# Design of a Q-Learning-based Client Quality Selection Algorithm for HTTP Adaptive Video Streaming

Maxim Claeys[1], Steven Latré[1], Jeroen Famaey[1], Tingyao Wu[2], Werner Van Leekwijck[2] and Filip De Turck[1]
[1] Department of Information Technology, Ghent University - iMinds
Gaston Crommenlaan 8/201, B-9050 Gent, Belgium, email: maxim.claeys@intec.ugent.be
[2] Alcatel-Lucent Bell Labs, Copernicuslaan 50, B-2018 Antwerpen, Belgium

## ABSTRACT

Over the past decades, the importance of multimedia services such as video streaming has increased considerably. HTTP Adaptive Streaming (HAS) is becoming the de-facto standard for adaptive video streaming services. In HAS, a video is split into multiple segments and encoded at multiple quality levels. State-of-the-art HAS clients employ deterministic heuristics to dynamically adapt the requested quality level based on the perceived network and device conditions. Current HAS client heuristics are however hardwired to fit specific network configurations, making them less flexible to fit a vast range of settings. In this article, an adaptive Q-Learning-based HAS client is proposed. In contrast to existing heuristics, the proposed HAS client dynamically learns the optimal behavior corresponding to the current network environment. Considering multiple aspects of video quality, a tunable reward function has been constructed, giving the opportunity to focus on different aspects of the Quality of Experience, the quality as perceived by the end-user. The proposed HAS client has been thoroughly evaluated using a network-based simulator, investigating multiple reward configurations and Reinforcement Learning specific settings. The evaluations show that the proposed client can outperform standard HAS in the evaluated networking environments.

## Categories and Subject Descriptors

I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence—*Intelligent Agents*; C.2.3 [**Computer-Communication Networks**]: Network Operations—*Network Management*

## General Terms

Algorithms, Design, Management, Experimentation

## Keywords

HTTP Adaptive Streaming, Reinforcement Learning, Agent Systems

## 1. INTRODUCTION

In recent years, multimedia services have gained a lot of popularity. This growth is largely due to video streaming services. These services can generally be divided into Internet Protocol Television (IPTV), offered by a network provider and managed through resource reservation, and Over-The-Top (OTT) services. HTTP Adaptive Streaming (HAS) techniques are becoming the de-facto standard for OTT video streaming, where video is delivered over the traditional best-effort Internet. One of the most popular examples of OTT video streaming services is YouTube[1]. These HTTP-based techniques split video content into small segments of typically 2 to 10 seconds, each of which are encoded at multiple quality levels. This approach allows video clients to dynamically adapt the requested video quality to fit the perceived network state, such as delay and throughput.

The use of HAS techniques comes with some important advantages. Not only is the video content delivered reliably over HTTP, HAS also allows seamless interaction through firewalls. On the downside, delivery over the best-effort Internet makes these techniques prone to network congestion and large bandwidth fluctuations due to cross traffic, which can be detrimental for the Quality of Experience (QoE), the quality as perceived by the end-users. HAS client behavior is therefore a crucial factor for the streaming service to be beneficial and to ensure a sufficient level of QoE for the end user.

Current HAS client heuristics are however hardcoded to fit specific network configurations. This makes current approaches less flexible to deal with a vast range of network setups and corresponding bandwidth variations. This paper proposes a Q-Learning based HAS client, allowing dynamic adjustment of streaming behavior to the perceived network state. The contributions of this paper are three-fold. First, a Q-Learning-based HAS client has been designed. This approach, in contrast to current heuristics, allows the client to dynamically learn the best actions corresponding to the network environment. The design of the HAS client includes the definition of a tunable reward function to consider multiple aspects of QoE. Second, the HAS client design is integrated in an existing network-based video streaming simulation framework. Third, the implementation is used to perform simulations to extensively evaluate the influence of different parameters of the client. The simulation results allow comparison with the proprietary Microsoft ISS Smooth Streaming algorithm.

The remainder of this paper is structured as follows: an overview of HAS is first given in Section 2. Next, Section 3 describes the proposed client quality selection algorithm, describing the Reinforcement Learning (RL) reward function definition, the environmental state and the evaluated exploration policies. In Section 4, we discuss how the constructed HAS client is evaluated. Furthermore, we show that the pro-

[1]http://www.youtube.com

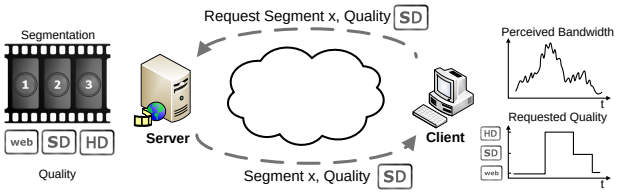



**Figure 1: Illustration of the HTTP Adaptive Streaming (HAS) concept.**

posed Q-Learning based HAS client outperforms traditional HAS heuristics in the simulated network environments. Section 5 gives an overview of related techniques, both on HAS and RL. Finally, Section 6 concludes this paper and presents some directions for future work.

## 2. HTTP ADAPTIVE STREAMING

HAS is the third generation of HTTP based streaming and is increasingly being used in OTT video services. Several industry players have proposed and make heavy use of HAS protocols, including Microsoft IIS Smooth Streaming [12], HTTP Live Streaming (HLS) by Apple [18] and Adobe's HTTP Dynamic Streaming [1]. With the standardized Dynamic Adaptive Streaming over HTTP (DASH) [20], MPEG tried to find the common ground between the multiple implementations available. In DASH however, only the interfaces and protocol data are standardized, leaving the algoritmic details up to the developer.

