Characterizing Adaptive Video Streaming Control Systems

Giuseppe Cofano, Luca De Cicco, Saverio Mascolo

Abstract— **Adaptive video streaming systems aim at providing the best user experience given the user device and the network available bandwidth. To the purpose, a controller selecting the video bitrate from a discrete set $\mathscr{L}$ has to be designed. The control goal is to maximize the video bitrate while avoiding playback interruptions and minimizing video bitrate switches. In the literature two different approaches, which we name *rate-based actuation* and *level-based actuation*, have been explored. The first one adapts both the received rate and the video bitrate, whereas the second acts only on the video bitrate. In this paper we advocate the adoption of level-based actuation controllers and we propose a hybrid dynamical system that models the essential features of such a class of controllers. With this model we are able to derive the minimum obtainable video bitrate switching frequency which can be considered as a benchmark for any level-based actuation controller. Finally, we show how to design the video level set $\mathscr{L}$ to obtain a performance trade-off between switching frequency and storage cost requirements at the servers. The theoretical results are validated through numerical simulation and experimental evaluation.**

I. Introduction

Video streaming services, such as YouTube and Netflix, generate an ever increasing fraction of the Internet traffic [4]. Providers have to deal with the challenge of delivering a seamless multimedia experience at the maximum obtainable Quality of Experience (QoE) across a wide range of devices and access networks. From the point of view of video providers, improving user engagement is the key requirement due to its direct connection to revenues. To this purpose the video content has to be made adaptive by employing a control algorithm designed to dynamically select the video bitrate. Today, the leading approach for implementing adaptivity is the *stream-switching* (or *multi bitrate*): the server encodes the video content at different bitrates, the *video levels*, and the control algorithm selects the video level to be sent. Due to its implementation and deployment simplicity, such an approach is today employed by leading video streaming services such as Netflix, Hulu, Vudu, Livestream, and YouTube. The two main adaptive streaming standards, MPEG Dynamic Adaptive Streaming over HTTP (DASH) and HTTP Live Streaming (HLS), adopt this approach. From the control architecture point of view, the leading choice is to place the controller at the client so that video distribution can be performed by means of standard HTTP servers and scalability can be achieved through CDNs.

In order to improve user engagement, the following goals have to be pursued: 1) avoiding playback interruptions; 2) maximizing video quality (level or bitrate); 3) minimizing the start up time; 4) minimizing the number of video level switches [12]. Two cooperating techniques are employed: 1) an algorithm to dynamically select the video level, which should ideally match the available bandwidth and 2) a playout buffer controller that is employed to absorb bandwidth variations and avoid playback interruptions. Playout buffer control algorithms can be designed by taking one of two different approaches: the buffer can be controlled by acting either on the received rate (*rate-based actuation* approach) or on the video level (*level-based actuation* approach). It is now well-known that the mainstream *rate-based actuation* approach leads to issues such as poor bandwidth utilization and unfairness in presence of concurrent flows [9], [2], whereas the *level-based actuation* approach can lead to a possible increase of the number of level switches at steady state.

In addition to the aforementioned requirements, content and network providers also aim at minimizing distribution costs such as server storage and network bandwidth utilization. This involves a fundamental trade-off: in fact, since the higher the video encoding quality the higher the video file size, user engagement can be increased only at the price of higher server storage and network bandwidth costs.

The contribution of this paper is twofold. Firstly, we advocate the use of *level-based actuation* controllers and we propose a model, in the form of a hybrid dynamical system, to generalize and analyze such controllers. Based on this model we derive the minimum switching frequency at steady state, which can be considered as a benchmark for any level-based actuation controller. Secondly, we take the provider point of view and we show how to design the video level set $\mathscr{L}$ to obtain a trade-off between switching frequency and storage costs.

