A Hybrid Model of Adaptive Video Streaming Control Systems

Giuseppe Cofano, Luca De Cicco, Saverio Mascolo

***Abstract*— Video streaming traffic over the Internet has significantly grown in the recent years. Adaptive video streaming control systems are employed to provide the best user experience given the user device and the network available bandwidth. The control goal is to maximize the video bitrate while avoiding playback interruptions. In this paper, we present a complete and accurate model of a generic adaptive streaming control system in the form of a hybrid dynamical system. The model describes all the system features, differently from previous models making the fluid-flow approximation, and allows to rigorously design video streaming controllers whose performance can be analytically assessed. The high accuracy of the model has been assessed by comparing numerical simulations to experimental data obtained through real network experiments. Given its accuracy and low computation cost, the proposed model provides a promising alternative to network experiments in order to aid the design and evaluation of adaptive video streaming systems.**

I. Introduction

Recent years have witnessed a remarkable change of the Internet traffic. If in the past years peer-to-peer traffic was dominant, today video streaming is considered the largest contributor of the global Internet traffic [5]. This phenomenon is mainly driven by the shift of a large portion of users which today prefer to consume video content over the Internet instead of using traditional broadcast TV channels. Consequently, video content providers are required to deliver a seamless multimedia experience across a heterogeneous mix of client devices (such as smart TVs, desktop PCs, smartphones) and access networks (such as wired cable/ADSL and wireless 3G/4G connections). When designing such systems, the Quality of Experience (QoE), i.e. the quality perceived by the user, has to be carefully taken into account due to its impact on user engagement, which is in turn directly connected to the revenues [9], [14]. For this reason, video streaming systems are required to design a control algorithm with the goal of maximizing the QoE under time-varying network conditions. Video delivery systems employing such control algorithms are defined as *adaptive video streaming systems.* Adaptivity can be implemented by throttling the video bitrate in real-time to track the time-varying and unpredictable Internet available bandwidth.

Due to its deployment and implementation advantages, the leading approach to implement adaptivity is the *stream-switching* (or *multi-bitrate*). The server encodes the video content at different bitrates, the *video levels*. Each video level is temporally divided into video segments of fixed duration. A playout buffer is employed at the client to absorb the instantaneous mismatches between the selected video bitrate and the network available bandwidth. The control algorithm dynamically selects the video level to be sent based on the knowledge of the state of the system. Such an approach is today used by all major video streaming services such as YouTube, Netflix, Hulu, Vudu, Livestream, and Akamai [4], and adopted by the two main adaptive streaming standards, i.e. the MPEG Dynamic Adaptive Streaming over HTTP (MPEG-DASH) and the HTTP Live Streaming (HLS).

The adaptive streaming control system has two fundamental goals: 1) to avoid the depletion of the playout buffer (rebuffering event), which has been shown to be highly detrimental for the QoE and the user engagement [3]; 2) to select the highest possible bitrate, possibly matching the available bandwidth. Several adaptive streaming controllers have been proposed in the literature, usually resorting to heuristic-based design whose performance can only be analyzed via experimental evaluations. The only models proposed in the literature make the fluid-flow approximation, which consists in assuming video segments of infinitesimal size. However, such models cannot describe the following features of the real system: 1) the playout buffer dynamics is characterized by a time-continuous draining process driven by the video player and an impulsive filling triggered by segments download completion; 2) the shaping of the received rate by inserting OFF periods, which is employed by an important class of controllers [6], [12]; 3) the mismatch between the encoded nominal video bitrates and the effective bitrates of each video segment.

In this work we make the following contributions. First, from a modeling standpoint, we propose a complete model of the control system in the form of a hybrid dynamical system overcoming all the above mentioned limitations. The model has been built by employing the well-established modeling framework for hybrid dynamical system presented in [10]. It allows to rigorously design video streaming controllers whose performance can be analytically assessed. Second, from a technological point of view, we show how to implement the proposed model and employ it as a tool which accurately simulates the behavior of the real system. The tool has a very low computational cost which allows carrying out simulations that are two orders of magnitude faster than the experimental runs. Considered the high costs generally involved to carry out experimental evaluations, which do not allow an exhaustive coverage of the whole design parameters space, our tool provides a promising alternative to aid the design and evaluation of adaptive video streaming systems.

The authors are with the Dipartimento di Ingegneria Elettrica e dell'Informazione, Politecnico di Bari, Via Orabona 4, Bari, Italy. Emails: {name.surname}@poliba.it

This work has been partially supported by the "Future in Research" project no. ACYBEH5 funded by the Apulia Region, Italy.

