Quality-Optimization Algorithm Based on Stochastic Dynamic Programming for MPEG DASH Video Streaming

Sergio García, Julián Cabrera and Narciso García

***Abstract*—In contrast to traditional push-based protocols, adaptive streaming techniques like Dynamic Adaptive Streaming over HTTP (DASH) fix attention on the client, who dynamically requests different-quality portions of the content to cope with a limited and variable bandwidth but aiming at maximizing the quality perceived by the user. Since DASH adaptation logic at the client is not covered by the standard, we propose a solution based on Stochastic Dynamic Programming (SDP) techniques to find the optimal request policies that guarantee the users' Quality of Experience (QoE). Our algorithm is evaluated in a simulated streaming session and is compared with other adaptation approaches. The results show that our proposal outperforms them in terms of QoE, requesting higher qualities on average.**

I. Introduction

As streaming technologies improve, consumers are more demanding and expect watching fluidly a certain video in every circumstance, either through a 3G connection or with broadband access, almost regardless of the quality. Traditional streaming approaches fail to provide high QoE for all the range of consumer electronics and access connections, especially when throughput fluctuates over time and involves issues like high latency or playback freezes. However, adaptive streaming techniques aim to minimize these shortcomings and guarantee a smooth streaming session for the users keeping the highest possible video quality [1]. This goal is feasible by encoding the same content with different qualities and dividing them in segments, making clients capable of dynamically switching between them.

Several proprietary adaptive streaming implementations, such as HTTP Live Streaming (HLS), have arisen, but the disparity of solutions and media formats has promoted the definition of the DASH standard by MPEG [2]. One of its key points is the open door for the adaptation logic that selects the next segment's quality, which is a clients' specific task.

Recent researches have been carried out to design algorithms based on either a heuristic [3] or a theoretical [4] approach. In this paper, we design an adaptation logic that maximizes the users' QoE by characterizing the stochastic nature of the system mathematically as a SDP problem. Then, applying SDP tools we compute offline the clients' optimal request policies, which can be easily included in current video consumer electronics such as smartphones or tablets. These techniques have already been used to control video-packets retransmissions in rate-limited environments [5] and match perfectly to the problem adaptive-streaming clients must face.

II. Stochastic Dynamic Programming Algorithm

When a client decides to request a DASH content, he must first download the manifest file which lists the different-quality versions stored at the server. Then the player has to sequentially request the segments in which the video has been split, deciding before each query which is the most adequate quality according to the network characteristics. Besides, it can even decide not to download any segment and delay the following request. After an amount of time that depends on the available bandwidth and the segment size, it will be downloaded and placed into the receiver's buffer. In the meanwhile, already-downloaded segments will be taken out of the buffer to be decoded and presented to the user.

Therefore we can formalize in terms of SDP this behavior as a dynamic system that evolves in stages influenced by random factors. Then, applying SDP techniques [7] we can compute optimal policies for each state, that is, the best quality-decisions according to the client's information. For that purpose, we need to define the state and action variables and their influence on the QoE-oriented cost function, as well as compute the state-transition probabilities.

Let k be the current stage and vector $s_k = (b_k, bw_k)$ the state of the system before requesting segment k, where b_k is the current buffer level, measured in segments and ranging from 0 to B_{MAX}, and bw_k is the last measured bandwidth, quantified in M discrete values $\{BW_1,\dots,BW_M\}$. Taking into account the state information, s_k, and the expected cost, the client must take a decision, a_k, on which quality has to be requested. The possibility of delaying the next request has also been included and denoted as a_k=0. Provided that there are R quality levels available at the server, the set of values for a_k is $\{0,Q_1,\dots,Q_R\}$.

The system progression is influenced by two random parameters: the requested-segment bit number, $\omega_k(a_k)$, and the change in the available bandwidth during next download.

From the system behavior we can derive system equation (1), noting that $[\bullet]$ represents the rounding operation, T stands for the segment duration in seconds and T_d is the delay time.

$$b_{k+1} = \begin{cases} \max\left\{ b_k + 1 - \left[\omega_k(a_k)/(T\cdot bw_{k+1})\right], 1\right\}, & a_k \neq 0 \\ \max\left\{ b_k - \left[T_d/T\right], 0\right\}, & a_k = 0 \end{cases} \quad (1)$$

The evolution of the variable bw_k is modeled as a first-order discrete-time Markov chain, simulating the throughput changes in a real environment [6]. The proposed chain is made up of the M discrete bandwidth values for bw_k and has a channel state-remaining probability of 0.8, whereas the transition probability to other channel states is 0.1 each. Additionally, we model the probability distribution of $\omega_k(a_k)$ as a sampled Gaussian function with mean $a_k \cdot T$.

The incurred cost for selecting a quality a_k when staying at state s_k is modeled considering two aspects: (i) the requested average bitrate should be close to and preferable below the measured bandwidth, and (ii) the buffer level should stay around a predefined target value B_{opt}, avoiding high and, especially, low values. We define the difference between the modulated bandwidth and the possible bitrates as d_k and the cost derived from the buffer level as c_k, defined in (2) and (3).

$$d_k = bw_k \frac{1 + b_k / B_{opt}}{2} - a_k \quad (2)$$

$$c_k = \left(b_k / B_{opt}\right)^2 - 2 \cdot b_k / B_{opt} + 1 \quad (3)$$

We therefore compute the cost for requesting quality a_k as

$$C(a_k \neq 0 / s_k) = \begin{cases} d_k + \alpha \cdot c_k, & d_k \geq 0 \\ \beta \cdot \left(1 - e^{d_k}\right) + \alpha \cdot c_k, & d_k < 0 \end{cases}.$$

In the case of delaying the decision, we penalize the cases where the buffer level is not close to the maximum value and where the bandwidth is close to the minimum, that is,

$$C(a_k = 0 / s_k) = \gamma \cdot \left\{ \left(\frac{b_k}{B_{MAX}} \right)^2 - 2 \cdot \frac{b_k}{B_{MAX}} + 1 \right\} + \delta \cdot \frac{BW_1}{bw_k} \cdot$$

Parameters α, β, γ and δ must be set to weight appropriately each cost component and obtain a natural policy behavior.

