A Q-Learning Solution for Adaptive Video Streaming

Dana MARINCA, Dominique BARTH
PRiSM, University of Versailles
Versailles, France
{Dana.Marinca, Dominique.Barth}@prism.uvsq.fr

Danny De Vleeschauwer
Alcatel-Lucent Bell Labs
Antwerp, Belgium
danny.de_vleeschauwer@alcatel-lucent.com

Abstract—**Adaptive streaming is a promising technique for video streaming service to cope with the quality degradation of mobile users' connections. In this paper we propose a service layer control mechanism for video flows based on a Reinforcement Learning (RL) paradigm that will gracefully degrade the video quality experienced by the end-user depending on the connection status. Using layered coded videos, the end-user should find the most appropriate quality level for its stream. The adaptive streaming problem can naturally be modeled as a Partial Observable Markov Decision Process (POMDP) because the end-user has partial information about the network state based on the received throughput, but this model cannot be applied on-line during streaming. We propose here an MDP modeling the adaptive streaming problem that can be solved on-line by the Q-Learning algorithm. Both models have identical solutions proving the validity of the proposed MDP model.**

Keywords-component: wireless users, adaptive video streaming, layered coded video, reinforcement learning, Q-Learning, MDP, POMDP

I. INTRODUCTION

Mobile data traffic is expected to increase 13-fold in the next 5 years and it will be dominated by video transfers [4]. This tendency is encouraged by an explosion of powerful mobile devices in the market that are capable of displaying high-quality video content and witnessed by the success of IPTV, catch-up services, online movie rentals [4]. The video contents are stored on Replication Servers (RS), spread out over the wired Internet and the contents are transferred from these RS to the end-users over a mixed wired and wireless infrastructure [19]. However, video communication over mobile broadband networks is challenging today due to the fluctuations in channel quality and difficulties in maintaining high reliability and low latency required by rich multimedia applications. Even with the migration from 3G to 4G networks, the increasing appetite for more bandwidth will continue to drive the network incongestion. Additionally, the lack of QoS over the wired Internet makes it difficult to achieve real-time and smooth video playback in periods of congestion because of lost packets and excessive delays during transfers.

High bandwidth video delivery necessitates examination of new ways to optimize future wireless and wired networks for delivering improved Quality of Experience (QoE) for a large set of video applications. One of the crucial video-enhancing solutions is adaptive streaming, which aims to optimize and adapt the video characteristics over time, considering varying channel conditions and wired network load, device capabilities, content characteristics.

Adaptive streaming mechanism aims for gracefully degrading quality of streamed video flows by adjusting the video bit rate to the temporary available throughput seen by the end-user and is efficient in better tackling the bandwidth limitations. The bit rate can be adjusted in various ways: 1) the encoder relies on feedback from the end-user to tune its parameters to set the right video bit rate (on-line); 2) the video is (off-line) encoded in multiple versions and depending on the network load the version with the right bit rate is chosen or 3) the video is encoded using layered coding (e.g., with H.264 Scalable Video Codec (SVC)) and depending on the network state the right number of layers is sent to the end-user. While we concentrate on an SVC-based system, the method proposed in this paper readily extends to the other two cases.

In this paper we propose a *service layer control mechanism* for video flows based on a Reinforcement Learning (RL) paradigm that will gracefully adapt the flow quality experienced by the end-user as its connection quality changes. The *end-user connection* covers the wireless and the wired network segments and the *connection status* is determined by the wireless channel quality and the wired network load. The goal of the end-user is to find the appropriate video bit-rate according to the connection status allowing him a smooth video playback. The underlying concept of RL is a Markov Decision Process (MDP). So the adaptive streaming problem should be modeled as an MDP. This allows the end-user to find the optimal solution. Alternatively, the adaptive streaming problem could naturally be modeled as a Partial Observable Markov Decision Process (POMDP) [6]: the end-user has partial information about the network state based just on the observed throughput of the received stream. The interest of the POMDP model is to compare its solution with the solution of proposed MDP model in order to evaluate the validity of our RL mechanism.

The remainder of this paper is organized as follows. Section II presents a survey of related work. In Section III we describe the POMDP model for adaptive streaming control problem and the HSVI solution. In Section IV we propose the RL adaptive stream control mechanism. The proposed MDP model is solved by Q-Learning, a well-known RL algorithm and both solutions are compared in Section IV.D. Section V analyzes the performances of proposed RL mechanism. Finally, in Section VI we draw the conclusions and present some perspectives.

