Quality of Experience of Commercially Deployed Adaptive Media Players

Christian Timmerer
Alpen-Adria-Universität Klagenfurt
Klagenfurt, Austria
christian.timmerer@itec.aau.at

Anatoliy Zabrovskiy, Evgeny Kuzmin,
Evgeny Petrov
Petrozavodsk State University
Petrozavodsk, Russia
{z_anatoliy, kuzmin, johnp}@petrsu.ru

***Abstract*—In the past decade we observed the transition from push-based, fully managed media streaming to pull-based, unmanaged adaptive HTTP streaming thanks to enhancements in media compression, network capacity, and client capabilities. Adaptive media players, specifically their algorithms, have been subject to research for a long time and lead to various approaches documented in the literature. In the past years we witnessed more and more commercial deployments taking into account findings presented in scientific papers but a quantitative evaluation and assessments of its performance is missing. In this paper, we propose means for the automated performance evaluation of commercially deployed adaptive media players with respect to *i)* objective, well-known metrics, such as bitrate, stalls, startup delay and *ii)* derived/calculated metrics (instability, inefficiency, average bitrate) previously proposed in the literature. Additionally, we propose a new metric (Bandwidth index) to measure the effectiveness of bandwidth utilization and together with existing QoE models for adaptive HTTP streaming (focusing on stalls, startup delay) we demonstrate its usefulness in this domain.**

I. Introduction

Streaming audio and video from Internet services is more and more replacing traditional broadcast television services and is being massively deployed thanks to the availability of international standards such as MPEG Dynamic Adaptive Streaming over HTTP (DASH) [9]. In MPEG-DASH, the media content is prepared in multiple versions (representation, e.g., different bitrates, resolutions) and divided in segments, each comprising a given amount of data measured in seconds allowing for a dynamic switching (typically at segment boundaries) between the different representations based on the given context conditions (e.g., fluctuating bandwidth in mobile networks).

In the past, we have witnessed a plethora of publications in this area, most of them focus on the adaptation algorithm and its evaluation (e.g., [4], [1], [12], [10], [11]) which in most cases determines the Quality of Experience in HTTP adaptive streaming [8]. However, these papers either propose a specific aspect of the issue (including a preliminary evaluation) or provide a comparison of a reduced set of different algorithms. None of them address real-world deployments with some exceptions [7] but a comprehensive evaluation of existing approaches is almost impossible due to the lack of the actual implementations and a common evaluation framework.

The aim of this paper is *i)* to investigate the streaming performance of adaptive media players (DASH), specifically focusing on those adopted within real-world deployments including – but not limited to – commercially available players and *ii)* to analyze available QoE models proposed in the context of DASH and how they could be utilized to assess the streaming performance. Therefore, we built an open, flexible evaluation architecture which enables the automated evaluation of DASH players under different network conditions. We have selected Web/HTML5 players as the main platform due to its widespread adoption and integrated a set of commercially deployed adaptive HTML5 players into our evaluation system. We analyzed the survey of Seufert et al. [8] for appropriate QoE models and have found two models which complement already existing, objective metrics – used in our evaluation system – and which are easy to integrate into our system. We performed a series of experiments under predefined network conditions to gather performance data and we will discuss findings in this paper. Further details about evaluation architecture and setup can be found here [13].

The remainder of this paper is organized as follows. Section II describes the system architecture developed for the automated QoE evaluation of adaptive HTML5 players and algorithms. The evaluation setup including an overview of the used players is described in Section III. Results of the evaluation are presented and discussed in Section IV and the paper is concluded in Section V.

II. Evaluation architecture

The system architecture of the proposed evaluation framework is shown in Fig. 1 and comprises a controlled environment enabling the automated evaluation of adaptive streaming systems. It defines a flexible system that allows adding new adaptive HTML5 players (and algorithms) relatively fast as it mostly relies on existing Web technologies based on HTML5.

The server infrastructure comprises three servers with distinct functionalities running Ubuntu OS (version 16.04 LTS) and the servers are connected using Gigabit Ethernet switches. The *Web server* is equipped with a standard HTTP server hosting the segmented video content including a MySQL database for storing all the measurements and metrics. The *network emulation server* comes with a customized Mininet (http://mininet.org/, last access: Feb 7, 2017) environment allowing for network emulation (e.g., bandwidth shaping, delay). It defines a controlled environment for dynamic network configurations fully managed by an external management interface. The *Selenium server* is an open source software testing

ISSN 2305-7254

framework for Web applications (http://www.seleniumhq.org/, last access Feb 7, 2017) which is used to automatically conduct our experiments with different adaptive HTML5 players running within a Web browser (in our case Chrome browser but also various mobile platforms are possible). Finally, the *Web management interface* provides two functions, *i)* one for configuring and conducting the experiments and *ii)* one which includes the player and provides real-time information about the currently conducted experiment. It is accessible from outside the controlled environment and everything else is within a controlled environment in order to avoid any cross-traffic that may influence the experiments.