II. Background

In this Section we provide background material focusing on HTTP adaptive video streaming control systems. The controllers can be classified based on their actuation variables [6]: (i) algorithms acting by limiting the received rate at the client by means of OFF periods are denoted as *rate-based*, whereas (ii) algorithms acting by adapting only the video level are denoted as *level-based*.

In the case of the *rate-based* approach OFF periods are inserted between the downloads of consecutive segments in order to shape the average received rate. The rationale is to set the received rate equal to (on average) the selected video level in order to avoid video level switches. Moreover, if the end-to-end bandwidth is constant, the queue tracks the set point. Despite its simplicity, this approach has two major drawbacks: (i) the available bandwidth is always underutilized; (ii) it has been experimentally shown that the ON-OFF traffic pattern causes the video flows to obtain a significantly smaller bandwidth share with respect to the fair one when competing with long-lived TCP flows [1], [12]. The first issue can degrade the perceived QoE remarkably in case the distance between the levels is large. The second issue, that is known in the literature as the *downward spiral effect*, can lead to an even worse degradation of the perceived QoE when concurrent long-lived TCP flows share the bottleneck with the video flow [1], [12]. Among the controllers proposed to address the described issues we mention [2], [18], [13].

In the case of the *level-based* approach (see for instance [7]), the video segments are downloaded back to back, thus eliminating the ON-OFF traffic pattern. In this way, video flows behave as any other TCP long-lived flow and, as a consequence, full utilization and fairness with TCP long-lived flows are achieved by design. The control is done by throttling the video bitrate in order to avoid rebuffering events. Well-known feedback control techniques such as PI regulators, Model Predictive Control, and feedback linearization can be employed [7], [17]. The drawback of this approach is that at steady state video level switches occur even when the available bandwidth is constant since the video bitrate belongs to a discrete set and cannot exactly match the available bandwidth.

III. Adaptive Video Streaming Control System

A. Overview

In this Section we describe a general adaptive video streaming system, in which a *client* plays a video that is sent by a remote *server* over the Internet. Such systems use the *stream-switching* (or *multi-bitrate*) approach to allow the video bitrate to be varied: the server encodes and stores the video content at different bitrate levels $l_1, \ldots, l_N$ forming a discrete set $\mathcal{L} = \{l_1, l_2, \ldots, l_N\}$. Each video level is temporally divided into a number of segments of fixed duration. A control algorithm dynamically selects 1) the video level to be streamed at each segment download and 2) the duration of idle times (OFF periods) inserted to shape the received rate. The video client employs a playout buffer to absorb the instantaneous mismatches between the selected video bitrate and the network available bandwidth that in best-effort Internet is unpredictable and time-varying.

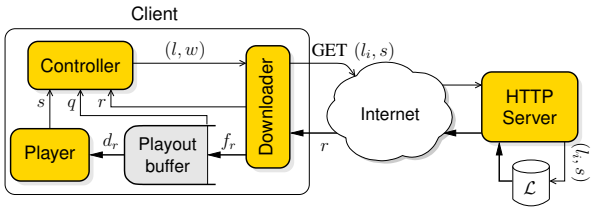


Fig. 1: An adaptive video streaming system

Fig. 1 shows the architecture of an adaptive video streaming system: control signals and data flows are represented with thin and thick lines respectively. In a nutshell, the client is made of four components: (i) a *controller* that computes the video level bitrate and the idle times; (ii) a *downloader* that retrieves the video segments from the server at the video level bitrate computed by the controller; (iii) a *playout buffer* that temporarily stores the video segments retrieved by *the* downloader; (iv) a *player* that drains the playout buffer, decodes the video frames, and renders the video on the screen.

In the following, we describe the workflow of a video streaming session by showing the interactions between components and their dynamics.

B. The Video Encoding

In adaptive video streaming systems, the video encoder generates N versions of the same video clip, each of them corresponding to a nominal encoding bitrate l_i. Each video version is made of video frames of fixed duration (usually a few milliseconds), which are the minimal part of information that a video player can process. Video frames are grouped to form video segments, which are the minimum transport unit and have a fixed playback duration (usually of few seconds). A video segment is identified by its index $s \in \mathcal{S}$, where $\mathcal{S} = \{1, 2, \ldots, \lfloor \hat{T}/\hat{\tau} \rfloor\}$ is the set of video segment indices, $\hat{\tau}$ is the segment duration and $\hat{T}$ is the video clip duration. The set of encoded video bitrates is depicted in Fig. 2. It is worth to notice that the effective encoding bitrate of each segment (height of each segment in Fig. 2) can be different from the nominal bitrate l_i set at the video encoder (dashed line in the Fig. 2). The effective encoded bitrate of the i-th segment is equal to $S_{i,s}/\hat{\tau}$, where $S_{i,s}$ is the size, measured in bytes, of the s-th segment of the i-th video level.

