# Empirical Evaluation of MDP-based DASH Player

Ayub Bokani, S. Amir Hoseini, Mahbub Hassan, Salil S. Kanhere
School of Computer Science and Engineering, The University of New South Wales, Sydney, Australia
Email: {abokani, amirhoseini, mahbub, salilk}@cse.unsw.edu.au
National ICT Australia (NICTA)

***Abstract*—Dynamic Adaptive Streaming over HTTP (DASH) is one of the most widely used adaptive streaming technique for watching online video content. DASH adapts to the varying network conditions by selecting the appropriate bitrate of the video stream. The bitrate adaptation is typically done by monitoring the playback buffer level and/or the network condition on client side. In this paper we empirically evaluate our *JavaScript* DASH player in which, Markov Decision Process (MDP) has been considered as the underlying optimization framework. This player uses Q-learning algorithm to learn the model and optimize the Quality of Service (QoS) after multiple streaming sessions. The *basic JavaScript* DASH player developed by DASH Industry Forum (DASHIF) is used as a benchmarking model in our evaluations. We use Google Chrome's 3G and 4G network emulators in our experiments and show that our MDP-based DASH player significantly outperforms the DASHIF player which uses *buffer control* and *rate adaptation* techniques simultaneously. Using real-time experiments, we show that for similar picture quality we can achieve about 18x fewer deadline misses and 5x fewer quality switches over a 3G network and 32x fewer deadline misses and 1.6x fewer quality switches over 4G.**

## I. Introduction

In recent few years we have witnessed an enormous growth of mobile data traffic all around the world. According to CISCO's report [1], at the end of 2014, the global mobile data traffic grew 69% compare to 2013. Considerable growth of cellular connection speed (e.x., 20% from 2013 to 2014) as well as the increasing capability of smart devices is enabling consumers to watch more multimedia contents while on move. Based on this report, the mobile video traffic exceeded 50% of the total mobile data traffic in 2012 and it is expected to increase to 72% by 2019.

Despite the improvements in peak data rates, the received mobile signal strength and therefore the available bandwidth in different times and locations are unstable and unpredictable. This severely affects the QoS of multimedia streaming on mobile devices, thus presenting an important challenge for content providers and network operators. To tackle this problem, a new standard for video streaming has recently emerged called Dynamic Adaptive Streaming over HTTP (DASH) [2]. The main concept of DASH is to encode the video files using multiple quality levels (bitrates) and store them as a series of small chunks which are typically 2-10 seconds in length. To maximize the QoS under varying network conditions, the streaming client fetches the most appropriate quality level for a given video chunk by using the standard HTTP GET command. An incorrect quality selection may cause deadline miss (if the available bandwidth cannot support the requested bitrate) or result in a lower quality stream being played out (if the available bandwidth is significantly greater than the chosen bit rate). It therefore becomes the DASH client's responsibility to dynamically select the 'right' quality for the next video chunk in order to have a smooth playback with the highest possible quality while also minimizing frequent quality switches. Providing such a streaming strategy, i.e., the client intelligence for maximizing the quality of experience (QoE) is left to the developers.

Most of the state-of-the-art streaming strategies [3], [4] monitor the client's buffer level and network throughput for selecting the appropriate streaming quality. Although these methods perform reasonably well, they do not optimise the trade-off between individual QoE metrics such as picture quality vs deadline miss, especially in vehicular environments which exhibit significant uncertainty in network bandwidth.

The primary goal of this paper is on extensive performance evaluation of our proposed DASH player in [5]. This model enables the streaming client to learn from experience how to select the most appropriate quality level for next video chunk in different circumstances. Q-learning [6] which is a well known model-free reinforcement learning technique to solve any given finite Markov decision process (MDP) [7] has been used as the underlying optimization framework in our player. In this model, after receiving each video chunk, consequence of selecting the latest quality level is recorded in a table called *Q-table*. Such a massive knowledge is then used to select the best possible quality in different conditions. We demonstrate that in practice and after Q-table is converged, our player outperforms the *basic* DASH player by missing 18x and 32x less deadlines on 3G and 4G networks respectively while minimizes the quality fluctuates by factors of 5 and 1.6.

The rest of the paper is organised as follows. Related works are discussed in Section II. In Section III we discuss the *basic JavaScript* DASH player and how we modified it to use MDP streaming strategies. In Section IV, we explain our experiment setup and the results are presented in Section V. We finally conclude the paper in Section VI.