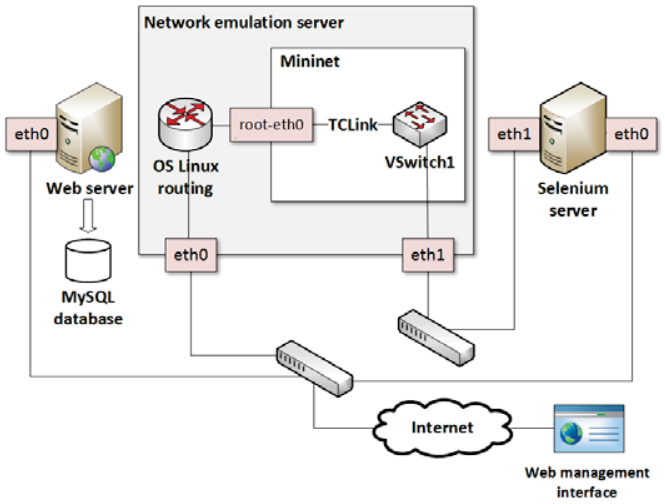

Fig. 1. System Architecture for the Evaluation

III. Evaluation setup

This section provides an overview of the evaluation setup including an overview of the adaptive HTML5 players used for the evaluation, the actual content and network configurations, and the evaluation metrics (objective, subjective) used in the experiments.

A. Overview of adaptive HTML5 players

All players used for this evaluation are implemented in Javascript and utilize the Media Source Extensions (MSE) available on all modern browser platforms. In general, each player has its own application programming interface (API) with methods, properties, and events which are used to obtain all metrics used in this paper. As they are not standardized, it requires a deep knowledge of the underlying technology to enable a comparison among each other. Table I comprises a list of players (in alphabetic order) used in this paper including version and Web site (URL).

B. Content and network configuration

In our evaluation we focus primarily on the streaming performance – in contrast to picture quality – and, thus, we adopted the Big Buck Bunny sequence which is also used in commonly used datasets in this domain [3]. The content is encoded and formatted according to MPEG-DASH

TABLE I. Overview of adaptive HTML5 players

Media player	Version	Web site (last access: Feb 7, 2017)
Bitmovin Player	7.0	https://bitmovin.com
dash.js	2.4.0	http://dashif.org
Flow Player	6.0.5	https://flowplayer.org
HAS Player	1.7	https://github.com/ Orange-OpenSource/hasplayer.js
JW Player	7.6.1	https://www.jwplayer.com
Radiant MP	3.10.8	https://www.radiantmediaplayer.com
Shaka Player	2.0.3	https://github.com/google/shaka-player
VideoJS Player	5.9.2	http://videojs.com

utilizing two different profiles. The first comprises a *FullHD* profile with five different representations: 426x238 pixels (400kbps), 640x360 (800), 854x480 (1200), 1280x720 (2400), and 1920x1080 (4800). For the second configuration we reverse-engineered the *Amazon* Prime video service which offers 14 different representations: 400x224 (100), 400x224 (150), 512x288 (200), 512x288 (300), 512x288 (500), 640x360 (800), 704x396 (1200), 704x396 (1800), 720x404 (2400), 720x404 (2500), 960x540 (2995), 1280x720 (3000), 1280x720 (4500), and 1920x1080 (8000). In both cases we adopted a segment length of four seconds as it provides the best trade-off with respect to streaming performance and coding efficiency [3] which is also used in commercial deployments like Netflix.

The network configuration comprises a bandwidth trajectory adopted from [11] providing both step-wise and abrupt adjustments in the available bandwidth to properly test all adaptive HTML5 players and its adaptation behavior under different conditions. It uses the following sequence: 750 kbps, 350 kbps, 2500 kbps, 500 kbps, 700 kbps, 1500 kbps, 2500 kbps, 3500 kbps, 2000 kbps, 1000kbps and 500 kbps. Such a scheme of bandwidth trajectory inevitably causes quality switches of MPEG-DASH streams used in our experiments. The network delay parameter was set to 70 milliseconds which corresponds to what can be observed within long-distance fixed line connections or reasonable mobile networks and, thus, is representative for a broad range of application scenarios. Finally, the duration of each experiment was set to 630 seconds.

