DASH Adaptation Algorithm Based on Adaptive Forgetting Factor Estimation

Miguel Aguayo , Luis Bellido , Carlos M. Lentisco , and Encarna Pastor

***Abstract*—The wide adoption of multimedia service-capable mobile devices, the availability of better networks with higher bandwidths, and the availability of platforms offering digital content has led to an increasing popularity of multimedia streaming services. However, multimedia streaming services can be subject to different factors that affect the quality perceived by the users, such as service interruptions or quality oscillations due to changing network conditions, particularly in mobile networks. Dynamic Adaptive Streaming over HTTP (DASH), leverages the use of content-distribution networks and the capabilities of the multimedia devices to allow multimedia players to dynamically adapt the quality of the media streaming to the available bandwidth and the device characteristics. While many elements of DASH are standardized, the algorithms providing the dynamic adaptation of the streaming are not. The adaptation is often based on the estimation of the throughput or a buffer control mechanism. In this paper, we present a new throughput estimation adaptation algorithm based on a statistical method named Adaptive Forgetting Factor (AFF). Using this method, the adaptation logic is able to react appropriately to the different conditions of different types of networks. A set of experiments with different traffic profiles show that the proposed algorithm improves video quality performance in both wired and wireless environments.**

***Index Terms*—Adaptive streaming over HTTP, adaptive forgetting factor, mobile communication, multimedia content delivery, throughput estimation.**

I. Introduction

THE Internet traffic nowadays is mostly real-time entertainment traffic (audio and video). A recent Internet usage report [1] shows that real-time entertainment traffic consumes 65.35 percent of the Internet backbone aggregate traffic. Video streaming traffic is expected to experiment a growth of 67 percent in mobile and 29 percent in fixed networks [2] in the future. The explosion of multimedia content has driven the industry and research community to create protocols and architectures to deliver video services to all users. The 3rd Generation Partnership Project (3GPP) has defined the use of Dynamic Adaptive Streaming over HTTP (DASH) [3] as the standard for multimedia delivery in mobile networks, specifically in Long Term Evolution (LTE) networks. In a previous work [4] we have proposed solutions to improve the delivery of a DASH-encoded multimedia content that is broadcast over LTE. In this paper, the focus is on multimedia unicast services based on DASH.

One of the advantages of using HTTP is that standard HTTP servers and content distribution techniques can be reused for storing and delivering multimedia content. The DASH standard defines how a multimedia content can be divided into small files or “chunks”, and how to store the description of chunks in a metadata file named Media Presentation Description (MPD) so a multimedia player can retrieve both metadata and the sequence of chunks using HTTP to play the multimedia content.

Different representations make it possible that the same content can be retrieved with different qualities, depending on the user terminal or on the available bandwidth. For each representation the MPD file contains information about media type, codec, video width and height, frame rate, average bitrate, and so on. The actual media files, or media segments, are identified by Uniform Resource Locators (URL).

A DASH player implements an adaptation algorithm monitoring network conditions (bandwidth, delay, and so on) and selecting the representation to be downloaded. The aim of the adaptation algorithm is to ensure that the client selects a representation with the most appropriate bitrate to obtain the highest quality, while avoiding stalling events during the media playback. A guideline with remarks on possible client behavior is provided as an annex in [5] but adaptation algorithms are not standardized.

Substantial research exists in adaptive algorithms for DASH. Many of the algorithms have been designed for specific scenarios, e.g., mobile wireless networks, so to establish the parameters to achieve the highest quality allowed by the available network conditions. However, the same adaptation algorithm in a different scenario can overestimate or underestimate the available network bandwidth. When an adaptation algorithm is estimating more bandwidth than the network is offering, video stalling happens, because of buffer under-runs. On the other hand, if an adaptation algorithm underestimates the network bandwidth, the video player retrieves video qualities which are lower than what the network conditions permit, hence affecting the quality perceived by the user.

In this paper, we propose to use the Adaptive Forgetting Factor (AFF) method to improve throughput estimation in DASH adaptation algorithms. Our proposal is based on the capability of AFF to quickly adapt to short-term fluctuations of the bandwidth, especially in wireless networks. Using AFF as a throughput estimation technique, a DASH multimedia player will be able to achieve better quality in the video playback in both wired and wireless scenarios.

The rest of the paper is organized as follows. Section II describes the state of the art on adaptation algorithms. Section III presents the adaptation algorithm based on AFF. Section IV describes the scenario used to test the proposed adaptation algorithm. Section V presents the obtained results. Section VI presents a fairness analysis of the AFF algorithm. Finally, Section VII presents the conclusions and future work.

II. Overview on Adaptation Algorithms

A DASH client uses an adaptation algorithm to handle the selection of the multimedia representation for each segment that needs to be downloaded. This selection is based on the network conditions, which are compared to a set of parameters on the client (e.g., buffer level) to make the choices about the highest possible quality when requesting the next media segment. This process of adaptation can be assisted by intermediate network nodes which have information on how much bandwidth is going to be allocated to the clients [6]–[8]. But, in general, adaptation algorithms implement buffer control and throughput estimation methods. Buffer control is designed so that the fluctuations of the bandwidth, especially in wireless environments, will not affect the playback of the video. Throughput estimation is designed to maximize the quality in terms of bitrate selection by estimating the available bandwidth. This paper focuses on throughput estimation algorithms.

Throughput estimation algorithms focus on providing an adequate estimation of the throughput that the client can obtain from the network. Most of the adaptation algorithms start by calculating the instant throughput [9], which is defined as the size of the last downloaded segment divided by the time taken to download it.

The problem with instant throughput is that it is not an appropriate throughput estimation method because measurements can fluctuate from one segment to another. Using it as a throughput estimator would have the effect of the multimedia player continuously adapting the bitrate to the instant throughput measurements, which would affect the quality of the playback. Adaptation algorithms that use instant throughput measurements as a throughput estimation method, combine it with other mechanisms. For example, BOLA [10], adapts the video bitrate using a buffer control method.

Some methods for estimating the throughput rely on the measurements of lower layers [11], [12], (e.g., from the physical layer). Those measurements are then compared to the instant throughput to make an adaptation decision. However, throughput measurements from the physical layer have the inconvenience of including all network services from the client.

Other throughput oriented algorithms calculate the average throughput for the last N segments of video obtained by the client [9]. However, this method has the disadvantage of not detecting short-term fluctuations of the available bandwidth that can occur, e.g., in wireless networks. If a short-term decrease of available bandwidth is not detected, this would lead to buffer starvation.

Lin *et al.* [13] proposed to use the mean, the standard deviation and the fluctuation of the throughput to estimate the bandwidth. This method has the problem of heuristically establishing the fluctuation parameter with values of 0 to 0.025 for wired networks and 0 to 0.1 for wireless networks.