II. RELATED WORK

Although there have been made some proposals based on RTP (Real-time Transport Protocol) and DCCP (Datagram Congestion Control Protocol), most of the latest work on

The research leading to these results has received funding from the European Union's Seventh Framework Program ([FP7/2007-2013]) under grant agreement n° 248775 as part of the FP7 STREP project OCEAN.

adaptive streaming was in the context of HTTP (Hypertext Transport Protocol). Some major companies have introduced their own system, e.g., Apple's HTTP Live Streaming [10] being the dominant one on mobile devices. They are all based on the same basic principles: a video is split into several segments which are encoded at different video bit rates, the video clients then dynamically adapt to the available throughput, based on metrics such as the measured throughput and the status of their video play-out buffer. Standardization efforts have tried to find the common ground [11] between these proprietary implementations, the most important being MPEG (Moving Pictures Experts Group) that standardized DASH (Dynamic Adaptive Streaming over HTTP) [12]. This standard allows the use of using multiple instances of a single layer media codecs, such as, H.264/AVC [13] and a layered media codecs, such as SVC [14] . The advantage of using SVC, in contrast to using multiple version of AVC-encoded video, where one decision has to be made at the beginning of each segment, the client can gradually build up the quality for a segment. It was also proven in [15] that the SVC-based approach can save storage space on a caching node of a CDN. Besides the commercial deployments mentioned above, several novel variants have been proposed in the literature. Jarnikov et al. [16] discuss several strategies to design a client that is robust against changing network conditions and provide guidelines with respect to the configuration of adaptation strategies. Adzic et al. [17] propose a client heuristic for mobile environments that uses a content-aware method to decide whether or not switching to a higher quality level is beneficial. Mastronarde et. al. [18] presents a RL technique to stream videos under low-delays and energy-efficiency constraints over a dynamic wireless environment.

III. POMDP Modeling For Adaptive Streaming

An end-user viewing a layered coded video wants to find the appropriate video quality level allowing him a good QoE, for each given state of its network connection.

The video adaptive streaming problem can be naturally modeled as a POMDP because the end-user acts under partial observability of the network load: it perceives the load through the ratio between the received throughput and the requested one. Solving the POMDP means to find the appropriate video quality level for each given network state in order to keep a good quality of visualization. Because the end-user cannot directly identify the state of the network, this problem does not fit the assumption of Markov property. The Markov property is an important requirement for an MDP model, ensuring its resolution. This property specifies that the state at time $t+1$ depends only on the state and the action taken at the time *t*. The aim of studying a POMDP model is to compare its solution with the solution of the MDP model of the RL mechanism proposed in Section IV.B.

A. POMDP Model

POMDPs provide a mathematical framework to reason about the user's perception of the environment and the effects of its actions in an incompletely known environment. To solve a POMDP we should derive from the POMDP an MDP model having the Markov property [6]. The solution of the associated MDP model represents also the solution of the POMDP model.

A POMDP is formally defined as a 7-tuple $\{S, A, T, R, \Omega, O, b_0\}$, and each element is defined as follows.

$S = \{CL_i, i \in [1, |S|]\}$ is the finite and discrete set of network states, more precisely different end-user Connection quality Levels (CL). We divide the interval [0,1] in several intervals, each one representing a CL (see (1)). We suppose that $CL_i \bigcap_{i \neq j} CL_j = \emptyset$ and $\bigcup_i CL_i = [0,1]$. A CL of 0 means very poor connection quality and a value of 1 means high CL.

$$CL_i = \begin{cases} [x_0, x_1], if \ 0 = x_0 < x_1 < 1, \\ (x_i, x_{i+1}], \ x_1 < x_i < x_{i+1} \leq 1 \end{cases}. \quad (1)$$

$A = \{Th_i, i \in [1, |A|]\}$ is the finite and discrete set of end-user actions. The elements $Th_i \in A$ represent the bit rates of the layered coded video associated to different quality levels. More precisely Th_i is the bit rate associated with the first *i* layers (base layer and *i-1* enhancement layers). We suppose that $Th_i < Th_{i+1} < Th_{i+2}$, for $1 \leq i \leq |A| - 2$. We denote $Th^{max} = \max_{1 \leq i \leq |A|}(Th_i)$ the maximum available bit-rate.

T represents the transition function characterizing the network dynamics. $T(s, a, s') = \Pr(s_{t+1} = s' | s_t = s, a_t = a)$ is the probability of reaching state s' at time $t+1$, from the state s at time t, executing the action a.

$R(s, a)$ is the reward received for executing the action $a \in A$ in the state $s \in S$. At time step t, the user being in the state s_t, asks the bit rate a_t and he receives the throughput Th_t^{rec}. The reward function is defined as follows:

$$R(s_t, a_t) = \begin{cases} Th_t^{rec} + Th^{max}, if \ Th_t^{rec}/a_t \geq \beta \\ Th_t^{rec}/(Th^{max})^3, otherwise \end{cases} \quad (2)$$

The constant $\beta \in [0.9, 1]$ defines the quality tolerance between the received and asked throughput. If $Th_t^{rec}/a_t \geq \beta$ we can consider that the user reception quality is not altered.