II. Adaptive Video Streaming

A video streaming system allows a client to play a video that is sent by a remote server through an Internet connection. The client employs a playout buffer to absorb the instantaneous mismatches between the encoding bitrate and the network available bandwidth that in best-effort Internet is unpredictable and time-varying. However, if the bandwidth gets below the video bitrate for a sufficiently long time, the buffer will eventually gets empty and a re-buffering phase will be triggered: the player gets paused for a time

The authors are with the Dipartimento di Ingegneria Elettrica e dell'Informazione, Politecnico di Bari, Via Orabona 4, Bari, Italy. Emails: giuseppe.cofano@poliba.it, luca.decicco@poliba.it, mascolo@poliba.it

This work has been supported by the Italian Ministry of Education, Universities and Research (MIUR) through the MAIVISTO project (PAC02L1_00061).

interval, the *re-buffering time*, allowing the buffer to reach a safety threshold before the playing can be resumed. The leading approach for implementing adaptivity is the stream-switching (or multi bitrate): the server encodes the video content at different bitrate levels forming a discrete set $\mathscr{L}$ and the adaptive control algorithm dynamically selects the video level to be streamed.

The goal of adaptive video streaming control algorithms is to maximize the users perceived Quality of Experience given the available bandwidth. Several factors concur to its maximization. In particular, the bitrate has to be maximized, possibly matching the available bandwidth, whereas the start up time, the frequency of re-buffering events and the number of video level switches have to be minimized. The control algorithm can act on the received rate and on the selected bitrate.

A. Plant model

In this Section we present a fluid-flow model of the playout buffer length. Given a video of total duration T_v, each video frame can be uniquely associated to a time instant $t_v \in [0, T_v]$. We define the *video encoding bitrate* as $l = dD/dt_v$, where dD is the amount of bytes required to store a portion of video of duration dt_v. Indeed, by definition the encoding rate is always strictly greater than zero. We denote, with some abuse of notation, the video level selected by the controller with $l(t)$. The *received rate* $r(t)$ can be defined as $r(t) = dD/dt$, i.e. the amount dD of bytes that are received in a time interval dt. We denote with $b(t)$ the end-to-end *available bandwidth* and with $q(t)$ the *playout buffer length*, i.e. the total duration of video stored in the playout buffer and measured in seconds[1].

As any storage element, we can model the playout buffer length as an integrator:

$$\dot{q}(t) = f(t) - d(t),$$

where $f(t)$ is the *filling rate* and $d(t)$ is the *draining rate*. If in a time dt an amount of video duration dt_v is received by the client and stored in the playout buffer, then – by definition – the instantaneous filling rate is equal to dt_v/dt. Thus, from $f(t) = (dt_v/dD) \cdot (dD/dt)$ it readily turns out:

$$f(t) = \frac{r(t)}{l(t)}. \tag{1}$$

The playout buffer is drained by the player: when the video is playing, in any τ seconds τ seconds of video are played, i.e. $d(t) = 1$; on the contrary, when the player is paused the draining rate is zero. Thus, the draining rate is given by:

$$d(t) = \begin{cases} 1 & \text{playing} \\ 0 & \text{paused} \end{cases} \tag{2}$$

Finally, combining (1) and (2) yields the playout buffer length model:

$$\dot{q}(t) = \frac{r(t)}{l(t)} - d(t). \tag{3}$$

B. Control approaches

Based on the actuation variable, video streaming control systems can be classified as 1) *rate-based* and 2) *level-based*. In the first case the actuation variables are both the video level $l(t)$ and the received rate $r(t)$, in the second one only the video level $l(t)$ is used.

Let us consider the rate-based actuation approach. For simplicity let us assume that the available bandwidth measured by the client is constant and equal to $\hat{B}$. In this case the video level $l(t)$ is typically selected as the maximum $l \in \mathscr{L}$ less than the available bandwidth $\hat{B}$ [1]. Since $l < \hat{B}$, it turns out that, according to (1), the filling rate would be greater than 1, i.e. the queue would always grow. Hence, with this approach, the received rate $r(t)$ is required to be set equal to $l(t)$ to make $q(t)$ match a threshold q_T. However, the client cannot set $r(t)$ to a desired value since, for each t, video segments are downloaded at a rate equal to the end-to-end available bandwidth $\hat{B}$. Thus, the only way to achieve, at least on average, the desired received rate $r(t)$ is to insert idle periods between the downloads of two consecutive video segments. In other words the client alternates between an ON and an OFF phase: during the ON period, the client receives at a rate $r(t) = \hat{B}$, whereas during the OFF period it stays idle, i.e. $r(t) = 0$. With this control approach the average received rate can be set equal to the selected video level by properly setting the OFF period duration. The advantage of this approach is that video level switches occur only when the bandwidth changes. However, this approach has two drawbacks: 1) the available bandwidth is always underutilized; 2) it has been experimentally shown that the ON-OFF traffic pattern causes the video flows to obtain a bandwidth share significantly less than the fair one when competing with long-lived TCP flows [2], [9], [11]. To tackle these issues, several adaptive streaming algorithms have been proposed so far. FESTIVE has been specifically designed to address the fairness issues arising in a multi-client scenario [10]. A different strategy to obtain $r(t) = l(t)$ is to shape the sending rate at the server as proposed in [3]. Interestingly, it has been shown that this strategy is also employed by the Akamai control algorithm [6], [7].

