Adaptive Video Streaming: Rate and Buffer on the Track of Minimum Re-buffering

Jordi Mongay Batalla, Piotr Krawiec, Andrzej Beben, Piotr Wisniewski, and Andrzej Chydzinski

***Abstract*—New trends of the Future Internet aim to overcome the *best effort* service and offer guaranteed service to the users. Guarantees should be acquired not only at the network level but also at higher layers in order to deliver the best quality of experience. This paper presents a new approach for HTTP-compliant adaptive applications. This solution fulfils guarantees of maximum rebuffering probability even in highly variable environments as the Future Internet seems to be. Guaranteed low rebuffering improves visibly the user's quality of experience during multimedia events. The tests performed confirmed that, unlike rate-based adaptation algorithms, our solution ensures maximum rebuffering and it decisively reduces rate switches in comparison to buffer-based adaptation algorithms.**

***Index Terms*—QoE, HTTP-compliant streaming, media adaptation, queueing models, stream switching, fluid flow modelling**

I. Introduction

ONE of the objectives of the future internet in multimedia streaming is to offer service guarantees, which are directly related with the willingness of the users to use the network for these services. Therefore, it is crucial to provide guarantees of the service in all the layers from network to application and for all the actors from service providers to consumers [1].

In this paper, we analyse the behaviour of adaptive video clients and argue that current adaptation algorithms are not committed to introduce guarantees during the video playout. Furthermore, we note that their role is limited to do their best in front of network variability and difficultly manage the bandwidth follow-up reacting improperly to the punctual variations. Taking into account that some mechanisms present in the future internet introduce high variability in the network (e.g., dynamic caching, multi-source multi-path streaming, etc.), advanced adaptation algorithms should be launched.

Our paper proposes a new adaptive solution, called Adaptation & Buffer Management Algorithm (ABMA), which ensures that the rebuffering probability is under given threshold during the video streaming. Rebuffering creates video image freezes, which are the main impediment in video quality of Experience (QoE). The proposed adaptation solution adjusts the queue size for absorbing short-time network variability while allowing streaming rate switching for adapting to long-time bandwidth variability.

The preliminary idea of ABMA was presented in [2]. However, the model presented there was not solved and the algorithm was based on numerical solution with high computational cost, which prevented its use in most of the devices. The complete version of ABMA (including mathematical analysis and solution) presented here may be fully implemented in medium-class computation devices since the operations to be performed are based on systems of linear equations. The proposed solution presents outstanding behaviour in scenarios with highly variable network conditions.

Manuscript received May 28, 2015; revised November 29, 2015. This work was supported in part by the European CHIST-ERA DISEDAN project under Grant ERA-NET-CHIST-ERA-II/01/2014.

J. Mongay Batalla, P. Krawiec and P. Wisniewski are with Warsaw University of Technology and National Institute of Telecommunications, Warsaw, Poland (e-mail of the correspondig author: jordim@interfree.it).

A. Beben is with Warsaw University of Technology, Warsaw, Poland.

A. Chydzinski is with Silesian University of Technology, Gliwice, Poland.

Digital Object Identifier ???

II. Related Work

Different solutions of adaptive streaming have been proposed during the last years. Stream-switching adaptive streaming (Akamai HD Video Streaming, Adobe Dynamic Streaming, Dynamic Adaptive Streaming over HTTP – DASH, etc.) has caught on thanks to the flexibility and the respect for the all-HTTP approach.

In stream-switching adaptive streaming, the adaptation algorithm selects the best (highest bitrate) representation of each chunk of content (also called segment), whose bitrate is not higher than the network download rate in order to avoid rebuffering situations. The buffer of the adaptation client is in charge of absorbing the variations of the network bandwidth in order to maintain stable video rate. Estimated download rate and measured buffer occupancy are the parameters used by the adaptation clients to assess the network conditions. In fact, the adaptation algorithms existing nowadays can be divided in two different groups: rate-based and buffer-based.

The majority of algorithms are based on estimated download rate, r. They select the representation rate of segment $k+1$: R_{k+1} as the maximum of the i possible representation rates, so that R_{k+1} is lesser or equal to the download rate estimated after the download of the segment $k(r_k)$: $R_{k+1} = \max\{R^i \in \mathfrak{R} : R^i \leq r_k\}$.

The differences between various rate-based algorithms come from the methods to estimate the rate. Several authors discussed the value of historical estimated rate values [3]–[7] to avoid oscillations, which could appea,r by using only the estimated rate of one segment [8]. Algorithms considering historical data react slower to the changes in network conditions, which may bring out rebuffering situations.

There are several issues related to rate-based decisions. One of them is the unfairness of bandwidth repartition when different players compete for bandwidth, i.e., the players

selecting higher bitrate are able to observe higher bandwidth [9]. However, the most important shortcoming of these algorithms is the impossibility of distinguishing long and short-time variations in the network, which leads to inappropriate adaptation decisions. One example of this is the sensitiveness of the algorithms to variable-size segments (due to Variable Bit Rate coding), which provokes rebuffering situations even in stable network conditions. Because of this, the rate-based adaptation algorithms are not able to offer QoE guarantees.

The other class of adaptation algorithms considers the state of the buffer after downloading segment k in order to decide which representation should be selected for the segment $k+1$. Basically, two main parameters can be analysed for taking adaptation decisions: the occupancy of the buffer (after downloading segment k) and buffer changes (i.e., variation rate of buffer occupancy). In the first case, the general idea is to increase the representation rate if the buffer occupancy is higher than give threshold $R_{k+1} = min\{R^i \in \mathfrak{R} : R^i > R_k\}$ if $B_k > B_{max}$. Similarly, the algorithm decreases the representation rate when the buffer is lower than B_{min}: $R_{k+1} = \max\{R^i \in \mathfrak{R} : R^i < R_k\}$ if $B_k < B_{min}$. Otherwise, the representation is the same as in segment k : $R_{k+1} = R_k$ if $B_{min} \leq B_k \leq B_{max}$.

In the last years, Huang et al. proposed an algorithm that selects the representation as a function of the buffer occupancy [10]. Such a function could be linear (first approach) or concave (more conservative).

In [11], the authors analyse buffer (counted in time) changes in order to include the effect of segment size variability into adaptation decisions. They compare the playout segment duration (Ω seconds) with the download time of segment k (called Segment Download Time or, briefly, SDT_k) and increase/decrease the representation rate when the relation Ω/SDT_k is higher/lower than respective thresholds.

Buffer-based algorithms are not able to differentiate long- and short-term variations of network conditions (any change in buffer occupancy may provoke a representation switching) and therefore, they fail in assuring stability during download process (representation switches are observed often).

III. Model of Client Buffer

This section presents the Adaptation & Buffer Management Algorithm which aims to ensure acceptable probability of rebuffering during the playout of the content. The ABMA proposes two parallel adaptations: on the one hand, the buffer size adapts to absorb short-term network variation (non-stationary variation) and, on the other hand, the representation bitrate adapts to long-term network variations (stationary variations). The combination of both adaptation actions may keep up target value of rebuffering probability during the transmission.

A. Functioning of Stream-switching Adaptive Client

The stream-switching adaptive download follows similar structure as depicted in Fig. 1. First, the client logic consecutively requests new video chunks (HTTP 1.1 avoids RTT delays by requesting more than one segment at once). Secondly, the TCP protocol downloads video chunks from server, whenever TCP socket is not full. Furthermore, the client logic

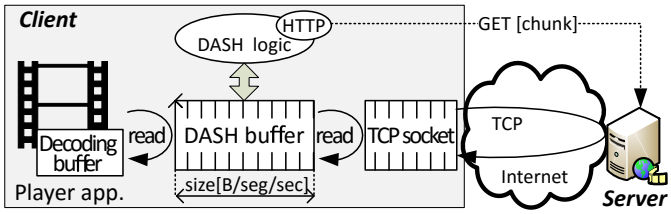


Fig. 1. Buffering chain at client side.

reads data from TCP socket, whenever its buffer is not full (and TCP socket is not empty). The TCP socket length depends on the operating system and may be time-variable (auto-tuning option); it usually spans from KB to MB (in Ubuntu minimum, default and maximum values of socket length are 4 KB, 87 KB and 6 MB, respectively). Finally, the player application fetches data, into its decoding buffer, from client buffer at variable rate following decoding/playout rate. Each reading/fetching operation is performed independently and asynchronously.