## II. Related works

Improving the quality of service (QoS) of HTTP-based adaptive video streaming has been the main focus of many researchers in recent few years. Using sender-driven rate adaptation [8] is one of strategies that has been frequently explored

978-1-4673-9348-5/15/$31.00 ©2015 IEEE

by researchers. Our study is related to the receiver-driven rate adaptation [4] which enables the client devices to control the streaming quality. With minor differences, the majority of such strategies consider certain buffer level thresholds for switching the quality level *Up* or *Down*. As such, these strategies are often slow to react to sudden changes in network conditions. In this paper we consider not only the buffer level occupancy but also other effective parameters such as number of deadline miss and quality changes to maximize the streaming quality.

Researchers used online network observation or historical bandwidth traces to predict or estimate the available throughput during fetching the next video chunk. In [9] and [10], researchers found that bandwidth can vary severely in different locations. In our previous studies we also found using location-based bandwidth statistics significantly improves the QoS in video streaming [11], [12]. However, none of these approaches considered learning from streaming sessions that is used in this paper.

In [13] and [14], researchers have modelled the decision making system in adaptive multimedia streaming as MDP problem. Cuetos et al. [15] also used MDP to optimise the scheduling and error concealment simultaneously in layered video. However, non of these studies used Q-learning algorithm to solve their MDP models which enables the streaming clients to learn from multiple streaming sessions and optimize the streaming performance.

In this paper, we discuss our implemented MDP-based *JavaScript* DASH player as well as the empirical testing results. We use Google Chrome's network emulator to empirically evaluate our DASH player and demonstrate how it significantly outperforms the *basic* DASH-player which uses *buffer control* and *rate adaptation* techniques for the bitrate adaptation.

## III. JavaScript DASH Players

The DASH Industry Forum introduced a *JavaScript* DASH player in which the optimization is done based on monitoring the available bandwidth as well as buffer occupancy level [16] at the client. In this work, We have modified the quality selection module of this player in a different way to employ our MDP-based rule as well as for benchmarking other DASH players (i.e., random and deterministic quality selection). In the following we briefly explain how the *basic* and MDP-based DASH players work.

### A. Basic DASH player

The algorithm used by DASH Reference Player considers two parameters for making decisions on bitrate adaptation: *Buffer level* and possible *download ratio*. In this player, a download ratio $\mu$ is computed by dividing the video chunk duration *VCD* by its fetching time *CFT* multiplied by a constant called safety factor ($\phi$) as:

$$\mu = \frac{VCD}{CFT} \times \phi \tag{1}$$

The safety factor $\phi$ which can vary between 0 and 1 is set to 0.75 by default to ensure that player behaves more conservatively and there is sufficient bandwidth before initating the *switch up* action. If the resulting $\mu$ is greater than 1, then the algorithm will decide to switch up to a higher bit rate. Otherwise it will initiate the *switch down* action. In addition, the player has to avoid an interruption to the stream (i.e. deadline miss). Thus regardless of the download ratio, if the buffer level is less than a certain threshold, the player will chose to take the *switch down* action.

### B. MDP-based DASH Player

The goal of MDP is to obtain a policy which returns the best possible action to take when a particular state is observed. To obtain such an optimal policy, we consider Q-learning algorithm in this work. To get better insights of our model, we first explain MDP formulation of DASH then we discuss how Q-learning model enables the streaming clients learn from multiple streaming sessions to maximize their QoE.

*1) MDP Formulation of DASH:* Our MDP model is characterized by a tuple $(S, A, R_{sa}, \gamma)$ [1], where $S$ is a set of *states*, $A$ is a set of *actions*, $R_{sa}$ is the immediate *revenue* for taking action $a$ in state $s$, and $\gamma = [0, 1)$ is a discounting factor for the revenues collected from future actions and states.

*System states and decision timings:* We observe the system state when a video chunk is completely downloaded. The system state $S(\rho, q)$ is jointly represented by the quality level ($q$) of the downloaded chunk and the amount of time available ($\rho$) before its playback deadline. There is a deadline miss if the chunk download is not completed before its deadline ($\rho < 0$), in which case the video is frozen for a while until the chunk is downloaded, and it is played immediately at that time. Therefore, for a deadline miss, $\rho$ is considered zero instead of negative.