C. Evaluation metrics

This subsection provides an overview of the metrics which are used for comparing the experimental results of the different adaptive HTML5 players. We cluster the metrics in *i)* those directly retrieved from the players' API, *ii)* those which are derived from the raw metrics, and *iii)* those used to predict the QoE score based on a given model.

Download video bitrate (or selected video quality) — the bitrate of the currently requested/downloaded video segment in kbps (higher is better).

Video buffer length (or video buffer level) — the value of buffered video data in seconds (depending on the policy, use case, higher/lower is better).

Video startup time — the time when the player enters the play state (i.e., when clicking the play button or autoplay is enabled) to the event which is fired when the current playback time has changed. For example, for the Bitmovin Player, the `onPlay` event is fired when the player enters the play state and the `onTimeChanged` is fired the first time when the playback

starts. Thus, video startup time (in milliseconds) equals to the timestamp of the `onTimeChanged` event minus timestamp of the `onPlay` event (lower/faster is better).

Stalls (or buffer underruns) — stage of the player playback occurring when the player buffer is getting empty and the playback stops (lower is better, zero is the best).

Quality switches — number of the switches between different representations encountered during the experiment (lower is better).

The second category of metrics are derived from those introduced above and defined as follows. N is the number of the measurements. $W_{i,t}$ is the network bandwidth defined by the network emulator at time t. $b_{i,t}$ is the bitrate selected for the video segment at time t.

Instability [2] — this metric is shown in Equation 1 and evaluates the ratio between the sum of all quality switches observed during the experiment and the sum of all bitrates in the experiment which were selected by the players' rate adaptation algorithm. Lower values of this metric reflect smoother video quality adaptation to the changing network characteristics (lower is better).

Inefficiency [2] — this metric is shown in Equation 2. Smaller values of this metric indicate that the player rate adaptation algorithm more efficiently utilize the available network throughput in order to deliver the media content to the application (lower is better).

Average video bitrate — is shown in Equation 3 and represents the average download video bitrate (or selected video quality) during the entire experiment. It reflects the media throughput delivered to the application (higher is better).

Bandwidth index — is shown in Equation 4 and establish a relationship among the average video bitrate, instability, and inefficiency. The higher the value, the more efficient the available bandwidth is utilized. c is a constant parameter used for fine-tuning and in our experiments c is set to 10^4 based on practical observations (higher is better).

$$\text{Instability} = \frac{\sum_{d=0}^{k-1} |b_{t-d} - b_{t-d-1}|}{\sum_{d=1}^{k} b_{t-d}} \quad (1)$$

$$\text{Inefficiency} = \frac{1}{N} \cdot \sum_{t} \frac{|b_{i,t} - W_{i,t}|}{W_{i,t}} \quad (2)$$

$$\text{Average video bitrate} = \frac{1}{N} \cdot \sum_{t} b_{i,t} \quad (3)$$

$$\text{Bandwidth index} = \frac{\text{Average video bitrate}}{\text{Instability} \cdot \text{Inefficiency} \cdot c} \quad (4)$$

Finally, the third category comprises QoE models that take into account start-up time and stalls as those are not considered in the previously introduced metrics. Together with the bandwidth index they provide means for a detailed analysis of the performance of each player and, more importantly, a comparison thereof. The following two models have been selected due to simplicity and ease of integration into the existing system.

Equation 5 provides the QoE model according to Mäki et al. [5] with number of stalls N_s and the total stalling time $T_{s,total}$. This model has been developed for sequences with a duration of 60s and, thus, we included k to cope with our duration of 630s resulting in $k = 630/60$. In this case we assume that stalls are uniformly distributed over the entire duration of the experiment.

$$QoE_{Mäki} = 4.56 - 0.36 \cdot \frac{N_s}{k} - 0.09 \cdot \frac{T_{s,total}}{k} \quad (5)$$

Equation 6 provides the QoE model from Mok et al. [6] with start-up time T_{init}, stalling frequency f_{rebuf}, and average duration of a stalling event $T_{rebuf} = T_{s,total}/N_s$. In this case we do not need to use k coefficient for f_{rebuf} and for T_{rebuf} because these are relative values.

$$QoE_{Mok} = 4.23 - 0.0672 \cdot T_{init} - 0.742 \cdot f_{rebuf} - 0.106 \cdot T_{rebuf} \quad (6)$$

Both models provide Mean Opinion Score (MOS) values on a scale from 1–5 which translates to bad, poor, fair, good, excellent service quality when adopting an absolute category rating (ACR).