We distinguish two main types of buffer management logics i.e. direct and progressive [3]. The direct logic, initially designed for VoD applications, assumes that the buffer has an infinite length (limited only by available memory) resulting in video being continuously downloaded at maximum available rate. In contrast, progressive logic (initially designed for live TV applications) assumes that the client buffer has a limited length and that a video is transferred gradually with download rate following playout rate.

The type of management logic has a direct impact on resource utilization on both network and server sides. As clients usually tend to watch only the first short fragment of video [12], the direct scheme highly overuses resources, since it unnecessary downloads big portion video. Moreover, direct download makes deeper the gap between adaptation decision and video playout, which usually results in inefficient utilization of the available resources. These issues, together with other commercial reasons (advertisements), caused that progressive scheme is the preferred choice for the multimedia streaming in the Future Internet, for both VoD and live TV applications.

The progressive management logic tries to keep the buffer full requesting new segments of video as soon as possible. As a consequence, a client can be in one of two non-volatile states: i) in "greedy" state, when TCP socket is not full and download rate equals to maximum available rate, or ii) in "immovable" state, when TCP socket is full and the video is not downloaded; in this state, the download is deferred until finishing the playout of the current segment.

B. Model of the System

The objective of the algorithm is to control the probability of rebuffering. Rebuffering may occur during the greedy state in progressive management logic but it cannot occur during the immovable state since in this state the client's buffer is full and play-out is not paused. Therefore, the state of the system is modelled during the greedy download and, in this situation, we can express the system state as the number of segments in

the queue and in service. The maximum number of segments waiting in the queue is $K-1$ (maximum buffer length) and the segments are served with deterministic service time (service time equals the segment duration, i.e., Ω seconds).

A rebuffering situation occurs when a segment is completely played out and the next segment did not arrive to the client's buffer. Therefore, to calculate the probability of rebuffering, we observe the system immediately after serving each packet. In these moments, we can calculate the steady-state probabilities of finding in the queue $0, 1, 2, \ldots, K-1$ segments, called $P^0, P^1, \ldots, P^{K-1}$. Let us remark that the system does not change in immovable state situations (full buffer) and, therefore, the probability of rebuffering is not affected by immovable situations.

During the service of one segment (Ω seconds), a number of segments arrive to the system, which is modeled by a discrete random variable $A(\Omega)$. The random variable (r.v.) $A(\Omega)$ considers the characteristics of the download process including variable segment size (due to variable bit rate video coding), TCP behavior and variable server and network conditions. We assume (see sub-section IV.C) that the segment arrival process is generic independent (i.e., there is no correlation between the segments arrived in one Ω-period and in the next one). According to the aforementioned assumptions, the system may be modeled by a single GI/D/1/K queue. Let us remark that GI/D/1/K is a loss system, instead in stream-switching the segments are not lost since the system passes to immovable state when the queue is full. Therefore, the model is not valid as far as loss probability is concerned. However, the probabilities in all the states $1, \ldots, K$ are equal in the system and in the model.

$$\begin{cases} P^0 = (P^0 + P^1) \times Pr\{A(\Omega) = 0\}, \\ \quad \vdots \\ P^n = P^0 \times Pr\{A(\Omega) = n\} \\ \qquad + \sum_{a=0}^{n} P^{n+1-a} \times Pr\{A(\Omega) = n\}, \quad n = 1, \ldots, K-2 \\ \sum_{j=0}^{K} P^j = 1 \end{cases} \tag{1}$$

By solving the equations (1), acquired by applying the imbedded Markov chain approach, we are able to calculate the probability of rebuffering P^0. It is indeed true that QoE is affected by both the rebuffering probability and the duration of the rebuffering situations (these both metrics will be measured in our tests), however low values of P^0 indicates fenced rebuffering situations and, thus, preserved QoE. $Pr\{A(\Omega) = n\}$ is the probability that exactly n segments arrive to the system during one segment playout. $Pr\{A(\Omega) = n\}$ is calculated below.

C. Calculation of $Pr\{A(\Omega) = n\}$

Let us consider Fig. 2. In time t, segment 0^{th} has been played out and the client initiates to play segment 1^{st}. At this moment, segment x_0 is being downloaded and finishes the download at time $t + y_1$. In order to calculate $Pr\{A(\Omega) = n\}$, we should consider how many segments arrive in time $\Omega - y_1$.

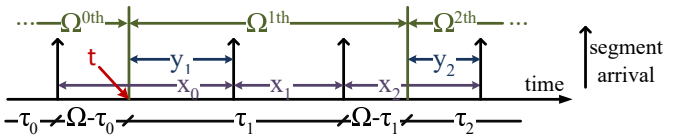


Fig. 2. Segment arrival process in time=Ω.

Let Y be the r.v. describing the residual time of the segment being downloaded when segment 0^{th} finishes to be played out (note that y_1 in Fig. 2 is a realization of r.v. Y). Let S be the r.v. describing the segment download time (i.e., in Fig. 2, $x_0, x_1, x_2, \ldots$ are realizations of S). We assume that S is independent and identically distributed. This assumption is discussed in Section IV.C. Then, we may calculate the probability that n segments arrive to the system during one segment playout (Ω) as the probability that the residual time (Y) is longer than Ω (in the case when $n = 0$) or the probability that the sum of residual time and $n-1$ new arrivals are lower than Ω knowing that any other segment arrival will arrive after Ω. Note that $n = 0$ is the case when the whole Ω period is occupied by the residual time y_1; this case causes rebuffering if the buffer did not contain any other segment (since the playout finishes and no segment has arrived). The formula (2) presents the formal definition of the explained $Pr\{A(\Omega) = n\}$:

$$Pr\{A(\Omega) = n\} = \begin{cases} Pr\{Y > \Omega\}, & n = 0 \\ \int_0^{\Omega} Pr\{Y + \underbrace{S + \cdots + S}_{n-1} = \tau\} \\ \quad \times Pr\{S > \Omega - \tau\}\, d\tau, & n = 1, \ldots, \infty \end{cases} \tag{2}$$

which can be re-written by using the distribution functions:

$$Pr\{A(\Omega) = n\} = \begin{cases} 1 - F_Y(\Omega), & n = 0 \\ F_{Y+\underbrace{S + \cdots + S}_{n-1}}(\Omega) \\ \quad - F_{Y+\underbrace{S + \cdots + S}_{n}}(\Omega), & n = 1, \ldots, \infty \end{cases} \tag{3}$$

where F_Y is the cumulative distribution function of r.v. Y. Formula (3) can be explained as follows: the probability that exactly n segments arrive to the system during Ω equals the probability that n or more segments arrive to the system (during Ω) minus the probability that more than n segments arrive to the system.

From Fig. 2, we can observe that Y is equal to y if and only if the sum of the residual time plus the SDTs of the segments downloaded in the current slot Ω, is equal to τ; whereas the download time of the next segment will be equal to $\Omega - \tau + y$. Therefore, we may calculate F_Y as the integral (over all values of τ and y) of the sum of probabilities such that $1, 2, \ldots, \infty$ segments are downloaded during the time slot. This is expressed in (4).

$$F_Y(y) = \int_0^{y} \int_0^{\Omega} \sum_{a=1}^{\infty} Pr\{Y + \underbrace{S + \cdots + S}_{a-1} = \tau\} \times Pr\{S = \Omega - \tau + t\}\, d\tau\, dt \tag{4}$$

Now we will demonstrate how the probability $Pr\{A(\Omega) = n\}$ can be computed for an arbitrary distribution of the segment download time and arbitrary n. Since the solution of formulas (2) and (3) is not immediate except for the case when SDT distribution function has memoryless property, we proposed a solution based on the combination of the renewal theory (see [13], chapters VI.6 and XI) together with the Laplace transform technique.

We will denote by $G(t, x)$ the time-dependent distribution of variable $y_1 = y_1(t)$ (see Fig. 2), i.e., $G(t, x) = Pr\{y_1(t) < x\}$.