The next step is computing the state-transition probabilities. Since state variables are independent of each other, we have

$$P\{s_{k+1} / s_k, a_k, \omega_k\} = P\{b_{k+1} / b_k, bw_k, a_k, \omega_k\} \cdot P\{bw_{k+1} / bw_k\}$$

Once our model is completely defined and considering our problem as an infinite horizon one, as the number of stages can be extremely high, we use the SDP policy iteration method to compute the optimal policies for every state s_k.

III. Simulation and Results

We compare our quality-adaptation algorithm based on the optimal SDP policies with another DASH algorithm [3] and with the HLS implementation in a commercial player.

We first encoded a two-hour-long HD video movie in 14 quality streams with average bitrates Q_i distributed between 0.1 and 4.5 Mbps following a similar scheme to the distributed DASH dataset [8]. Each stream was segmented in 2-seconds chunks and stored in a server connected to our computer through a LAN, along with the DASH and HLS manifest files.

In order to dynamically limit the available bandwidth and simulate the most frequent throughput variations in a shared network, we used the open source tool DummyNet. We followed the proposed Markov chain model with BW_i ranging from 0.25 to 5 Mbps in steps of 250 kbps and evaluated the state-change possibility every second. Then we repeated the same sequence of bandwidth changes in all the simulations.

The numerical values used to compute offline the optimal policies before performing the simulations are B_{MAX} =10, B_{opt} =7, T=2, T_d=2, α=0.5, β=4.14, γ=100 and δ=100.

We used a self-implemented DASH client and an HLS player for the streaming sessions and registered the requested bitrate for every segment. An extract of the quality evolution for the simulations is presented in Fig. 1, along with the available throughput. Our algorithm manages to request bitrates which are on average higher than the bandwidth, at the expense of involving more quality switches.

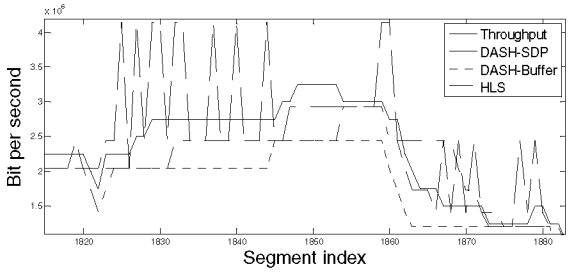

Fig. 1. Evolution of the throughput and the requested bitrates.

We finally carried out a numerical comparison based on previously proposed QoE measures [9]. Users' satisfaction is modeled as a function directly proportional to the average segment quality (factor Q) penalized by the frequency and length of video freezes (factor F) and quality switches (factor S). The average simulation results are presented in Table I.

TABLE I
Comparison of QoE Results of Simulations

Solution	Algorithm	QoE	Q (%)	F (%)	S (%)
DASH	SDP-based	3.738	75.37	6.9	4.8
DASH	Buffer-based [3]	3.360	69.59	10.2	0.6
HLS	Player's	3.639	67.61	2.1	2.2

As expected from Fig. 1, our SDP-based algorithm requests a higher average quality Q. However, as extracted from factor S, it has the highest quality-switch rate, which may be considered negative by users. Regarding the influence of freezes (F), our logic achieves an intermediate punctuation.

In terms of global QoE, which is computed by weighting the previous factors, our proposed logic outperforms the evaluated algorithms. It achieves a tradeoff between the requested quality and the resulting video freezes, so it can be simply and easily implemented in existing consumer electronic devices to improve the users' satisfaction relative to current solutions.

IV. References

[1] A.C. Begen, T. Akgul, and M. Baugher, "Watching video over the web: part 1: streaming protocols," *IEEE Internet Computing*, vol. 15, no. 2, pp. 54–63, Mar.-Apr. 2011.

[2] T. Stockhammer, "Dynamic adaptive streaming over HTTP – standards and design principles," *Proceedings of MMSys*, pp. 133-144, Feb. 2011.

[3] K. Miller, E. Quacchio, G. Gennari and A. Wolisz, "Adaptation algorithm for adaptive streaming over HTTP," *IEEE 19th International Packet Video Workshop*, pp. 173-178, May 2012.

[4] T. C. Thang, J. W. Kang, A. T. Pham, "Adaptive Streaming of audiovisual content using MPEG DASH," *IEEE Trans. Consumer Electronics*, vol. 58, no. 1, pp. 78-85, March 2012.

[5] V. Miguel, J. Cabrera, F. Jaureguizar and N. García, "Distribution of high-definition video in 802.11 wireless home networks," *IEEE Trans. Consumer Electronics*, vol. 57, no. 1, pp. 53-61 Feb. 2011.

[6] T. Andelin, V. Chetty, D. Harbaugh, S. Warnick and D. Zappala, "Quality selection for dynamic adaptive streaming over HTTP with scalable video coding," *Proc. of MMSys*, pp. 194-154, Feb. 2012.

[7] D. P. Bertsekas. *Dynamic programming: deterministic and stochastic models* (1st ed.), Prentice-Hall, 1987.

[8] S. Lederer et al., "Distributed DASH dataset," *Proceedings of MMSys*, pp. 131-135, Feb. 2013.

[9] M. Claeys et al., "Design of a Q-learning-based client quality selection algorithm for HTTP adaptive video streaming," *AAMAS*, May 2013.