$\Omega = \{Th_i^{obs}, i \in [1, |\Omega|]\}$ is the set of user's discrete observations. In our context an element $o_t \in \Omega$ is the throughput of received stream at time *t*. If $Th_{i-1}^{obs} \leq Th_t^{rec} \leq Th_i^{obs}$, the received throughput is approximated by Th_{i-1}^{obs}.

O is the observation probability function, modeling the received stream, depending on the connection quality level and requested bit-rate. $O(o, a, s') = Pr(o_{t+1} = o | a_t = a, s_{t+1} = s')$ is the probability of observing *o* when executing action *a* and reaching the state s'.

Solving this model means to find, in a Markovian framework, an optimal policy $\pi: S \to A$, that maps states to actions such that the long term reward is maximized. One possibility for building a Markovian framework, from non-Markovian one, is to reformulate the problem in the space of probability distributions over the history of state-observations, known as *beliefs states* (see [6] for more details). A belief state is a vector $b = [b(s)]$, for all s ∈ S, where $b(s) = \Pr(s|h)$ is the probability of being in every state $s \in S$. So $b(s) \in [0,1]$ and $\sum_{s \in S} b(s) = 1$. The *belief states set*, denoted B, defines a Markovian framework, but the difficulty is that B is an infinite set. The model should also provide a distribution over initial states denoted $b_0(s_0) = \Pr(s_0)$ where $s_0 \in S$.

Formally, the POMDP defined above could be redefined as an MDP over the belief states spaces as a 4-tuple: $\{B, A, \tau, R_B\}$, where: (1) B is the *infinite set* of all possible belief states over S; (2) A is the set of agent actions as in the original POMDP; (3) τ is the belief states' transition function: $\tau(b, a, b')$. It is the probability of starting at belief b, executing action a and reaching the new belief b': $\tau(b, a, b') = \sum_{o\in\Omega} \Pr(o|b, a)$. The expected reward from executing action a in state s is $R_B = \sum_{s\in S} b(s)R(s, a)$. Solving this associated MDP solves the initial POMDP.

B. POMD Values for HSVI Algorithm

The belief-states' MDP is difficult to solve because of the infinite nature of B. Real applications showed that many belief states are not relevant to find an optimal or near-optimal policy [6][9]. The point-based solving algorithms compute a value function over a significant and reachable subset of the beliefs' states space. They maintain and improve a value function over this subset finding an optimal or near-optimal solution. Without lack of generality we will use the HSVI algorithm [1][2] and its simulator [3] to solve the MDP associated model.

The input data values are defined hereafter. We define a set of $|S| = 10$ connection quality levels intervals: $S = \{CL_i = [0.1 * (i-1), 0.1 * i], \forall i \epsilon \{1,2, \dots 10\}$. The actions' set contains $|A| = 3$ values expressed in Mbps: $A = \{Th_1 = 0.5;\ Th_2 = 1;\ Th_3 = 1.5\}$. The reward function is defined by (2), where $\beta = 0.95$ and $Th^{max} = 1.5$. We suppose without loss of generality the following transition model for the network dynamics:

$$T(s, a, s') = \begin{cases} 0.7, if\ s = s' = CL_i\ , for\ i = 2..9 \\ 0.85, if\ s = s' = CL_i, for\ i = 1\ or\ i = 10 \\ 0.15, if\ s = CL_i\ and\ s' = CL_{i+1}, for\ i = 1..9 \\ 0.15, if\ s = CL_i\ and\ s' = CL_{i-1}, for\ i = 2..10 \end{cases}$$

The client observations, Ω, contains received throughputs values (expressed in Mbps) : $\Omega = \{0.1;\ 0.2; \dots 1.0;\ 1.1; \dots 1.5\}$. If $obs_i \in \Omega$, a throughput value Th^{rec}, such as $obs_i < Th^{rec} < obs_{i+1}$ will be approximated to obs_i . Client probability observations, $O(o, a, s')$, are defined as bi-dimensional matrix O^a of type $|\Omega| \times |S|$, for each $a \in A$:

$$O^a(i, j) = \begin{cases} \frac{1}{10}, for\ i \leq a, \forall j \\ 0, otherwise \end{cases}$$

We suppose without lack of generality that the initial state is CL_1, so the belief state at t_0 is $b_0 = [1,0,0 \dots 0]$.

C. HSVI Solution

The HSVI algorithm evaluates two elements. The first one is an upper and lower bound of a state-action function, denoted Q-value, which are adjusted during the algorithm running. The convergence of the two values proves that the algorithm found an optimal Q-value, indeed an optimal policy. We can see the algorithm convergence in Figure 1.