Using the full probability formula with respect to the first segment download time, the integral equation for $G(t, x)$ can be written:

$$G(t, x) = F(t + x) - F(t) + \int_0^t G(t - u, x)\, \mathrm{d}F(u), \quad (5)$$

where $F(x) = Pr\{S < x\}$.

By Theorem 1, p. 185 of [13], the solution of this equation has the form:

$$G(t, x) = \int_0^t (F(t + x - u) - F(t - u))\, \mathrm{d}H(u), \quad (6)$$

where $H(u)$ is the renewal function defined as $H(u) = \sum_{k=0}^{\infty} F^{k*}(u)$, where $F^{0*}(u) = 1$, $F^{1*}(u) = F(u)$ and $F^{k*}(u)$, $k \geq 2$ is the k-fold convolution of the distribution function F with itself, i.e.:

$$F^{k*}(u) = \int_0^u F^{k-1}(u - v)\, \mathrm{d}F(v). \quad (7)$$

The steady-state distribution of y_1, i.e., the distribution for $t \to \infty : G(x) = \lim_{t\to\infty} G(t, x)$, can be obtained from the key renewal theorem (see p. 363 of [13]) and equals:

$$G(x) = \frac{1}{m} \int_0^x (1 - F(u))\, \mathrm{d}u, \quad (8)$$

where $m = E(S) = \int_0^{\infty} (1 - F(u))\, \mathrm{d}u$.

The steady-state (independent of the time slot considered) probability that in the time interval of length Ω the number of finished segment downloads is n equals:

$$D_n(\Omega) = Pr\{A(\Omega) = n\} = Pr\{Y + \underbrace{S + \cdots + S}_{n-1} < \Omega\} - Pr\{Y + \underbrace{S + \cdots + S}_{n} < \Omega\} \quad (9)$$

Therefore, we have

$$D_n(\Omega) = G_n(\Omega) - G_{n+1}(\Omega), \quad n = 0, 1, 2, \ldots \quad (10)$$

where $G_0(x) = 1$, $G_1(x) = G(x)$, and

$$G_n(x) = G_{n-1} * F(x) = \int_0^{\infty} G_{n-1}(x - v)\, \mathrm{d}F(v), \quad n \geq 2.$$

The values of $D_n(\Omega)$ can be obtained using the Laplace transform. Namely, denoting

$$d_n(s) = \int_0^{\infty} e^{-sx} D_n(x)\, \mathrm{d}x, \quad (11)$$

$$g_n(s) = \int_0^{\infty} e^{-sx}\, \mathrm{d}G_n(x), \quad (12)$$

$$f(s) = \int_0^{\infty} e^{-sx}\, \mathrm{d}F(x), \quad (13)$$

we have:

$$g(s) = g_1(s) = \int_0^{\infty} e^{-sx}\, \mathrm{d}G(x) = \frac{1}{m} \int_0^{\infty} e^{-sx} (1 - F(x))\, \mathrm{d}x = \frac{1 - f(s)}{ms} \quad (14)$$

$$g_n(s) = g(s) f^{n-1}(s), \quad n \geq 1, \quad (15)$$

and

$$d_0(s) = \frac{g_0(s) - g_1(s)}{s} = \frac{1 - g(s)}{s} = \frac{f(s) + ms - 1}{ms^2}, \quad (16)$$

while for $n \geq 1$:

$$d_n(s) = \frac{g_n(s) - g_{n+1}(s)}{s} = g(s) f^{n-1}(s) \frac{1 - f(s)}{s} = \frac{f^{n-1}(s)(1 - f(s))^2}{ms^2} \quad (17)$$

The values of $D_n(\Omega)$ can be computed effectively using one of the available methods for the Laplace transform inversion. For instance, using the method of [14] we obtain:

$$D_n(\Omega) \cong \frac{e^{C/2\Omega}}{2l\Omega} \sum_{k=0}^{p} \sum_{j=0}^{q+k} \binom{p}{k} 2^{-p} (-1)^j b_j(n, \Omega), \quad (18)$$

where:

$$b_0(n, \Omega) = d_n\left(\frac{C}{2l\Omega}\right) + 2 \sum_{j=1}^{l} \mathrm{Re}\left[d_n\left(\frac{C}{2l\Omega} + \frac{ij\pi}{l\Omega}\right) e^{ij\pi/\Omega}\right], \quad (19)$$

$$b_k(n, \Omega) = 2 \sum_{j=1}^{l} \mathrm{Re}\left[d_n\left(\frac{C}{2l\Omega} + \frac{ij\pi}{l\Omega} + \frac{ik\pi}{\Omega}\right) e^{ij\pi/\Omega}\right], \quad k \geq 1, \quad (20)$$

and p, q, C, l are parameters of the inversion method. Their typical values proposed in [14] try to balance the accuracy and the algorithm response time and are: $p = 38$, $q = 11$, $C = 19$ and $l = 1$. In [14], Abate, Choudhury and Whitt present a detailed discussion on the accuracy of the inversion method. In particular, it is suggested that if the resulting accuracy is not good enough, then the parameters p and l should be increased.

Below we present examples of a few particular distributions of the segment download time. Let us remark that the segment download time is the sum of two effects: chunk size and network variability (causing jitter) and, therefore, it is not possible to foresee its distribution a priori. Below, we perform measurements for analysing the distribution of the SDT in real conditions.

Exponential distribution. For any r.v. S with memoryless property (exponential or geometrical distribution), formulas (2) and (3) are easily solved. Let us remark that S is not memoryless in the reality, so it cannot have an exponential distribution function.

Folded normal distribution. Assuming that the probability density function of the segment download time has the form:

$$F'(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} + \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x+\mu)^2}{2\sigma^2}}, x > 0, \sigma > 0,$$

we have:

$$f(s) = \frac{1}{2}e^{\frac{s^2\sigma^2}{2}-s\mu}\left[1+\text{Erf}\left(\frac{\mu-s\sigma^2}{\sigma\sqrt{2}}\right)\right] + \frac{1}{2}e^{\frac{s^2\sigma^2}{2}+s\mu}\left[1-\text{Erf}\left(\frac{\mu+s\sigma^2}{\sigma\sqrt{2}}\right)\right],$$
$$m = \sigma\sqrt{\frac{2}{\pi}}e^{-\frac{\mu^2}{2\ \sigma^2}} - \mu\ \text{Erf}\left(-\frac{\mu}{\sqrt{2}\sigma}\right),$$

where:

$$\text{Erf}\,(z) = \frac{2}{\sqrt{\pi}}\int_0^z e^{-t^2}\,\text{d}t.$$

As a consequence, we obtain:

$$d_0(s) = \frac{\frac{1}{2}e^{\frac{s^2\sigma^2}{2}-s\mu}\left[1+\text{Erf}\left(\frac{\mu-s\sigma^2}{\sigma\sqrt{2}}\right)\right]}{m\ s^2} + \frac{\frac{1}{2}e^{\frac{s^2\sigma^2}{2}+s\mu}\left[1-\text{Erf}\left(\frac{\mu+s\sigma^2}{\sigma\sqrt{2}}\right)\right]+ms-1}{m\ s^2}$$
$$= \frac{1}{s} + \frac{e^{\frac{s^2\sigma^2}{2}}\left[e^{-s\ \mu}\left(1+\text{Erf}\left(\frac{\mu-s\ \sigma^2}{\sqrt{2}\sigma}\right)\right)\right]}{2\ \ s^2\left[\sigma\sqrt{\frac{2}{\pi}}e^{-\frac{\mu^2}{2\ \sigma^2}} - \mu\ \text{Erf}\left(-\frac{\mu}{\sqrt{2}\sigma}\right)\right]} + \frac{e^{\frac{s^2\sigma^2}{2}}\left[e^{s\ \mu}\left(1-\text{Erf}\left(\frac{\mu+s\ \sigma^2}{\sqrt{2}\sigma}\right)\right)\right]-2}{2\ \ s^2\left[\sigma\sqrt{\frac{2}{\pi}}e^{-\frac{\mu^2}{2\ \sigma^2}} - \mu\ \text{Erf}\left(-\frac{\mu}{\sqrt{2}\sigma}\right)\right]},$$