C. The Controller

Control laws are designed with the overall goal of maximizing the user QoE. Even though a quantitative model of the QoE based on key performance parameters is still missing [16], it is today well-known that quality degradation is due to, in decreasing order of importance, the following factors: (i) rebuffering events [3]; (ii) low average video bitrate; (iii) video bitrate oscillations. The control law $u(\cdot)$ is made of two components: 1) $u_1(\cdot)$ is the *video bitrate* $l \in \mathcal{L}$ to be downloaded; 2) $u_2(\cdot)$ is the *idle time* w to wait

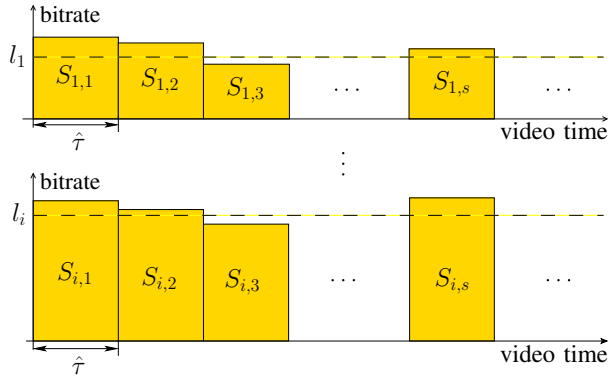


Fig. 2: The mismatch between nominal and effective segments bitrates for the levels $l_1, \ldots, l_i$

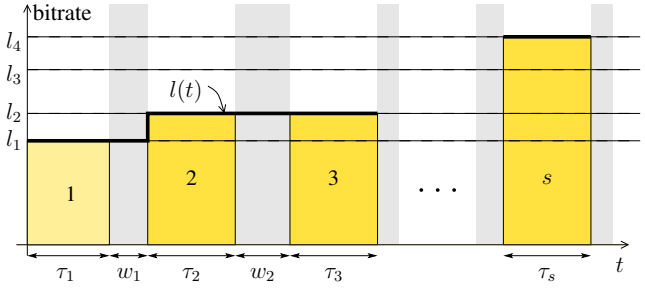


Fig. 3: The video bitrate selection in an adaptive video streaming system. OFF periods are depicted in gray.

before downloading the next segment, which is employed to shape the received rate.

The control law is computed after the completion of the segment download. The selected bitrate assumes values in the discrete set $\mathcal{L}$, thus $l(t)$ is a piece-wise constant signal. The idle time w belongs to $\mathbb{R}_{\geq 0}$. Controllers can be classified as (i) *level-based*, which download segments back to back and set the idle time equal to 0 (i.e. they do not shape the received rate and (ii) *rate-based*, which update the idle time to shape the received rate. An example of a generic control action is shown in Fig. 3.

D. The Downloader

The downloader component is the actuator of the control system. It retrieves from the server the video segments at the bitrate decided by the controller and keeps track of the progression of s. Let us define t_s as the time instant at which the download of the segment s starts. The downloader measures the download time τ_s, i.e. the time elapsed from t_s to the completion of the download of segment s. Then, the downloader waits w_s seconds before starting to retrieve the next segment $s+1$. As a consequence, it holds

$$t_{s+1} = t_s + \tau_s + w_s.$$

E. The Player

The player is the component that drains the playout buffer, decodes, and renders the video on the user screen. It is worth noting that if the controller works at segments timescale, the player works at frames timescale, i.e., the minimum amount of video drained by the player is one frame. Since frame duration is negligible compared to segment duration[1], we consider the player to drain the buffer with continuity. The player has two possible states: (i) *playing*: during this state the player drains the playout buffer, meaning that for any given time interval of duration ΔT, ΔT seconds worth of video are drained by the playout buffer; (ii) *paused*: during this state the playback is stopped and no video is drained from the playout buffer; the player is in such state either when the buffer is empty or during the initial phase of a video streaming session.