The second element is the optimal policy, if it exists. For each system state, the algorithm computes a value associated to each possible action and the optimal action is the action having the largest value. The optimal policy founded by HSVI is shown in Figure 2. We can see the best actions for different connection quality levels: $\{\text{CL}_1 \text{ to } \text{CL}_5\} \rightarrow \{a = Th_3\}$; $\{\text{CL}_6, \text{CL}_7\} \rightarrow \{a = Th_2\}$; $\{\text{CL}_8, \text{CL}_9\} \rightarrow \{a = Th_1\}$; For $\{\text{CL}_8, \text{CL}_9, \text{CL}_{10}\}$ the action Th_3 was evaluated to 0, meaning that the system should avoid this action. For CL_{10} the values associated to actions Th_1 and Th_2 are inferior to values computed for CL_8 and CL_9 showing that these actions are possible but they are not recommended for this state.

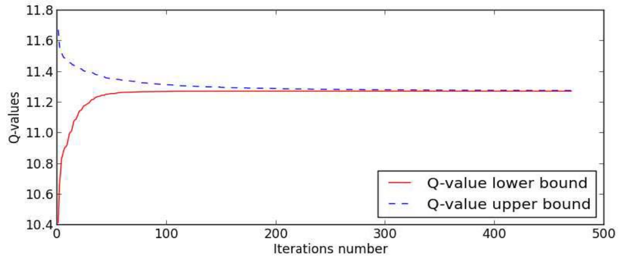

Figure 1. The convergence of POMDP model with HSVI

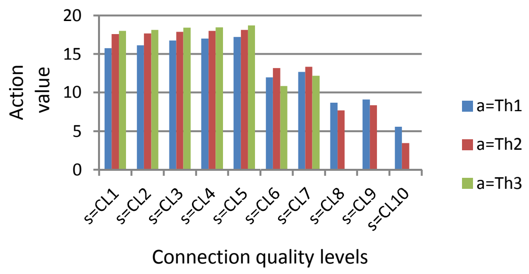

Figure 2. The best actions identified by HSVI

IV. RL Mechanism for Adaptive Streaming

The inconvenience of the POMDP model described in the previous section is that it cannot be solved on-line, during streaming service. In this section we propose a method allowing the end-user to map the information about the received stream into a CL. Using this mapping, the adaptive streaming problem could be defined directly as an MDP. The MDP model, integrated in the generic mechanism presented here could be solved on-line, during streaming service, using the well-knowing Q-Learning algorithm (see Section IV.C). The proposed mechanism learns to adapt CL variation covering channel quality fluctuation and/or network load variations without exploiting the Channel Quality Indicator (CQI) feedbacks from mobile end-user equipments, available at the Medium Access Control (MAC) layer [20].

A. Client-Driven Mechanism

We propose in this section a generic stream-control mechanism consisting of two modules: a client side module and a server side module. After selecting a video content to view, the client side module measures periodically the "quality" of his connection and maps it into a CL, computes a desired bit rate according to the identified CL and sends to the server module a command asking the new computed bit rate. The MDP model detailed in the Section B will be integrated on the client side. The server side module receives regularly the client's commands requesting a desired bit-rate. It selects the appropriate video quality level and sends it to the client respecting the required bit rate. This mechanism adapts dynamically to network variability. It can be integrated in a congestion control protocol, as for example DCCP [5], designed for applications with timing constraints on delivery.

B. *MDP Modeling*

The MDP model that we associate to adaptive streaming problem consists of the 4-tuple $\{S, A, T, R\}$. Each element was defined in Section III.A. In order to correctly identify the CL at each time step the client should use the procedure described in the next paragraph. The model will be solved using one-step Q-Learning algorithm (Section IV.B.2).

1) *Connection Quality Level Detection*

The client module acts periodically, as show Figure 3. We denote by T the constant interval between two time steps: $[t, t+1]$. The server delivers contents with different bit rates, but using constant size packets, denoted *S*. We suppose that the server respects the sending bit rate during T: it sends $T * Th_i/S$ packets. We also suppose that the server transmits to the client, at the beginning of the streaming session, two values, D_{min}, respectively D_{max}, representing a minimum respectively maximum delay observed on the route between the server and the client, maintained by the traffic monitoring system.