An alternative throughput estimation method is to calculate the harmonic mean of a certain number of past measurements. The FESTIVE algorithm [14] uses the last twenty measurements while the ELASTIC algorithm [15] uses the last five. The harmonic mean method functions optimally when having steady bandwidth measurements because it can discriminate some of the outlier measurements. However, in wireless environments, there are short-term bandwidth fluctuations that can cause this method to overestimate or underestimate the available bandwidth.

Zhou *et al.* [16] proposed the use of a rate adaptation algorithm based on Markov decision processes that uses the mean and a temporal variance as a mechanism for bitrate switching. The problem of continuous bitrate switching is addressed by establishing two buffer thresholds. Using an algorithm that is also based on Markov decision processes, [17] proposes the use of a reinforcement learning method with a reward function to program new segment petitions.

The EWMA method is proposed in [18], [19]. EWMA behaves as a low-pass filter for the throughput measurements. EWMA applies weighting factors so the weighting of each older instant throughput measurement decreases exponentially. The problem with this method is that the initial weight parameter needs to be fixed to a different value depending on the type of network. Usually weight values of 0 to 0.1 are proposed for wired networks and 0.2 to 0.3 for wireless networks. This can cause the method to not behave adequately (e.g., producing stalling events) when the weight parameters are not set adequately for different network scenarios.

Li *et al.* [20] proposed PANDA, an algorithm that uses EWMA with an exponential weight of 0.2 as a throughput estimator, which would be adequate for wireless networks. The algorithm also proposes the use of an Additive Increase Multiplicative Decrease (AIMD) bitrate selection algorithm and a random scheduler algorithm to avoid playback synchronization for simultaneous players.

Another example of an EWMA based throughput estimation method adapted to a specific type of network can be found in [21], which proposed the use of DASH in a dense wireless network scenario, using proportional-integral-derivative (PID) controllers and an EWMA throughput estimation in every wireless client to manage the quality selection and client scheduling.

Thang *et al.* [22] presented a method that combines the use of EWMA and the Round Trip Time (RTT) estimation method of TCP but presents the problem of setting a fixed weight parameter depending on the type of network used. Jeong and Chung [23] proposed to use EWMA, RTT and a Media Segment Duration (MSD) measurement, but a fixed weight parameter depending on the access network is needed, like [18], [19], [22]–[24].

Lai *et al.* [25] proposed the use of EWMA and two correcting parameters that are used to calculate the weight used in EWMA, and a safety margin of three times the standard deviation in the EWMA formula.

A combination of EWMA and a dynamic fluctuation factor in [26] tries to compensate for the error between the last throughput measurement and the next. However, this method is based on comparing the last throughput measurement to the previous one, making the exponential parameter to adapt too slowly when there are bandwidth fluctuations.

The AFF method proposed in this paper to address the problem of bandwidth fluctuations is explained in the following section.

III. Adaptive Forgetting Factor for DASH

The AFF method was originally designed as a recursive least-square adaptive filter to recover data from corrupted signals [27]. Similar methods can be found in various fields like medicine, finance, computer networks monitoring or astronomy.

In this paper we propose the use of an AFF method originally designed by Bodenham [28] as a throughput estimation method for network security. This method is based on statistical process control to analyze streams of data and detect changes using the mean and the variance.

AFF shares with EWMA the idea of using a weight parameter that decreases the impact of older measurements exponentially (functioning as a smoothing parameter) on the estimation of the mean. But while EWMA uses a fixed weight factor, AFF calculates the value dynamically. This allows AFF to react quicker to short-term fluctuations of the bandwidth. The estimated throughput in the AFF method is shown in (1). The AFF mechanism is $\overrightarrow{\lambda}$, where $\overrightarrow{\lambda} = (\lambda_0, \lambda_1, ..., \lambda_N)$ and $\lambda_i \in (0,1)$.

$$\overline{THR}_{N,\overrightarrow{\lambda}} = \frac{m_{N,\overrightarrow{\lambda}}}{w_{N,\overrightarrow{\lambda}}},\ N \geq 1 \tag{1}$$

$$m_{N,\overrightarrow{\lambda}} = \lambda_{N-1} m_{N-1,\overrightarrow{\lambda}} + THR,\ N \geq 1 \tag{2}$$

$$w_{N,\overrightarrow{\lambda}} = \lambda_{N-1} w_{N-1,\overrightarrow{\lambda}} + 1,\ N \geq 1 \tag{3}$$

Equation (1) is the throughput estimation, (2) represents the accumulated instant throughput measurements. (3) shows the accumulated of the number of segments. Where $m_{0,\overrightarrow{\lambda}} = 0$, $w_{0,\overrightarrow{\lambda}} = 0$ and $\lambda_0 = 1$ respectively.

As shown in (2) and (3), the key component of the AFF method is how the weight factor is updated, namely $\lambda_N \rightarrow \lambda_{N+1}$. To obtain λ_N, it is necessary to apply an online optimization to minimize a cost function $L_{N+1,\overrightarrow{\lambda}}$.This is possible by applying a first order optimization algorithm such as the one-step gradient descent (4).

$$\lambda_{N+1} = \lambda_N - \eta \frac{\partial}{\partial \overrightarrow{\lambda}} L_{N+1,\overrightarrow{\lambda}} \tag{4}$$

$$L_{N+1,\overrightarrow{\lambda}} = \left[\overline{THR}_{N,\overrightarrow{\lambda}} - THR_{N+1}\right]^2 \tag{5}$$

In (4) η is the step size and $\eta \ll 1$ so that the algorithm can react faster when solving the optimization. The cost function, shown in (5), compares the average throughput values to the new measurement to assure that the estimated value is as close as possible to the new measurement.

Utilizing the chain rule method in (5), the result is a derivative of (1), that translates in derivatives of (2) and (3). To solve those derivatives a differentiation from first principle (delta method) is applied. This method aims to find the instant rate of change of (2) and (3) with respect of $\overrightarrow{\lambda}$.