Regardless of specific implementation details, all of these HAS protocols follow some general architectural principles. First, video content is encoded using multiple quality levels and resolutions. Each of these encoded video streams are subsequently divided into segments by a stream segmenter. Typically, one video segment constitutes of several seconds of video. At the client side, a manifest file is provided, containing information about the different quality levels and video segments. The manifest file can thus be used to link the different video segments into a single video stream. For each segment, which can be transported as a single file over HTTP, the client uses the information in the manifest file to decide on the segment and the quality level to download. The selection heuristic depends on the specific implementation. Based on the perceived network state, these heuristics dynamically adapt the requested quality level. Current HAS clients are however deterministic, making them less flexible to fit a vast range of network environments. Each segment is downloaded in a progressive manner, while a buffer at the client side is used to take care of temporary anomalies such as a late arrival of a video segment. The successive segments, stored in the buffer, are played back smoothly as a single video stream. The general HAS concept is illustrated in Figure 1.

## 3. Q-LEARNING HAS CLIENT

### 3.1 Architectural overview

A general sequential HAS client requests a video segment on arrival of the previous segment. Prior to a request, the heuristic is called to select the quality level to request. The proposed client uses a RL agent in this heuristic.

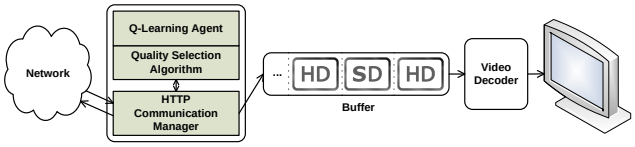



**Figure 2: Schematic overview of the RL HAS client design.**

RL is a machine learning technique in which an agent only has limited knowledge about the environment, leading to a high degree of uncertainty concerning how the environment will react to the performed actions. However, interaction with the environment is the only way for the agent to learn. At each state in the environment, the agent perceives a numerical reward, providing feedback to the agent's actions. The agent's goal is to learn which action to take in a given state of the environment in order to maximize the cumulative numerical reward [7].

A commonly used RL algorithm is Q-Learning [21]. Using Q-Learning, knowledge regarding both reward prospects and environmental state transitions are obtained through interaction with the environment. The proposed HAS client uses a Q-Learning agent to guide the quality selection heuristic.

When the heuristic is called, the current environmental state (Section 3.3) is updated in the agent. In Q-Learning, a Q-function is used to measure the "Quality" of a state-action combination, based on the perceived rewards. Equation 1 shows how the Q-values are updated when action $a$ is taken in state $s$, yielding a reward $r$. In this equation, $(s, a)$ is the state-action pair and $\alpha \in [0; 1]$ and $\gamma \in [0; 1]$ are the learning rate and the discount factor respectively. The learning rate $\alpha$ determines to what extent the newly acquired information overrides the old information, while the discount factor $\gamma$ is a measure of the importance of future rewards.

$$Q(s,a) = Q(s,a) + \alpha \left[ r + \gamma \max_b Q(s',b) - Q(s,a) \right] \quad (1)$$

Based on the learned Q-values, the quality level to request is selected. The specific selection tactic depends on the used policy (see Section 3.4). The current action is assigned a reward to update the learning values, based on the reward function described in Section 3.2. Whenever a new segment arrives, it is buffered at the client side. This buffer provides some protection for temporary unexpected anomalies. The segments in the buffer are smoothly played back sequentially.

### 3.2 Reward function

Since the reward function is the fundamental guide for the RL agent to learn the desired policy, we want the reward function to be a measure for the QoE. Nonetheless, defining good behavior of a video streaming service is very subjective. For example, it is hard tell if oscillating between two video quality levels is better than streaming a single, intermediary quality level. Therefore configurability is a desirable property for the RL reward function, giving the service provider a means to indicate his preferences. For the construction of this reward function, three aspects of quality could be identified [13]: (i) the current video quality level, (ii) the oscillation in quality levels during the video playout and (iii) buffer starvations, leading to video freezes. These aspects have led to a reward function being the weighted sum of four components, detailed below.

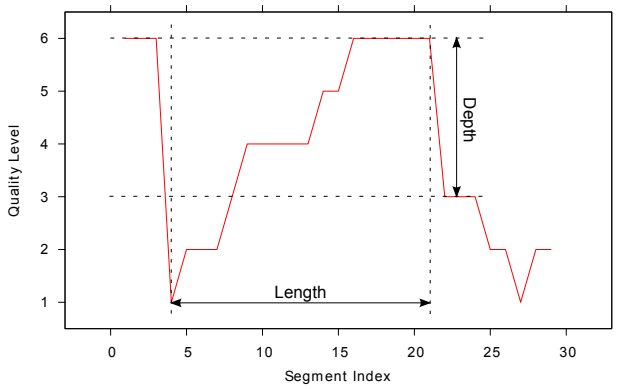



**Figure 3: Illustration of the oscillation length and depth definitions.**

### 3.2.1 Segment quality level

It has been shown that a quasi linear relationship exists between the objective PSNR values and the subjective Mean Opinion Score (MOS) evaluations [8, 14]. Using this knowledge, we model the quality level component of the reward as a simplistic linear function. The quality level component $R_{quality}$ of the reward function $R$ for streaming quality level $QL_i$ of a video file, consisting of $N$ quality levels, can be calculated using Equation 2.

$$\mathrm{R}_{\mathrm{quality}} = \frac{QL_i - 1}{N - 1} * 2 - 1 \tag{2}$$

### 3.2.2 Quality level oscillations

In HAS, the requested quality level will vary over time if the network environment, such as the available bandwidth, fluctuates. Single quality level changes in general do not yield negative experience. For example, if the available bandwidth has increased, switching to a higher quality level will not be perceived as bad behavior. Quality level switches however impact the QoE when oscillations occur. An oscillation can be defined as when a previous increase is followed by a decrease, or vice versa. The penalisation for the oscillation will be imposed only at the moment it is observed. To model the cost of an oscillation, we introduce the *length* and *depth* of the oscillation. An illustration of these properties is shown in Figure 3.