IV. Evaluation Results

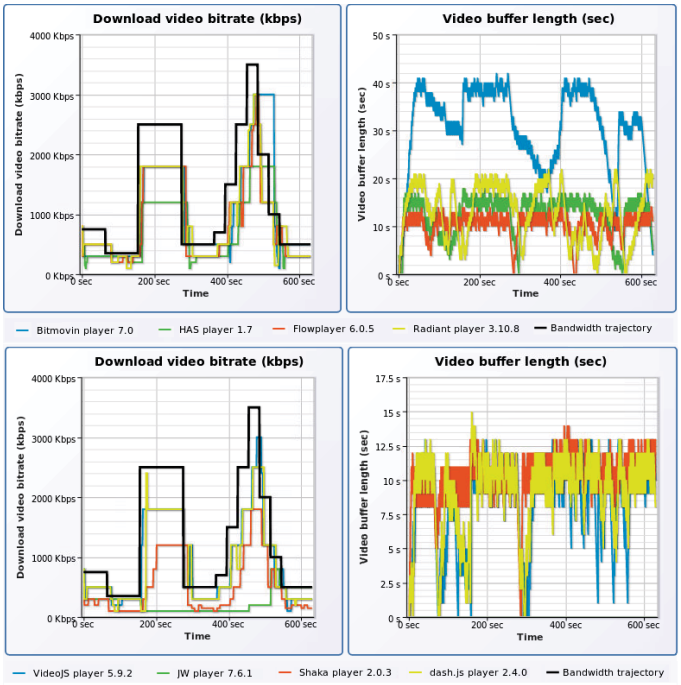

Fig. 2. Download Video Bitrate and Buffer Length for the Amazon Profile

In this section, we present and discuss the results of our evaluation. Each experiment was conducted ten times and the average is presented here (i.e., 160 experiments; each 630s; total duration 28h). An excerpt for the *Amazon* profile is shown in Fig. 2 comprising the download video bitrate and video buffer length of all players under the predefined