Thus, the player draining rate $d_r(t) = d\hat{\tau}/dt$, defined as the duration of video $d\hat{\tau}$ drained in a time interval dt, can be modeled as follows

$$d_r(t) = \begin{cases} 1 & \text{playing,} \\ 0 & \text{paused.} \end{cases} \tag{1}$$

F. The Playout Buffer

The playout buffer is employed to store the video content. Its length $q(t)$ is the video duration measured in seconds. This means that, if the filling rate is zero, the video playback that can still be ensured for $q(t)$ seconds. The playout buffer is fed by the downloader and drained by the player. The playout buffer dynamics is characterized by a time-continuous draining process driven by the video player according to (1) and an impulsive filling of amplitude $\hat{\tau}$ triggered by segments download completion, which occurs at time $t_s + \tau_s$ for each segment s.

Thus, the net increment of the queue length due to the download of s-th segment is given by

$$q(t_s + \tau_s) - q(t_s) = \hat{\tau} - \int_{t_s}^{t_s+\tau_s} d_r(\xi) d\xi \tag{2}$$

since in τ_s seconds the queue has been filled by a quantity $\hat{\tau}$ and drained by

$$\int_{t_s}^{t_s+\tau_s} d_r(\xi) d\xi \tag{3}$$

seconds according to integration of (1).

IV. The Model of the Control System

In this Section we present a mathematical model of a generic adaptive video streaming control system. To the purpose, the hybrid dynamical systems framework proposed in [11] has been employed. Such a mathematical framework allows modeling dynamical systems which combine time-continuous and time-discrete dynamics [10].

The adaptive video streaming system presented in Section III is made of two components which form a classic closed-loop system: the *plant* to be controlled which is composed of the player, the downloader, and the playout buffer presented in Section III, and the *controller* computing the control law $u(\cdot)$.

Before describing the model, let us briefly introduce the hybrid dynamical system framework. Further details are

[1]The duration of one frame is in the range 16ms-40ms whereas segments are typically in the range 1s-10s.

providede in [11]. The key idea of this framework is to separate the continuous part of the dynamics of a dynamical system from its discrete part.

A hybrid system $\mathscr{H}$ can be described using four elements $(\mathcal{C}, \mathcal{D}, f, g)$

$$\mathscr{H} : \begin{cases} \dot{x} = f(x), & x \in \mathcal{C} \\ x^+ = g(x), & x \in \mathcal{D} \end{cases}$$

where x is the state, $\mathcal{C}$ is the *flow set* where x evolves according to a continuous equation, f is a function describing the continuous evolution of the state, $\mathcal{D}$ is the *jump set* where jumps are enabled, g is a function describing the discrete evolution of the state. When x belongs to the flow set $\mathcal{C}$, the system evolves with a time-continuous dynamics according to the differential equations f (*flow map*). When x belongs to the jump set $\mathcal{D}$, the system evolves with a time-discrete dynamics according to the finite-difference equation g (*jump map*).

A. The Hybrid Model of the Playout Buffer

In this Section the hybrid model $\mathscr{B}$ of the playout buffer dynamics is proposed. The state of the system is given by $x = [q, l, D, d_r, \tau, w, s, \sigma]^T \in \mathcal{X} \subseteq \mathbb{R}^8$, where: 1) q is the playout buffer length, which is measured in seconds; 2) l is the selected video bitrate taking values in $\mathcal{L}$; 3) D is the amount of bytes of the current segment which have already been downloaded; 4) d_r is the player draining rate; 5) τ is the timer tracking the current segment download time; 6) w is the duration of the OFF period; 7) s is the current segment index; 8) σ is a logical state, which has value 1 during ON periods and 0 during OFF periods. The system has three input variables: the control laws $u_1(\cdot)$ and $u_2(\cdot)$, the received rate $r(t)$. The control laws are computed by the controller, the received rate is considered as a disturbance.

For convenience of notation, we define the following sets:

$$\begin{aligned} \mathcal{D}_1 &= \{x \in \mathcal{X} : D = S_{i,s}\}, \\ \mathcal{D}_2 &= \{x \in \mathcal{X} : \tau = w \wedge \sigma = 0\}, \\ \mathcal{D}_3 &= \{x \in \mathcal{X} : q = 0\}, \\ \mathcal{D}_4 &= \{x \in \mathcal{X} : q = q_{\min} \wedge d_r = 0\}. \end{aligned}$$

The condition defining $\mathcal{D}_1$, i.e. the downloaded bytes D are equal to the current segment size $S_{i,s}$, represents the event occurring at the end of an ON period, which corresponds to the completion of a video segment download. The set $\mathcal{D}_2$ represents the event occurring when an OFF period ends, i.e. the timer τ is equal to the idle time w set by the controller. The condition defining $\mathcal{D}_3$ is met when a rebuffering event occurs, i.e. the playout buffer gets empty. The condition defining $\mathcal{D}_4$ is met when the playout buffer has been filled again after a rebuffering event, i.e. $q(t)$ gets above a configured threshold $q_{\min}$ and the player is not active ($d_r = 0$) because of the rebuffering.