In the level-based actuation approach, the video segments are downloaded back to back, thus eliminating the ON-OFF traffic pattern, i.e. $r(t)$ is always equal to $\hat{B}$. As a consequence, full utilization and fairness with greedy TCP flows can be achieved. The control is done by throttling $l(t)$ in order to keep $q(t)$ in a range $[q_L, q_H]$. With this approach, in fact, it would be impossible to steer $q(t)$ to a target q_T because of the quantized nature of $l(t)$. The drawback of this approach is that at steady state video level switches occur even with constant bandwidth $\hat{B}$. In [14] a PI is employed to steer the buffer level to a setpoint. In [5] feedback linearization is used to design ELASTIC, a controller that steers the buffer level to the target. ELASTIC

[1]For the rest of the paper we will use the terms "buffer" and "queue" equivalently.

has been experimentally compared with several *rate-based actuation* controllers and it has been shown that it is able to overcome *rate-based actuation* controllers issues such as bandwidth underutilization and unfairness with greedy TCP flows at the price of a possible increase of video level switches number at steady state.

III. LEVEL-BASED ACTUATION CONTROL

The goal of this section is to characterize the behavior of the family of the stream-switching controllers acting only on the video level $l(t)$. To the purpose, we employ a queue-based controller which is representative of this family. In Section III-A we provide a formal model of the closed-loop system. In Section III-B we prove that an asymptotically stable limit cycle exists and, based on this analysis, we derive some key properties quantifying the effect of the design parameters on the QoE. From now on, we assume a piecewise constant available bandwidth input function, which can be considered in practice a worst case scenario. Thus, without loss of generality, we analyze the system behavior in response to a step input of amplitude B.

A. Hybrid Model

The control goal of a generic adaptive streaming control system is to obtain: G1) full utilization of the available bandwidth at steady state, i.e. $r(t) = B$; G2) preventing re-buffering events ($q(t) = 0$) while keeping the queue as low as possible to minimize network bandwidth, client memory and to enforce liveness in the case of live streaming; G3) minimizing video level switches at steady state. It is easy to show that it is not possible to completely avoid video level switches at steady state while matching G1 and G2. However, when a video level switch from l_i to l_j occurs, the larger the distance between l_i and l_j the higher is the QoE impairment [12]. For this reason, though we cannot avoid switches at steady state, we should at least avoid switches between non-adjacent video levels. Thus, at steady state any optimal controller in the sense above defined has to switch the video level $l(t)$ between the two adjacent video levels $\underline{l}$ and $\overline{l}$ such that $\underline{l} < B < \overline{l}$. In particular:

$$\underline{l} = \underset{l \in \mathscr{L} \text{ s.t. } l < B}{\operatorname{argmax}\, l}\ , \tag{4}$$

$$\overline{l} = \underset{l \in \mathscr{L} \text{ s.t. } l > B}{\operatorname{argmin}\, l}\ . \tag{5}$$

Among all the optimal controllers, we consider the simplest controller yet capturing the desired optimal steady state behavior. In particular, the proposed controller employs two thresholds q_L and q_H ($q_L < q_H$). When $q(t)$ gets above the higher threshold q_H, the video level is increased by selecting $l(t) = \overline{l}$, whereas when $q(t) < q_L$ the video level is decreased to $\underline{l}$. When $q(t) \in [q_L, q_H]$, the video level is not changed. It is worth to notice again that, due to the sampled and quantized nature of the actuation on $l(t)$, any queue-based control law causes the oscillation of the queue level $q(t)$ within a range (excluding the trivial case that the bandwidth is equal to one of the video levels).