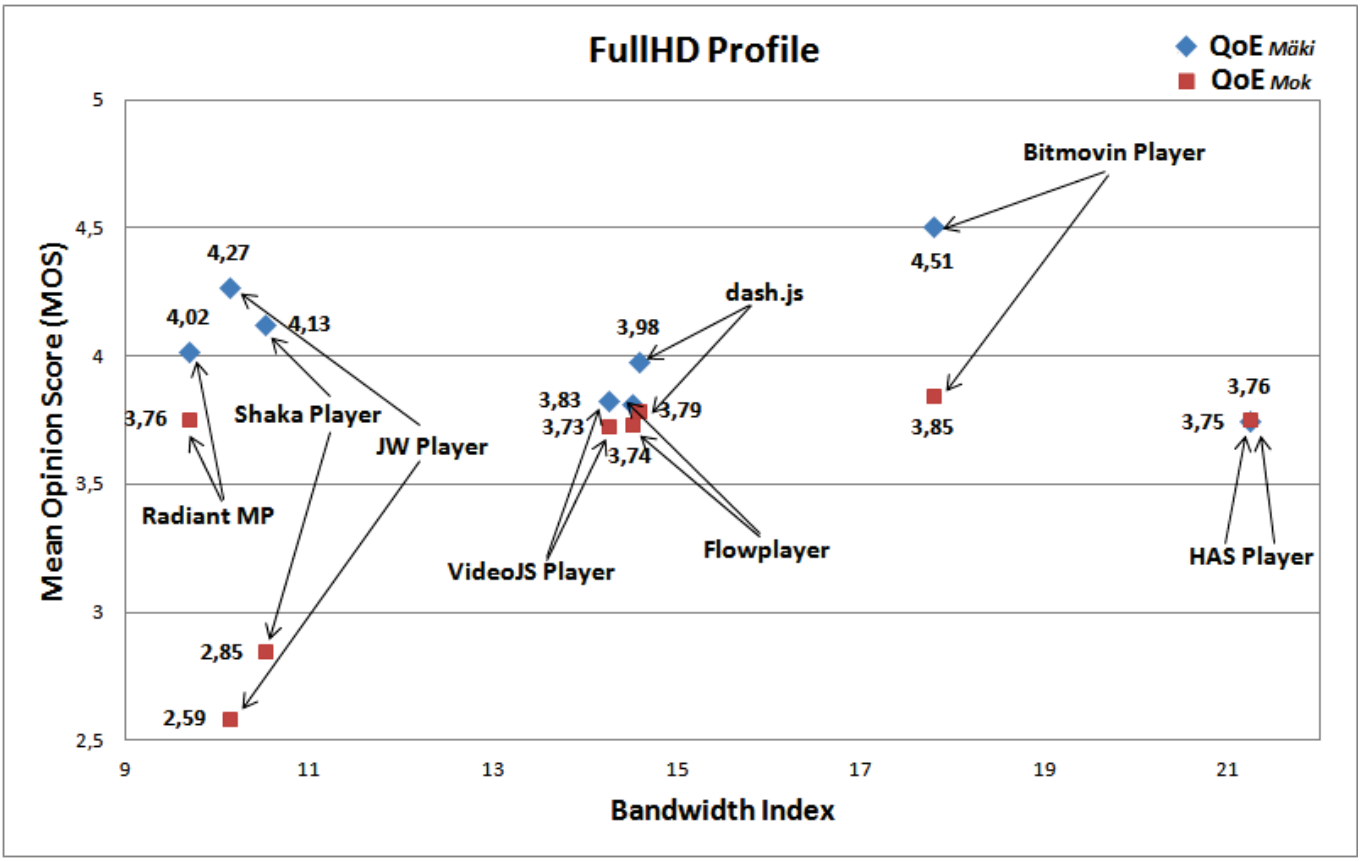

Fig. 3. $QoE_{Mäki}$ (blue) and QoE_{Mok} (red) vs. Bandwidth Index for *FullHD* Content Profile

bandwidth trajectory. It clearly shows that players provide different streaming performances but, in general, all are able to follow and adapt to the available bandwidth. The video buffer length shows that some players produced stalls while others could avoid stalls at all.

In the following we present the results for both content configurations, i.e., *FullHD* and *Amazon*, using QoE values according to the given models ($QoE_{Mäki}$ Equation 5 and QoE_{Mok} Equation 6) on the y-axis and the bandwidth index (Equations 4) on the x-axis.

The results for the *FullHD* content profile is shown in Fig. 3 and the results for the *Amazon* content profile is shown in Fig. 4 respectively.

In general, $QoE_{Mäki}$ always reports higher MOS than QoE_{Mok} which can be explained by the fact that QoE_{Mok} also includes startup time whereas $QoE_{Mäki}$ solely considers stalls. Different players show quite a different behavior with respect to the number and the duration of stalls. In some cases the difference between the two QoE models is higher (e.g., Shaka and JW players) and sometimes it is very close to each other (e.g., HAS player). In this case, Shaka and JW players have much higher startup time than all other players (including HAS player) which explains the (huge) gap.

For both content profiles and QoE models, the Bitmovin player achieves the highest MOS scores as it also has the highest video download bitrate, lowest number/duration of stalls, and lowest startup time compared to all other players. Interestingly, the HAS player has a much higher bandwidth index using the *FullHD* profile than all others due to a very low instability. This low instability results from a low number of switches during playback. However, we noticed also a high number/duration of stalls which inevitably leads to switches during stalls but they are not considered within the instability metric. This is the main reason why HAS player has a higher bandwidth index but lower MOS value. Additionally, the player only performs step-wise switches (up/down) which also contributes to the lower instability.

The bandwidth index for the *Amazon* profile covers a broader range [2..16] than for the *FullHD* profile [10..18] (excluding HAS player) as it offers almost three times more representations which allow for a more fine-grained, flexible adaptation to changing bandwidth conditions.

The interested reader will apparently observe that dash.js, Flowplayer, and VideoJS player form a sort of cluster for both QoE models as they share the same code base. In particular, Flowplayer and VideoJS player as well as Radiant MP and HAS players are all based on dash.js but, interestingly, the latter two show slightly different performances. dash.js is the official reference client of the DASH Industry Forum (DASH-IF, http://dashif.org/) which is open source and allows its usage also for commercial purposes. It is equipped with a pluggable adaptation logic which explains the difference in the performance or, alternatively, Radiant MP and HAS players