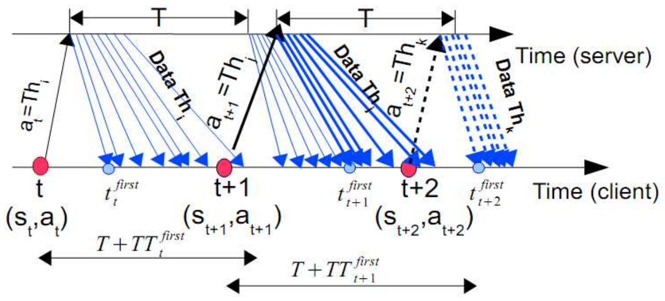

Figure 3. The client evaluates periodicaly the connection quality level

At each time step *t,* the environment being in the state $s_t = CL_i \in S$, the client requests to the server a stream respecting a throughput $a_t = Th_i$. According to Figure 3. the one-way trip time for the first data packet related to the command a_t is:

$$TT_t^{first} = (t_t^{first} - t)/2 \tag{3}$$

During the interval $[t, t + T + TT_t^{first}]$ the client should receive $T * Th_i/S$ data packets. We denote by Np_t^{rec} the number of packets really received during this interval. Delayed packets related to client's commands before a_t should not be taken into account. We define the rate of lost or delayed packets, p as follows:

$$p = \frac{S * Np_t^{rec}}{T * Th_i}. \tag{4}$$

We denote by σ_t the standard deviation of the inter-arrival time for data packets received during $[t, t + T + TT_t^{first}]$, related to the requested throughput $a_t = Th_i$. Using the relations (3) and (4), the client computes the value C_t as follows, $k_1, k_2, k_3 \geq 0$ being some tuning parameters:

$$C_t = k_1 \frac{TT_t^{first} - D_{min}}{D_{max} - D_{min}} + k_2 \sigma_t + k_3(1-p). \tag{5}$$

The client maps the value C_t to a interval CL_i, where $x_i < C_t \leq x_{i+1}$. If the value $C_t > 1$, the associated CL will be the highest one: $CL_{|S|}$.

2) *Q-Learning Algorithm*

One-step Q-Learning is a RL algorithm [7] based on the state-action value function $Q(s, a)$, evaluating the efficiency of each action a in each state s. At the time step t the agent executes the action a_t in the state s_t and receives an immediate reward $R(s_t, a_t)$. A policy $\pi: S \to A$, maps a state to one action and $Q^\pi(s, a)$ evaluates the expected value of the long term discount reward following π. The algorithm should find the optimal policy π^*, giving the best value $Q^*(s, a)$, following all policies. The interest of Q-Learning resides in the fact that it is able to find the optimal policy without knowing the environment transition model. At each time step *t*, the Q-value $Q(s_t, a_t)$, is updated following the relation (7), where $\alpha \in (0,1)$ represents the learning rate influencing the rate of Q-values updating, and $\gamma \in (0,1)$ represents the discount factor. The value $Q(s_t, a_t)$ computed by (7) directly approximates $Q^*(s, a)$, regardless of the followed policy [7].

$$Q(s_t, a_t) \leftarrow$$
$$Q(s_t, a_t) + \alpha * [R(s_t, a_t) + \gamma * max_a Q(s_{t+1}, a) - Q(s_t, a_t)] \tag{7}$$

An RL algorithm should balance its behavior between the exploration phases and exploitation phases. Different action choice policies are available [7]. In the next section we use *ε-greedy* policy to compute the Q-learning solution.

C. *Q-learning Solution*

To validate the proposed MDP model we simulated under NS-2 the architecture is composed of 4 nodes: *S↔N1↔N2↔C*. *S* represents the server sending the RL controlled video flow and *C* the mobile user receiving this flow. *N1* and *N2* represent intermediate nodes connected by a link at 100 Mbps. The links *S↔N* and *N2↔C* present a throughput of 10Mbps. N1 generates CBR/UDP traffic of about 94 Mbps and burst Pareto On/Off traffic. N1 send the generated traffic to N2.

The client module, acting on *C* node, applies the RL mechanism, presented in Sections IV.B. The elements *S, A, T* and *R* assume the values defined in Section III.B. For the CL detection procedure we used the values: $k_1 = 0.9, k_2 = 0, k_3 = 1$. For the Q-Learning algorithm we used the values: α=0.15, γ=0.2, *ε=1/n*, where *n=1000* represents the iteration number. We simulated 10 scenarios, each scenario $Scen_i$ corresponding to the connection quality level CL_i.

We present in the following figures the Q-values profile for each action. In each scenario $Scen_i$, after the learning period, the highest Q-value indicates the best action to do in that state. In all scenarios we observed that after a learning period generally inferior of 3-7 minutes, we can identify the best action. After a longer period of about 8-10 minutes, the Q-value no longer changes, meaning that the learning method converges. 0illustrates the Q-value for the three actions, in the scenario $Scen_5$ (CL_5). The best Q-value is associated to the action $Th_3 = 1.5\ Mbps$. Due to space limitations we do not show the Q-values associated to scenarios CL_1 to CL_4, but they have a similar profile. We can also see in Figure 4.b that the Q-value associated to the best action, Th_3, in the three states converges to the same value. Figure 5. shows the Q-values for all actions in CL_7. We observe that the best action is $Th_2 = 1$ Mbps. The action Th_3 has a much lower Q-value than the actions Th_2 and Th_1, meaning that this action is not recommended at all in these scenarios. Figure 5. shows the Q-values profile in the state CL_9 and the best action is $Th_1 = 0.5$ Mbps.