In the following we propose a hybrid dynamical model $\mathscr{H}$ of the considered system by employing the framework proposed in [8]. The formal model allows us to rigorously prove that the queue length keeps within the range $[q_L, q_H]$ at steady state regardless of the initial conditions. Moreover, it allows us to provide some key properties of the system.

Before defining the model, we have to consider some trivial cases. If $B \in \mathscr{L}$, there is no switching between adjacent levels at steady state. However, this is a purely mathematical condition that never holds in the practice: we exclude it by imposing that $B \notin \mathscr{L}$. If $q(t)$ grows above a threshold $q_{max} \gg q_H$, the control algorithm reacts with a safety mechanism by employing the ON-OFF pattern to reduce the rate and prevent the download of the entire video. Therefore, the proposed model holds only as long as $q(t) < q_{max}$.

Let us now define $\mathscr{H}$. The state of the system is given by $x = [q, l] \in X = [0, q_{max}] \times \mathscr{L}$. For convenience of notation we define the sets:

$$c_L = \{x \in X \,:\, q < q_L\}$$
$$c_H = \{x \in X \,:\, q > q_H\}$$
$$c_T = \{x \in X \,:\, q_L \leq q \leq q_H\}$$
$$c_{sup} = \{x \in X \,:\, l = \overline{l}\}$$
$$c_{inf} = \{x \in X \,:\, l = \underline{l}\}$$

The flow set C and jump set D are then given by:

$$C = (c_L \wedge c_{inf}) \vee c_T \vee (c_H \wedge c_{sup})$$
$$D = (c_L \wedge c_{sup}) \vee (c_H \wedge c_{inf})$$

The flow map is defined as:

$$f(x) = \left[\frac{B}{l} - 1, 0\right] \tag{6}$$

The jump map is given by:

$$g(x) = \begin{cases} [q, \underline{l}] & \text{if } x \in (c_L \wedge c_{sup}) \\ [q, \overline{l}] & \text{if } x \in (c_H \wedge c_{inf}) \end{cases}$$

B. System properties

The following theorem ensures that the queue length keeps within the range $[q_L, q_H]$ regardless of the initial conditions.

Theorem 1: The set $A = c_T$ is uniformly globally pre-asymptotically stable (UGpAS) for the hybrid system $\mathscr{H}$.

Proof: the proof is omitted due to space limitations. ■

We can further characterize the dynamical behavior of the system with the following simple proposition.

Proposition 1: Given $B \in (l_i, l_{i+1})$, the evolution of the queue of $\mathscr{H}$ at steady state is a triangular wave with *switching period given by:*

$$T_s = (q_H - q_L)\left(\frac{l_i}{B - l_i} + \frac{l_{i+1}}{l_{i+1} - B}\right). \tag{7}$$

Proof: the proof is omitted due to space limitations.

Figure 1 shows the limit cycle dynamics.

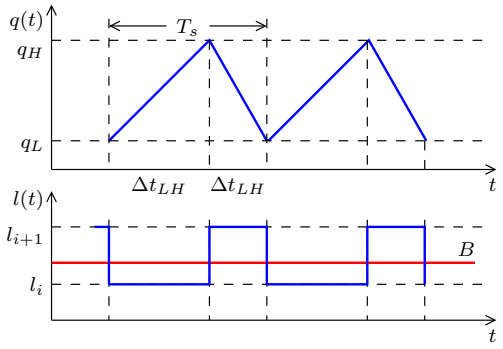

Fig. 1: Limit cycle

The level switching period T_s should be as large as possible to provide a good user experience. The following Proposition characterizes the worst case switching period, which corresponds to the minimum value taken by T_s. ■

Proposition 2: The minimum switching period $\overline{T}_s$ is given by:

$$\overline{T}_s = \frac{(q_H - q_L)D_i}{D_i + 2 - 2\sqrt{D_i + 1}}, \tag{8}$$

with $D_i = (l_{i+1} - l_i)/l_i$ (the video level relative distance), when $\overline{B} = \sqrt{l_i l_{i+1}}$.