If there is not enough space remaining in the buffer for another chunk after storing a downloaded chunk, the decision making and start of downloading the next chunk stall until there is enough room in the buffer. The value of $\rho$ therefore assumes the value at the time of decision making (when there is space in the buffer) instead of when the last chunk was downloaded. This provides an upper bound for $\rho$, which is basically controlled by the buffer length. For example, if we have a buffer with a capacity to hold 7 chunks each 2 seconds long, then the upper bound for $\rho$ is 14 seconds. Note that chunks have different sizes based on the quality while in most practical systems a buffer will have a maximum size in terms of bytes. One way to address this issue would be to configure the buffer length in bytes using the maximum chunk size, but measure buffer occupancy in units of chunks stored in the buffer.

Although $\rho$ is a continuous number between zero and upper bound, we propose to use a discrete interval system to achieve a finite number of MDP states. We divide each second into $n$ intervals and use an integer to represent the value of $\rho$. For example, for a 7-chunk buffer holding only 2-sec chunks, $n = 2$ would give us $7 \times 2 \times 2 = 28$ different values for $\rho$.

*Actions:* At each state, the decision taken is referred to as an action. For our adaptive HTTP streaming system, an action is

[1] Transition probability is not required with our model-free MDP.

basically a decision about the quality level for the next chunk. If we have $N$ quality levels to choose from, then we have $N$ possible actions.

*Revenue function:* The *Revenue function* $R(s, a, s')$ uses some rewards and penalties to evaluate the outcome when action $a'$ is chosen at state $(s, a)$:

$$R(s, a, s') = u(a') - D \tag{2}$$

where $u(a')$ is a reward to watch a chunk in quality $a'$ and $D$ is a penalty if a deadline is missed [5]. Note that $u(a')$ can be derived immediately from predefined tables without observing or knowing the next state. The deadline miss penalty $D$, however, is driven after observing the next state as follows:

$$D = \begin{cases} 0 & if \quad no\ deadline\ miss \\ D_M & otherwise \end{cases} \tag{3}$$

where $D_M$ is a constant that can be used to tune the MDP model.

*2) Q-learning Algorithm:* In this technique, a $Q$ matrix which defines the values of every state $s$ if action $a$ is taken is initialised with zero [17]. Whenever the DASH client starts out in state $s$, takes action $a$, and ends up in state $s'$, it updates $Q(s, a)$ as:

$$Q(s, a) \leftarrow (1 - \alpha)Q(s, a) + \alpha[R(s, a, s') + \gamma max_a Q(s', a')] \tag{4}$$

where $\alpha$ is the learning rate. When state $s$ is observed, the $Q$ matrix, or the particular row in the matrix for state $s$, is used to make the decisions in the following way: choose the action that provides the maximum value based on the current estimates in $Q$ for *most of the time*, but a *random action the rest of the time*. We use the *Boltzmann distribution* function [17] to make a balance between maximum value decision (exploitation) and random decision (exploration):

$$f(s, a) = \frac{e^{Q(s,a)/\theta}}{\sum_j e^{Q(s,a_j)/\theta}} \tag{5}$$

where $f(s, a)$ is the probability of selecting action $a$ when in state $s$, and $\theta$, often referred to as temperature, controls the degree of randomness in choosing an action. With large $\theta$, actions will be fairly randomly selected irrespective of the values in $Q$, but for a small $\theta$ closer to zero, the best action based on $Q$ values will be selected with high probabilities. For a DASH client it is useful to begin with a large $\theta$ so the client makes mostly random decisions and learns from its observations. As it continues to learn, the $Q$ values are used more often to make the decisions. Our player's rule is defined based on Algorithm 1.