$$d_n(s) = \left(\frac{1}{2}e^{\frac{s^2\sigma^2}{2}-s\mu}\left[1+\text{Erf}\left(\frac{\mu-s\sigma^2}{\sigma\sqrt{2}}\right)\right] + \frac{1}{2}e^{\frac{s^2\sigma^2}{2}+s\mu}\left[1-\text{Erf}\left(\frac{\mu+s\sigma^2}{\sigma\sqrt{2}}\right)\right]\right)^{n-1} \times\left(1-\frac{1}{2}e^{\frac{s^2\sigma^2}{2}-s\mu}\left[1+\text{Erf}\left(\frac{\mu-s\sigma^2}{\sigma\sqrt{2}}\right)\right] - \frac{1}{2}e^{\frac{s^2\sigma^2}{2}+s\mu}\left[1-\text{Erf}\left(\frac{\mu+s\sigma^2}{\sigma\sqrt{2}}\right)\right]\right)^2\Bigg/\left(m\ s^2\right)$$
$$= \frac{e^{\frac{s^2\sigma^2(n-1)}{2}}}{2^{n-1}\ s^2}\left[e^{-s\ \mu}\left(1+\text{Erf}\left(\frac{\mu-s\ \sigma^2}{\sqrt{2}\sigma}\right)\right) + e^{s\ \mu}\left(1-\text{Erf}\left(\frac{\mu+s\ \sigma^2}{\sqrt{2}\sigma}\right)\right)\right]^{n-1} \times\left[1-\frac{1}{2}e^{\frac{s^2\sigma^2}{2}}\left(e^{-s\ \mu}\left(1+\text{Erf}\left(\frac{\mu-s\ \sigma^2}{\sqrt{2}\sigma}\right)\right) + e^{s\ \mu}\left(1-\text{Erf}\left(\frac{\mu+s\ \sigma^2}{\sqrt{2}\sigma}\right)\right)\right)\right]^2\Bigg/\left(\sigma\sqrt{\frac{2}{\pi}}e^{-\frac{\mu^2}{2\ \sigma^2}} - \mu\ \text{Erf}\left(-\frac{\mu}{\sqrt{2}\sigma}\right)\right), \quad n \geq 1.$$

Gamma distribution. Assuming that the probability density function of the segment download time has the form:

$$F'(x) = \frac{e^{-\frac{x}{\lambda}}x^{\alpha-1}\lambda^{-\alpha}}{\Gamma(\alpha)}, \quad x>0, \alpha>0, \lambda>0,$$

we have:

$$f(s) = \left(s+\frac{1}{\lambda}\right)^{-\alpha}\lambda^{-\alpha},$$
$$m = \alpha\lambda,$$

which gives:

$$d_0(s) = \frac{\lambda^{-\alpha}\,(s+1/\lambda)^{-\alpha} + s\alpha\lambda - 1}{\alpha\ \lambda\ s^2},$$
$$d_n(s) = \frac{\lambda^{-\alpha(n-1)}\,(s+1/\lambda)^{-\alpha(n-1)}\left(1-\lambda^{-\alpha}\,(s+1/\lambda)^{-\alpha}\right)^2}{\alpha\ \lambda\ s^2}, \quad n \geq 1.$$

Uniform distribution. Assuming that the probability density function of the segment download time has the form:

$$F'(x) = \frac{1}{b-a}, \qquad 0 \leq a < x < b,$$

we have:

$$f(s) = \frac{e^{-as}-e^{-bs}}{s(b-a)},$$
$$m = \frac{b-a}{2},$$

which gives:

$$d_0(s) = \frac{(b-a)\,s + \frac{2\left(e^{-as}-e^{-bs}\right)}{(b-a)s} - 2}{(b-a)\ s^2},$$
$$d_n(s) = \frac{2\left(1-\frac{e^{-as}-e^{-bs}}{(b-a)s}\right)^2\left(\frac{e^{-as}-e^{-bs}}{(b-a)s}\right)^{n-1}}{(b-a)\ s^2}, \quad n \geq 1.$$

The proposed solution is computationally simple, which makes feasible the implementation of the adaptation algorithm in any device (mobile phone, tablet, etc.).

D. Definition of the Adaptation and Buffer Management Algorithm (ABMA)

The objective of ABMA is to determine the highest representation rate for the segment $k+1$: $R_{k+1} = R^i \in \mathfrak{R}$ and the client's buffer size: $B_{k+1} < M$, which ensure the rebuffering probability $P^0(B_{k+1})$ to be lower than given threshold ε. The value M [seconds] is the maximum possible buffer size limited by the conditions of the service as, e.g., the maximum video length stored by the application without playing out (this depends on commercial and advertisements policies) [12]. The value of M is fixed at the beginning of the transmission in our model and, as explained above, it depends on the application.

The algorithm is run when a segment k has been downloaded and a new SDT value (SDT_k) has been measured. The initialisation phase (step 1 in Fig. 3) assumes that the representation of segment $k+1$: R_{k+1}, is equal to the representation of the downloaded segment k, i.e., $R_{k+1} = R_k$, as shown in Fig. 3. In this step, ABMA calculates the moments (typically mean and variance) of the SDT distribution (from the last N probes stored, including SDT_k). The estimation based on N probes introduce an error (estimation error), which will be discussed in Section IV.D. However, the measurements performed in our

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JSAC.2016.2577360, IEEE Journal on Selected Areas in Communications

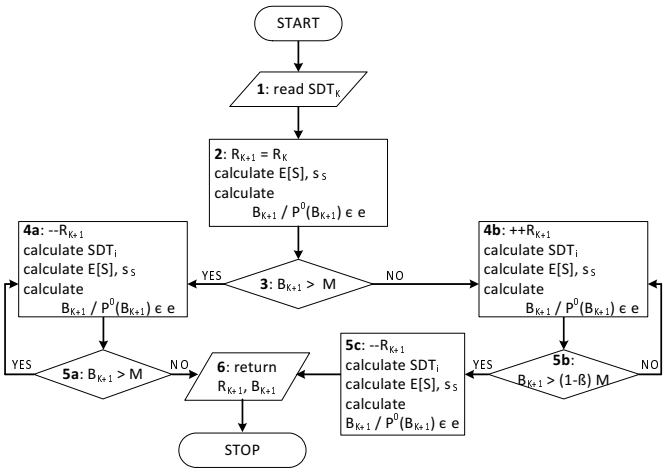


Fig. 3. Flowchart of ABMA.

implementation for several real scenarios showed that $N = 50$ probes introduces enough accuracy in the estimation of the statistical moments of the SDT distribution (i.e., the difference of the standard deviation between 50 probes and 500 probes is negligible). The distribution function (shape) of segment download time should be assumed a priori (this assumption is discussed in Section IV.A).

At last, the algorithm calculates the buffer in segment $k+1$, i.e., it calculates $B_{k+1}|P^0(B_{k+1}) \leq \varepsilon \wedge P^0(B_{k+1}-1) > \varepsilon$. The equations (1) allow to calculate the probability of rebuffering for given buffer size. In order to calculate the buffer size for given probability of rebuffering, we need to apply a binary search algorithm.

In the step 2, the algorithm checks whether the calculated B_{k+1} is larger than maximum buffer size M. In affirmative case, we should decrease the assumed representation rate (R_{k+1}) since such high representation rate would need a too long buffer to avoid rebuffering. In the negative case, the algorithm may try to increase the representation rate for gaining video quality.

The decrease of R_{k+1} (step 3) means that the algorithm selects the next representation $R^i \in \Re$ with closest lower representation rate. By changing R_{k+1}, the algorithm must scale the measured values of SDT (all the N probes) so that, if the old values of SDT_j $(j = 1, \ldots, N)$ were measured with rate R_j, then the new SDT_i $(i = 1, \ldots, N)$ scaled to R_j, are calculated as: $SDT_i = SDT_j \times R_i/R_j$. Once the new values of SDT_i are calculated, then the algorithm re-calculate the moments of new SDT distribution function and estimates the buffer size $B_{k+1}|P^0(B_{k+1}) \leq \varepsilon \wedge P^0(B_{k+1}-1) > \varepsilon$.