The length of an oscillation is defined as the number of video segments that have passed since the previous oscillation. For the reward calculation, a limit of $OL_{max}$ video segments is used, meaning that an oscillation of length higher or equal to $OL_{max}$ will receive a reward of 0. The number of quality levels of the immediate increase or decrease at the moment of the oscillation is defined as the depth. Oscillations are known to have a bigger impact on the QoE as the length becomes smaller. Similarly, quality switches are more noticeable when the switch amounts multiple quality levels. Using these definitions and observations of length ($OL_i$) and depth ($OD_i$), the oscillation component $R_{oscillation}$ of the reward function $R$ can be calculated as

$$\mathrm{R}_{\mathrm{oscillation}} = \begin{cases} 0 & : \text{no oscillation} \\ \frac{-1}{OL_i^{\frac{2}{OD_i}}} + \frac{OL_i - 1}{(OL_{max} - 1) * OL_{max}^{\frac{2}{OD_i}}} & : \text{oscillation} \end{cases} \tag{3}$$

### 3.2.3 Buffer starvations

Buffer starvations, leading to video freezes, are a third aspect known to have a considerable influence on the QoE. When a client buffer depletes, the video playout is interrupted and rebuffering takes place. After a while, the video playout continues. For example, such artifacts can sporadically be observed when watching a YouTube video. A freeze during the video playout is dramatic for the user quality perception. The inclusion of video freezes in the RL reward function however is not trivial. The reward function must capture the quality of the decision on a single segment quality level. However, a video freeze will typically not be the direct result of the decision on the last segment, but will be the result of earlier actions. Straightforward punishment of video freezes thus punishes the wrong action.

An indication of a negative evolution of the client side results, possibly leading to a video freeze, can however be found in the buffer filling level: a freeze is more likely when the buffer filling level is lower. Since a decreasing buffer increases the probability of a buffer to deplete and thus introduce a video freeze, also the change in buffer filling level is considered.

As for the quality level reward component, a simple linear function is used for the actual buffer filling level reward component. When the buffer filling level $B_i$ is below 10% of the total buffer size $B_{max}$, the lowest reward of -1 is rewarded. A buffer filling level below this threshold entails such a high chance of buffer starvation that a panic alarm should be enforced, imposing maximal penalisation. For higher buffer filling levels, a linear function with values between -1 and 1 is used, as shown in Equation 4.

$$\mathrm{R}_{\mathrm{bufferfilling}} = \begin{cases} -1 & : B_i \leq 0.10 * B_{max} \\ \frac{2 * B_i}{(1-0.1) * B_{max}} - \frac{1+0.1}{1-0.1} & : B_i > 0.10 * B_{max} \end{cases} \tag{4}$$

Concerning the change in the buffer filling level, the goal is to reward increasing buffer filling, while punishing decaying buffers. The reward calculation has been constructed in a way that rewards or punishes the buffer filling change more when the actual filling is lower. The specification can be found in Equation 5.

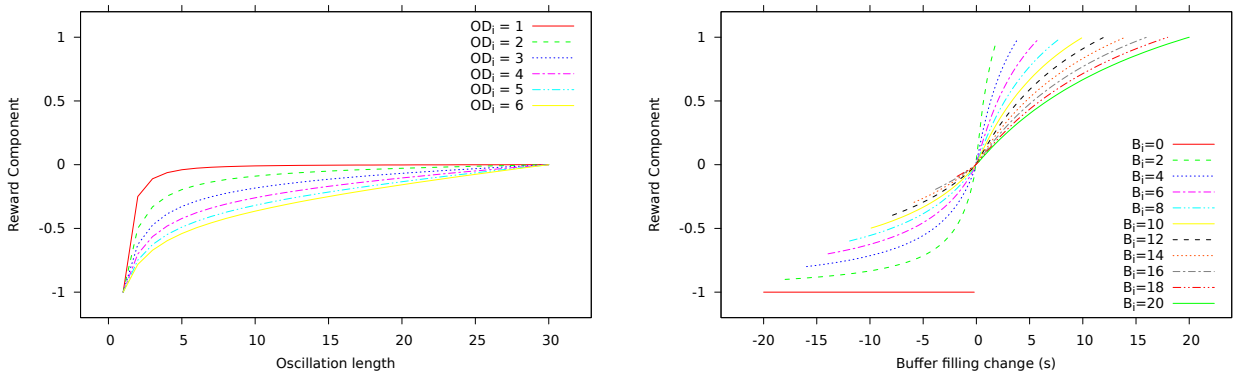
$$\mathrm{R}_{\mathrm{bufferchange}} = \begin{cases} \frac{B_i - B_{i-1}}{B_{i-1}} & : B_i \leq B_{i-1} \\ \frac{B_i - B_{i-1}}{B_i - \frac{B_{i-1}}{2}} & : B_i > B_{i-1} \end{cases} \tag{5}$$