D. *Comparision of POMDP and MDP solutions*

We described adaptive streaming problem as two different models, a POMDP and an MDP, having common elements: *{S, A, T, R}*. The POMDP solution computed by HSVI algorithm (Section III.C) is an optimal policy identified as: (1) for the states $s = \{CL_1, .., CL_5\}$ the best action is $a = Th_3$; (2) for the states $s = \{CL_6, CL_7\}$ the best action is $a = Th_2$; for the states $s = \{CL_8, CL_9\}$ the best action is $a = Th_1$. The same solution was founded the proposed MDP model by for Q-Learning algorithm (Section IV.C) showing the validity of the proposed RL mechanism.

V. Performance Evaluation For Reinforcement Learning Mechanism

In this section we evaluate the impact of Q-learning tuning, the time until convergence for the RL process and the stream throughput at the end-user.

A. *Q-Learning tuning*

We analyzed through several simulations the impact of different values for the learning rate α and the discount factor γ (section IV.B.2). We observed that for both parameters, values superior to 0.4 determines too large updates of Q-values, and implicitly too fast and incoherent learning process. Values inferior to 0.15 determines too slow learning process. The best values for α and γ are in the interval *[0.15, 0.4]*. In our simulations we considered α=0.15 and γ=0.2.

B. *Number of iterations until convergence*

We calculated the mean number of iterations until convergence, on 20 runs of each elementary scenario $Scen_1$ to $Scen_9$. We regrouped the elementary scenarios depending on the best action identified. We denote the grouped scenarios as: $Scen_{1\text{-}5}=\{Scen_1, .., Scen_5\}$, $Scen_{6,7}=\{Scen_6, Scen_7\}$ and $Scen_{8,9} = \{Scen_8, Scen_9\}$. The mean number of iterations for each grouped scenario is computed as the mean between the mean values of elementary scenarios, as follows: $Scen_{1,5}\rightarrow 494$, $Scen_{6,7}\rightarrow 582$, $Scen_{8,9}\rightarrow 471$. In our simulations we considered each iteration being 1 second long, so the proposes scenarios finish the learning process after: $Scen_{1,5}\rightarrow$8.23minutes, $Scen_{6,7}\rightarrow$9.7minutes, $Scen_{8,9}\rightarrow$ 7.85minutes.

Indeed, we saw in 0 and Figure 5. that the best action is faster identified. In the worst case, the best action is identified after a number of iterations as follows: $Scen_{1,5}\rightarrow 196$, $Scen_{6,7}\rightarrow 84$, $Scen_{8,9}\rightarrow 72$. So, when the connection quality level stays constant, the learning process could be stopped after a maximum 200 iterations.

C. *Flow throughput at client side*

Figure 6. , Figure 7. Figure 8. respectively Figure 8. show the requested and the received throughput at the client in *Scen₅*, *Scen₇*, respectively *Scen₉*. We can observe in all scenarios that even after the identification of the best action, occasionally the client request a bit rate different from the optimal one. This effect is due to the exploration phase of the *ε-greedy* policy used for the choice of the action. To reduce the effect of the exploitation phase after the desired number of iterations, the value of ε should become very low, i.e. $\varepsilon=10^{-5}$.

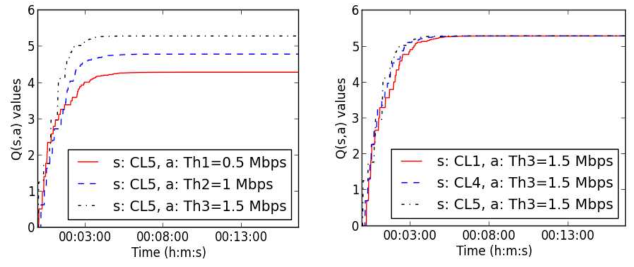

Figure 4. Q-values for (a) CL_5 (b) the best action Th_3

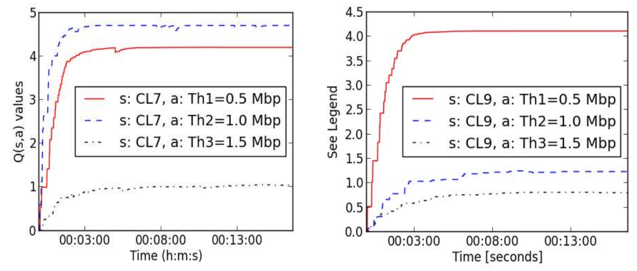

Figure 5. Q-values for (a) CL_7 and (b) CL_9

Figure 9. illustrates a scenario where the system changes its states respecting the transition model specified in Section III.B. During the simulation all $CL_i \in S$CL were generated. We can see on the graphic the learning phase during the first 9 minutes. The client is in the exploration phase, requesting different bit rates while trying to find the optimal actions. During this period the received flow varies a lot. After the learning period the optimal throughputs being identified, the received flow presents less variations at the client, the existing ones being generated by the exploration phase of ε-greedy policy. This scenario shows that the proposed RL mechanism is able to learn an optimal policy even if the system changes successively its states. Yet we identified a restriction: the system should stay in the same state during a certain number of iterations: empirically we founded a minimum number of iterations of 15. If the network changes its state at more quickly, the learning process is instable.