**Algorithm 1**

1: Initialize $Q$ to zero
2: Download the first chunk using the lowest quality and observe next state
3: REPEAT
4: Update $Q$
5: Calculate Boltzmann probability and use it to select quality $q$ of next chunk
6: Download the next chunk using quality $q$ and observe next state
7: CONTINUE

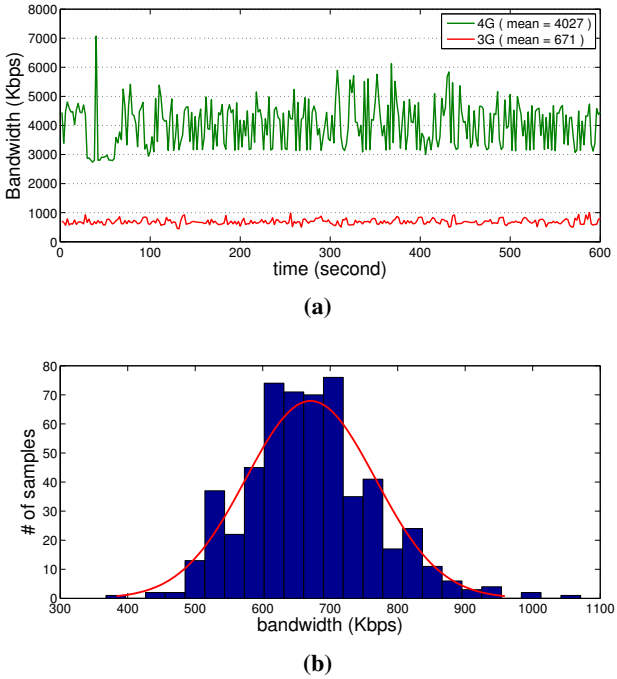


**Fig. 1:** Bandwidth monitoring on Google Chrome's network emulator: Regular 3G ($\mu = 671$, $\sigma = 95.9$) and regular 4G ($\mu = 4027$, $\sigma = 760.7$) - bandwidth fluctuations (a) , normality check (b)

## IV. Experiments Set-up

### A. Network Emulation

Google Chrome's developer mode provides a network emulator which allows to emulate a variety of network connections, including DSL, 3G, and even offline. There are several connection type choices to apply network throttling and latency manipulation. Throttling artificially limits the maximum download throughput and latency manipulation applies a minimum delay in connection (RTT). Fig. 1 illustrates our observed bandwidth statistics while using emulated 3G and 4G models in our experiments. The bandwidth fluctuation in both models presented in Fig.1a and Fig. 1b show that bandwidth samples are normally distributed.

### B. Video Statistics

To test our MDP-based DASH player, we chose to use the existing manifest files provided by DASHIF [16]. Table I presents the available bitrates and resolutions of the two video clips *Big Buck Bunny* (in 10 qualities) and *Envivo* (in 5 qualities) that are used in our experiments over 4G and 3G networks respectively.

**TABLE I:** Available quality levels

| | Big Buck Bunny | | | Envivo | | |
|---|---|---|---|---|---|---|
| **Q** | **Kbps** | **Width** | **Height** | **Kbps** | **Width** | **Height** |
| 1 | 230 | 1920 | 1080 | 350 | 320 | 180 |
| 2 | 331 | 1600 | 900 | 600 | 480 | 270 |
| 3 | 477 | 1280 | 720 | 1000 | 704 | 396 |
| 4 | 688 | 992 | 560 | 2000 | 1024 | 576 |
| 5 | 991 | 768 | 432 | 3000 | 1280 | 720 |
| 6 | 1427 | 592 | 332 | | | |
| 7 | 2056 | 448 | 252 | | | |
| 8 | 2962 | 368 | 208 | | | |
| 9 | 5027 | 284 | 160 | | | |
| 10 | 6000 | 224 | 128 | | | |

**TABLE II:** Reward function

| *quality level (q)* | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** |
|---|---|---|---|---|---|---|---|---|---|---|
| $u(q)$ | 10 | 20 | 30 | 40 | 50 | 60 | 70 | 80 | 90 | 100 |

### C. MDP Parameters

Recall that the tunable MDP parameters allow us to customize the streaming quality. In our experiments we used a deadline miss penalty ($D$) to penalize the *state/action* pairs that cause deadline misses. In our previous study [18] we presented a wide range of results by tuning $D$ in order to have fair comparison with different methods. In this study, we first run the *basic* DASH player. Then, we run multiple tests with different $D$ values in our MDP-based player to find such a setting to achieve Average Quality (AQ) similar to *basic* DASH player. By fixing the AQ from both players, we can compare them based on their total quality changes and deadline misses. We keep the reward parameters fixed as shown in Table II, returning higher values for taking higher actions to encourage the Q-learning algorithm to maximize the AQ. Other MDP parameters are set as following:

$D$=100, $N$ = 10 & 5, $M$ = 6, T = 2 [2], $n$ = 1, $\alpha = 0.9, \gamma = 0.9$, $\theta$ is initiated with 15 and $\epsilon = 0.005$.

