# Design and Evaluation of a Self-Learning HTTP Adaptive Video Streaming Client

Maxim Claeys, *Student Member, IEEE*, Steven Latré, *Member, IEEE*, Jeroen Famaey, *Member, IEEE*, and Filip De Turck, *Senior Member, IEEE*

***Abstract*—HTTP Adaptive Streaming (HAS) is becoming the de facto standard for Over-The-Top (OTT)-based video streaming services such as YouTube and Netflix. By splitting a video into multiple segments of a couple of seconds and encoding each of these at multiple quality levels, HAS allows a video client to dynamically adapt the requested quality during the playout to react to network changes. However, state-of-the-art quality selection heuristics are deterministic and tailored to specific network configurations. Therefore, they are unable to cope with a vast range of highly dynamic network settings. In this letter, a novel Reinforcement Learning (RL)-based HAS client is presented and evaluated. The self-learning HAS client dynamically adapts its behaviour by interacting with the environment to optimize the Quality of Experience (QoE), the quality as perceived by the end-user. The proposed client has been thoroughly evaluated using a network-based simulator and is shown to outperform traditional HAS clients by up to 13% in a mobile network environment.**

***Index Terms*—Streaming media, learning systems, intelligent agent, quality of service**

## I. Introduction

In recent years, Over-The-Top (OTT) video delivery, where content is transported over the best-effort Internet, has gained a lot of popularity. For OTT video, HTTP Adaptive Streaming (HAS) is becoming the de facto standard. The general HAS concept is illustrated in Fig. 1. A HAS video file consists of multiple segments with a typical length of 2 to 10 seconds, encoded at multiple quality levels and resolutions. At the client side, the information about the video segments and quality levels is provided in the form of a manifest file. A standard HAS client requests a video segment on arrival of the previous segment. Based on the perceived network state and the information in the manifest file, a quality selection heuristic dynamically adapts the requested quality level. Each segment is downloaded in a progressive manner, while at the client side a buffer is used to bridge temporary anomalies such as a late arrival of a video segment during a small time window. Finally, the video segments, stored in the buffer, are played back as a single continuous video stream.

Transporting the video segments over HTTP provides both seamless interaction through firewalls and reliable delivery. On the other hand, the best-effort nature of the Internet makes these HTTP-based techniques prone to bandwidth fluctuations and network congestion. Given the detrimental impact on the Quality of Experience (QoE), the intelligence of the quality selection heuristic is crucial for the quality of HAS techniques.

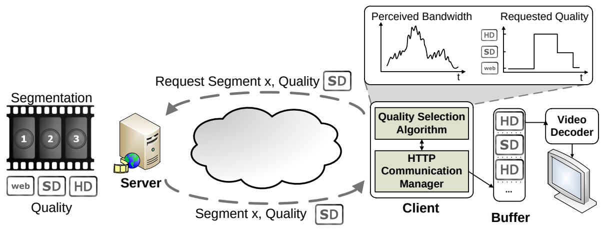

Fig. 1. Schematic overview of the HTTP Adaptive Streaming (HAS) concept.

Several large industrial players, including Microsoft[1], Apple[2] and Adobe[3] have commercial HAS implementations. MPEG, in collaboration with other standard groups, such as 3GPP, standardized the HAS interfaces and protocol data in Dynamic Adaptive Streaming over HTTP (DASH) in 2011 [1]. In this way, a common ground was established between the vast amount of available implementations. The bitrate adaptation heuristics are, however, not standardized, and thus implementation specific.

While current quality adaptation algorithms are hard-wired to fit specific network configurations [2], in this letter, a Q-Learning-based HAS client is proposed to dynamically adjust its behaviour by interacting with the network environment. Applying learning enables the client to adapt its behaviour to network conditions that were not under consideration when designing the typical deterministic quality selection algorithms. Using a network-based video streaming simulation framework, we performed extensive simulations of the proposed client for multiple video sequences with different dynamic scene variations. The simulations allow evaluation of the self-learning client in terms of convergence speed, performance and applicability. Furthermore, the evaluations are used to compare the performance of the proposed self-learning client to the traditional Microsoft ISS Smooth Streaming (MSS) algorithm.