Proof: the proof is omitted due to space limitations. ■

Remark 1: Eq. (8) can be employed to tune video levels and queue thresholds such that a target worst case switching period is obtained. Since the function (8) is monotonically decreasing with a vertical asymptote in $D_i = 0$, the worst case switching period decreases as the relative distance of two adjacent levels increases. Importantly, (8) also expresses that, if D_i is fixed, i.e. $D_i = D\ \forall i$, $\overline{T}_s$ is independent on i. This is a key design choice that provides a predictable performance across the entire range of bandwidths $B \in [l_0, l_{N-1}]$.

It is also interesting to compare it with the switching period computed in the case of a video level set $\mathscr{L} = \{l_0, l_0 + P, l_0 + 2P, \ldots, l_0 + (N-1)P\}$, which corresponds to set adjacent video levels with a constant distance of P. Since in this case $D_i = P/l_i$, we obtain from (8):

$$\overline{T}_s(l_i) = \frac{\Delta q}{P}(P + 2l_i + 2\sqrt{Pl_i + l_i^2}), \tag{9}$$

which means that $\overline{T}_s$ is not independent on i and can be approximated as a linear function when $l_i \gg P$.

IV. Optimal Video Level Design

Let us now consider the problem of the design of video level set $\mathscr{L}$. In an adaptive streaming system the minimum and the maximum video levels l_0 and l_M are a design requirement that fixes the range within which the actuation signal can vary. Moreover, by taking into account the Remark 1, we can proceed by fixing the target T_s and Δq[2] in (8) to get a unique value of D, which is independent on each l_i. Notice that D is bounded by $D_{max} = (l_M - l_0)/l_0$. Once D and l_0 are fixed, each video level l_i can be expressed as:

$$l_i = (1 + D)^i l_0 \tag{10}$$

The number N of levels is a function of D equal to:

$$N(D) = \left\lceil \frac{\log l_M - \log l_0}{\log(D+1)} \right\rceil + 1 \tag{11}$$

To give an example, let us take a typical use case (see for instance [7]) in which $l_0 = 300$kb/s (240p) and $l_M > 4500$kb/s (1080p) and we require a worst case period of 150s with a hysteresis range of $q_H - q_L = 15$s. From (8) we get roughly $D = 0.5$, which means that we need to use the following 8 levels:

$$\mathscr{L} = \{0.3, 0.45, 0.675, 1.0, 1.52, 2.28, 3.42, 5.12\}\text{Mb/s}.$$

However, the storage cost for a video of duration T is proportional to the sum of the size of all the video versions:

$$S(D) = T \sum_{k=0}^{N(D)-1} l_i = T l_0 \frac{(1+D)^{\left\lceil \frac{\log l_M - \log l_0}{\log(D+1)} \right\rceil + 1} - 1}{D} \tag{12}$$

Thus, we need to consider the trade-off between the minimization of the switching frequency, which is inversely proportional to D, and the minimization of the storage cost, which instead is increasing with D. To the purpose, we formulate the following optimization problem:

$$\min_{D \in (0, +\infty)} J(D) = S(D) + \alpha f_s(D),$$

where $J(D)$ is a one variable function with a freely adjustable weighting parameter $\alpha \geq 0$. We expect that, when α is small, the storage component dominates and large values of D will be obtained, meaning that few and far apart levels should be employed. In particular, when $\alpha = 0$, the trivial solution $\mathscr{L} = \{l_0, l_M\}$ is obtained. On the contrary, with increasing values of α the QoE component dominates and D gets smaller and smaller converging to 0.

Let us now analyze the derivatives of the two functions. The derivative of the switching frequency $f_s = 1/T_s$ is equal to:

$$\frac{\partial}{\partial D}\frac{1}{\Delta t} = \frac{1}{\Delta q}\frac{D + 2 - 2\sqrt{D+1}}{D^2\sqrt{D+1}}. \tag{13}$$

It can be shown that it is a monotonically decreasing function which is equal to $1/(4 \cdot \Delta q)$ for $D = 0$ and asymptotically converges to 0.

The derivative of (12) turns out to be:

$$\frac{\partial S}{\partial D} = \frac{1 - (1+D)^{\frac{\log l_M - \log l_0}{\log(D+1)}}}{D^2} \tag{14}$$

that is monotonically increasing and goes from $-\infty$ to 0.