Equations (7) and (8) present the results of applying the first principle or delta method to (2) and (3). For lemma proof or explanations regarding the mathematical methods applied in this research consult [29].

$$\Delta_{N,\overrightarrow{\lambda}} = \lambda_{N-1} \Delta_{N-1,\overrightarrow{\lambda}} + m_{N-1,\overrightarrow{\lambda}} \tag{6}$$

$$\Omega_{N,\overrightarrow{\lambda}} = \lambda_{N-1} \Omega_{N-1,\overrightarrow{\lambda}} + w_{N-1,\overrightarrow{\lambda}} \tag{7}$$

In (6) and (7) the initial measurements are defined as: $\Delta_{1,\overrightarrow{\lambda}} = 0$ and $\Omega_{1,\overrightarrow{\lambda}} = 0$. Next we proceed to solve the derivative of (1) which is shown in (8).

$$\frac{\partial}{\partial \overrightarrow{\lambda}} \overline{THR}_{N,\overrightarrow{\lambda}} = \frac{\Delta_{N,\overrightarrow{\lambda}} m_{N,\overrightarrow{\lambda}} - \Omega_{N,\overrightarrow{\lambda}} w_{N,\overrightarrow{\lambda}}}{\left(w_{N,\overrightarrow{\lambda}}\right)^2} \tag{8}$$

The equations formerly explained were implemented as shown in Algorithm 1. The aim of the AFF method is to correctly select the representation whose bitrate matches the bandwidth conditions that the network offers to achieve the highest quality in the video playback. This is achieved by estimating adaptively the throughput of the video streaming, and when encountering short-term fluctuations, placing greater weights on more recent observations and thereby "forgetting" older measurements faster.

The algorithm compares if the actual measurement is the first, because in that case the calculation is done with different initialization variables. Afterwards the algorithm calculates the estimated throughput using the next instant throughput measurement in the AFF equations.

The AFF algorithm is executed every time a new instant throughput measurement is obtained, i.e., every time a video segment is downloaded. Once the average throughput is estimated, the algorithm selects a new video representation by performing a sequential search on the video bitrates of the different representations, from highest to lowest, to select the highest bitrate that is below the average throughput.

Bodenham in [29], performing numerous simulations, reached the conclusion that truncating the range of $\overrightarrow{\lambda}$ to $\overrightarrow{\lambda} \in (0.6, 1)$ can make the method react faster when encountering short-term fluctuations.

Bodenham also defined that the step size η variable should be between $\eta \in (0.001, 0.1)$ so that the algorithm can return to a steady behavior faster when detecting a fluctuation. In this proposal a value of $\eta = 0.1$ is used.

The next section presents the scenario in which the AFF adaptive algorithm was evaluated.

Algorithm 1: AFF throughput estimation algorithm

```
Input: $maxbitrateindex, bitratebw(i),$
    $SegmentIndex = n, THR_n$
Output: $switchbitrate(i)$
    $Initialization: m_0 = w_0 = 0; \lambda_0 = 0;$
    $\Omega_1 = \Delta_1 = 0; \eta = 0.1;$
1:  if ($SegmentIndex == 1$) then
2:      $m_1 = (\lambda_0 * m_0) + THR_1$
3:      $w_1 = (\lambda_0 * w_0) + 1$
4:      $\overline{THR_1} = \frac{m_1}{w_1}$
5:      $\lambda_1 = \lambda_0 - \eta * 2 * (\overline{THR_1} - THR_1) *(\frac{\Delta_1 * w_1 - \Omega_1 * m_1}{w_1^2})$
6:  else
7:      $\Delta_n = (\lambda_{n-1} * \Delta_{n-1}) + m$
8:      $\Omega_n = (\lambda_{n-1} * \Omega_{n-1}) + w$
9:      $m_n = (\lambda_{n-1} * m_{n-1}) + THR_n$
10:     $w_n = (\lambda_{n-1} * w_{n-1}) + 1$
11:     $\overline{THR_n} = \frac{m_n}{w_n}$
12:     $\lambda_n = \lambda_{n-1} - \eta * 2 * (\overline{THR_n} - THR_n) *(\frac{\Delta_n * w_n - \Omega_n * m_n}{w_n^2})$
13: end if
14: for $i = maxbitrateindex$ to 0 do
15:   if ($\overline{THR_n} \geq bitratebw(i)$) then
16:       return $switchbitrate(i)$
17:       end for
18:   end if
19: end for
```

IV. Implementation Scenario

To test the proposal, a DASH player has been modified to add AFF as the throughput estimation algorithm. The dash.js player [30] has been selected. This player is a JavaScript implementation of a DASH player that can run in a web browser. The original dash.js reference player implements a throughput algorithm that consists of calculating the average of the throughput obtained for the last three video segments. This algorithm will be referred to as *avg-last-3*. The dash.js player also implements a buffer control algorithm so that it can adapt differently when certain established levels are reached. For instance, when the buffer level drops to 8 seconds, lower bitrate segments are requested independently of how much bandwidth the throughput algorithm estimates.

Fig. 1 presents a block diagram of the adaptation logic implemented in dash.js. There are four elements in the adaptation algorithm. From those four elements, the complexity of the adaptation logic resides mainly on the Buffer Module and the Throughput Module, which are the modules implementing the algorithms managing the selection of the next segment bitrate.

The Rules Entity defines all the necessary parameters and levels such as the minimum buffer level or if the streaming is live or pre-stored. The Throughput Module is where the instant throughput is measured and where the avg-last-3 method to estimate throughput is implemented. This module has been modified to implement the AFF method and the EWMA method

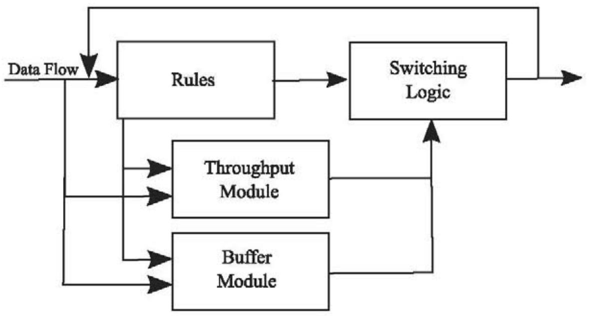

Fig. 1. Block diagram of the adaptation logic of dash.js.

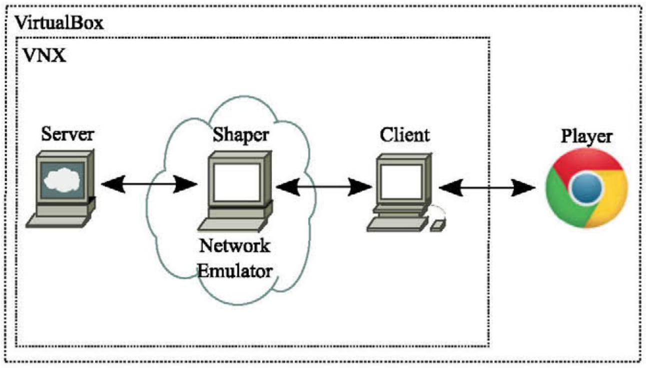

Fig. 2. Experiment scenario.

so the three different throughput estimation methods can be compared.

The Buffer Module implements a heuristic algorithm that consists of monitoring the buffer level to force a bitrate change when the buffer is under a minimum level. Finally, the Switching Logic controls the request for new video segments choosing a video representation bitrate that fits the conditions coming from the Buffer Module and the Throughput Module.