The remainder of this letter is structured as follows: first, Section II describes the proposed client quality selection algorithm (Section II-A) and the design of the environmental state and reward definition (Section II-B). In Section III we define how the self-learning approach is evaluated (III-A) and how the experiments have been performed (III-B). The description of the evaluation results follows in Section III-C. Finally, Section IV presents the conclusions.

Manuscript received Month dd, 2013.

M. Claeys, J. Famaey, and F. De Turck are with the Dept. of Information Tech., Ghent Univ. - iMinds, Belgium (email: maxim.claeys@intec.ugent.be).

S. Latré is with the Dept. of Mathematics and Computer Science, University of Antwerp - iMinds, Belgium.

Digital Object Identifier 00.0000/XXXXX.0000.000000.000000

[1]http://www.iis.net/downloads/microsoft/smooth-streaming

[2]http://tools.ietf.org/html/draft-pantos-http-live-streaming-10

[3]http://www.adobe.com/products/hds-dynamic-streaming.html

## II. Q-Learning HAS Client

### A. Approach

Recently, the first steps in the application of a Reinforcement Learning (RL) agent in a HAS quality selection heuristic have been investigated [3], [4]. In RL, an agent learns by iterative interaction with its environment. The agent perceives the state of the environment and selects one of the available actions to take in this state. This action possibly causes a state transition and the agent receives a numerical reward to rate the quality of this transition. Based on the reward values, the agent gradually learns the optimal action to take in a specific state. From the vast amount of available RL algorithms, this work applies the commonly used Q-Learning algorithm [5]. In Q-learning, estimates of the "Quality" of performing an action in a particular state, called Q-values, are constructed for every state-action pair based on the numerical rewards. These Q-values are used by an exploration policy to select which action to perform in a given state, defining the trade-off between exploiting obtained knowledge and exploring new actions. Even though multiple exploration policies have been investigated, simulations have shown that Softmax yielded the best performance in the HAS use case. More information on Softmax, Q-Learning and its parameters can be found in [5].

### B. Use case modelling

To apply RL on the HAS use case, the available actions, the environmental state and the reward function have to be modelled. The agent's action is to select one of the available quality levels to request for the next video segment, given the perceived network state. A state definition is needed to model the behaviour of the networking environment the agent interacts with. When defining the state definition, care has to be given to the introduced number of states. Too many states will slow down learning and introduce the danger of overfitting, while too few states hamper the ability to correctly model the environment, leading to unsatisfactory results.

The initial approach to a self-learning HAS client [3] consisted of an environment model with more than 2.5 million states, leading to issues in terms of convergence. Furthermore, the vast environment model made the client unapplicable in situations with variable bandwidth. In the proposed learning agent, the state is constructed by only two parameters, found to be essential to model the networking environment: the available bandwidth perceived by the client and the current client buffer filling level, i.e. the total duration of the segments stored in the client buffer. Both of them are continuous values, which need to be discretized to be modeled as state variables. The value ranges and number of discretization levels of these state elements are shown in Table I. In this table, $B_{max}$ denotes the maximum client buffer size in seconds while $T_{seg}$ and $N$ respectively denote the segment duration in seconds and the number of quality levels. $BW_{max}$ is the highest possible throughput, e.g. the physical link capacity.

Since the agent's goal is to maximize the numerical reward, we want the reward function to be a measure of the QoE. There are three factors that impact the video quality as perceived by the user [6]: (i) the average segment quality level, (ii) the switching behaviour of quality levels and (iii) video freezes, caused by buffer starvations. For each of these aspects, a simple linear reward component has been constructed, as shown in (1), (2) and (3).