$\frac{\partial S}{\partial D} + \alpha \frac{\partial 1/\Delta t}{\partial D} = 0$ has only one zero depending on α. Figure 2 shows the optimal value of D as a function of α for several values of $A = \log l_M - \log l_0$. In particular, the optimal value of D monotonically decreases converging to

[2]Recall that Δq cannot be made too large (see the control goal G2 in Section III-A).

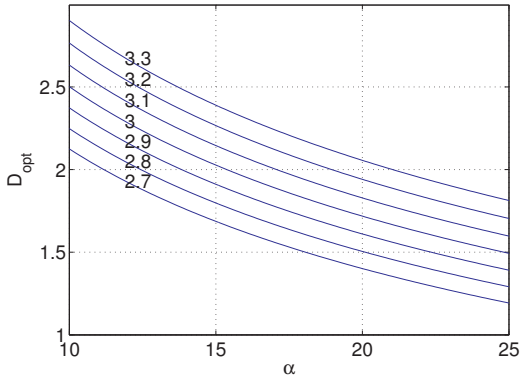

Fig. 2: Family of functions $D_{opt}(\alpha)$ when the parameter $A = \log l_M - \log l_0$ varies

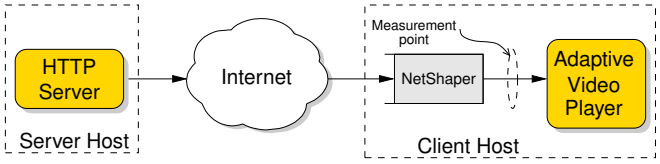

Fig. 3: Testbed setup.

0 when α goes to $+\infty$, due to the fact that with a small α the storage term dominates, whereas with a large α the switching frequency term dominates, as expected.

V. Simulations and Experimental Evaluation

In this section we validate the model and the properties of the *level-based actuation* control algorithm proposed in Section III by comparing numerical simulations with experimental data. Simulations of the proposed hybrid model have been done through the Matlab *Hybrid Equations (HyEq) Toolbox* [13]. The proposed control algorithm has been then implemented on the video player `Adaptive Video Player (AVP)` and tested over a real network scenario.

Figure 3 shows the employed testbed: an Apache HTTP server has been installed on a Debian Linux server, whereas a Ubuntu Linux client machine runs the `Adaptive Video Player (AVP)`. `AVP` has been implemented using the GStreamer[3] libraries and supports the HTTP Live Streaming (HLS) format[4]. In order to perform bandwidth shaping and to set propagation delays, we have used the tool `NetShaper`, a tool we have developed that is similar to `Dummynet`. The video sequence "Sintel"[5], encoded at five different bitrates $\mathscr{L} = \{300, 600, 900, 2500, 4000\}kb/s$, has been employed. In all the runs the available bandwidth B has been set to $1500kb/s$ unless otherwise specified.

Figure 4 compares the dynamics of the system state $x = [q(t)\ l(t)]^T$ in the case of simulations (left) and experimental runs (right) obtained when $\Delta q = 12$s. The simulated system accurately models the real control system. The only difference is that in the real system the queue gets slightly below (above) the lower (higher) threshold before a video level switch is triggered, whereas in the model the video level switch is triggered exactly when the queue gets equal to the thresholds. This difference is due to the fact that the model does not capture the presence of the chunks, which make the control actuation in the real system discrete.

[3]http://gstreamer.freedesktop.org/
[4]http://tools.ietf.org/html/draft-pantos-http-live-streaming-07
[5]http://www.sintel.org/