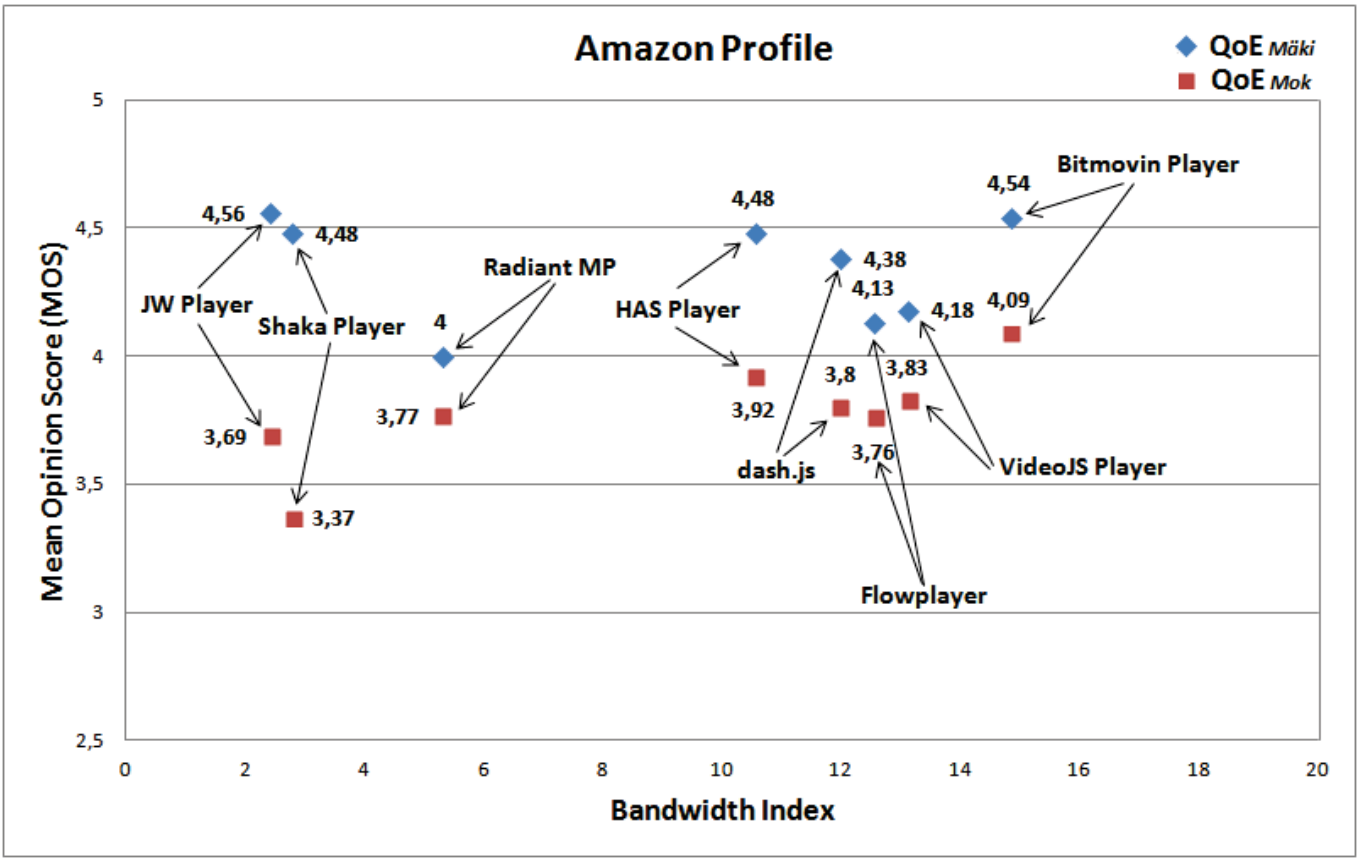

Fig. 4. $QoE_{Mäki}$ (blue) and QoE_{Mok} (red) vs. Bandwidth Index for *Amazon* Content Profile

adopted a different version from dash.js as others. Among the players based on dash.js, Radiant MP shows the lowest performance and we observed that it sometimes requests the highest representation although this does not fit at all into the available bandwidth.

Finally, we observe that the MOS values barely are below three which means that the quality of such a service would be always considered to be fair or good although some players show high number of stalls (> 10), high stall duration ($> 20s$), and high startup time ($> 3s$). Thus, absolute MOS values should be considered very careful but when taking into account the bandwidth index and comparing the results among each other, it provides a relatively good indication about the performance of each player. In general, the number of stalls, stall duration, and startup time is higher for the *FullHD* profile than for the *Amazon* profile which is explained due to the different number of content representations. It is much higher for the *Amazon* profile and, thus, provides more flexibility for the adaptation logic resulting into a lower number of stalls and shorter stall durations. The bitrates at the lower end are also much smaller for the *Amazon* profile which allows for a faster startup time. In practice, the number of representations used for such a service directly depends on business decisions as the higher the number the higher the costs.