TABLE I
Proposed environmental state definition.

| State element | Range | Levels |
|---|---|---|
| Buffer filling | [0 ; $B_{max}$]sec | $\frac{B_{max}}{T_{seg}}+1$ |
| Bandwidth | [0 ; $BW_{max}$]bps | $N+1$ |

$$\mathrm{R}_{\text{quality}} = Q_k - N \tag{1}$$

$$\mathrm{R}_{\text{switches}} = -1.0 * |Q_k - Q_{k-1}| \tag{2}$$

$$\mathrm{R}_{\text{bufferfilling}} = \begin{cases} -100 & : B_k = 0 \\ B_k - B_{max} & : B_k > 0 \end{cases} \tag{3}$$

In these equations, $Q_k$ and $B_k$ respectively denote the quality level requested for segment $k$ and the buffer filling at the time of that request. The components drive the agent to higher quality levels, less switches and higher buffer filling. Since the highest priority is to avoid video freezes at any time, a strong penalization is given to an empty buffer. The exact penalization value is of less importance, but has to be significantly higher than the other components of the reward function. The total reward function can be defined as specified in (4).

$$\mathrm{R} = \mathrm{R}_{\text{quality}} + \mathrm{R}_{\text{switches}} + \mathrm{R}_{\text{bufferfilling}} \tag{4}$$

## III. Performance Evaluation

### A. Evaluation Metric

In order to allow fair comparison between the deterministic MSS algorithm and the proposed approach, an objective video quality metric is required. De Vriendt et al. [7] define the QoE of video delivered using HAS to be dependent on the average segment quality and the standard deviation of the segment quality. Using this *quality level model*, its parameters were tuned based on the results of a small subjective test by experts in the field of multimedia streaming.

Furthermore, Mok et al. [6] define the influence of the frequency and duration of video freezes on the estimated Mean Opinion Score (MOS). While the proposed formulation only considers three discrete levels of freeze frequency and length, we use a continuous interpolation of these levels to measure the influence of freezes on QoE. The result of this interpolation is given in (5), where $F_{freq}$ and $F_{avg}$ represent the number of freezes relative to the number of segments and the average duration over all freezes respectively.

$$\phi = \frac{7 * \max\left(\frac{\ln(F_{freq})}{6} + 1, 0\right) + \left(\frac{\min(F_{avg}, 15)}{15}\right)}{8} \tag{5}$$

Based on these considerations, the MOS for the playout of a video with $K$ segments, $N$ quality levels and played quality level $Q_k$ for segment $k$ can be estimated by (6).

$$\mathrm{eMOS} = \max(5.67 * \mu - 6.72 * \sigma - 4.95 * \phi + 0.17, 0) \tag{6}$$

In this equation, $\mu$ and $\sigma$ represent the normalized average quality level ($\frac{\sum_{k=1}^{K} \frac{Q_k}{N}}{K}$) and standard deviation

TABLE II
QUALITY LEVEL (QL) BITRATES (IN KBPS) FOR THE USED VIDEO TRACES: BIG BUCK BUNNY (B.B.B.), ELEPHANTS DREAM (E.D.), STAR WARS (S.W.), TOKYO OLYMPICS (T.O.), SONY DEMO (S.D.) AND SILENCE OF THE LAMBS (S.O.T.L.)

| QL | B.B.B. | E.D. | S.W. | T.O. | S.D. | S.o.t.L. |
|---|---|---|---|---|---|---|
| 1 | 262 | 261 | 150 | 165 | 229 | 165 |
| 2 | 333 | 328 | 276 | 338 | 505 | 337 |
| 3 | 521 | 523 | 424 | 548 | 863 | 544 |
| 4 | 789 | 796 | 672 | 907 | 1510 | 892 |
| 5 | 1030 | 1032 | 1371 | 1959 | 3379 | 1876 |
| 6 | 1242 | 1230 | 2222 | – | – | – |
| 7 | 2128 | 2117 | – | – | – | – |
| $\omega$ | 1.081 | 1.276 | 3.441 | 4.061 | 2.408 | 7.705 |

($\sqrt{\frac{\sum_{k=1}^{K}(\frac{Q_k}{N}-\mu)^2}{K-1}}$) respectively. One can verify that the theoretical range of this metric is $[0; 5.84]$. During the simulations however, a practical metric range $[0.00; 5.41]$ was observed, which corresponds to the typical levels of an estimated MOS.