Our experimental platform consists of a client-server model which is interconnected by a network emulator, as shown in Fig. 2. The scenario has been implemented using an open-source virtualized platform [31].

This scenario is built over two layers of virtualization; the first layer consists of an open-source tool named Virtual Networks over linuX (VNX) [32]. This tool is based on Linux Containers (LXC) and an XML file to create virtual network scenarios. The XML file contains the information on the number of network elements, the interconnections among them and commands to perform specific tasks. The second layer of virtualization consists of hosting the VNX scenario in a virtual machine running Ubuntu inside VirtualBox [33]. This layer provides the portability for the scenario to run in any OS with the use of an Open Virtual Appliance (OVA).

The server stores the Big Buck Bunny video [34] at a resolution of 720p. The video is encoded in H.264/Advanced Video Coding (AVC) format with variable bitrate (VBR) using the x264 tool [35] with four different bitrates (250, 500, 1000 and 2000 Kbps). MP4Box [36] is used to split each video file in segments and generate the MPD file. Each video segment has a duration of 2 seconds.

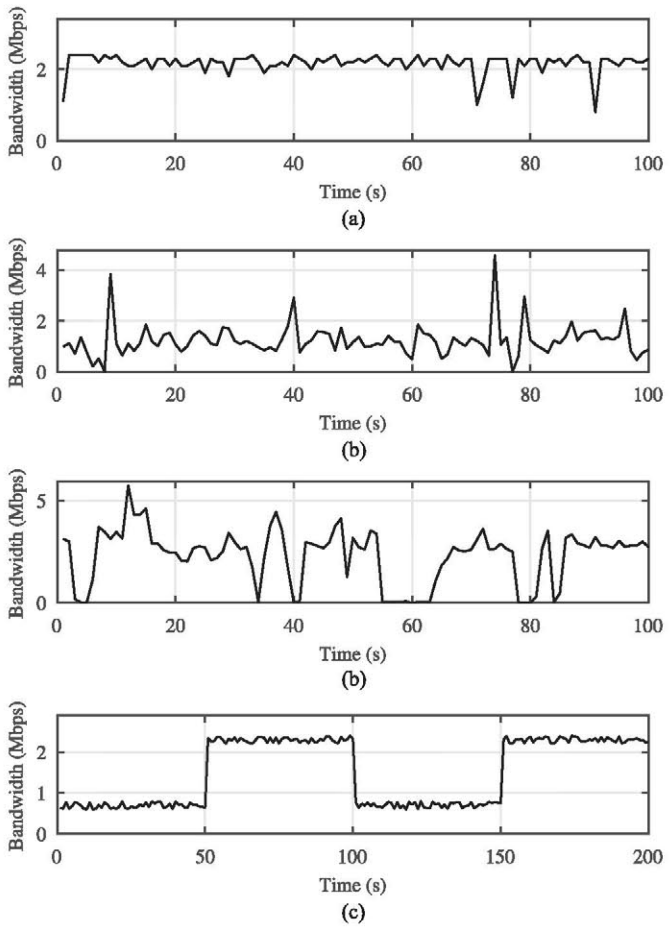

Fig. 3. Bandwidth profiles used.

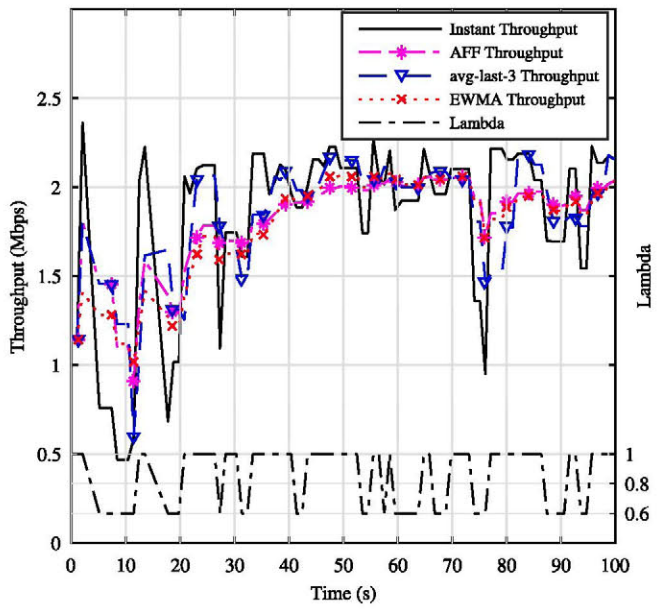

Fig. 4. Estimated throughput of the methods and lambda in bandwidth profile Test 1.

TABLE I
STATISTICS OF THE BANDWIDTH PROFILES IN MBPS

Test	Bandwidth Type	Max.	Min.	Avg.	S.D.
Test 1	High bandwidth, low fluctuation.	2.40	0.80	2.17	0.2765
Test 2	Low bandwidth, high fluctuation.	4.57	0.01	1.23	0.6374
Test 3	High bandwidth, high fluctuation.	5.73	0.01	2.31	1.3317
Test 4	High-low bandwidth, low fluctuation.	2.39	0.60	1.50	0.8063

The shaper behaves as a network emulator that modifies the network conditions in three aspects: available bandwidth, network delay and packet loss. The shaper is based on a shell script that reads a Comma-separated Value (CSV) network profile file defining the network conditions on different time periods. This file is used to change the configuration of network interfaces during the video streaming experiments, using the traffic control and the network emulation (NetEm) tools of Linux. More specifically, the shaper uses the Hierarchy Token Bucket (HTB) queuing discipline to classify the incoming TCP traffic and enforce the bandwidth parameter read from the network profile file.

The client has a caching proxy, that can be used to support broadcast streaming services. In our scenario, as depicted in Fig. 3, the player consists of a Chrome web client running dash.js to access the video segments transparently using the client proxy. The player is also responsible for collecting the data which is used to analyze the behavior of the throughput estimation algorithm; i.e., buffer length, video bitrate and instant throughput.

The AFF throughput estimation method has been compared to the dash.js avg-last-3 estimation method [30] and to our own implementation of the EWMA method described in [18], [24]. For EWMA, the fixed weight parameter is set to 0.2, which is the value for wireless network proposed by [18], since the profiles are based on LTE traffic measurements.

The experiments are carried out using four different bandwidth profiles, as shown in Fig. 4. The three first profiles use real LTE network data obtained in a previous research [31]. The fourth profile was created to analyze the behavior of the throughput estimation methods when the available bandwidth changes abruptly.