In step 4 the buffer size is compared to M once again and, in the case $B_{k+1} < M$, the calculated values of R_{k+1} and B_{k+1} are valid and the algorithm finishes.

In the increasing adaptation loop, the operations are the same but, at the beginning, the representation rate R_{k+1} is increased (to closest higher representation rate). The algorithm calculates the new SDT values, the new moments of the SDT distribution function and the buffer B_{k+1}. The last is compared with the value $(1-\beta) \times M$. β is an anti-oscillation parameter destined to avoid representation swinging in the case that download rate varies near to two representation rates. At last, the algorithm repeats the previous actions, if needed.

TABLE I
χ^2 TEST RESULTS FOR DISCRIMINATION AGAINST DISTRIBUTION FUNCTIONS

	Exponential	**Folded normal**	**Gamma**	**Uniform**
χ^2_{exp}	59.413	28.125	37.657	78.649
$\chi^2_{0.9,df}$	55.090	54.230	54.230	55.090
df	43	42	42	43

IV. DISCUSSION ON THE ASSUMPTIONS

This section provides a discussion on each one of the assumptions of the ABMA.

A. Distribution Function of SDT

The SDT is the sum of delays of all the packets of one segment within the network. Many papers have been published about the effect of the network into the delay of packets and the general conclusion is that the network in stationary conditions causes normal-like delays [15], especially if the number of hops transferred by the packets is numerous.

Since the SDT value is the sum of different packet delays, the "normality" tendency should be more stressed. In order to show this point, we downloaded 5 long contents from the Internet (the servers were located in various European countries) and measured the SDT values for 2-seconds segments. The SDT values were classified in 45 intervals and the frequencies were compared with the theoretical exponential, folded normal, gamma, and uniform distributions by using the χ^2 test.

The parameters of the theoretical functions were obtained from the SDT samples. Since each function needs to calculate different parameters, then the degrees of freedom (df) of the χ^2 test are different from one function to another. For example, folded normal needs the values of average (μ) and variance (σ), so the df are equal to 42, corresponding to the 45 samples -1 (due to the fact that samples are frequencies and the last frequency is not independent from the others) -2 (calculation of two parameters from the samples).

The results of the five tests were similar regardless of the origin of the content, which reasserts the idea that the shape of the SDT distribution function is independent of the scenario.

Table I presents the results of one (example) test. Concretely, the table shows the values of the discriminant function (χ^2_{exp}) between the histogram of SDT values and each one of the four theoretical distribution functions. Moreover, Table I shows the df of each discriminant function and the tabulate 90% point ($\chi^2_{0.9,df}$).

As we may observe, the distribution function that better fits the SDT measurements is the folded normal (the lowest value of χ^2_{exp}). For uniform and exponential, the null hypothesis ("there are no differences between measured values and theoretical distribution") must be rejected at the 0.9 confidence level. In the cases of gamma and folded normal functions, the null hypothesis cannot be rejected and we may accept

TABLE II
VALIDATION OF "IID" ASSUMPTION. COMPARISON RESULTS

SDT characteristics		Prob. of video rebuffering [$x10^{-7}$]	
μ [s]	σ [s]	Calculation	Simulation
0.25	0.10	4.64	0.87 ± 0.08
	0.20	5.14	1.13 ± 0.10
	0.30	6.35	1.41 ± 0.07
0.5	0.10	10.04	3.14 ± 0.28
	0.20	10.87	3.94 ± 0.31
	0.30	11.46	4.89 ± 0.35
0.75	0.10	12.85	8.41 ± 0.24
	0.20	13.41	9.27 ± 0.16
	0.30	14.27	10.51 ± 0.31

that the measurements follow these distribution functions. The differences of the measurements and theoretical values come from the non-stationary conditions of the network (each test took two hours to finish).

The conclusion is that folded normal better fits real SDT measurements histogram in the Internet. This was the distribution function used in our ABMA implementation.

B. Error of Inversion (of Laplace) Method

In the analysis performed, we applied the approximation made by Abate et al. [14] regarding the parameters used for calculating the inverse of Laplace, i.e., parameters p, q, C, l of equations (18) and (19). Even when the typical values should be valid for our solution, we performed numerical calculation of formula (5) for folded normal arrival process and compared the results with the theoretical ones. For all 100 tests (with different values of μ and σ) there were no differences observed between numerical calculation and theoretical solution. Therefore, we may conclude that the inversion method yields (almost) exact results.

C. Independent and Identically Distributed Segment Arrival Process

The solution of the model assumes that r.v. S (segment arrivals) is independent and identically distributed (iid). Such an assumption is satisfactory only for arrival processes with memoryless feature (i.e., Poissonian or geometrical). In the case of folded normal or other processes, the number of segments arrived during Ω is correlated to the number of segments in the previous period (due to the residual time y_1 shown in Fig. 2), i.e., if during the previous period few segments arrived, then y_1 is probably large and the probability that many segments arrive during the present period Ω is low.

In order to understand the range of the error introduced by this assumption, we simulated the G/D/1/K system (folded normal arrival process) and measured the rebuffering probability (P_0). To calculate the confidence intervals, the simulations were repeated 10 times and each simulation counted 10^9 arrived segments. We compared the results with the value calculated by our model. The time service was equal to 1 ($D = 1$) and the queue size was equal to 9 ($K = 10$).

The high differences in P_0 between simulation and the theoretical calculation (see Table II) are caused by the fact that the calculation does not consider the effect of the residual time; quite the opposite of the simulations, where the service process and the arrival process are independent causing residual times into the service process. Moreover, we may observe that the differences are higher for lower mean values, which may be explained by the fact that the residual time is longer when fewer segments arrive during one service unit.

The rebuffering probability is always higher in the model than in the reality. This means that, for given P_0, the model gives a required buffer size which is higher than it is actually necessary. This is logical since the residual time is not negative. Therefore, we may conclude that the theoretical model overestimates the buffer size, although the proposed buffer sizes ensure assumed probability of rebuffering.

D. Estimation Lag of SDT Distribution

SDT distribution is estimated by means of moments of the set of N most recent SDT probes (assessed for a given representation). Please note that the SDT time series is, in general, non-stationary. As a consequence ABMA may run on outdated data, especially in case of rapid changes of downloading conditions.

The buffered-time reservoir BR needed to counterweight the adaptation lag is bounded by the time required to accommodate N new SDT probes. This accommodation time depends on severity and rapidity of non-stationary bandwidth deterioration, current and target representation rates and representation rate granularity, among others. Moreover, the time BR is directly correlated with estimated SDT and the number of used probes N. As a consequence, we heuristically set buffer reservoir according to $BR = \gamma \times N \times \widehat{SDT}$, where γ is the nonstationarity factor and $\widehat{SDT}$ is the SDT moving average. The BR is considered into the ABMA by reducing the effective value of M, such as $M_{eff} = M - BR$.

The estimation latency of instantaneous downloading conditions is common to almost all adaptation approaches (with the exception of methods based on instant estimators) and, because of this, all adaptation approaches exploit some form of buffer reservoir to compensate for the estimation latency.

V. PERFORMANCE EVALUATION

There are several approaches to assess the performance of adaptation algorithms. The more common approach is to analyse (at the adaptive client level) specific parameters that have direct impact on the quality of the user's experience (at the playout level). Assessment methods following this approach are called no-reference methods. We apply this approach and select a set of parameters that accurately compare the ABMA with other two algorithms: (i) Rate-Based Algorithm (RBA) [16], which selects the highest video representation rate that is not greater than the most recently estimated download rate, and (ii) Buffer Based Algorithm (BBA) [10] which selects the representation based on linear rate map function based on actual buffer occupancy.

For fair comparison of adaptation algorithms, we perform experiments under the same conditions, i.e., same download rate traces and same video files. We developed a fluid flow simulation tool which enables repeating our tests in fully controlled manner. This is possible since the system state is defined by analytical equations that determine values of the system variables in consecutive time instants. The HTTP streaming system is observed just after downloading each video segment, when the adaptation algorithm has just updated information about downloaded segment and is prepared to select the representation for the next video segment. In these time instants, the system state is expressed by three variables: time instant (t_k) in seconds, buffer occupancy (B_k) in playout seconds and representation rate (R_k) in bps.