## B. Experimental Setup

All of the experiments have been performed using the NS-3-based simulation framework described by Bouten et al. [8]. The simulated network topology consists of a single HAS server and client, between which the available bandwidth models a 3G bandwidth trace, described by Riiser et al. [9]. Based on these 3G bandwidth traces with a total duration of 220 minutes, measured on the bus path between Ljan and Oslo central station, Norway[4], we constructed a bandwidth trace with 800 unique episodes of 10 minutes. The resulting trace had an average bandwidth of 2163kbps with a standard deviation of 1268kbps. The minimum and maximum available bandwidth are 22bps and 6335kbps respectively. It is important to note that this bandwidth trace not only yields high variability within an episode, but also across the episodes. At the client side, a buffer of 20 seconds is available.

Over this topology, multiple video sequences with varying content types were streamed. Each video sequence, encoded using the H.264 Advanced Video Coding (AVC) codec, has a duration of about 10 minutes, a segment length of 2 seconds and quality levels as shown in Table II. The different bitrates are obtained by applying a different Quantization Parameter (QP) in the encoding. The average relative rangewidth $\omega$ is defined as an indication of the dispersion of segment bitrates, calculated as the average over all quality levels $i$ of the relative rangewidth $\frac{\max_k br_k^i - \min_k br_k^i}{\text{avg}_k br_k^i}$. In this formula, $br_k^i$ denotes the bitrate of segment $k$ in quality level $i$. For the learning agent to be able to converge, each simulation consisted of 400 episodes of streaming a 10 minutes video trace. To evaluate the results in the converged state, only the last 50 episodes are considered.

To compare the behaviour of the learning client to current deterministic HAS algorithms, we used the traditional Microsoft ISS Smooth Streaming (MSS) algorithm[5] and the DASH client proposed by Miller et al. [10]. In MSS, three buffer thresholds are used to configure the client behaviour. In our experiments, the values of 25%, 40% and 80% have been used for the panic, lower and upper buffer thresholds respectively, as experimentally defined by Famaey et al. [11]. The parameter values used for the client designed by Miller et al. are the ones proposed by the authors [10].

[4]Dataset available from:
http://home.ifi.uio.no/paalh/dataset/hsdpa-tcp-logs/bus.ljansbakken-oslo/

[5]Original source code available from:
https://slextensions.svn.codeplex.com/svn/trunk/SLExtensions/AdaptiveStreaming

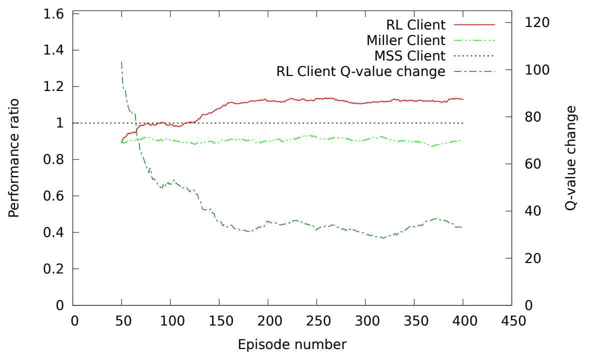

Fig. 2. Convergence of the learning client performance for the *Big Buck Bunny* video trace.

## C. Obtained Results

This section describes the evaluation results of the proposed self-learning HAS client. Since the behaviour of a learning agent depends on the configuration of multiple parameters, a parameter analysis has been performed to find the most appropriate parameter configuration out of a set of 500 possible configurations. The analysis showed that the best performance is obtained setting the learning rate, discount factor, Softmax inverse temperature and eligibility factor to 0.1, 0.1, 5.0 and 0.6 respectively. Since simulations have shown that this configuration yields satisfactory results in a wide range of network conditions, these values will be applied throughout the remainder of the evaluations.