The first bandwidth profile was selected to represent the behavior of a steady network such as a residential Internet connection. The second profile aims to study the performance of the throughput estimation methods for limited bandwidths that present short-term fluctuations. The third profile is inspired by test pattern one (TP1) in [26], depicting a user walking during daytime in an urban environment. The fourth profile was created to analyze the behavior of the different throughput estimation methods when a sudden loss of bandwidth occurs, especially to measure how a high change in the bandwidth affect the algorithms that carry past measurements when estimating the available throughput.

Table I shows the main statistics for each of the bandwidth profiles used to test the performance of the AFF algorithm: maximum, minimum, average, and standard deviation for the bandwidth of each profile.

TABLE II
STATISTICS OF THE ALGORITHMS

Test	Method	Bitrate Changes	Stalling Events	Time (s)	Mean Bitrate (Kbps)
Test 1	AFF	**7**	0	–	**1387.00**
	avg-last-3	41	0	–	1002.10
	EWMA	21	**2**	1.28, 0.9	822.12
Test 2	AFF	34	0	–	492.00
	avg-last-3	35	**1**	5.01	471.83
	EWMA	**14**	0	–	**563.00**
Test 3	AFF	**8**	0	–	**1216.80**
	avg-last-3	15	0	–	1174.00
	EWMA	25	0	–	775.00
Test 4	AFF	**26**	0	–	730.00
	avg-last-3	33	**2**	0.3, 2.23	**880.00**
	EWMA	30	0	–	814.00

V. PERFORMANCE EVALUATION

In this section we present the experimental results of evaluating the different throughput estimation methods avg-last-3, EWMA and AFF for each of the bandwidth profiles.

The first subsection presents an analysis of the three different throughput estimation methods using the first bandwidth profile, and the behavior of λ for the AFF method. The next four subsections present the results for each bandwidth profile in the form of a Cumulative Distribution Function (CDF) for the video bitrate selection and the buffer level obtained for the three throughput estimation methods. Finally, the last subsection discusses the Quality of Experience (QoE) indicators for each of the scenarios that are summarized in Table II.

A. *Comparison of Throughput Estimation Methods*

Fig. 4 shows measurements of the instant throughput that the DASH player calculates, the estimates that each method obtains from the instant throughput measurements, and the lambda value that the AFF method calculates adaptively. Since AFF and EWMA share a similar weighting mechanism, they present a similar behavior, and different from avg-last-3 that just calculates the mean value of the three past measurements.

Fig. 4 also shows how the lambda AFF factor evolves during the playback of the video. When the instant throughput behaves in a steady manner, $\lambda = 1$, the highest value. But when there is a short-term fluctuation of the throughput, lambda drops to the lowest value of $\lambda = 0.6$, which makes the AFF algorithm to forget the latest measurements faster.

B. *High Available Bandwidth With Low Short-Term Fluctuations*

This test was conducted to compare the three algorithms in a steady environment with enough bandwidth to reach the highest representation most of the time. This type of bandwidth profile describes the typical bandwidth of a residential Internet connection and it is similar to test pattern two (TP2) in [26].

Fig. 5 presents the CDF of the bitrate selection for each of the algorithms, showing that the AFF algorithm is more stable

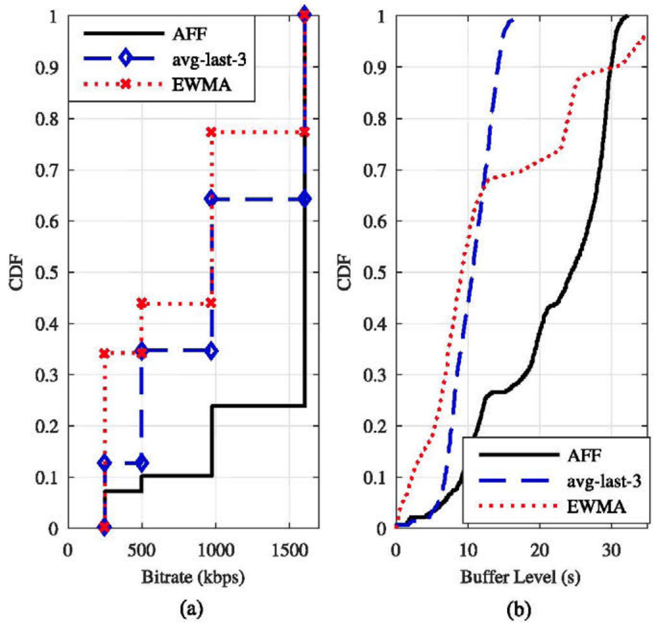

Fig. 5. (a) CDF of bitrate and (b) CDF of buffer level in Test 1.

than the other algorithms since the best quality is chosen most of the time.

Fig. 5 also presents the CDF of the buffer level. The AFF algorithm maintains a buffer level above ten seconds, which allows the AFF throughput estimation method to choose the bitrates instead of the buffer control algorithm. It also shows that the avg-last-3 and EWMA methods present more bitrate changes.

C. *Low Available Bandwidth With High Short-Term Fluctuations*

In the second test, a low bandwidth and high number of short-term bandwidth fluctuations profile is used. This profile affects video quality because of the sudden drops of available bandwidth. It may also cause stalling during the playback because of the continuous fluctuation in bandwidth. This profile was selected to evaluate the behavior of the different throughput estimation methods in an environment where the bandwidth is low.

Fig. 6 shows that in this case the EWMA method presents better results than the AFF method and avg-last-3 method. Both the AFF and the avg-last-3 methods select the lowest bitrate quality most of the playback time. However, the dahs.js method present a stalling event of five seconds that affects the quality of the playback. The buffer level behaved in a similar manner with the three methods, mostly under 10 seconds because of the low bandwidth.

D. *High Available Bandwidth With High Short-Term Fluctuations*

The high bandwidth and high short-term fluctuations of this profile are expected to show whether the buffer can minimize the impact of having severe drops from high bandwidth measurements, as well as the behavior of the different throughput

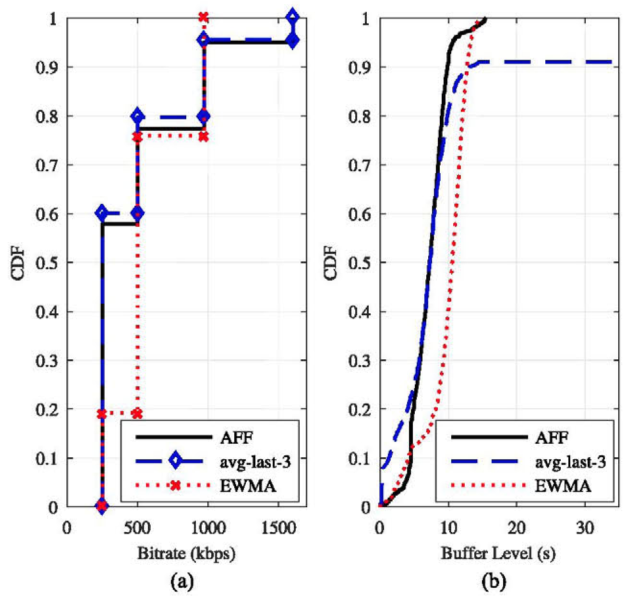

Fig. 6. (a) CDF of bitrate and (b) CDF of buffer level in Test 2.