The variable t_k describes the time instant when the download of k^{th} segment has finished. It is defined by equation (21).

$$t_k = t_{k+1} + SDT_k + \max\left(B_{k-1} + \Omega - B; 0\right) \tag{21}$$

where SDT_k denotes the download time of k^{th} segment, function max(.) determines the duration of download deferring periods occurring when the playout buffer is full, B is the maximum buffer size (expressed in seconds) and Ω is the segment playout time. The k-segment download time SDT_k depends on both the segment file size in bits, and the download rate r_k:

$$SDT_k = \frac{SegSize_k\left(Fadapt(.)\right)}{r_k} \tag{22}$$

The segment size depends on the video representation selected by the adaptation algorithm: $Fadapt(.)$. $Fadapt(.)$ is algorithm-specific and each algorithm uses specific arguments. For example, RBA algorithm uses average download rate estimator, BBA algorithm uses actual buffer occupancy and ABMA uses segment download time characteristics. At last, r_k denotes the average download rate experienced by k^{th} segment. The second variable B_k describes the buffer occupancy observed just after the k^{th} video segment has been downloaded. B_k, expressed by equation (23), defines the total playout time of video frames stored in the playout buffer.

$$B_k = \max\left[B_{k-1} - SDT_k - \max\left(B_{k-1} + \Omega - B; 0\right); 0\right] + \Omega \tag{23}$$

where the first function max[.] is used for modelling the buffer occupancy after re-buffering events, and the second function max(.) defines the download deferring periods. Once the buffer becomes empty, the video remains frozen until the download of current segment will be finished.

The last variable describes the video representation selected for the next downloaded segment, R_{k+1}. The representation is selected by the adaptation specific function and depends on algorithm-specific arguments like r_k, B_k, SDT or others (depending on the adaptation algorithm), see (24).

$$R_{k+1} = F_{adapt}\left(r_k, B_k, \widehat{SDT}, \dots\right) \tag{24}$$

Depending on the applied adaptation algorithm, different information is used, e.g., estimated download rate, buffer occupancy or rebuffering probability.

The fluid flow simulation follows three steps:

Step 1: Initialize the system state. In this step we set $t_0 = 0$, the initial buffer occupancy $B_0 = 0$ and the representation of the first segment, R_1, is set to the representation with the minimum rate, $R_1 = \min_i \{R_i \in \mathfrak{R}\}$.

Step 2: Calculate the values of system variables for consecutive segments. Starting from the first segment, SDT_1 is calculated based on the values of *segment_size* and r_1 following formula (22). Next, we calculate the time instant when the first segment is downloaded by applying equation (21) and using initial buffer occupancy. At last, we calculate the buffer occupancy observed after downloading the first segment by applying equation (23) and the representation for the next segment based on equation (24), which is different for each one of the three algorithms.

Step 3: Calculate the values of performance metrics based on the values of system variables derived in the step 2.

The performance metrics selected to compare the algorithms are: the Representation Selection Efficiency (RSE), the Representation Switch Ratio (RSR), the Representation Switch Amplitude (RSA), the Rebuffering Event Ratio (RER) and, at last but not least, the Rebuffering Event Average Duration (RED). Rebuffering events and efficiency in selecting the representation have high influence on the QoE of video reception but also continuous rate switches decrease the engagement of the users [17], [18]. The selection of such performance metrics instead of MOS metrics agrees with last trend in QoE analysis [19] and is the response to the increasing complexity of video streaming in the Internet.

Specifically, RSE indicates the algorithm capacity of matching the available bandwidth selecting the adequate representation. It is calculated as the relation between the rate of the selected representations and the minimum of the bottleneck and the highest available representation. RSR is the rate of switches between representations. RSA is the mean value of the switches (of representation) performed during the download. RER is the probability of rebuffering in relation to the number of segments. At last, RED alludes to the length (in time) of the rebuffering situations. The formal description of all the parameters is presented in Table III. All of them consider $k = 1, \dots, K$ segments downloaded with representation rate R_k by a network with bandwidth BW (counted in bps). Note that the download rate r_k of segment k may be, in general, lower than BW and it can be, as a maximum, equal to BW.

In the presented experiments, we simulated a single video client playing "Big Buck Bunny" film encoded in different representations ranging from 45 kbps up to 15 Mbps, as defined in the manifest file. The segment playout duration (Ω) was 2s. The maximum playout buffer in video client was dimensioned to 32 segments, which is typically used in commercial clients (about 1 min of continuous video playout). For RBA and BBA adaptation algorithms, we used the default values of parameters that were recommended by their authors. In particular, for RBA, we used moving average estimator with the range of last 50 probes. In case of BBA, we used T^{min} threshold (reservoir) of 5 segments and linear function between 5 and 30 segments as the map between rate and buffer size. In ABMA algorithm, we assumed target video rebuffering probability ε equal to 10^{-4}, SDT sample size N equal to 50

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JSAC.2016.2577360, IEEE Journal on Selected Areas in Communications

TABLE III
FORMAL DESCRIPTION OF THE METRICS USED FOR PERFORMANCE EVALUATION OF ADAPTATION ALGORITHMS

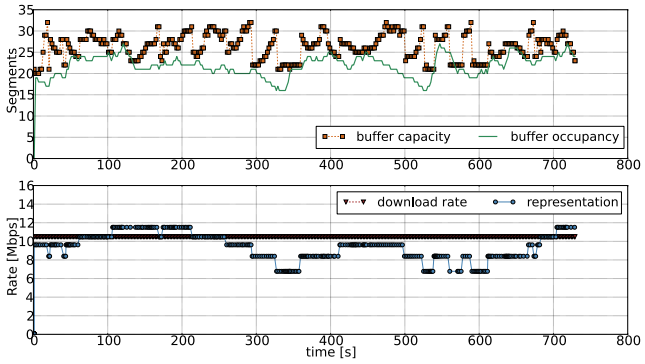
Parameter	Formula
RSE =	$\frac{\frac{1}{K}\times\sum_{k=1}^{K} R_k}{\min\left\{\max_j R_j, BW\right\}}$
RSR =	$\frac{1}{K-1}\times\sum_{k=2}^{K}\begin{cases}1, & \text{if } R_k \neq R_{k-1}\\ 0, & \text{if } R_k = R_{k-1}\end{cases}$
RSA =	$\frac{1}{K\times RSR}\times\sum_{j=2}^{K}\lvert R_j - R_{j-1}\rvert$
RER =	$\frac{1}{K}\times\lvert B(t)=0 \wedge B(t^-)\neq 0\rvert$, where $\lvert\text{function}\rvert$ indicates the cardinality of function
RED =	$\frac{1}{K\times RER}\times\sum_{j=1}^{K\times RER}(t_2-t_1)\ \rvert_j$, where $B(t_2)\neq 0 \wedge B(t_2^-)=0 \wedge B(t_1)=0 \wedge B(t_1^-\neq 0)$

probes (the same value as used for the rate estimator in RBA), nonstationarity factor γ equal to 0.3, and anti oscillation factor β equal to 0.9.

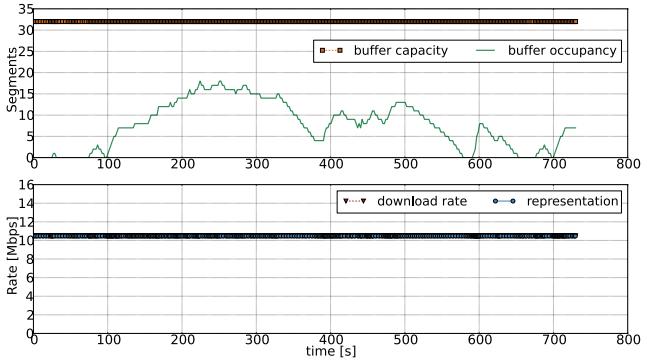
A. Test #1: Constant Download Rate Conditions

In this test, we analyse how ABMA, RBA and BBA work under constant download rate conditions. The algorithms must compensate variations of SDT caused by variable size of downloaded segments. In order to stress adaptation algorithms, we use one of the highest video representation, i.e. 10.4 Mbps and fix a little higher download rate equal to 10.6 Mbps, which causes that adaptation algorithms work in stressed conditions. Moreover, the high rate of video representation assures relatively high segment file size variation.