*1) Performance Convergence:* On the left axis, Fig. 2 shows the relative performance of the RL-based client and the client proposed by Miller et al. compared to the traditional MSS client in terms of average MOS for the *Big Buck Bunny* video sequence. To be able to observe the general trend, a moving average of the metric values of the last 50 episodes is used. The graph shows that after a learning period of about 75 episodes, the RL-based client is able to achieve the same level of performance as the MSS client. After about 200 episodes, the increasing trend stabilizes. The convergence of the learning agent can also be seen in the flattening out of the Q-value changes, plotted on the right axis. Considering the converged state in the last 50 episodes, the self-learning HAS client outperforms the traditional MSS client with on average 13.05%. Furthermore, it can be seen that the performance of the client proposed by Miller et al. is overall about 10% lower than the MSS client. For this reason, the remainder of the results will be compared to the MSS client.

*2) Performance Reconvergence:* An important property of a learning agent is its ability to adapt to an environment change. Therefore, we evaluated the client performance when streaming different video sequences during a single learning session. For this purpose, we selected the *Big Buck Bunny* and

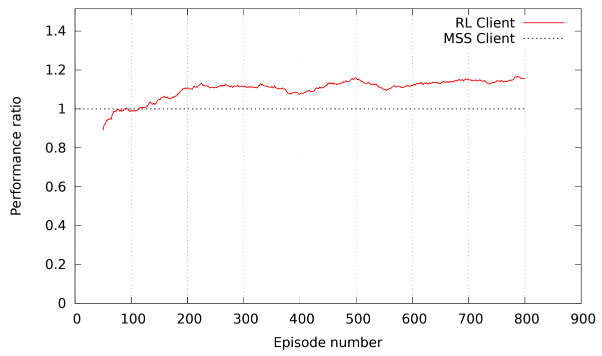

Fig. 3. Reconvergence of learning client performance when shifting between the *Big Buck Bunny* and *Elephants Dream* video sequences every 100 episodes.

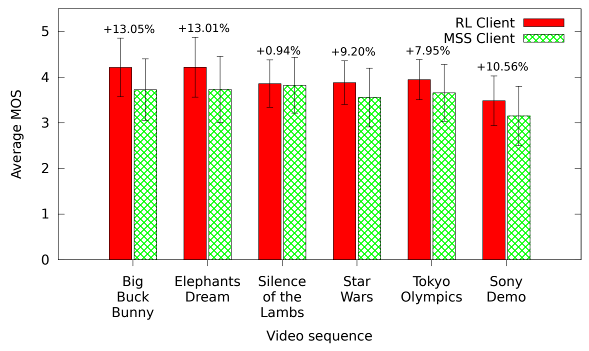

Fig. 4. Performance evaluation of the RL-based client for multiple video sequences in the converged state of the last 50 episodes.

*Elephants Dream* video sequences since these have similar quality level bitrates. This is a plausible assumption since a content provider will most likely offer all videos encoded using the same settings. Fig. 3 shows the performance ratio of the RL-based client, compared to the MSS client, in a situation where the video sequence shifts every 100 episodes. The graph shows that the self-learning client is able to instantly adapt to a new video sequence without performance loss. The previously obtained knowledge can be reused when learning on a new video sequence. It is clear that after a learning phase of comparable length as in the single video case, the self-learning client is able to outperform the MSS client by 15.48% in the converged state of the last 50 episodes.

*3) Multiple Video Sequences:* To show the general applicability of the RL-based client, experiments have been performed using the six video sequences shown in Table II. This set of video sequences contains multiple types of content with differences in scene variations. For each of the sequences, 400 episodes have been streamed. Fig. 4 shows the average performance and its standard deviation in the converged state of the last 50 episodes for each of the videos. On average, the RL-based client is able to outperform the MSS client by 9.12% in terms of average MOS, while reducing its standard deviation on average by 16.65%. Observations show an inverse relationship between the performance gained by the RL-based client and the dispersion of segment bitrates within a quality level. High values of $\omega$ affect the performance of the learning agent. This is illustrated by the lower performance increase for the *Silence of the Lambs* video with high bitrate dispersion and the high performance increases for the other videos with moderate dispersion.