For a better understanding of the above conclusions we include detailed results for the number of stalls, stall duration, and download video bitrate for the *Amazon* profile (Fig. 5, Fig. 6, Fig. 7). The download video bitrate of Shaka and JW players is much lower than for others which explains the low bandwidth index. The performance of dash.js-based players is similar except that Radiant MP, Flowplayer, and VideoJS players produce more and longer stalls whereas HAS player performs better than dash.js regarding stalls but has a lower download video bitrate than others. The Bitmovin player has the highest download video bitrate and almost zero stalls which is also reflected in the MOS.

V. Conclusions

In this paper we presented our evaluation system for adaptive HTML5 players which is flexible and easy to use, new players (and also algorithms) can be integrated easily, and it can be used for the automated performance testing on various platforms (i.e., operating systems, browsers, mobile, desktop, etc.). We conducted a series of experiments with various players which are fully deployed in existing applications and services. We evaluated the results adopting *i)* objective metrics, such as bitrate, stalls, startup delay, *ii)* derived/calculated metrics (instability, inefficiency, average bitrate), and *iii)* proposed *a)* the bandwidth index – taking into account the aforementioned derived/calculated metrics – together with *b)* existing QoE models – focusing on stalls, startup delay – as a main tool to determine the streaming performance of each player. The findings presented in this paper demonstrate the applicability of these metrics for the evaluation of adaptive media players. Overall, the Bitmovin player offered the best

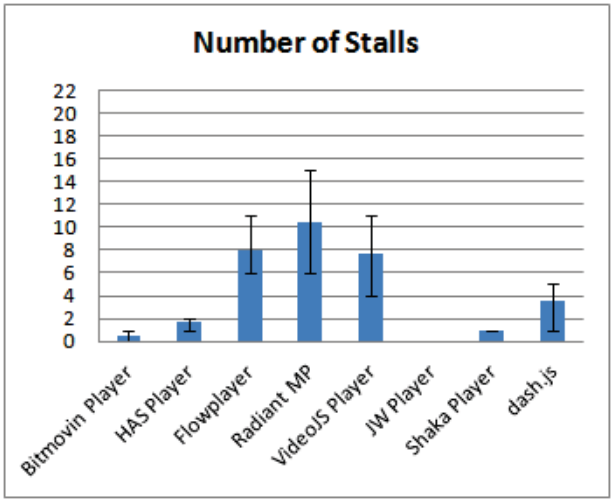

Fig. 5. Number of Stalls for the *Amazon* Profile

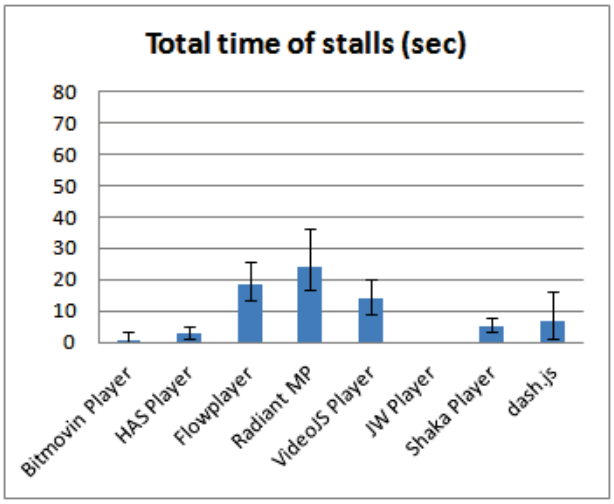

Fig. 6. Total Time of Stalls for the *Amazon* Profile

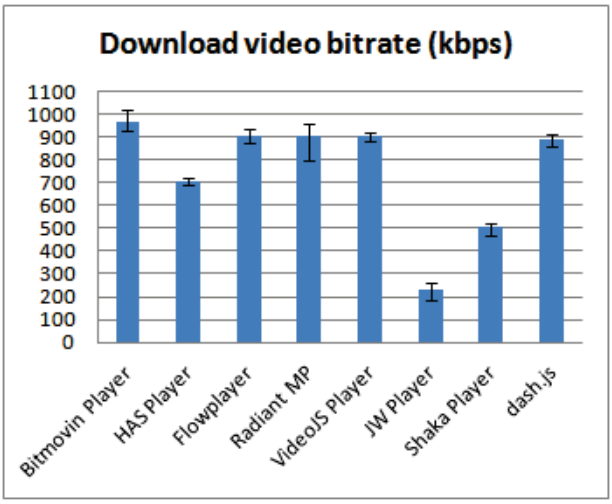

Fig. 7. Download Video Bitrate for the *Amazon* Profile

performance followed by those players which are based on the DASH-IF reference client dash.js.