In our proposed model the flow set $\mathcal{C}$ and jump set $\mathcal{D}$ are given by

$$\mathcal{D} = \bigcup_{i=1}^{4} \mathcal{D}_i \ ; \ \mathcal{C} = \mathcal{X} \setminus \mathcal{D}.$$

The flow map f is defined as

$$f(x) = \begin{cases} \begin{bmatrix} -d_r, & 0, & r, & 0, & 1, & 0, & 0, & 0 \end{bmatrix}^T, & \sigma = 1 \\ \begin{bmatrix} -d_r, & 0, & 0, & 0, & 1, & 0, & 0, & 0 \end{bmatrix}^T, & \sigma = 0 \end{cases}$$

where f_1 models the time-continuous playout buffer depletion, i.e $\dot{q} = -d_r$; f_3 models $\dot{D} = r$, following from $D(t) = \int_{t_s}^{t} r(\zeta) d\zeta$, i.e. the amount of downloaded bytes of the current segment is equal to the integral of the received rate r; f_5 models the timer, i.e. $\dot{\tau} = 1$; finally, f_2, f_4, f_6, f_7, f_8 do not vary during the time-continuous dynamics. In case the control law employs OFF periods, the received rate is set to 0 during such intervals ($\sigma = 0$), since the download process is paused. It is worth to notice that the dynamics of the received rate r can be freely set to simulate any rate pattern. A model of the interaction between r and the available bandwidth B can be plugged in f_3 to analyze underutilization issues (i.e. $r < B$) that might affect video streaming controllers, as shown in [1], [12]. The modeling of such an interaction, however, is outside the scope of this work.

The jump map g is given by

$$\begin{aligned} g_1(x) &= [q + \hat{\tau},\, u_1(\cdot),\, 0,\, d_r,\, 0,\, u_2(\cdot),\, s + 1,\, 0]^T, \\ g_2(x) &= \begin{bmatrix} q, & l, & D, & d_r, & 0, & w, & s, & 1 \end{bmatrix}^T, \\ g_3(x) &= \begin{bmatrix} q, & l, & D, & 0, & \tau, & w, & s, & \sigma \end{bmatrix}^T, \\ g_4(x) &= \begin{bmatrix} q, & l, & D, & 1, & \tau, & w, & s, & \sigma \end{bmatrix}^T. \end{aligned}$$

Let us analyze separately each jump map g_i which is triggered when $x \in \mathcal{D}_i$.

g_1 is triggered when c_1 is met, i.e., a segment has been downloaded. Thus, the queue q is increased by the segment duration $\hat{\tau}$, the control law $u(\cdot)$ is executed to select the bitrate l of the next segment through $u_1(\cdot)$ and the duration w of the OFF period through $u_2(\cdot)$, the amount of downloaded bytes D and the timer τ are reset to 0, the player draining rate is left to its current value d_r, the logical state σ is set to OFF (i.e. 0), the index of the current segment s is increased by 1.

g_2 is triggered when c_2 is met, i.e., the OFF period ends. All variables are left in their current states except for the timer τ, which is reset to 0, and the logical state of the system σ, which is set to ON, i.e. 1.

g_3 is triggered when c_3 is met, i.e., when a rebuffering event occurs. All variables are kept in their current states except for the player draining rate d_r, which is set to 0. With this choice the playout buffer is not drained anymore in the time-continuous dynamics, allowing to be refilled.

g_4 is triggered when c_4 is met, i.e., when the playout buffer has been refilled after a rebuffering event. All variables are kept in their current states except for the player draining rate d_r, which is set to 1 to resume the playing.

V. Experimental Validation

In this Section we validate the hybrid model $\mathscr{B}$ proposed in Section IV. The validation is carried out by comparing

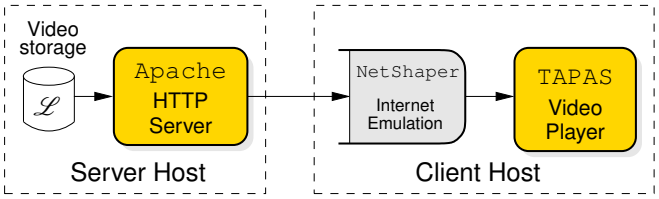


Fig. 4: Testbed employed for the experimental evaluation

numerical simulations with experimental data. Moreover, we have compared the hybrid model $\mathscr{B}$ to the fluid-flow model in order to show the improved accuracy of the proposed model.