### 3.2.4 Total reward function

The behavior of the constructed reward components for oscillations and buffer filling changes is illustrated in Figure 4. Since the quality and buffer filling level reward components are modelled as simple linear functions, their behavior is omitted due to space limitations. The illustration is based on the setup presented in Section 4.1, using a video trace with 7 quality levels, fixed segment length of 2 seconds and a maximum client buffer of 20 seconds. Based on the presented reward components, the total reward function can be defined as specified in Equation 6.

$$\begin{aligned} \mathrm{R} &= C_1 * \mathrm{R}_{\mathrm{quality}} + C_2 * \mathrm{R}_{\mathrm{oscillation}} \\ &+ C_3 * \mathrm{R}_{\mathrm{bufferfilling}} + C_4 * \mathrm{R}_{\mathrm{bufferchange}} \end{aligned} \tag{6}$$

By assigning different weights $C_1$-$C_4$, different behavior can be perceived, making the reward function tunable to fit mul-




(a) Oscillation reward

(b) Buffer change reward

**Figure 4: Illustration of the oscillation and buffer change reward components behavior.**

tiple situations. As discussed earlier, this tunability is required because of the subjectivity of the QoE.

## 3.3 Environmental state

When constructing the environmental state, the learning agent is provided with all information used to calculate the reward function. The state has therefore been defined using five elements, modelling information about the current quality level, buffer filling level and quality oscillations. Even though the available bandwidth is not an indicator of the perceived quality level, it is of crucial importance when deciding on the quality level to download. The environmental state has thus been extended with the available network bandwidth. The proposed environmental state definition is summarized in Table 1, where $T_{seg}$ represents the segment length in seconds.

| State element | Range | Levels |
|---|---|---|
| Buffer filling | $[0\ ;\ B_{max}]$sec | $\frac{B_{max}}{T_{seg}}$ |
| Buffer filling change | $[-B_{max}\ ;\ B_{max}]$sec | $2 * \frac{B_{max}}{T_{seg}}$ |
| Quality level | $[1\ ;\ N]$ | $N$ |
| Bandwidth | $[0\ ;\ BW_{max}]$bps | $N+1$ |
| Oscillation length | [0 ; 30]segments | 31 |
| Oscillation depth | $[0\ ;\ N-1]$ | N |

**Table 1: Proposed environmental state definition.**

## 3.4 Exploration Policy

One of the most challenging tasks in RL can be found in balancing between exploration and exploitation [24]. Where too much exploration may have negative impact with regard to the short-term reward, focus on exploitation can prevent maximizing long-term reward in uncertain environments since the learned actions possibly remain suboptimal [21]. Furthermore, this balancing task has a major influence of the convergence of the learning agents' behavior.

An often used approach to this tradeoff is the $\epsilon$-greedy method [27]. Using this method, exploration comes down to random action selection and is performed with probability $\epsilon$. The best known action is thus exploited with probability $1-\epsilon$. Even though this method often outperforms more complex approaches, it has some practical drawbacks. Since the optimal configuration of the $\epsilon$-parameter is very application dependent, rigorous tuning is required to obtain desirable results.

Another commonly used exporation method is Softmax [21]. In contrast to the $\epsilon$-greedy method, which is balancing optimal and random action selection, with Softmax action-selection is always performed in a probabilistic way. A Boltzmann distribution is used to rank the learned Q-values, based on which selection probabilities are calculated. As with the $\epsilon$-greedy method, parameters have to be tuned to fit the specific application.

Tokic et. al. propose the Value-Difference Based Exploration with Softmax action selection (VDBE-Softmax) policy [23, 24]. The VDBE-Softmax policy only performs exploration in situations when knowledge about the environment is uncertain, indicated by fluctuating Q-values during learning. The Softmax-method is extended by introducing a state-dependent exploration probability $\epsilon(s)$, being updated through the learning process based on the difference in learned Q-values before and after a learning step. Large differences indicate a high degree of uncertainty about the environment, shifting the policy to exploration. When differences decrease and the agent gains certainty, exploitation becomes more probable. In case of exploration, the Softmax policy is used to avoid bad performance when many actions yield relatively high negative reward, as was the case with the Value-Difference Based Exploration (VDBE) policy.

The proposed HAS client has been evaluated using both the Softmax and the VDBE-Softmax exploration policy.

# 4. PERFORMANCE EVALUATION

## 4.1 Experimental setup

The experiments have been performed using the NS-3 [16] based simulation framework described in [4], streaming the *Big Buck Bunny* video trace. A single episode of the video trace consists of 299 segments, each with a fixed length of 2 seconds. This comes down to a total episode duration of about 10 minutes. Each video segment has been encoded in 7 quality levels, with bitrates ranging from 300kbps to 2436kbps, as shown in Table 2. To ensure the RL agent has time to converge, 350 episodes of the video trace are simulated.

