Evaluation of Q-Learning approach for HTTP Adaptive Streaming

Virginia Martín, Julián Cabrera, and Narciso García

***Abstract*—We propose a Q-Learning-based algorithm for an HTTP Adaptive Streaming (HAS) Client that maximizes the perceived quality, taking into account the relation between the estimated bandwidth and the qualities and penalizing the freezes. The results will show that it produces an optimal control as other approaches do, but keeping the adaptiveness.**

I. Introduction

HAS is becoming a widely adopted solution to deliver video streaming to end-users in highly dynamic networks. The streaming client requests each content segment at the most appropriate quality using its particular adaptation algorithm. The standard Dynamic Adaptive Streaming over HTTP, also known as MPEG-DASH [1], defines the format of the Media Presentation Description and the segment format, however it does not specify the control algorithm for the quality selection. Several approaches have been proposed to solve the quality decision in a scenario with a single HAS client. Miller et al. [2] try to keep the buffer level close to a target value B_{opt}, being its main priority to avoid playback freezes. Also the BBA-0 algorithm [3] controls the buffer occupancy, although it uses a bandwidth estimation as well during the startup phase. Liu et al. [4] compute a smoothed measure of the throughput in order to detect network congestion. Juluri [5] includes the segment size, due to the impact of variable bit rate encoding, in addition to the estimated bandwidth and the buffer filling to predict more accurately the download time of the next segment.

Other research studies model the problem as an optimization control problem and use mathematical tools to solve it. In that sense, García et al. [6] propose an algorithm based on Stochastic Dynamic Programming (SDP), known as DASH-SDP. They compute off-line optimal control policies based on a priori models of the system. Although this scheme produces good results, it lacks of adaptiveness if there is a mismatch between the priori model and the real system behavior. In this paper, we present our adaptation algorithm based on Q-Learning approach, DASH-QL. This reinforcement learning technique [7] allows the client to learn, through its experience, which qualities are the most appropriate in accordance with several environment conditions. Previous researches have addressed the use of these techniques [8] [9]. Our algorithm uses a suitable number of states variables achieving to capture the dynamics of the system. It also employs a novel reward function that combines the main Quality of Experience (QoE) factors: quality of the requested segments, quality switches and freezes. In order to validate our formulation, we compare our control logic with the SDP-based algorithm. The results will show that our method produces an optimal control as the SDP approach, but keeping the adaptiveness of the Q-Learning approach.

This work has been partially supported by the Ministerio de Economía y Competitividad of the Spanish Government under TEC2013-48453 (MR-UHDTV)

II. Description of the DASH-QL algorithm

Q-Learning [7] is a **model-free reinforcement learning technique** that allows a dynamic system to learn which is the most appropriate action for the next stage depending on specific environment conditions at each time. It combines an exploration mode where random actions are selected and the algorithm learns from their outcome, and an exploitation mode in which the logic selects the most suitable action according to what it has learned up to that moment.

A. Elements of the DASH-QL approach

Below we describe the main elements of our formulation:

State, s***:*** In our model, it contains the following information about the environment at each stage k: $s_k = (buf_k, bw_k, q_{k-1})$

- buf_k is the level of the client's buffer in seconds.
- bw_k is the estimated available bandwidth in Kbps.
- q_{k-1} is the bitrate of the previous requested quality in Kbps.

The updating of bw_k (1) and buf_k (2) through the stages is computed as follows:

$$bw_k = \frac{q_k \cdot T_{seg}}{\Delta t_{k-1}} \quad (1) \qquad buf_k = buf_{k-1} + T_{seg} - d_k \cdot T_{seg} \quad (2)$$

T_{seg} is the segment duration and d_k is the number of segments extracted from the buffer during the download time, Δt_k.

Action: The different available qualities of the segments.

Reward function: Indicates how good the decision taken is.

$$r = R_{quality} + R_{switches} + R_{freezes} \quad (3)$$

where:

$$R_{quality} = -1.5 \cdot \left| bw_k \frac{1 + (buf_k / B_{opt})}{3 - (bw_1/q_1)} - q_k) \right| \quad (4)$$

$$R_{switches} = -|q_{k-1} - q_k| \quad (5) \qquad R_{freezes} = -|bw_k \cdot f_l| \quad (6)$$

$R_{quality}$(4), whose coefficients have been empirically selected, aims at favoring the selection of higher qualities, but taking into account the buffer level in relation to its optimal value

(B_{opt}) and the relationship between the lowest bandwidth and the lowest quality to control the freezes in the case of the bandwidth is insufficient even for the lowest video quality. $R_{switches}$(5) penalizes the quality switches between two consecutive requests depending on the jump magnitude. Lastly, $R_{freezes}$(6) is the penalization factor of the video freezes taking into account their duration (f_l).

Q-table, $Q(s_k, a)$*:* Indicates the learned benefit that the system will get taking action a in state s_k. In the exploitation mode, the algorithm will select the action with the highest $Q(\cdot)$ value for a given state, s. The updating of this matrix (learning process) after each quality decision is described below (7):

$$Q(s_k, a) \to (1-\alpha)\cdot Q(s_k, a) + \alpha \cdot \left[r + \gamma \cdot max_b Q(s_{k+1}, b)\right] \quad (7)$$

where α is the learning rate, γ weighs the contribution of the immediate and future rewards and b is the action that produces the highest Q-value for the next state s_{k+1}.