The fluid-flow model is described in details in [6]. In a nutshell, the playout buffer dynamics is simply described by $\dot{q} = r/l - d_r$. The model is remarkably simpler, yet it does not capture important features such as the impulsive playout buffer filling, the OFF periods, and the mismatch between nominal and effective video bitrates.

The simulations have been carried out by implementing the models with the Matlab *Hybrid Equations (HyEq) Toolbox* [15], which is capable of simulating individual and interconnected hybrid systems. Regarding the network experiments, instead, the control algorithms have been implemented using `TAPAS` [8], an open-source tool written in Python that allows video streaming control algorithms to be implemented and tested in real networks.

Fig. 4 shows the employed testbed that is composed of two hosts connected through a 1 Gbps switch: the *server host* is a workstation with a Debian Linux operating system equipped with the software `Apache`[2] that acts as the HTTP server; the server host also stores the video sequence "Sintel"[3], encoded at five nominal bitrates $\mathcal{L} = \{240, 500, 900, 1400, 2600\}$ kbps; the *client host* is a Debian Linux machine that runs the `TAPAS` tool in order to download video segments from the HTTP server and play the video; the `TAPAS` tool collects several variables such as the video level $l(t)$ selected by the controller and the playout buffer length $q(t)$ and stores them in log files [8]; moreover, the client host runs `NetShaper`, a tool developed by us that allows setting the bandwidth and the delay of the link connecting the client to the server host to emulate an Internet connection. We have tested step-like bandwidth changes. In all runs, which last 400 s, the bandwidth $B(t)$ is either increased (step-up) or decreased (step-down) after 200 s following a step function. In order to show the accuracy of the proposed model for several control laws, we have considered two controllers, the *Hysteresis* controller and the *Conventional* controller, which are representative respectively of the *level-based* and *rate-based* approaches [6]. The *Hysteresis* controller, proposed in [6], dynamically updates the video level based on the bandwidth estimate and the queue length so that the average video bitrate matches the received rate. The playout buffer length is kept within two thresholds, q_L and q_H, which have been set, respectively, to 12 s and 24 s. The validation of the *Conventional* controller, however, is not shown in the following due to space constraints.

The results obtained with the Hysteresis controller are shown in Fig. 5 and Fig. 6. We point out that further experiments were carried out during the validation process, but we show only two of them due to space constraints.

Let us first examine the experiment considering a bandwidth that drops from 2000 kbps to 1200 kbps at $t = 200$ s. Fig. 5 shows the dynamics of the queue length and the selected bitrate for the fluid-flow model (Fig. 5 (a)), the proposed hybrid model $\mathscr{B}$ (Fig. 5 (b)), and the real system (Fig. 5 (c)). The dashed lines represent the thresholds q_L and q_H in top figures and the available bandwidth $B(t)$ in bottom figures. The figure shows that the dynamics of the model $\mathscr{B}$ accurately matches the real system dynamics gathered through a real network experiment. The accuracy improvement of $\mathscr{B}$ compared to the fluid-flow model is clear. In fact, it can be observed that, in the case of the fluid-flow model, at 200 s a switch-up event occurs, setting the video level to $l_5 = 2600$ kbps. On the other hand, both the hybrid model $\mathscr{B}$ and the real system do not exhibit this event. In summary, this scenario shows that the proposed hybrid model $\mathscr{B}$ is able to accurately model the mismatch between nominal and effective bitrates.