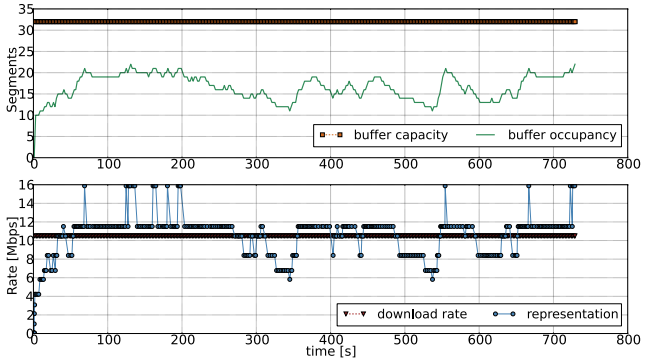
Fig. 4 shows time plots of actual buffer occupancy vs. buffer capacity (upper plot) as well as the rate of video representation selected by adaptation algorithms vs. current download rate (lower plot). The figure presents the results for the three algorithms: ABMA, RBA and BBA. We can observe, in Fig. 4a, that ABMA keeps the buffer far away from empty state avoiding rebuffering events. This comes from continuous estimation of SDT characteristics. In the case when SDT characteristics become worse, then the ABMA increases the playout buffer capacity to avoid rebuffering events, so in such conditions, the maximum buffer size of ABMA is M (32 segments), like the other algorithms. This may be observed in many moments of the transmission, see Fig. 4a. In the case that this action is not enough, then ABMA selects lower representation. From Fig. 4a, we may conclude that ABMA behaves a bit conservative since, from time to time, it selects representation below assumed download rate. This conservativeness comes from assumed relatively low rebuffering threshold (10^{-4}). On the other hand, the RBA algorithm correctly selects representation and it keeps it unchanged over the simulation time. However, we can observe that buffer becomes empty several times causing rebuffering events. This negative effect comes from the fact that rate-based adaptation algorithm does not consider the impact of variable segment size. The BBA algorithm (presented in Fig. 4c) avoids rebuffering events, however it suffers from frequent and significant representation switching. Even under constant download rate conditions, the buffer occupancy significantly



(a) ABMA



(b) RBA



(c) BBA

Fig. 4. Performance of adaptation algorithms under constant download rate conditions.

TABLE IV
THE VALUES OF PERFORMANCE METRICS IN TEST #1

Metric	Adaptation algorithm		
	ABMA	RBA	BBA
RSE [%]	86.9	99.8	101.1
RSR [%]	6.74	0.0	18.74
RSA [Mbps]	1.4	0.0	2.34
RER [%]	0.0	3.7	0.0
RED [s]	0.0	0.8	0.0

varies due to variable size of downloaded segments, thus BBA changes the video representation. Table 4 presents values of performance metrics collected in Test #1.

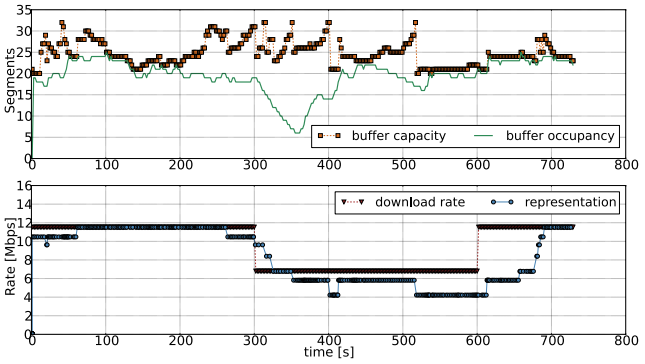
We can observe that ABMA achieves the objective of minimum rebuffering. Moreover, Table IV shows that ABMA has slightly lower RSE than the other algorithms, but it has relatively small representation switching ratio. The RBA does not change representation at all, but it suffers from significant rebuffering ratio (almost every 2 min). On the opposite, BBA suffers from significant and frequent representation switching (up to 18%). The representation is changed almost every 10s, which significantly degrades the quality experienced by users. However, the BBA is characterized by the higher efficiency.

B. Test #2: Highly Variable Download Rate Conditions

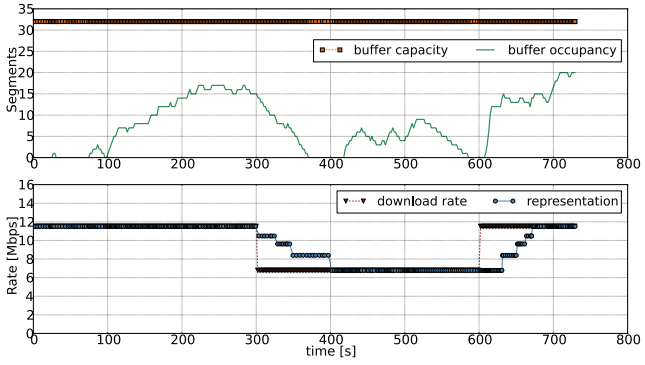
In the second test, we aim to evaluate how the algorithms adapt video representation after sudden change of network conditions. Therefore, we assume that download rate periodically alternates between high (11.53 Mbps) and low (6.8 Mbps) bit rate every 300 s. The length of each phase was tuned to assure semi-stationary conditions. The ratio of high to low bit rate is about 1.7, which assures high variability. Both selected bit rates are slightly higher (few kbps) than representation rates available in the manifest file.

Similarly as above, Fig. 5 presents time plots of actual buffer occupancy vs. buffer capacity (upper plot) and rate of video representation vs. current download rate (lower plot). We can observe that the ABMA selects video representation following the download rate changes with slight delay. This delay is caused by SDT estimation used in ABMA. Moreover, we can observe that ABMA quickly responses to download rate degradation, while it is more preventive in the case of download rate increase. Similarly to previous test, ABMA avoids rebuffering events but it has conservative tendency. Also RBA follows rate changes with slight delay, which is caused by the estimation algorithm used to assess download rate. However, the RBA algorithm suffers from rebuffering events. Moreover, we can observe that this algorithm reacts slower on reduced download rate and faster on increased download rate. This is an undesirable effect since it increases the risk of rebuffering events. In the case of BBA, the buffer occupancy behaves quite smoothly and follows the download rate changes. It also avoids rebuffering events as in the case of ABMA, but it suffers significant representation rate changes occurring in a short time. For instance in the 550^{th} second, the representation rate went from 4 to 12 Mbps and then it went back to 4 Mbps during just a few segments. Thus, we may conclude that the variable network conditions will boost the negative effect of representation switching as it was previously reported in Test #1.

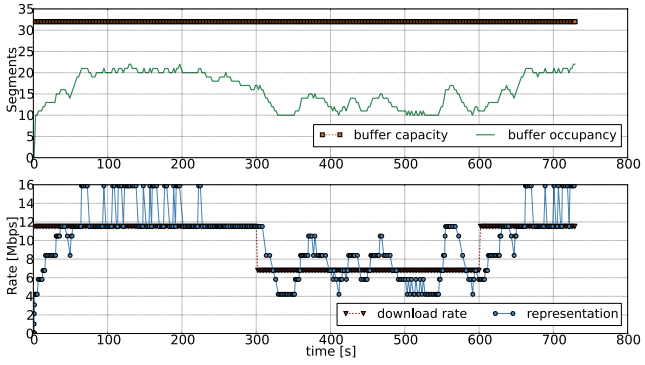
Table V presents the values of performance metrics collected in Test #2. We may observe that rebuffering is avoided (for ABMA and BBA) even in highly variable conditions. Basically, the main conclusions from their analysis are similar to those presented in Test #1. The main differences are the following: (i) the efficiency of ABMA algorithm as well as other algorithms is slightly lower than in Test #1. This effect comes from the fact that we increase the variability in the network, so all adaptation algorithms are more stressed, and (ii) the BBA algorithm significantly increases the RSR, which causes that representation is changed almost every 3 segments.



(a) ABMA



(b) RBA



(c) BBA

Fig. 5. Performance of adaptation algorithms under highly variable network conditions.