A basic network topology has been modelled, consisting of a single HAS client and server. For the construction of a bandwidth model, a cross traffic generator was used. The generated traffic is a sequence of fixed bitrate levels, sent over a 3Mbps link. Each bitrate level was uniformly dis-

| Quality level | Bitrate |
|---|---|
| 1 | 300kbps |
| 2 | 427kbps |
| 3 | 608kbps |
| 4 | 866kbps |
| 5 | 1233kbps |
| 6 | 1636kbps |
| 7 | 2436kbps |

**Table 2: Quality level bitrates for the Big Buck Bunny video trace.**

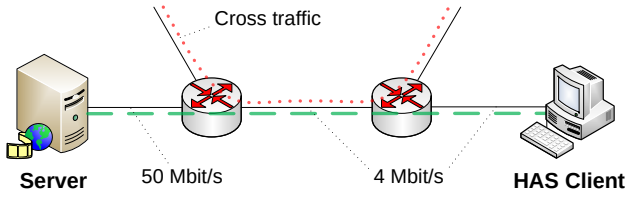



**Figure 5: Overview of the simulated topology.**

tributed between 0kbps and 2760kbps and persisted for a uniformly distributed amount of time ranging from 1s to 300s. The remaining bandwidth perceived by the client was used as a bandwidth trace, yielding an available bandwidth ranging from 203kbps to 2355kbps with an average of 1513kbps. An overview of this topology can be found in Figure 5. At the client side, a buffer of maximum 20 seconds (equal to 10 segments) is available.

## 4.2 Quality metric for evaluations

The reward function, described in Section 3.2, has been constructed to be a measure for the quality of a decision on a single segment quality level. To evaluate the different approaches however, a measure of the total video playout quality has to be used. Since objective video quality metrics are still an open research topic, an intuitive metric has been used in this work.

As stated earlier, the overall video quality experience depends on three aspects of the video playout [13]: (i) the average segment quality level, (ii) video freezes and (iii) switches in segment quality levels. The used metric has thus be constructed as a linear combination of these three aspects.

In Section 3.2.1, we discussed that the QoE scales linearly with the average segment quality level. The metric quality component will therefore be expressed as $\frac{QL_{avg}}{N}$, where $N$ is the total number of available quality levels.

The influence of video freezes on the perceived video quality depends on both the number of freezes and the average length of the freezes. In [13], a calculation has been proposed, using only three discrete levels of video freeze frequency and average freeze time. Based on an interpolation of these levels, a continuous function has been used for the metric in this work. The resulting function metric component can be found in Equation 7, where $F_{freq}$ and $FT_{avg}$ represent the freeze frequency and the average freeze time respectively.

$$F = \frac{7}{8} * \left( \frac{\ln(F_{freq})}{6} + 1 \right) + \frac{1}{8} * \left( \frac{\min(FT_{avg}, 15)}{15} \right) \quad (7)$$

The last aspect influencing the perceived video quality is the switching between segment quality levels. Each switch has an associated depth, being the number of quality levels bridged by the switch. The numerical impact on the quality can be expressed based on the number of switches $S$ and the average switch depth $SD_{avg}$, as shown in Equation 8 for a video trace containing $M$ segments and $N$ quality levels.

$$S = \frac{S * SD_{avg}}{M * (N-1)} \quad (8)$$

The metric can now be expressed as a linear combination of these components. To capture the coefficients of the linear combination, a small subjective test panel of 6 subjects has been composed. The test subjects were presented different traces of streaming behavior and were asked to judge the perceived quality. Linear regression has been applied to the results to extract the optimal coefficients, leading to the overall metric expression shown in Equation 9.

$$4.85 * \frac{QL_{avg}}{N} - 4.95 * F - 1.57 * S + 0.50 \quad (9)$$

One can verify that the theoretical range of this metric is $[-3.76; 5.35]$. However, since the minimal value would require a video freeze of at least 15 seconds after each segment while streaming the lowest quality, this range is not valid in practice. The practical range for realistic scenarios can be defined as $[0.0; 5.35]$.

## 4.3 Obtained results

### 4.3.1 *Reward function selection*

As described in Section 3.2, the reward function is defined as the weighted sum of 4 components. It is clear that this composite function yields an unlimited number of configurations. To be able to evaluate the results, only a small subset of these configurations has been considered. To support the selection process, 10 weight elements were divided among the 4 coefficients. This results in a total number of 286 configurations. For each of these reward configurations, a simulation of 350 episodes was executed and the average metric value of the last 50 episodes was calculated. This process was executed for both the Softmax and the VDBE-Softmax exploration policy. Based on the average metric values, a reward configuration has been selected, performing well for both exploration policies. The simulations were performed using a standard learning rate $\alpha = 0.3$ and a discount factor $\gamma = 0.95$. The resulting reward configuration assigns following weights according to the definition introduced in Equation 6: $C_1 = 2$, $C_2 = 1$, $C_3 = 4$ and $C_4 = 3$.

Further evaluations have been performed using the selected reward configuration to allow meaningfull comparison between the Softmax and the VDBE-Softmax exploration policy. All simulations are executed using fixed settings for the Softmax inverse temperature parameter $\beta$ and the VDBE-Softmax parameter $\sigma$. The inverse temperature $\beta$ has been set to 1.0, yielding an exploration/exploitation balancing relative to the intermediary Q-values. Lowering $\beta$ will force the balance towards random exploration while the opposite effect is achieved by increasing $\beta$. Based on a combination of results described in literature [24] and preliminary experiments, the VDBE-Softmax parameter $\sigma$ has been fixed to 0.2. For the oscillation reward component $R_{oscillation}$, the maximal oscillation length has been set to $OL_{max} = 30$.