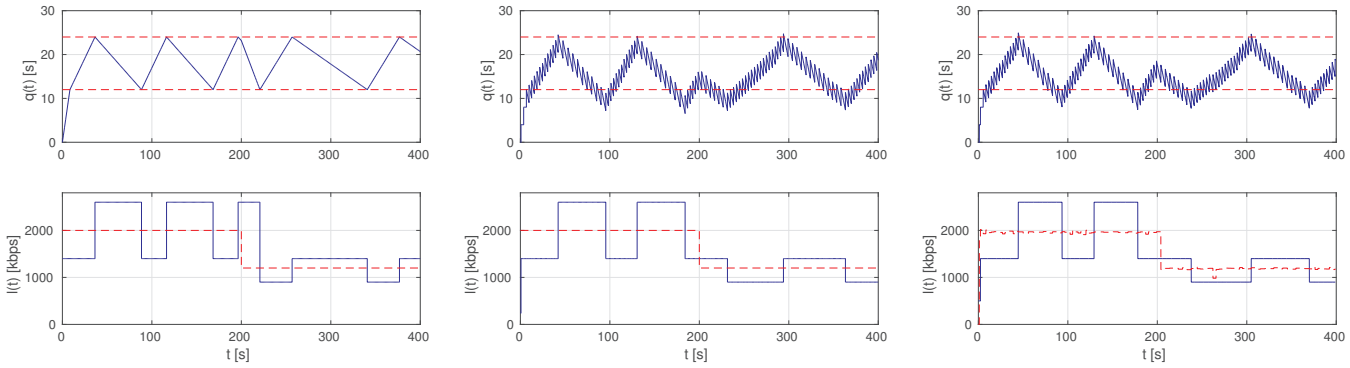
Let us now consider the experiments with step-up bandwidth change, shown in Fig. 6. The bandwidth is increased from 1200 kbps to 2000 kbps. The fluid-flow model provides a slightly better approximation of the real system dynamics compared to the previous scenario, even though a switch-up to $l_4 = 1400$ kbps is triggered at around 170 s, which is instead not exhibited by the real system. On the other hand, the dynamics of the proposed model is almost equivalent to the one of the real system.

The validation has shown that the proposed hybrid model is accurate in reproducing the dynamics of the real system and clearly outperforms the fluid-flow model. The benefits of the hybrid model are particularly evident when the actual video bitrate is close to the received rate.

VI. Conclusions

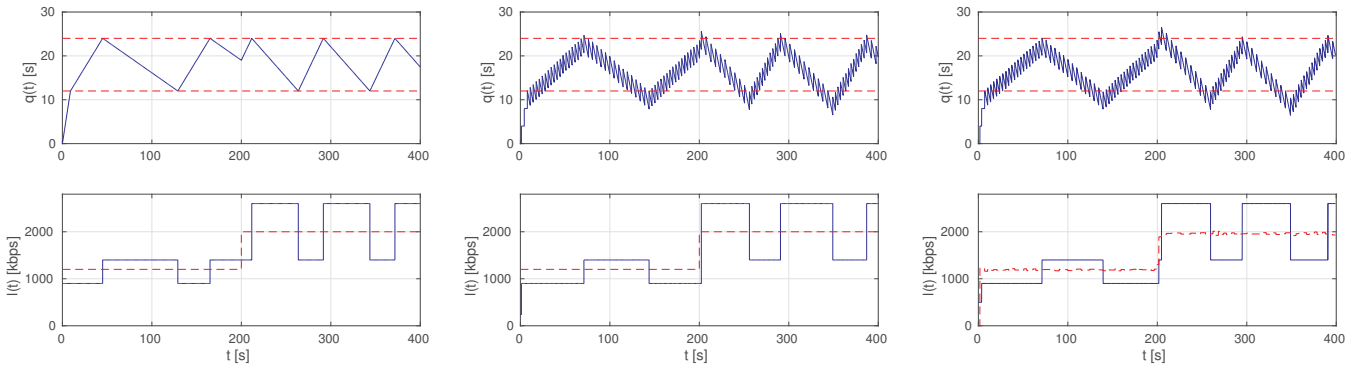
In this work we have proposed a complete and accurate model of a generic adaptive streaming control system in the form of a hybrid dynamical system. Among the improvements of the proposed model over the fluid-flow model previously proposed in the literature, we mention: (i) the accurate modelling of the hybrid playout buffer dynamics made of a time-continuous draining process and an impulsive filling process; (ii) the possibility to model *rate-based* controllers which shape the received rate by inserting OFF periods; (iii) the possibility to model the effect of the mismatch between the encoded nominal video bitrates and the actual bitrates of each video segment. The validation of the proposed model has been carried out by comparing simulations results to experimental data obtained through a testbed implementing a real video streaming system. The proposed model allows to carry out simulations that are two orders of magnitude faster than the experimental runs

[2] `http://httpd.apache.org/`
[3] `http://www.sintel.org/`



(a) Fluid-flow model (b) Hybrid model $\mathscr{B}$ (c) Real system

Fig. 5: Dynamics of the system with the Hysteresis controller and bandwidth drop from 2000 kbps to 1200 kbps



(a) Fluid-flow model (b) Hybrid model $\mathscr{B}$ (c) Real system

Fig. 6: Dynamics of the system with the Hysteresis controller and bandwidth increase from 1200 kbps to 2000 kbps

with very low implementation effort. We argue that this low computational cost is particularly beneficial during the development cycle when the designer has to iterate through the design of the control system, the parameters tuning, and the performance evaluation.