## V. Results

In this section we demonstrate the efficiency of MDP-based DASH player by presenting the empirical results. We consider the buffer level, number of quality changes and the number of deadline misses as evaluation metrics. We also use *deterministic* and *random* quality selection strategies as benchmarks.

### A. 3G

We first present results for experiments conducted over the emulated 3G network with average bandwidth of 671 Kbps. Note that the lower the buffer level, the higher the deadline miss risk becomes. Therefore, we compare the buffer level while streaming the *Envivo* video clip for all 4 strategies in Fig. 2 . As observed from Fig. 2a, with the *deterministic* quality selection, selecting Q1 and Q2 for the entire streaming session will result in the maximum possible buffer level (12 seconds) for most of the times. With Q3 there is a very high risk for deadline miss most of the times while with higher qualities of Q4 and Q5 the player misses every playback deadlines.

In Fig. 2b, we can see the *random* strategy achieves similar result as *deterministic* quality 3 since the average quality with *random* method is close to 3. However, with both *basic* and MDP-based DASH players we can observe considerably higher buffer levels compare to the ones from *random* and *deterministic* quality 3 though their average qualities are also around 3 (3.09 and 2.78 , Fig 3a). The highest buffer level that is achieved with our MDP-based DASH player indicates lower deadline miss risk compare to the benchmarking algorithms. Presented results are the average over 10 different experiments.

In Fig. 3, we compare the *basic* DASH and MDP-based DASH players over three quality dimensions in 10 separate experiments. Recall that we fix the AQ of these players to evaluate them based on their number of quality changes and deadline misses. We found that for very close AQs, the MDP-based player significantly decreases the number of deadline misses (1.3 from 24 DM, i.e., 18.5x reduction) (Fig. 3b, f). As it's shown in Fig. 3c, the quality fluctuation is also significantly minimized with our player. On average of 10 tests, the *basic* DASH player switches the streaming quality 46.1 times when MDP-based player only switches 8.1 times (i.e., 5.6x less). It can be also captured that the *basic* DASH player does not provide consistent streaming performance in different tests compare to our MDP-based player. This can be explained as the bitrate adaptation in *basic* DASH player is done based on network and buffer level conditions which can be different in each separate test.

### B. 4G

Next, we present results for an emulated 4G network (average bandwidth 4027 Kbps). In Fig. 4 we can see *deterministic* qualities of Q9 and Q10 drain the buffer while choosing other qualities there is lower risk of deadline miss. With *random* quality selection strategy we always achieved a high buffer level. Similar to 3G, MDP-based DASH player guarantees higher buffer level compared to *basic* DASH player while their AQ are very close (7.3 and 7.6).

As we can see from Fig. 5, our player misses only 1.4 deadlines on average of 10 tests which is 32.6 times less than 45.7 deadline misses from *basic* DASH player. In term of quality change, although our MDP-based DASH player fluctuates the quality 81 times on average of 10 runs which is more than what we achieved over 3G network, yet it's significantly less than *basic* DASH player with 130 times quality changes meaning 1.6x less fluctuation. All the results are achieved over 10 different tests.

## VI. Conclusions

Using the publicly available *JavaScript* DASH player, we have created an MDP-based player which enables the video player to learn from multiple streaming sessions. We used Google Chrome's network emulator to empirically evaluate our player. We have found that comparing to the *basic* DASH player, our player can reduce the number of deadline misses by a factor of 32 over 3G and a factor of 18 over 4G networks. Our