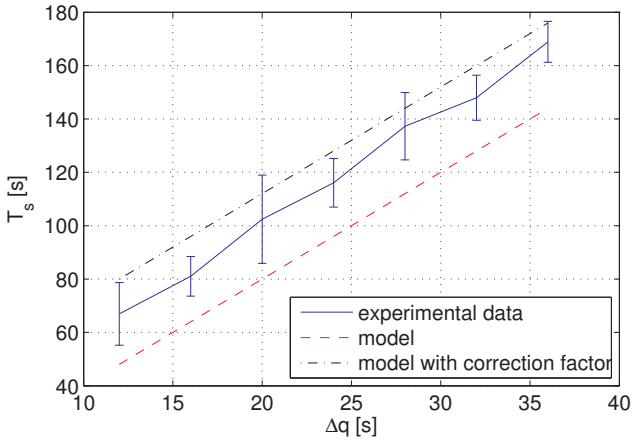

Fig. 5: Comparison between experimental and model data

We have then validated (7) on the real system. Several runs, each one with a different $\Delta q \in \mathscr{Q} = \{12, 16, 20, 24, 28, 32, 36\}$s, have been carried out. In each run the average switching period T_s has been measured. Figure 5 shows that (7) (dashed line) quite accurately fits the average measured T_s of the real system (solid line). In order to compensate for the discrete behavior of the real system due to the presence of the chunks, Δq in (7) has been increased by adding a correction factor equal to $2T_{chunk}$, where $T_{chunk} = 4$s is the duration of the single chunk. The dash-dotted line in Figure 5 shows that the corrected switching period T_s better fits the experimental data.

Finally, we have experimentally compared the worst case $\overline{T}_s$ obtained when video levels are designed in the same range $[300, 4000]$kb/s either according to the optimal procedure or the equally spaced one described in Section III-B. In these experiments the available bandwidth is set equal to the worst case bandwidth $\overline{B}$ for each couple of adjacent levels (l_i, l_{i+1}) according to $\overline{B} = \sqrt{l_i l_{i+1}}$. With the optimal procedure the set $\mathscr{L}_A = \{300, 573, 1094, 2090, 4000\}$kb/s is obtained. Figure 6 compares the average worst case measured $\overline{T}_s$ (thick solid line) to the one computed with (8) (thick dashed line). It has to be noticed that encoders are not able to produce video levels that match precisely the target levels for low bitrates. In fact, we have measured that the actual bitrates of l_0 and l_1 are equal to, respectively, 342kb/s and 595kb/s, with a relative error equal to 0.14 and 0.03 *wrt* the target levels. Despite of this, (7) nicely predicts the worst case switching period $\overline{T}_s$. With the equally spaced design procedure (thin line), $\mathscr{L}_B = \{300, 1225, 2150, 3075, 4000\}$kb/s is obtained and $\overline{T}_s$ grows approximately linearly, as expected from (9) (thin dashed line).

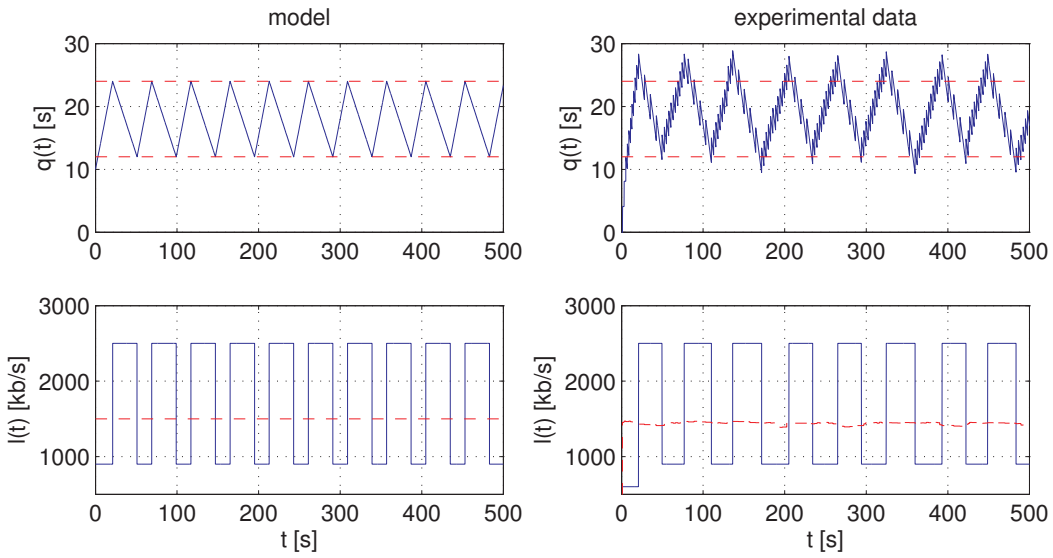

Fig. 4: Comparison between simulations (left) and experimental results (right)