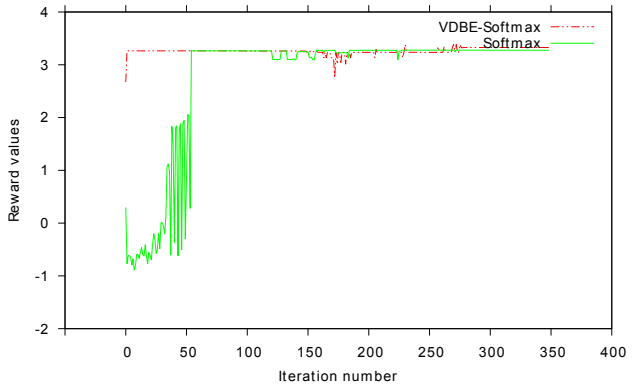
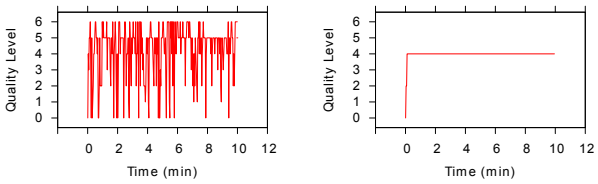



**Figure 6: Influence of the exploration policy on the learning agent convergence.**




(a) Episode number 20 (b) Episode number 350

**Figure 7: Convergence of the client behavior using the Softmax exploration policy and parameters $\alpha = 0.3$, $\gamma = 0.95$.**

### 4.3.2 *Influence of the exploration policy on convergence*

An important property of a learning agent is its convergence. Therefore, as a first evaluation, we analyse the convergence of the Softmax and the VDBE-Softmax exploration policy. Figure 6 shows the evolution of the average reward value per episode as a function of the number of simulated episodes for both the Softmax and the VDBE-Softmax policy. The figure shows that both policies converge to a comparable average reward value. However, faster convergence is achieved using the VDBE-Softmax policy due to the shifting to optimal action selection. The Softmax exploration policy on the other hand stays in exploration mode for a longer period. Since for the VDBE-Softmax policy the exploitation probability does not reach 100%, sporadic application of Softmax exploration is still possible. This can be seen in the fluctuations of the average reward around the $175^{\text{th}}$ episode.

### 4.3.3 *Evolution of client behavior*

The convergence can not only be seen in terms of absolute reward values, but can also be perceived when analysing the HAS client behavior. As shown in Figure 7(a), visualising the requested quality level per segment for the $20^{\text{th}}$ episode, the client behavior is still very unstable for early episode iterations. After convergence (episode nr. 350), the client behavior is distinct and more stable, as can be seen in Figure 7(b). Both graphs show the behavior of the HAS client, using the Softmax exploration policy with parameter settings $\alpha = 0.3$ and $\gamma = 0.95$.

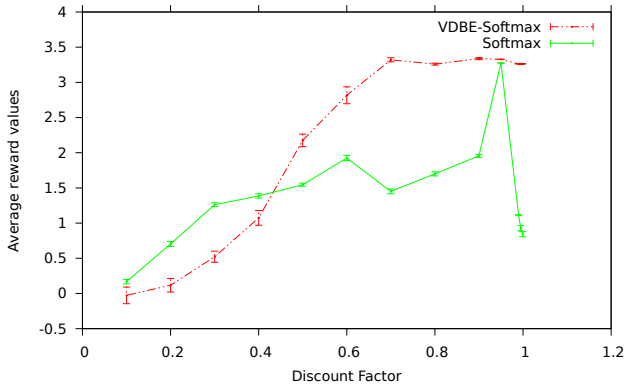



**Figure 8: Influence of the discount factor on the client performance.**

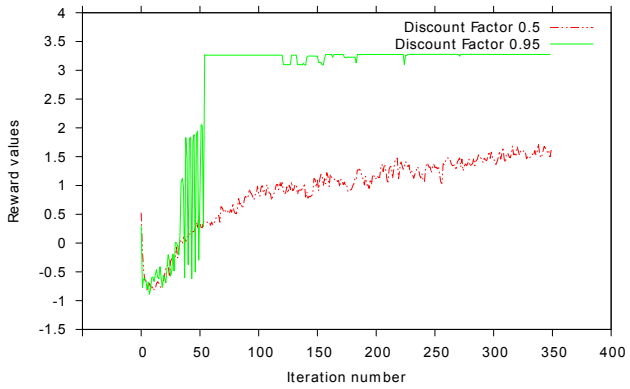



**Figure 9: Influence of the discount factor on the convergence of the Softmax learning agent.**

### 4.3.4 *Influence of the Q-Learning discount factor*

In order to evaluate the influence of the Q-Learning discount factor $\gamma$, simulations have been performed using 9 discount factors of 0.1 to 0.9. Seen the improving behavior for higher discount factors, additional simulations have been performed for discount factors 0.95, 0.99, 0.995 and 0.999. Each of these simulations were performed using a default learning rate of $\alpha = 0.3$. The results are summarized in Figure 8 for both exploration policies. The graph shows the impact of the discount factor $\gamma$ on the average reward values and corresponding 95% confidence intervals over the last 50 episodes (episodes 301-350). It can be seen that higher values of $\gamma$ yield higher reward values, with an optimal discount factor of $\gamma = 0.95$ for both the Softmax and the VDBE-Softmax exploration policy. Limited performance for small discount factor could be due to the large environmental state space. Smaller growth of Q-values can lead to slower convergence for large state spaces. For example, Figure 9 shows the convergence of the Softmax learning agent with learning rate $\alpha = 0.3$ and discount factors $\gamma = 0.5$ and $\gamma = 0.95$. It can be seen that convergence is significantly slower for smaller discount factors.