[2] $M$ = 6 & $T$ = 2 $\rightarrow$ buffer length = 12 second

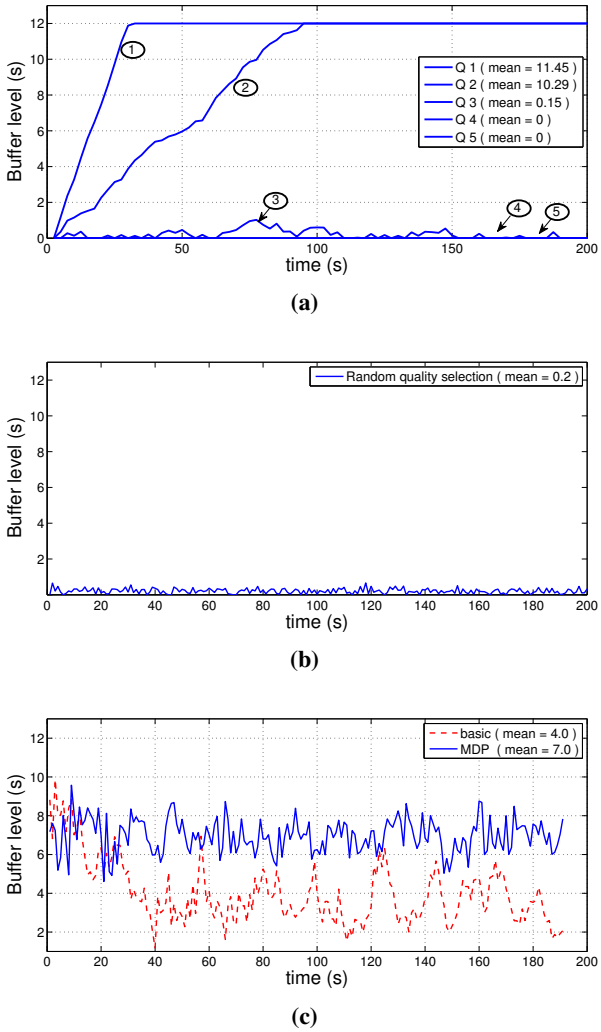


**Fig. 2:** Buffer level (average over 10 separate experiments): (a) Deterministic, (b) Random, (c) *basic* and MDP-based DASH players - Video clip: *Envivo*, 5 available qualities ; Network Emulator: Regular 3G

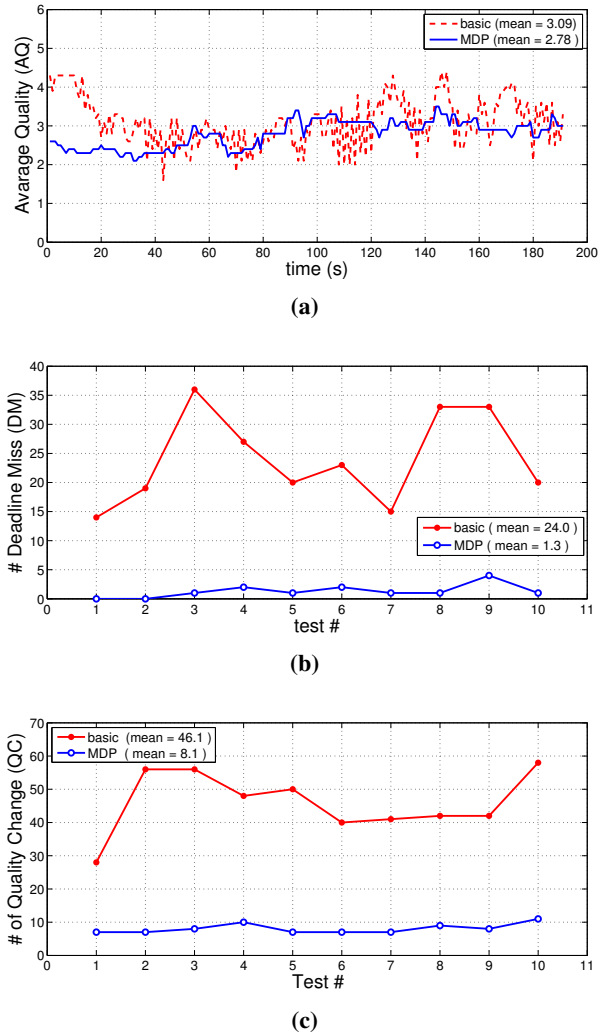


**Fig. 3:** Comparing basic and MDP-based DASH players: (a) Average picture quality, (b) Deadline misses, (c) Quality change - Network emulator: 3G

DASH player also decreases the amount of quality fluctuations of basic DASH player 5x and 32x over 3G and 4G networks respectively. In next step we will evaluate the performance of our MDP-based DASH player in real road traffic over real mobile networks.

## REFERENCES

[1] Cisco, "Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update 20142019 White Paper," [Online accessed 04-June-2015], URL: http://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/white_paper_c11-520862.html.