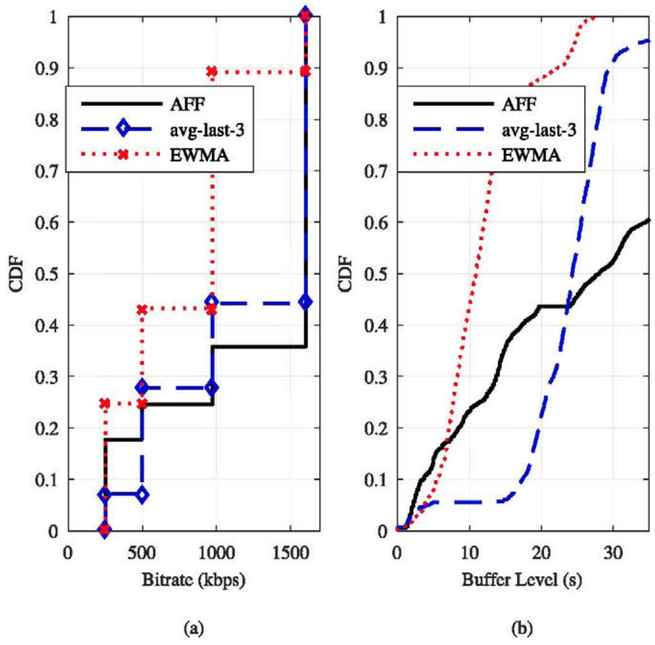

Fig. 7. (a) CDF of bitrate and (b) CDF of buffer level in Test 3.

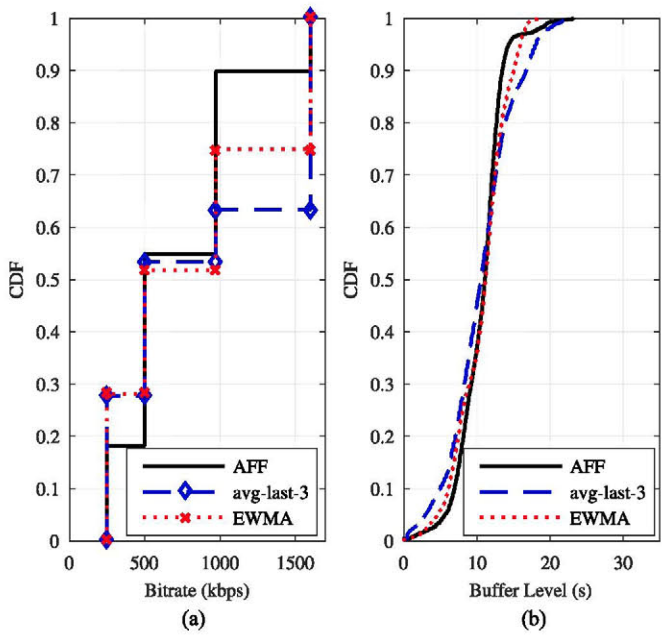

Fig. 8. (a) CDF of bitrate and (b) CDF of buffer level in Test 4.

estimation methods. This profile represents the bandwidth behavior of an LTE user who is walking in an urban environment, and it is similar to TP1 in [26].

As shown in Fig. 7, since the bandwidth profile has many high short-term fluctuations with high bandwidth, the AFF and the avg-last-3 methods selects the highest bitrate most of the time. However, the AFF method selects the highest bitrate quality ten percent more than the avg-last-3 method. The EWMA method shows a poor behavior, selecting the third bitrate quality most of the time. The buffer level behaves differently for each method; however, it is shown that the EWMA method presented the most bitrate switches because the buffer level stays below ten seconds most of the time, which triggers the buffer level mechanism. The results from the first and third tests show that the AFF method is suited to perform proficiently in wired and wireless environments.

E. Low and High Bandwidth Variations

This profile combines low bandwidth and high bandwidth with low number of short-term fluctuations. This profile was created to analyze the impact of older bandwidth estimations on the accuracy of the throughput estimation methods when there is an abrupt bandwidth change.

Fig. 8 shows that the AFF method selects the second and third bitrate quality most of the time, and the smallest number of quality changes. The avg-last-3 method reaches the highest mean bitrate, but suffers from two stalling events of 0.3 seconds and 2 seconds respectively, as Table II shows. The EWMA method presents a better mean bitrate than the AFF method but selects the lowest quality more often and presents more bitrate quality switches. It also shows that the buffer level is lower than ten seconds half of the time, which means that the buffer control algorithm affects the throughput estimation by constantly lowering the bitrate chosen for this bandwidth profile.

F. QoE Results

Table II shows, for each of the methods, a set of quality of experience indicators: number of bitrate changes, number of stalling events (due to buffer under-runs), the duration of each stalling event in seconds and the mean bitrate in Kbps. In the first test the AFF method presented a better-quality performance since there were less bitrate changes and the mean bitrate was higher than for the other methods. It also shows that the EWMA method presented two stalling events that lasted 1.28 and 0.9 seconds respectively, having a great impact on the QoE of the play-

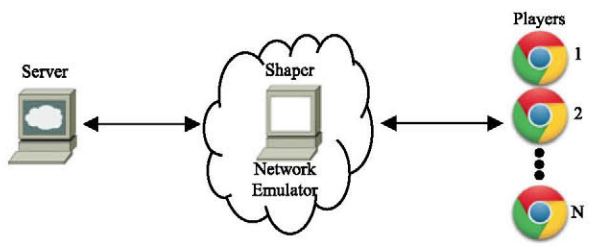

Fig. 9. Fairness experiment scenario.

TABLE III
FAIRNESS ANALYSIS NETWORK PROFILE

Interval (s)	Bandwidth (Mbps)
0–100	22
100–200	12
200–300	6
300–360	22

back. In the second bandwidth profile, the EWMA method presented a steadier behavior than the other two methods; however, the mean bitrate of each method is not far from one another. It also shows that the avg-last-3 method presented a stalling event that lasted 5 seconds. For the third bandwidth profile, the results show that the AFF method presents less bitrate changes and a better mean bitrate. In the fourth test, the AFF method presents less bitrate changes, however, the avg-last-3 method had the best mean bitrate. The problem with the avg-last-3 method is that it had two stalling events, one lasted three seconds and the other one two seconds, making this method not suited to be used in environments with significant drops in bandwidth.