Figure 10. shows the RL controlled flow and a TCP flow at the client, during the same network load variation. The TCP flow presents high throughput variations, inadequate for a video stream. The client and intermediate nodes should allocate buffers to absorb these variations. The RL flow presents a smoothed throughput and better adapts to the real playing bit rate and to network load, allowing lower size buffers at the client and in the intermediate nodes.

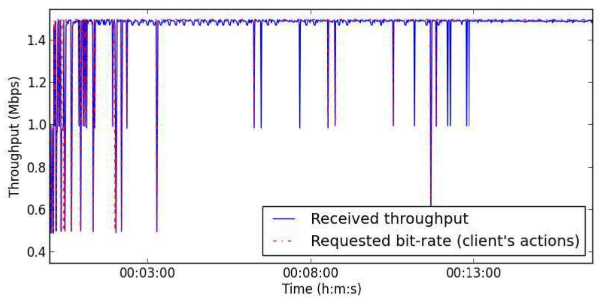

Figure 6. Throughput of video flow in state CL_5

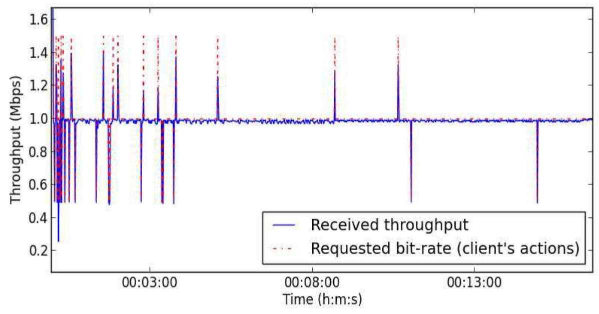

Figure 7. Throughput of video flow in state CL_7

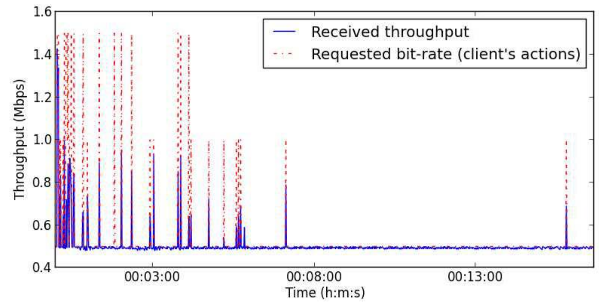

Figure 8. Throughput of video flow in state CL_9(b)

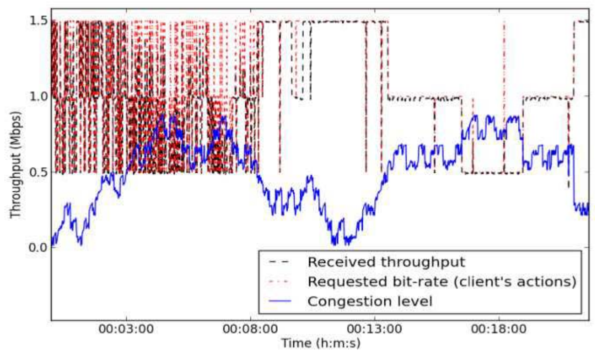

Figure 9. Throughput of video flow for different connections quality levels

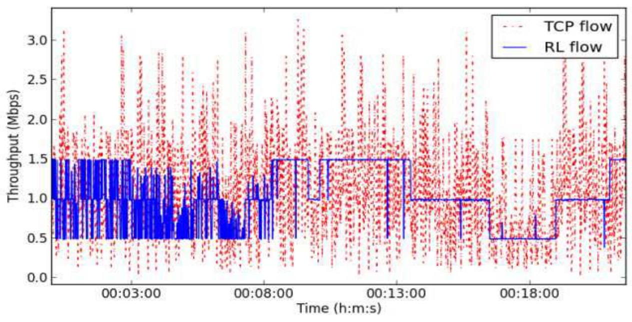

Figure 10. Throughput of RL flow vs.TCP flow

VI. Conclusion and Perspectives

We proposed a service level mechanism for the on-line flow control of layered coded videos. This mechanism is based on a MDP modeling of adaptive streaming problem and its solution is founded by Q-Learning algorithm. The same problem is modeled as a POMDP model and its solution is computed by HSVI. The two solutions are identical, showing the validity of the proposed mechanism. Some tuning parameters are analyzed and the learning duration phase was evaluated.