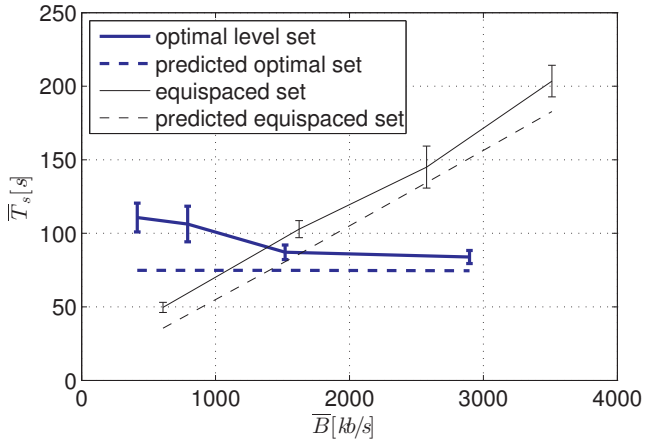

Fig. 6: Comparison between optimal level set $\mathscr{L}_A = \{300, 573, 1094, 2090, 4000\}$kb/s and equally spaced levels $\mathscr{L}_B = \{300, 1225, 2150, 3075, 4000\}$kb/s

VI. Conclusions

In this paper we have considered an important class of adaptive video streaming control systems motivated by its fundamental advantages in terms of bandwidth utilization and fairness. We have provided tuning rules of the system parameters to minimize video level switches, which is the main drawback of employing such class of controllers. To the purpose, we have proposed a hybrid dynamical model of a simple threshold-based controller that is representative of such a class. Based on the model, we have derived the minimum switching frequency at steady state as a function of the thresholds. Moreover, we have shown how to design the video level set $\mathscr{L}$ to obtain an optimal trade-off between switching frequency and storage cost requirements. The theoretical findings have been validated by comparing numerical simulations and experimental results.

References

[1] S. Akhshabi et al. An experimental evaluation of rate-adaptation algorithms in adaptive streaming over HTTP. In *Proc. of ACM MMSys 2011*, pages 157–168, 2011.

[2] S. Akhshabi et al. What Happens When HTTP Adaptive Streaming Players Compete for Bandwidth? In *Proc. of ACM NOSSDAV '12*, pages 9–14, 2012.

[3] S. Akhshabi et al. Server-Based Traffic Shaping for Stabilizing Oscillating Adaptive Streaming Players. In *Proc. of ACM NOSSDAV '13*, pages 19–24, 2013.

[4] Cisco. Cisco Visual Networking Index:Forecast and Methodology 2013-2018. 2013.

[5] L. De Cicco et al. ELASTIC: a Client-side Controller for Dynamic Adaptive Streaming over HTTP (DASH). In *Proc. of Packet Video Workshop '13*, December 2013.

[6] L. De Cicco et al. A Hybrid Model of the Akamai Adaptive Streaming Control System. In *Proc. of IFAC World Congress '14*, 2014.

[7] L. De Cicco and S. Mascolo. An Adaptive Video Streaming Control System: Modeling, Validation, and Performance Evaluation. *IEEE/ACM Transaction on Networking*, 22(2):526–539, April 2014.

[8] R. Goebel et al. *Hybrid Dynamical Systems: modeling, stability, and robustness*. Princeton University Press, 2012.

[9] T.Y. Huang et al. Confused, timid, and unstable: picking a video streaming rate is hard. In *Proc. of ACM IMC '12*, 2012.

[10] J. Jiang et al. Improving fairness, efficiency, and stability in HTTP-based adaptive video streaming with FESTIVE. In *Proc. of CoNEXT '12*, pages 97–108, 2012.

[11] T. Kupka et al. Performance of On-Off Traffic Stemming From Live Adaptive Segmented HTTP Video Streaming. In *Proc. of IEEE Conference on Local Computer Networks*, October 2012.

[12] X. Liu et al. A case for a coordinated internet video control plane. In *Proc. of ACM SIGCOMM 2012*, pages 359–370, 2012.

[13] R. Sanfelice et al. A toolbox for simulation of hybrid systems in Matlab/Simulink: hybrid equations (HyEQ) toolbox. In *Proc. of Hybrid Systems: Computation and Control '13*, pages 101–106, 2013.

[14] G. Tian and Y. Liu. Towards agile and smooth video adaptation in dynamic http streaming. In *Proc. of CoNEXT 2012*, pages 109–120, New York, NY, USA, 2012.