VI. FAIRNESS ANALYSIS

In adaptive video streaming, fairness metrics are used to determine if the adaptation algorithm is able to deliver a fair share of the network bandwidth to different clients. In order to analyze the fairness of the AFF algorithm we have created an experimental scenario, as shown in Fig. 9, to measure the bandwidth obtained by several simultaneous players sharing a bottleneck link.

This scenario is similar to the scenario explained in Section IV, but in this case the equipment is not virtualized. The HTTP server, the multimedia content and the multimedia player are described in Section IV. The shaper is now implemented using a Raspberry Pi 3 with two network interfaces. The shaper uses a network profile as specified in Table III. This profile was created to measure the fairness of the AFF estimation algorithm and the avg-last-3 algorithm, in order to detect how the algorithm responds to changes in the bandwidth when competing with other video players using the same algorithm.

The experiments consist of ten players initiating the multimedia playback randomly within the first 15 s of the experiment. The results are obtained measuring the average bandwidth of each client in the interval 50-350 s. These values are used to calculate the Jain Fair index (JFI) [37] and the total average throughput, as shown in Table IV.

TABLE IV
FAIRNESS ANALYSIS RESULTS

Test	Algorithm	JFI	Avg. Thr. (Kbps)
Test 5	AFF	.9969	**1281.7**
Test 6	avg-last-3	**.9995**	1188.5

The results show that both algorithms, with a JFI value close to 1, present a fair use of the bandwidth. Thus, the AFF estimation algorithm shows fairness results that are similar to already existing algorithms, making the AFF algorithm suited to be used as the throughput estimation mechanism in combination with different adaptation algorithms. The results also show that the AFF method manages to achieve a slightly higher average throughput for the clients.

VII. CONCLUSION AND FUTURE WORK

Adaptive bitrate streaming solutions rely on throughput estimation algorithms that might need fine-tuning depending on the characteristics of the network. In this paper, we propose to use the Adaptive Forgetting Factor method for throughput estimation in DASH adaptation algorithms. This method relies on instant throughput measurements that are aggregated using exponential weights adaptively, so it overcomes the problems of short-term fluctuations in network throughput. By using this method it is possible to improve the QoE obtained by a DASH player over different types of networks, avoiding the fine-tuning of parameters of other throughput estimation algorithms. Using different bandwidth profiles, the AFF proposal has been tested and compared to alternative throughput estimation methods such as EWMA and average-last-3. The results show that AFF might obtain slightly lower average bitrates than the alternative methods. However, it presents a better behavior in regard to video stalling and number of bitrate switches, which are two of the key parameters for QoE in adaptive streaming. Finally, the results of the fairness experiments show that the AFF algorithm is able to deliver a fair share of the network bandwidth to different clients. Therefore, we propose AFF as a valid throughput estimation method that can work adequately for DASH players over different types of networks.

A future step in our research is to work on DASH adaptation algorithms that can exploit a better communication between throughput estimation and buffer control methods, to provide the best QoE with different network conditions.

REFERENCES

[1] Sandvine, Incorporated, "Sandvine global Internet phenomena report Oct. 2016," [Online]. Available: https://www.sandvine.com/downloads/general/global-internet-phenomena/2016/global-internet-phenomena-report-latin-america-and-north-america.pdf, Accessed on: Nov. 2016.
[2] C. V. Forecast, "Cisco visual networking index: Global mobile data traffic forecast update 2015–2020," *Cisco Public Information*, vol. 9, Feb. 2016.
[3] *Transparent End-to-End Packet-Switched Streaming Service (PSS); Progressive Download and Dynamic Adaptive Streaming Over HTTP*, Eur. Telecommun. Standards Inst., Sophia Antipolis, France, 3GPP TS 26.247 v14.1.0, 2017.

