received segment throughput in the decision. Additionally, ARBITER also considers the application state by integrating the buffer occupancy and actual video segment rates in the representation rate selection decision. Our simulations, derived using real video content and cellular network traces, show that ARBITER strikes a balance between achieved rate, stalls dynamics and switching behaviour. ARBITER achieves the highest QoE in comparison to state-of-the-art algorithms by a relatively large margin.

Acknowledgment

This publication has emanated from research conducted with the financial support of Science Foundation Ireland (SFI) under Grant Number 13/IA/1892. The authors wish to thank Jason Quinlan for providing us with access to the DASH video dataset.

References

[1] F. Fund et. al., "Performance of DASH and WebRTC Video Services for Mobile Users," in *Proc. PV'13*, San Jose, CA, Dec 2013.

[2] I. Sodagar, "The MPEG-DASH Standard for Multimedia Streaming Over the Internet," *IEEE MultiMedia*, vol. 18, no. 4, pp. 62–67, Oct. 2011.

[3] T.-Y. Huang et. al, "A Buffer-based Approach to Rate Adaptation: Evidence from a Large Video Streaming Service," in *Proc. of the 2014 ACM SIGCOMM '14*, Chicago, Illinois, USA, 2014, pp. 187–198.

[4] J. Jiang et. al., "Improving Fairness, Efficiency, and Stability in HTTP-based Adaptive Video Streaming with FESTIVE," in *Proc. of CoNEXT '12*, Nice, France, 2012, pp. 97–108.

[5] X. Yin et. al., "A Control-Theoretic Approach for Dynamic Adaptive Video Streaming over HTTP," *SIGCOMM Comput. Commun. Rev.*, vol. 45, no. 5, pp. 325–338, Aug. 2015.

[6] L. De Cicco et. al., "ELASTIC: A Client-Side Controller for Dynamic Adaptive Streaming over HTTP (DASH)," in *Proc. of PV '13*, San Jose, CA, Dec 2013.

[7] Z. Li et. al., "Probe and Adapt: Rate Adaptation for HTTP Video Streaming At Scale," *IEEE J. Sel. Area in Commun.*, vol. 32, no. 4, pp. 719–733, 2014.

[8] X. Xie et. al., "piStream: Physical Layer Informed Adaptive Video Streaming over LTE," in *Proc. of MobiCom '15*, Paris, France, 2015, pp. 413–425.

[9] A. Bokani et. al., "HTTP-Based Adaptive Streaming for Mobile Clients using Markov Decision Process," in *Proc. of PV'13*, San Jose, CA, Dec 2013.

[10] A. H. Zahran et. al., "OSCAR: An Optimized Stall-Cautious Adaptive Bitrate Streaming Algorithm For Mobile Networks," in *Proc. MoVid'16 (to appear)*, 2016.

[11] A. Saamer et. al., "What Happens when HTTP Adaptive Streaming Players Compete for Bandwidth?" in *Proc. of NOSSDAV'12*, ser. NOSSDAV '12, 2012, pp. 9–14.

[12] H. Riiser et. al., "Commute Path Bandwidth Traces from 3G Networks: Analysis and Applications," in *Proc. of MMSys '13*, Oslo, Norway, 2013, pp. 114–118.

[13] J. J. Quinlan et. al, "Datasets for AVC (H.264) and HEVC (H.265) for Evaluating Dynamic Adaptive Streaming over HTTP (DASH)," in *Proc. of MMsys dataset track (to appear)*, 2016.

[14] Y. Liu et. al., "Deriving and Validating User Experience Model for DASH Video Streaming," *IEEE Trans. on Broadcasting*, vol. 61, no. 4, pp. 651–665, Dec 2015.

[15] S. S. Krishnan and R. K. Sitaraman, "Video Stream Quality Impacts Viewer Behavior: Inferring Causality Using Quasi-experimental Designs," in *Proc. of ACM IMC'12*, Boston, Massachusetts, USA, 2012, pp. 211–224.