[2] T. Stockhammer, "Dynamic Adaptive Streaming Over HTTP: Standards and Design Principles," in *Proceedings of the second annual ACM conference on Multimedia systems (MMSys)*, USA, 23 Feb 2011.

[3] T.-Y. Huang, R. Johari, and N. McKeown, "Downton Abbey Without the Hiccups: Buffer-Based Rate Adaptation for HTTP Video Streaming," in *Proceedings of the 2013 ACM SIGCOMM workshop on Future human-centric multimedia networking*, Hong Kong, China, 16 August 2013.

[4] C. Liu, I. Bouazizi, and M. Gabbouj, "Rate Adaptation for Adaptive HTTP Streaming," in *Proceedings of the second annual ACM conference on Multimedia systems (MMSys' 11)*, New York, USA, 23 February 2011.

[5] A. Bokani, M. Hassan, S. Kanhere, and X. Zhu, "Optimizing HTTP-Based Adaptive Streaming in Vehicular Environment using Markov Decision Process," *IEEE Transactions on Multimedia (ACCEPTED)*, 2015.

[6] C. J. Watkins and P. Dayan, "Q-Learning," *Machine learning*, vol. 8, no. 3-4, pp. 279–292, 1992.

[7] M. L. Puterman, *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. Wiley.com, 2009, vol. 414.

[8] L. Lam, J. Y. Lee, S. C. Liew, and W. Wang, "A Transparent Rate Adaptation Algorithm for Streaming Video Over the Internet," in *The*

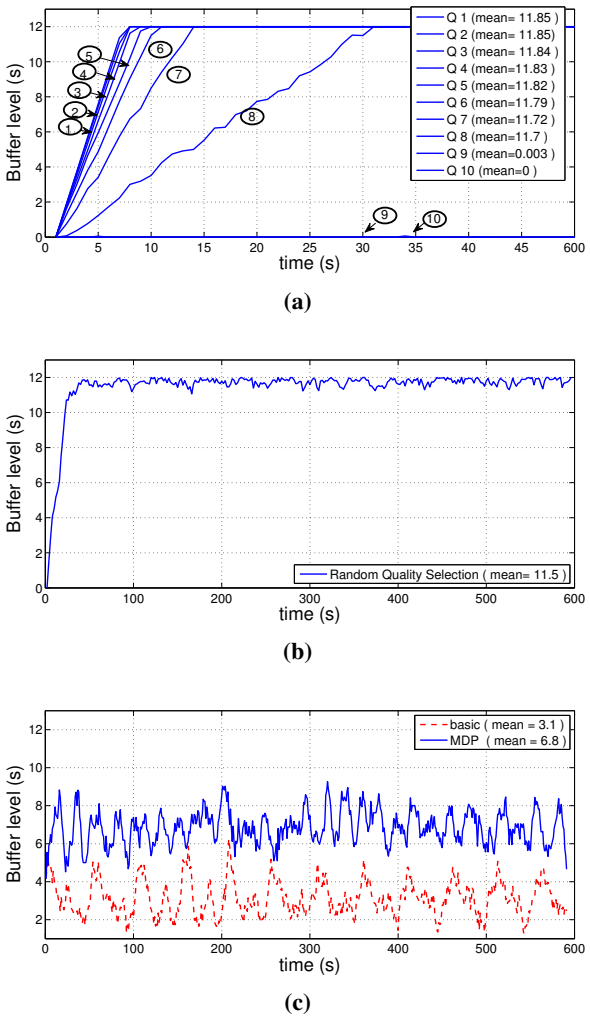


**Fig. 4:** Buffer level (average over 10 separate experiments): (a) Deterministic, (b) Random, (c) *basic* and MDP-based DASH players - Video clip: *Big Buck Bunny*, 10 available qualities ; Network Emulator: Regular 4G