### 4.3.5 *Comparison with traditional HAS*

Using the optimal settings $\gamma = 0.3$ and $\alpha = 0.95$, the performance of the Q-Learning HAS client can be compared to traditional HAS behavior. Table 3 shows the average metric

values and standard deviations over the last 50 episodes for the proposed Q-Learning HAS client, using both the Softmax and the VDBE-Softmax exploration policy, compared to the traditional Microsoft ISS Smooth Streaming client [2]. The data shows that the Q-Learning-based HAS client outperforms traditional HAS client algorithms with 9.7% for the quality metric constructed in Section 4.2. Note that the increased MOS score caused a shift in classification from impairments being slightly annoying to noticable but not annoying. Furthermore, the negligible standard deviations confirm the converged behavior of the learning agent.

| Client | Quality metric | Stdev |
|---|---|---|
| MSS | 3.59601 | 0 (deterministic) |
| Softmax | 3.94450 | 0.00116 |
| VDBE-Softmax | 3.94343 | 0 (total stability) |

**Table 3: Average metric values and standard deviation per client algorithm.**

# 5. RELATED WORK

## 5.1 HAS client algorithms

As described in Section 2, multiple proprietary HAS algorithms are available, most of which are AVC-based. Recently, several new client approaches have been described in literature. Liu et al. propose a client heuristic to handle parallel HTTP connections, based on the segment download time [10]. By comparing the perceived segment download time with the expected segment download time, bandwidth fluctuations can be estimated appropriately. Jarnikov et al. discuss several guidelines on configuring robust HAS clients with regard to changing network conditions [6]. The opportunities of HAS in the domain of live streaming services are investigated by Lohmar et al. [11]. The work focusses on the influence of client buffering on the end-to-end delay. Many recent research topics focus on the applicability of HAS in mobile environments by exploiting additional information. The heuristic described by Riiser et al. [19] uses Global Positioning System (GPS) information to obtain more accurate information on the available bandwidth. Furthermore, Adzic et al. have proposed content-aware heuristics [2]. These approaches require metadata to be embedded in the video description. The additional consequences of quality selection in mobile environments have been shown by Trestian et al. [25]. The research shows that lowering the requested quality can significantly reduce energy consumption of Android devices.

In contrast to the above described approaches, we focus on an increased adaptivity and self-learning behavior of the client heuristic through the design of a RL-based client heuristic. To the best of our knowledge, no previous research proposed the application of Reinforcement Learning (RL) in the design of a HTTP Adaptive Streaming (HAS) client heuristic.

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

## 5.2 RL in network management

Reinforcement Learning (RL) has previously been successfully applied to various network management problems. In [5], Cao et. al. propose an agent-based network fault diagnosis model in which the agent uses RL to improve its fault diagnosis performance. Their research shows this approach can outperform traditional fault diagnosis models. Bagnasco et. al. propose the application of RL to dynamically adapt a hierarchical policy model to perform autonomous network management [3]. They argue that an autonomic system must have a degree of flexiblity to adapt to changes in goals or resources, which is hard to achieve by means of static policies. In the area of resource allocation, successfull applications of RL can be found. Vengerov presents a general framework for adaptive reconfiguration of distributed systems using a combination of RL and fuzzy rulebases [26]. Dynamic resource allocation of entities sharing a set of resources is used as an example. On the other hand, Tesauro et. al. propose a hybrid approach, gaining performance in the combination of RL and deterministic queuing models for resource allocation [22]. In this hybrid system RL is used to train offline on collected data, hereby avoiding possible performance loss during the online training phase. Furthermore, multiple approaches have been proposed, focussing on the resource allocation aspect in wireless mesh networks [9, 15]. Another area of network management where RL has been applied previously is Quality of Service (QoS) routing. Especially in wireless sensor networks, the network topology may change frequently, yielding inherently imprecise state information, which impedes QoS routing. In [17], Ouferhat et. al. propose a Q-Learning based formalism to optimise QoS scheduling.

# 6. CONCLUSION

In this paper, we proposed a Q-Learning based HAS client, allowing dynamical adjustment of the streaming behavior to the perceived network state in order to maximize the QoE. The RL agent has been provided with a tunable reward function, considering multiple aspects of QoE. Experiments have been performed using a NS-3 based simulation framework in order to analyse the influence of various RL setups on this specific use-case. Based on these simulation results, we were able to evaluate the proposed HAS client in comparison to proprietary clients such as Microsoft ISS Smooth Streaming based on an objective video quality metric. We have shown that, depending on the client configuration, the RL-based HAS client obtains a quality metric, which is on average 9.7% better than traditional HAS heuristics in the simulated network environments.

In future work, the evaluations will be extended to cover more realistic network environments. This includes network setups with multiple clients and additional bandwidth patterns. Furthermore, we will investigate the possibilities of multi-agent RL for multiple HAS clients to cooperate on optimizing their behavior.

# 7. ACKNOWLEDGMENTS

Maxim Claeys is funded by grant of the Agency for Innovation by Science and Technology in Flanders (IWT). The research was performed partially within the iMinds MISTRAL project (under grant agreement no. 10838). The Alcatel-Lucent research was performed partially within IWT project 110112.

## 8. REFERENCES


[1] Adobe. HTTP Dynamic Streaming: Flexible delivery of on-demand and live video streaming. Online. `http://www.adobe.com/products/hds-dynamic-streaming.html`, Last accessed: Dec. 2012.

[2] V. Adzic, H. Kalva, and B. Furht. Optimized adaptive HTTP streaming for mobile devices. *Applications of Digital Image Processing XXXIV*, page 81350T, 2011.

[3] R. Bagnasco and J. Serrat. Multi-agent Reinforcement Learning in Network Management. In *Scalability of Networks and Services*, volume 5637 of *Lecture Notes in Computer Science*, pages 199–202. Springer Berlin Heidelberg, 2009.