REFERENCES

[1] S. Akhshabi, L. Ananthakrishnan, A. C. Begen, and C. Dovrolis. What Happens When HTTP Adaptive Streaming Players Compete for Bandwidth? In *Proc. of ACM NOSSDAV*, pages 9–14, 2012.

[2] S. Akhshabi, L. Ananthakrishnan, A. C. Begen, and C. Dovrolis. Server-Based Traffic Shaping for Stabilizing Oscillating Adaptive Streaming Players. In *Proc. of ACM NOSSDAV*, pages 19–24, 2013.

[3] A. Balachandran, V. Sekar, A. Akella, S. Seshan, I. Stoica, and H. Zhang. Developing a predictive model of quality of experience for internet video. In *Proc. of ACM SIGCOMM*, pages 339–350, 2013.

[4] L. De Cicco, S. Mascolo, and C. T. Abdallah. An experimental evaluation of Akamai adaptive video streaming over HSDPA networks. In *2011 IEEE International Symposium on Computer-Aided Control System Design (CACSD)*, pages 13–18, Sept 2011.

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

[6] G. Cofano, L. De Cicco, and S. Mascolo. Characterizing adaptive video streaming control systems. In *Proc. American Control Conference*, pages 2729–2734, 2015.

[7] L. De Cicco, V. Caldaralo, V. Palmisano, and S. Mascolo. ELASTIC: a Client-side Controller for Dynamic Adaptive Streaming over HTTP (DASH). In *Proc. of Packet Video Workshop*, pages 1–8, 2013.

[8] L. De Cicco, V. Caldaralo, V. Palmisano, and S. Mascolo. TAPAS: A Tool for rApid Prototyping of Adaptive Streaming Algorithms. In *Proc. of Workshop on Design, Quality and Deployment of Adaptive Video Streaming (VideoNext)*, pages 1–6, 2014.

[9] F. Dobrian, V. Sekar, A. Awan, I. Stoica, D. Joseph, A. Ganjam, J. Zhan, and H. Zhang. Understanding the impact of video quality on user engagement. In *Proc. of the ACM SIGCOMM*, pages 362–373, 2011.

[10] R. Goebel, R. G. Sanfelice, and A. R. Teel. Hybrid dynamical systems. *IEEE Control Systems Magazine*, 29(2):28–93, 2009.

[11] R. Goebel, R. G. Sanfelice, and A. R. Teel. *Hybrid Dynamical Systems: modeling, stability, and robustness*. Princeton University Press, 2012.

[12] T.Y. Huang, N. Handigol, B. Heller, N. McKeown, and R. Johari. Confused, timid, and unstable: picking a video streaming rate is hard. In *Proc. ACM IMC*, 2012.

[13] J. Jiang, V. Sekar, and H. Zhang. Improving fairness, efficiency, and stability in HTTP-based adaptive video streaming with FESTIVE. In *Proc. of ACM CoNEXT*, pages 97–108, 2012.

[14] S. S. Krishnan and R. K. Sitaraman. Video stream quality impacts viewer behavior: inferring causality using quasi-experimental designs. *IEEE/ACM Transactions on Networking*, 21(6):2001–2014, 2013.

[15] R. Sanfelice, D. Copp, and P. Nanez. A toolbox for simulation of hybrid systems in Matlab/Simulink: hybrid equations (HyEQ) toolbox. In *Proc. of Hybrid Systems: Computation and Control*, 2013.

[16] M. Seufert, S. Egger, M. Slanina, T. Zinner, T. Hossfeld, and P. Tran-Gia. A Survey on Quality of Experience of HTTP Adaptive Streaming. *IEEE Communications Surveys and Tutorials*, 17(1):469–492, 2015.

[17] X. Yin, V. Sekar, and B. Sinopoli. Toward a Principled Framework to Design Adaptive Streaming Algorithms over HTTP. In *Proc. of the ACM Hotnets*, 2014.

[18] L. Zhi, X. Zhu, J. Gahm, R. Pan, H. Hu, A. C. Begen, and D. Oran. Probe and adapt: Rate adaptation for HTTP video streaming at scale. *IEEE JSAC*, 32(4):719–733, 2014.