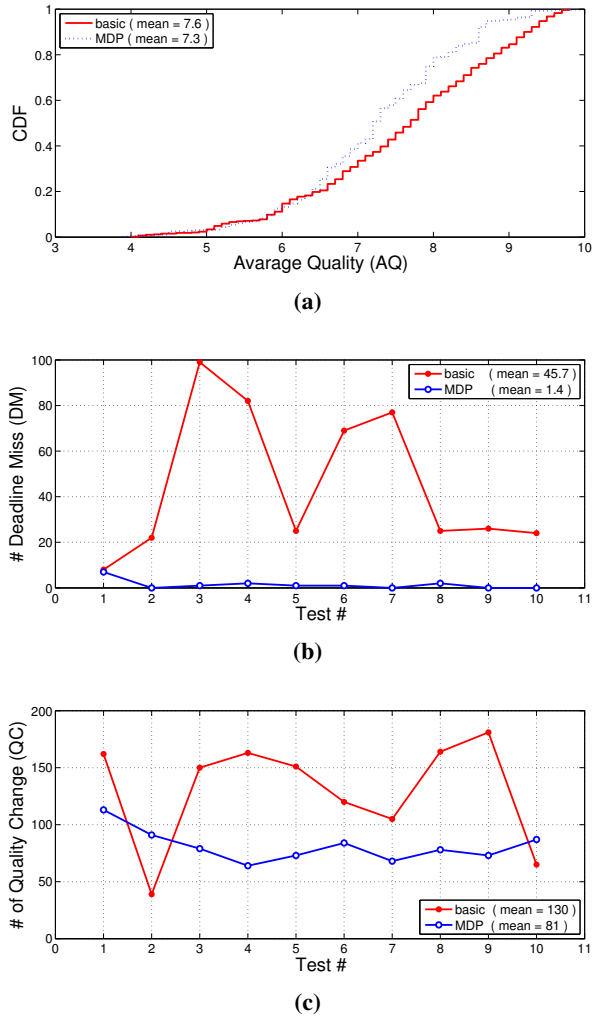


**Fig. 5:** Comparing basic and MDP-based DASH players: (a) Average quality, (b) Deadline misses, (c) Quality change - Network emulator: Regular 4G

*18th International Conference on Advanced Information Networking and Applications (AINA)*, Fukuoka, Japan, 29-31 MArch 2004.

[9] J. Yao, S. S. Kanhere, and M. Hassan, "An Empirical Study of Bandwidth Predictability in Mobile Computing," in *Proceedings of the third ACM international workshop on Wireless network testbeds, experimental evaluation and characterization (MobiCom-WiNTECH)*, San Francisco, USA, 14-19 September 2008.

[10] P. Deshpande, X. Hou, and S. R. Das, "Performance Comparison of 3G and Metro-Scale WiFi for Vehicular Network Access," in *Proceedings of the 10th ACM conference on Internet measurement*, Melbourne, Australia, 13 November 2010.

[11] A. Bokani, "Location-Based Adaptation for DASH in Vehicular Environment," in *Proceedings of the 2014 CoNEXT on Student Workshop ACM '2014*, Sydney, Australia, 2-5 December 2014.

[12] G. Zhong and A. Bokani, "A Geo-Adaptive JavaScript DASH Player," in *Proceedings of the VideoNEXT '2014 Workshop on Design, Quality and Deployment of Adaptive Video Streaming ACM '2014*, Sydney, Australia, 2-5 December 2014.

[13] C. C. Wüst and W. F. Verhaegh, "Quality Control for Scalable Media Processing Applications," *Journal of Scheduling*, vol. 7, no. 2, pp. 105–117, 2004.

[14] D. Jarnikov and T. Özçelebi, "Client Intelligence for Adaptive Streaming Solutions," *Signal Processing: Image Communication*, vol. 26, no. 7, pp. 378–389, 2011.

[15] P. de Cuetos and K. W. Ross, "Optimal Streaming of Layered Video: Joint Scheduling and Error Concealment," in *Proceedings of the eleventh ACM international conference on Multimedia (MM '03)*, Berkeley, USA, 02-08 November 2003.

[16] D. I. Forum, "JavaScript DASH Player," [Online accessed 01-March-2014], URL: http://dashif.org/reference/players/javascript/1.3.0/samples/dash-if-reference-player/.

[17] C. J. C. H. Watkins, "Learning From Delayed Rewards." Ph.D. dissertation, University of Cambridge, 1989.

[18] A. Bokani, M. Hassan, and S. Kanhere, "HTTP-Based Adaptive Streaming for Mobile Clients using Markov Decision Process," in *Proceedings of the 20th Packet Video Workshop (PV)*, San Jose, USA, 12 Dec 2013.