III. Experimental Results and Conclusions

A. Experimental Setup

In this section, we describe the experiments that have been carried out in order to compare the performance of DASH-SDP and our proposed control strategy, DASH-QL. We have conducted simulations of 150 episodes with 300 segments each episode. However, results have been measured over the last 50 episodes in order to evaluate the performance of the DASH-QL algorithm when its convergence has been achieved. We have used the following parameter values: a typical segment duration $T_{seg} = 2$ sec and each segment has been encoded at $N_q = 14$ quality levels with bitrates Q_i distributed between 100 and 4500 Kbps following a similar scheme to the distribution used in [6]. The buffer size (B_{max}) is 6 sec and we have determined experimentally that a reasonable target value to keep during the streaming session is $B_{opt} = 2/3 \cdot B_{max} = 4$ sec. The available channel throughput has been modeled by different Markov chains with N_{bw} quantified levels and a remaining probability of $p = 0.2$ with the aim of having highly dynamic channel instances. Regarding the fixed parameters of Eq. (7), we have taken the values that Claeys describes as the best configuration [9], $\alpha = 0.1$ and $\gamma = 0.1$. In order to assess the performance of the algorithms, we utilize the QoE measurement (8) defined by Claeys [8]:

$$QoE = 4.85 \cdot Q - 4.95 \cdot F - 1.57 \cdot S + 0.5 \qquad (8)$$

where Q is the average segment quality, F is the penalization by freezes and S captures the degradation by quality switches.

B. Comparison with DASH-SDP

In the first test, a single channel model has been used for the training and simulation episodes, with $N_{bw} = 15$ states with quantified values distributed between 100 and 4750 Kbps. As can be observed in the Table I, the QoE values obtained by both approaches are quite similar. This result validates our formulation since we have obtained a comparable result to that of [6]. Actually, DASH-QL obtains a slightly higher QoE value, due to the fact that our algorithm reduces the number of quality switches and their switch depth, hence a lower S value is obtained. To that extent, our algorithm requests slightly lower quality representations (Q value is lower for DASH-QL) but it outperforms DASH-SDP in terms of QoE.

TABLE I
Results with a single channel model

Algorithm	QoE	Q (%)	F (%)	S (%)
DASH-SDP	2.0228	34.88	0	10.75
DASH-QL	2.1012	34.64	0	5.01

In order to evaluate the inherent adaptiveness of our QL approach, we have conducted a second test in which we have trained both algorithms with the previous channel model, but we have measured their performance with a lower bandwidth channel. More specifically, we have used for the simulation phase a channel model with quantified values distributed between 50 and 2100 Kbps. Results are shown in Table II.

TABLE II
Results with different channel models

Algorithm	QoE	Q (%)	F (%)	S (%)
DASH-SDP	-0.093	17.92	27.39 [455]	6.67
DASH-QL	0.0362	15.39	24 [358]	1.43

As can be observed, there is a significant drop of QoE for both algorithms compared to the previous experiment. This is consequence of having a lower bandwidth channel that forces to request lower qualities and produces video freezes. Nevertheless, DASH-QL outperforms again DASH-SDP showing the adaptiveness of the QL approach. During the simulation episodes, DASH-QL is able to adapt to the new conditions of the channel modifying its control policy (whereas DASH-SDP follows the same policy). In that sense, DASH-QL aims at decreasing the occurrence of freezes requesting significantly lower qualities than DASH-SDP, preventing the video interruptions due to a greedy behavior that might lead to the buffer starvation, but getting finally a higher QoE value.

References

[1] Dynamic adaptive streaming over HTTP (DASH) – Part 1: Media presentation description and segment formats. http://standards.iso.org/ittf/PubliclyAvailableStandards/index.html.

[2] K. Miller, E. Quacchio, G. Gennari, and A. Wolisz, "Adaptation algorithm for Adaptive Streaming over HTTP," in *Packet Video Workshop (PV), 2012 19th International.* IEEE, 2012, pp. 173–178.

[3] 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*, ser. SIGCOMM '14, 2014, pp. 187–198.

[4] C. Liu, I. Bouazizi, and M. Gabbouj, "Rate adaptation for adaptive http streaming," in *ACM MMSys 2011, Special Session: Modern Media Transport, Dynamic Adaptive Streaming over HTTP (DASH*, pp. 23–25.

[5] P. Juluri, V. Tamarapalli, and D. Medhi, "SARA: Segment Aware Rate Adaptation Algorithm for Dynamic Adaptive Streaming over HTTP," June 2015.

[6] S. Garcia, J. Cabrera, and N. Garcia, "Quality-optimization algorithm based on stochastic dynamic programming for mpeg dash video streaming," in *Consumer Electronics (ICCE), 2014 IEEE International Conference on*, Jan 2014, pp. 574–575.

[7] A. G. Barto, *Reinforcement learning: An introduction (Adaptive Computation and Machine Learning).* MIT press, 1998.

[8] J. F. T. W. W. V. L. Maxim Claeys, Steven Latré and F. D. Turck, "Design of a Q-Learning-based client quality selection algorithm for HTTP adaptive video streaming," in *Adaptive and Learning Agents Workshop, part of AAMAS2013 (ALA-2013)*, 2013, pp. 30–37.

[9] M. Claeys, S. Latré, J. Famaey, T. Wu, W. Van Leekwijck, and F. De Turck, "Design and optimisation of a (FA) Q-learning-based HTTP adaptive streaming client," *Connection Science*, vol. 26, pp. 25–43, 2014.