[4] N. Bouten, J. Famaey, S. Latre, R. Huysegems, B. De Vleeschauwer, W. Van Leekwijck, and F. De Turck. QoE optimization through in-network quality adaptation for HTTP Adaptive Streaming. In *Network and Service Management (CNSM), 2012 8th International Conference on*, pages 336–342. IEEE, 2012.

[5] J. Cao. Using reinforcement learning for agent-based network fault diagnosis system. In *Information and Automation (ICIA), 2011 IEEE International Conference on*, pages 750–754, June 2011.

[6] D. Jarnikov and T. Özçelebi. Client intelligence for adaptive streaming solutions. In *Multimedia and Expo (ICME), 2010 IEEE International Conference on*, pages 1499–1504, July 2010.

[7] L. P. Kaelbling, M. L. Littman, and A. W. Moore. Reinforcement Learning: A Survey. *Journal of Artificial Intelligence Research*, 4:237–285, 1996.

[8] J. Klaue, B. Rathke, and A. Wolisz. EvalVid - A Framework for Video Transmission and Quality Evaluation. In *In Proc. of the 13th International Conference on Modelling Techniques and Tools for Computer Performance Evaluation*, pages 255–272, 2003.

[9] M. Lee, D. Marconett, X. Ye, and S. Yoo. Cognitive Network Management with Reinforcement Learning for Wireless Mesh Networks. In *IP Operations and Management*, volume 4786 of *Lecture Notes in Computer Science*, pages 168–179. Springer Berlin Heidelberg, 2007.

[10] C. Liu, I. Bouazizi, and M. Gabbouj. Parallel Adaptive HTTP Media Streaming. In *Computer Communications and Networks (ICCCN), 2011 Proceedings of 20th International Conference on*, pages 1–6, Aug. 2011.

[11] T. Lohmar, T. Einarsson, P. Frojdh, F. Gabin, and M. Kampmann. Dynamic adaptive HTTP streaming of live content. In *World of Wireless, Mobile and Multimedia Networks (WoWMoM), 2011 IEEE International Symposium on a*, pages 1–8, June 2011.

[12] Microsoft. Smooth Streaming: The Official Microsoft IIS Site. Online. `http://www.iis.net/downloads/microsoft/smooth-streaming`, Last accessed: Dec. 2012.

[13] R. Mok, E. Chan, and R. Chang. Measuring the Quality of Experience of HTTP video streaming. In *Integrated Network Management (IM), 2011 IFIP/IEEE International Symposium on*, pages 485–492, May 2011.

[14] O. Nemethova, M. Ries, M. Zavodsky, and M. Rupp. PSNR-Based Estimation of Subjective Time-Variant Video Quality for Mobiles. In *Proc. of MESAQIN 2006*, 2006.

[15] D. Niyato and E. Hossain. A Radio Resource Management Framework for IEEE 802.16-Based OFDM/TDD Wireless Mesh Networks. In *Communications, 2006. ICC '06. IEEE International Conference on*, volume 9, pages 3911–3916, June 2006.

[16] ns 3. The Network Simulator ns-3. Online. `http://www.nsnam.org`, Last accessed: Dec. 2012.

[17] N. Ouferhat and A. Mellouk. A QoS Scheduler Packets for Wireless Sensor Networks. In *Computer Systems and Applications, 2007. AICCSA '07. IEEE/ACS International Conference on*, pages 211–216, May 2007.

[18] R. Pantos and W. May. HTTP Live Streaming. Online. `http://tools.ietf.org/html/draft-pantos-http-live-streaming-10`, Last accessed: Dec. 2012.

[19] H. Riiser, P. Vigmostad, C. Griwodz, and P. Halvorsen. Bitrate and video quality planning for mobile streaming scenarios using a GPS-based bandwidth lookup service. In *Multimedia and Expo (ICME), 2011 IEEE International Conference on*, pages 1–6, July 2011.

[20] T. Stockhammer. Dynamic adaptive streaming over HTTP: standards and design principles. In *Proceedings of the second annual ACM conference on Multimedia systems*, MMSys '11, pages 133–144, 2011.

[21] R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction (Adaptive Computation and Machine Learning)*. The MIT Press, Mar. 1998.

[22] G. Tesauro, N. K. Jong, R. Das, and M. N. Bennani. On the use of hybrid reinforcement learning for autonomic resource allocation. *Cluster Computing*, 10(3):287–299, Sept. 2007.

[23] M. Tokic. Adaptive $\epsilon$-greedy exploration in reinforcement learning based on value differences. In *Proceedings of the 33rd annual German conference on Advances in artificial intelligence*, KI'10, pages 203–210. Springer-Verlag, 2010.

[24] M. Tokic and G. Palm. Value-difference based exploration: adaptive control between epsilon-greedy and softmax. In *Proceedings of the 34th Annual German conference on Advances in artificial intelligence*, KI'11, pages 335–346. Springer-Verlag, 2011.

[25] R. Trestian, A.-N. Moldovan, O. Ormond, and G.-M. Muntean. Energy consumption analysis of video streaming to Android mobile devices. In *Network Operations and Management Symposium (NOMS), 2012 IEEE*, pages 444–452, Apr. 2012.

[26] D. Vengerov. A Reinforcement Learning Approach to Dynamic Resource Allocation. Technical report, Sun Microsystems Laboratories, 2005.

[27] C. Watkins. *Learning from Delayed Rewards*. PhD thesis, University of Cambridge, England, 1989.