## IV. Conclusions

In this letter, a Q-Learning-based HTTP Adaptive Streaming (HAS) client is presented, able to dynamically adjust its behaviour to the perceived networking environment. By using only two state elements, a self-learning client was built that outperforms the deterministic traditional Microsoft ISS Smooth Streaming (MSS) algorithm by up to 13% in a mobile network environment. Furthermore, the Q-Learning-based client is shown to be very well suited to react to video shifts. Since previously obtained knowledge can be reused when learning on a new video sequence, the learning agent can be trained offline in a variable bandwidth environment before being applied in an online HAS client. After an offline training phase of about 200 episodes, the self-learning HAS client is able to instantly adapt to network and video changes for all considered scenarios, making it applicable in a practical environment.

## Acknowledgment

M. Claeys is funded by grant of the Agency for Innovation by Science and Technology in Flanders (IWT). The research was performed partially within the ICON MISTRAL project (under grant agreement no. 10838). This work was partly funded by Flamingo, a Network of Excellence project (318488) supported by the European Commission under its Seventh Framework Programme.

## References

[1] T. Stockhammer, "Dynamic adaptive streaming over HTTP: standards and design principles," in *ACM Conference on Multimedia Systems (MMSys)*, 2011.
[2] S. Akhshabi, A. C. Begen, and C. Dovrolis, "An Experimental Evaluation of Rate-adaptation Algorithms in Adaptive Streaming over HTTP," in *ACM Conference on Multimedia Systems (MMSys)*, 2011.
[3] M. Claeys, S. Latré, J. Famaey, T. Wu, W. Van Leekwijck, and F. De Turck, "Design of a Q-Learning-based Client Quality Selection Algorithm for HTTP Adaptive Video Streaming," in *Workshop on Adaptive and Learning Agents (ALA)*, 2013.
[4] V. Menkovski and A. Liotta, "Intelligent control for adaptive video streaming," in *IEEE International Conference on Consumer Electronics (ICCE)*, 2013.
[5] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction (Adaptive Computation and Machine Learning)*. MIT Press, Mar. 1998.
[6] R. Mok, E. Chan, and R. Chang, "Measuring the Quality of Experience of HTTP video streaming," in *IFIP/IEEE International Symposium on Integrated Network Management (IM)*, 2011.
[7] J. De Vriendt, D. De Vleeschauwer, and D. Robinson, "Model for estimating QoE of Video delivered using HTTP Adaptive Streaming," in *IFIP/IEEE Workshop on QoE Centric Management (QCMAN)*, 2013.
[8] N. Bouten, J. Famaey, S. Latré, R. Huysegems, B. De Vleeschauwer, W. Van Leekwijck, and F. De Turck, "QoE optimization through in-network quality adaptation for HTTP Adaptive Streaming," in *International Conference on Network and Service Management (CNSM)*, 2012.
[9] H. Riiser, P. Vigmostad, C. Griwodz, and P. Halvorsen, "Commute Path Bandwidth Traces from 3G Networks: Analysis and Applications," in *ACM Conference on Multimedia Systems (MMSys)*, 2013.
[10] K. Miller, E. Quacchio, G. Gennari, and A. Wolisz, "Adaptation algorithm for adaptive streaming over HTTP," in *International Workshop on Packet Video (PV)*, 2012.
[11] J. Famaey, S. Latré, N. Bouten, W. Van de Meerssche, B. De Vleeschauwer, W. Van Leekwijck, and F. De Turck, "On the Merits of SVC-Based HTTP Adaptive Streaming," in *IFIP/IEEE International Symposium on Integrated Network Management (IM)*, 2013.