[4] C. M. Lentisco, L. Bellido, and E. Pastor, “Reducing latency for multimedia broadcast services over mobile networks,” *IEEE Trans. Multimedia*, vol. 19, no. 1, pp. 173–182, Jan. 2017.
[5] *Inf. technol. – Dynamic Adaptive Streaming Over HTTP (DASH) – Part 1: Media Presentation Description and Segments Formats*, ISO/IEC 23009-1, 2014.
[6] C. M. Lentisco, L. Bellido, and E. Pastor, “Seamless mobile multimedia broadcasting using adaptive error recovery,” *Mobile Inform. Syst.*, vol. 2017, Feb. 2017, Art. no. 1847538.
[7] A. E. Essaili, D. Schroeder, E. Steinbach, D. Staehle, and M. Shehada, “QoE-based traffic and resource management for adaptive HTTP video delivery in LTE,” *IEEE Trans. Circuits Syst. Video Technol.*, vol. 25, no. 6, pp. 988–1001, Jun. 2015.
[8] N. Bouten, S. Latré, J. Famaey, W. V. Leekwijck, and F. D. Turck, “In-network quality optimization for adaptive video streaming services,” *IEEE Trans. Multimedia*, vol. 16, no. 8, pp. 2281–2293, Dec. 2014.
[9] G. Tian and Y. Liu, “On adaptive HTTP streaming to mobile devices,” in *Proc. 2013 20th Int. Packet Video Workshop*, Dec. 2013, pp. 1–8.
[10] K. Spiteri, R. Urgaonkar, and R. K. Sitaraman, “BOLA: Near-optimal bitrate adaptation for online videos,” in *Proc. IEEE INFOCOM 2016–35th Annu. IEEE Int. Conf. Comput. Commun.*, Apr. 2016, pp. 1–9.
[11] V. Ramamurthi and O. Oyman, “Link aware HTTP adaptive streaming for enhanced quality of experience,” in *Proc. 2013 IEEE Global Commun. Conf.*, Dec. 2013, pp. 1675–1680.
[12] V. Ramamurthi, O. Oyman, and J. Foerster, “Using link awareness for HTTP adaptive streaming over changing wireless conditions,” in *Proc. 2015 Int. Conf. Comput., Netw. Commun.*, Feb. 2015, pp. 727–731.
[13] Q. Lin *et al.*, “Bandwidth estimation of rate adaption algorithm in DASH,” in *Proc. 2014 IEEE Globecom Workshops*, Dec. 2014, pp. 243–247.
[14] J. Jiang, V. Sekar, and H. Zhang, “Improving fairness, efficiency, and stability in HTTP-based adaptive video streaming with festive,” *IEEE/ACM Trans. Netw.*, vol. 22, no. 1, pp. 326–340, Feb. 2014.
[15] L. De Cicco, V. Caldaralo, V. Palmisano, and S. Mascolo, “ELASTIC: A client-side controller for dynamic adaptive streaming over HTTP (DASH),” in *Proc. 2013 20th Int. Packet Video Workshop,*, 2013, pp. 1–8.
[16] C. Zhou, C. W. Lin, and Z. Guo, “mDASH: A markov decision-based rate adaptation approach for dynamic HTTP streaming,” *IEEE Trans. Multimedia*, vol. 18, no. 4, pp. 738–751, Apr. 2016.
[17] A. Bokani, M. Hassan, S. Kanhere, and X. Zhu, “Optimizing HTTP-based adaptive streaming in vehicular environment using markov decision process,” *IEEE Trans. Multimedia*, vol. 17, no. 12, pp. 2297–2309, Dec. 2015.
[18] S. Akhshabi, A. C. Begen, and C. Dovrolis, “An experimental evaluation of rate-adaptation algorithms in adaptive streaming over HTTP,” in *Proc. 2nd Annu. ACM Conf. Multimedia Syst.*, Feb. 2011, pp. 157–168.
[19] T. C. Thang, Q. D. Ho, J. W. Kang, and A. T. Pham, “Adaptive streaming of audiovisual content using MPEG DASH,” *IEEE Trans. Consum. Electron.*, vol. 58, no. 1, pp. 78–85, Feb. 2012.
[20] Z. Li *et al.*, “Probe and adapt: Rate adaptation for HTTP video streaming at scale,” *IEEE J. Sel. Areas Commun.*, vol. 32, no. 4, pp. 719–733, Apr. 2014.
[21] K. Miller, D. Bethanabhotla, G. Caire, and A. Wolisz, “A control-theoretic approach to adaptive video streaming in dense wireless networks,” *IEEE Trans. Multimedia*, vol. 17, no. 8, pp. 1309–1322, Aug. 2015.
[22] H. T. Le, H. N. Nguyen, N. Pham Ngoc, A. T. Pham, and T. C. Thang, “A novel adaptation method for HTTP streaming of VBR videos over mobile networks,” *Mobile Inform. Syst.*, vol. 2016, Jun. 2016, Art. no. 2920850.
[23] U. Jeong and K. Chung, “Video quality adaptation to improve the quality of experience in DASH environments,” *Int. J. Comput. Sci. Netw. Security*, vol. 14, no. 8, pp. 22–29, Aug. 2014.
[24] T. C. Thang *et al.*, “Adaptive video streaming over HTTP with dynamic resource estimation,” *J. Commun. Netw.*, vol. 15, no. 6, pp. 635–644, Dec. 2013.
[25] C. F. Lai, H. Wang, H. C. Chao, and G. Nan, “A network and device aware QoS approach for cloud-based mobile streaming,” *IEEE Trans. Multimedia*, vol. 15, no. 4, pp. 747–757, Jun. 2013.
[26] Y.-H. Kim, J. Shin, and J. Park, “Design and implementation of a network-adaptive mechanism for HTTP video streaming,” *ETRI J.*, vol. 35, no. 1, pp. 27–34, Feb. 2013.
[27] J. Cooper and K. Worden, “On-line physical parameter estimation with adaptive forgetting factors,” *Mech. Syst. Signal Process.*, vol. 14, no. 5, pp. 705–730, May 2000.
[28] D. A. Bodenham and N. M. Adams, “Continuous monitoring of a computer network using multivariate adaptive estimation,” in *Proc. 2013 IEEE 13th Int. Conf. Data Mining Workshops*, Dec. 2013, pp. 311–318.
[29] D. A. Bodenham, “Adaptive estimation with change detection for streaming data,” Ph.D. dissertation, Imperial College London, London, U.K., 2014.
[30] DASH Industry Forum, MPEG-DASH reference player dash.js. [Online]. Available: http://github.com/Dash-Industry-Forum/dash.js, Accessed on: May 2017.
[31] C. M. Lentisco *et al.*, “A virtualized platform for analyzing LTE broadcast services,” in *Proc. 2015 Eur. Conf. Netw. Commun.*, Jun. 2015, pp. 512–516.
[32] D. Fernández *et al.*, “Enhancing learning experience in computer networking through a virtualization-based laboratory model,” *Int. J. Eng. Educ.*, vol. 32, no. 6, pp. 2569–2584, Dec. 2016.
[33] Oracle. VirtualBox. [Online]. Available: https://www.virtualbox.org, Accessed on: May 2017.
[34] Blender Foundation. Big buck bunny movie. [Online]. Available: https://peach.blender.org/, Accessed on: Dec. 2016.
[35] L. Merritt and R. Vanam. x264: A high performance H.264/AVC encoder, 2006. [Online]. Available: http://akuvian.org/src/x264/overview_x264_v8_5.pdf
[36] J. L. Feuvre. Gpac multimedia open source project. [Online]. Available: http://gpac.wp.mines-telecom.fr/mp4box, Accessed on: May. 2017.
[37] R. Jain, D. Chiu, and W. Hawe, “A quantitative measure of fairness and discrimination for resource allocation in shared computer system,” Digital Equipment Corporation, Maynard, MA, USA, Tech. Rep. DEC-TR-301, Sep. 1984.

Miguel Aguayo received the B.E. degree in telematic engineering from the Universidad de Colima, Colima, México, in 2002, and the M.S. degree in electronics and telecommunications from the Ensenada Center for Scientific Research and Higher Education, Ensenada, México, in 2004. He is currently a Ph.D. candidate in the Department of Telematics Systems Engineering, the Universidad Politécnica de Madrid, Madrid, Spain. His research interests include multimedia communications, internetworking, mobile networks, and quality of experience.

Luis Bellido received the M.S. and Ph.D. degrees in telecommunications engineering from the Universidad Politécnica de Madrid (UPM), Madrid, Spain, in 1994 and 2004, respectively. He is currently an Associate Professor at UPM, specializing in the fields of computer networking, Internet technologies, and quality of service. His current research interests include mobile networks, multimedia applications, and virtualization.

Carlos M. Lentisco received the telecommunications engineering degree from the Universidad Carlos III de Madrid, Madrid, Spain, in 2013, and the M.S. degree in networks and telematic services engineering from Universidad Politécnica de Madrid (UPM), Madrid, Spain. He is currently a Ph.D. candidate in Telematics Systems Engineering Department, UPM. His research interests include multimedia streaming, mobile networks, and virtualization.

Encarna Pastor received the M.S. and Ph.D. degrees in computer science from the Universidad Politécnica de Madrid (UPM), Madrid, Spain, in 1980 and 1988, respectively. She is currently a Full Professor at UPM, specializing in the fields of computer networking, multimedia applications, and Internet technologies. Her current research interests include content delivery networks, multimedia networking, and quality of experience.