Future work in this context will include new players (as they become available) and also proponents proposing new algorithms are invited to provide a Javascipt implementation to allow for effective comparison with state-of-the-art deployed systems. Additionally, we will investigate further QoE models and subjective user studies (using crowdsourcing) in order to verify the results and fine-tune the models itself. Finally, we will investigate whether and how these players perform under competition with respect to fairness.

VI. Acknowledgments

This work was supported in part by the Austrian FFG AdvUHD-DASH and PROMETHEUS projects.

References

[1] 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. In *Proceedings of the 2014 ACM Conference on SIGCOMM*, SIGCOMM '14, pages 187–198, New York, NY, USA, 2014. ACM.

[2] J. Jiang, V. Sekar, and H. Zhang. Improving Fairness, Efficiency, and Stability in HTTP-based Adaptive Video Streaming With Festive. *IEEE/ACM Trans. Netw.*, 22(1):326–340, Feb. 2014.

[3] S. Lederer, C. Müller, and C. Timmerer. Dynamic Adaptive Streaming over HTTP Dataset. In *Proceedings of the 3rd Multimedia Systems Conference*, MMSys '12, pages 89–94, New York, NY, USA, 2012. ACM.

[4] Z. Li, X. Zhu, J. Gahm, R. Pan, H. Hu, A. C. Begen, and D. Oran. Probe and Adapt: Rate Adaptation for HTTP Video Streaming At Scale. *IEEE Journal on Selected Areas in Communications*, 32(4):719–733, April 2014.

[5] T. Mäki, M. Varela, and D. Ammar. A Layered Model for Quality Estimation of HTTP Video from QoS Measurements. In *2015 11th International Conference on Signal-Image Technology Internet-Based Systems (SITIS)*, pages 591–598, Nov 2015.

[6] R. K. P. Mok, E. W. W. Chan, and R. K. C. Chang. Measuring the Quality of Experience of HTTP Video Streaming. In *12th IFIP/IEEE International Symposium on Integrated Network Management (IM 2011) and Workshops*, pages 485–492, May 2011.

[7] R. Roverso, S. El-Ansary, and M. Högqvist. On HTTP Live Streaming in Large Enterprises. In *Proceedings of the ACM SIGCOMM 2013 Conference on SIGCOMM*, SIGCOMM '13, pages 489–490, New York, NY, USA, 2013. ACM.

[8] M. Seufert, S. Egger, M. Slanina, T. Zinner, T. Hofeld, and P. Tran-Gia. A Survey on Quality of Experience of HTTP Adaptive Streaming. *IEEE Communications Surveys Tutorials*, 17(1):469–492, Firstquarter 2015.

[9] I. Sodagar. The MPEG-DASH Standard for Multimedia Streaming Over the Internet. *IEEE MultiMedia*, 18(4):62–67, 2011.

[10] T. C. Thang, H. T. Le, A. T. Pham, and Y. M. Ro. An Evaluation of Bitrate Adaptation Methods for HTTP Live Streaming. *IEEE Journal on Selected Areas in Communications*, 32(4):693–705, April 2014.

[11] C. Timmerer, M. Maiero, and B. Rainer. Which Adaptation Logic? An Objective and Subjective Performance Evaluation of HTTP-based Adaptive Media Streaming Systems. *arXiv.org [cs.MM]*, abs/1606.00341:11, jun 2016.

[12] X. Yin, A. Jindal, V. Sekar, and B. Sinopoli. A Control-Theoretic Approach for Dynamic Adaptive Video Streaming over HTTP. In *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication*, SIGCOMM '15, pages 325–338, New York, NY, USA, 2015. ACM.

[13] A. Zabrovskiy, E. Petrov, E. Kuzmin, and C. Timmerer. Evaluation of the Performance of Adaptive HTTP Streaming Systems. *ArXiv e-prints*, Oct. 2017.