The proposed mechanism dynamically learns to adapt to the connections quality variations without exploiting the CQI feedbacks from end-user equipments available only at the MAC layer. In future work, a cross layer mechanism between the service layer and the MAC layer will be proposed to enhance and accelerate the learning process. Another important item for further study is to evaluate the impact on the network traffic when several clients use the RL mechanism.

References

[1] T.Smith, R.Simmons, “Heuristic Search Value Iteration for POMDPs”, Proceedings of the 20th conference on UAI’04, Pages 520-527

[2] T.Smith, R.Simmons, “Point-Based POMDP Algorithms: Improved Analysis and Implementation”, Proceedings of the 21th conf. on UAI’05

[3] https://github.com/trey0/zmdp

[4] http://www.cisco.com/en/US/solutions/collateral/ns341/ns525/ns537/ns705/ns827/white_paper_c11-520862.html

[5] E. Kohler, M. Handley, S. Floyd, “Datagram Congestion Control Protocol (DCCP)”, IETF RFC4340, 2006

[6] G.Shani, J.Pineau, R.Kaplow, “A survey of point-based POMDP solvers”, Jurnal Autonomous Agents and Multi-Agent Systems, Springer, July 2013, Volume 27, Issue 1, pp 1-51

[7] R. Sutton, A.G.Barto“Reinforcement Learning: An Introduction”, ISBN-13: 978-0262193986, MIT Press, Cambridge, MA, 1998

[8] A.Gosavi, “Reinforcement Learning: A Tutorial Survey and Recent Advances”, INFORMS Journal on Computing Spring, vol. 21/ 2, 178-192, 2009

[9] J.Pineau, G.Gordon,S.Thrun, “Point-based value iteration: An anytime algorithm for POMDPs”, International joint conference on artificial intelligence. Vol. 18, 2003.

[10] R. Pantos and W. May, “HTTP Live Streaming,” Internet Draft, http://tools.ietf.org/html/draft-pantos-http-live-streaming-10, 2012

[11] T. Stockhammer, “Dynamic adaptive streaming over HTTP: standards and design principles,” in Proc. of the second annual ACM conference on Multimedia systems, MMSys’11, pp. 133–144, ACM, 2011

[12] “Dynamic adaptive streaming over HTTP (DASH) -- Part 1: Media presentation description and segment formats”, ISO/IEC 23009-1:2012

[13] Thomas Wiegand, Gary J. Sullivan, Gisle Bjontegaard, and Ajay Luthra: Overview of the H.264/AVC Video Coding Standard, IEEE Transactions on Circuits and Systems for Video Technology, vol. 13, no. 7, pp. 560-576, July 2003

[14] R. Huysegems, B. De Vleeschauwer, T. Wu, and W. Van Leekwijck, “SVC-based HTTP adaptive streaming,” Bell Labs Technical Journal, vol. 16, no. 4, pp. 25–41, 2012

[15] Y. Sanchez, T. Schierl, C. Hellge, T. Wiegand, D. Hong, D. De Vleeschauwer, W. Van Leekwijck, Y. Le Louedec, “Improved caching for HTTP-based video on demand using scalable video coding”, in Proceedings of the Eighth Annual IEEE Consumer Communications and Networking Conference – Special Session IPTV and Multimedia CDN – (CCNC’2011 – SS IPTV), Las Vegas (NV), January 9-12, 2011

[16] D. Jarnikov and T. Özçelebi, “Client intelligence for adaptive streaming solutions,” Signal Processing: Image Communication, vol. 26, no. 7, pp. 378 – 389, 2011

[17] V. Adzic, H. Kalva, and B. Furht, “Optimized adaptive HTTP streaming for mobile devices,” in Applications of Digital Image Processing XXXIV (A. G. Tescher, ed.), vol. 8135, p. 81350T, SPIE, SPIE, 2011

[18] N. Mastronarde, M. van der Schaar, "Energy-Efficient Delay-Critical Communication in Unknown Wireless Environments", IEEE COMSOC MMTC E-Letter, Vol.7, No.8, pp. 8-11, November 2012

[19] C. Schwartz, J. Eisl, A. Halimi, A. Rafetseder, K. Tutschku “Interconnected Content Distribution in LTE Networks”, GLOBECOM Workshop on Advances in Communications and Networks 2010

[20] S. Singh, O. Oyman, A. Papathanassiou, D. Chatterjee J. G. Andrews “Video Capacity and QoE Enhancements over LTE”, IEEE International Conference on Communications 2012