TABLE V
THE VALUES OF PERFORMANCE METRICS IN TEST #2

Metric	Adaptation algorithm		
	ABMA	RBA	BBA
RSE [%]	82.3	96.2	102.6
RSR [%]	4.5	2.38	28.4
RSA [Mbps]	1.45	1.18	2.48
RER [%]	0.0	3.3	0.0
RED [s]	0.0	1.27	0.0

C. Test #3: Experiments Under Real Traffic

In this test, we aim to analyse the performance under lifelike download rate conditions. We collected download rate traces

TABLE VI
THE VALUES OF PERFORMANCE METRICS IN TEST #3

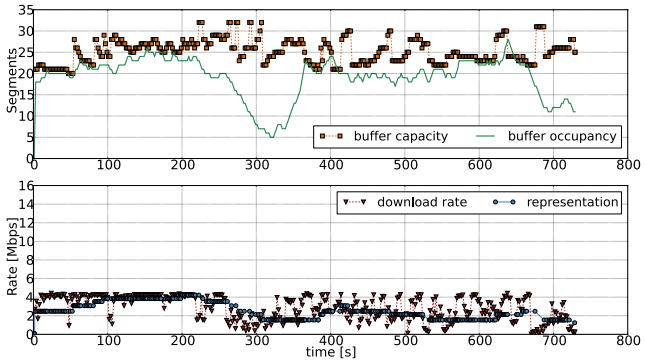
Metric	Adaptation algorithm		
	ABMA	RBA	BBA
RSE [%]	75	82	98
RSR [%]	3.91	5	44
RSA [Mbps]	0.45	0.421	1.16
RER [%]	0.0	0.9	0.0
RED [s]	0.0	0.97	0.0

(every two seconds) by downloading the content by a client connected through WiFi network (shared with other users). Our measurements took 6 hours to get more than 10^5 probes. Since the available measurements of video download rate in the Internet are scarce, we decided to make them available for researchers at http://wp2.tele.pw.edu.pl/disedan/software/traces.

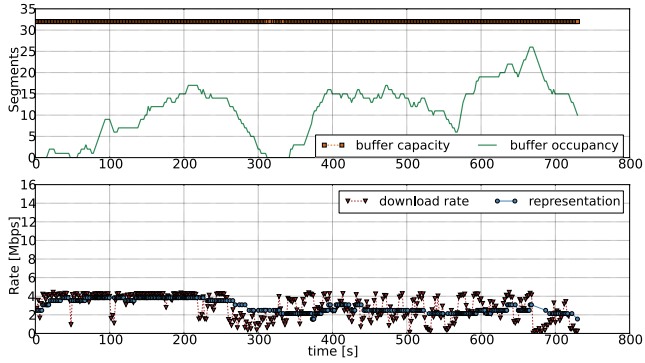
The above measurements were introduced in our fluid-flow simulation environment and the algorithms run under these download rate conditions. Note that the ideal algorithm should select representation of each video segment in such a way that its download time is close to the playout time.

Fig. 6 presents similar set of plots as seen in the previous experiments. In this case, the download rate could be higher than representation rate bringing about the overprovisioned conditions. From these results, we conclude the following: the ABMA algorithm (Fig. 6a) correctly follows the download rate changes and minimizes the representation changes by estimating the distribution of SDT. The SDT distribution provides much more knowledge about the arrival process than just a single value of average rate or buffer occupancy exploited by the other algorithms. In the case of the RBA algorithm (Fig. 6b), we observe rebuffering events and significant variation of buffer occupancy (see situation about 850^{th} second). The BBA algorithm (Fig. 6c) suffers, once again, significant representation switches, which degrade the quality experienced by the end-users.

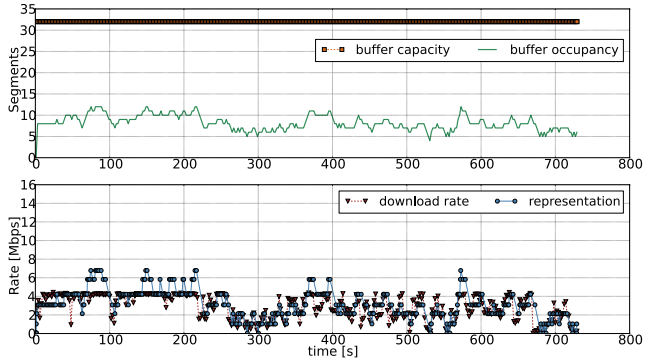
Table VI presents the values of performance metrics collected in Test #3. We observe that the efficiency of all the algorithms is further slightly decreased. Moreover, the representation switching in BBA is significantly increased and the representation changes occur almost every two segments. These effects come from the high variability of download rate caused by WiFi network. However, the most important conclusion is that rebuffering is avoided (in ABMA and BBA) in real conditions scenario.



(a) ABMA



(b) RBA



(c) BBA

Fig. 6. Performance of adaptation algorithms under lifelike download rate conditions.

VI. INTERNET EXPERIMENTS

The experiments conducted over the Internet aim to verify whether the proposed algorithm is capable to compensate variability of downloading conditions and to prevent buffer depletion during play out in real, highly variable network environment.

We implemented our algorithm as a C++ module in order to achieve effortless integration with different multimedia players and other applications as well as easiness in creating further extensions.

The player application was the open-source VLC media player, which allows DASH adaptive streaming by means of dedicated plug-in [20]. Our adaptation module was incorporated into the DASH plug-in. The source code of our prototype and modified VLC DASH plug-in is available on the web page: http://wp2.tele.pw.edu.pl/disedan/software/abma.

A. ABMA Behavior in the Internet

The first experiment was provided to analyse ABMA in case of streaming over the Internet. In particular, we wanted to verify if the proposed algorithm: (i) is capable to compensate

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JSAC.2016.2577360, IEEE Journal on Selected Areas in Communications

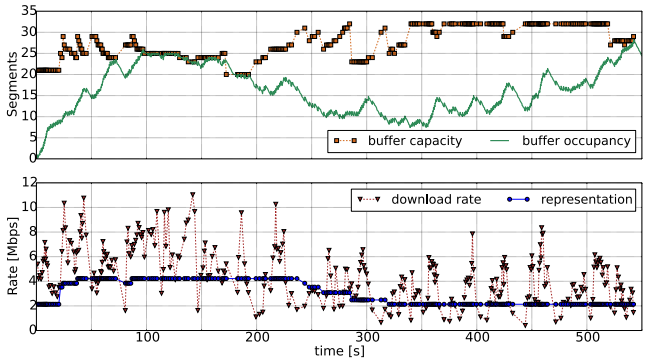


Fig. 7. ABMA during trials over Internet.

variability of SDT, (ii) responds properly to download rate variation, (iii) prevents buffer exhaustion during media playout and (iv) determines optimal media representation rate.

The client (located in Poland) uses WiFi access network and downloads a DASH video sequence from remote server located in Austria. We used "Big Buck Bunny" cartoon [21] in 1080p resolution and the following representation rates: 2.1, 2.5, 3.1, 3.5, 3.8 and 4.2 Mbps. The video was divided into segments with constant length equal to 2 seconds. ABMA was configured with the same parameters as the ones described in Section V.

In parallel with the test client streaming, a second WiFi terminal operated in background and run a Full HD YouTube session. In this way, the bandwidth available for the test client was throttled to the range of the DASH video sequence. In addition, another background session started at about 230^{th} second, what introduced deterioration of the available bandwidth.

Fig. 7 presents the obtained time plot for buffer capacity and buffer occupancy (the upper subplot), and download rate jointly with chosen representation (the lower subplot). In initial phase, ABMA quickly moved towards the highest representation, and next stayed at this level. The buffer capacity stabilized around 25 segments, which was enough to absorb detected download rate fluctuations. After 230^{th} second, when the new competing background stream was launched, the available bandwidth declined and buffer occupancy fell. Therefore, ABMA started to consecutively decrease media representation due to bandwidth starvation. Moreover, the value of buffer capacity determined by ABMA increased in order to compensate higher variability in download rate. Both factors became stable at about 330^{th} second. From that point, ABMA kept media representation unchanged to the end of the experiment, although the download rate fluctuated between 1 and 8 Mbps. Since about 350^{th} second we can observe gradual growth of buffer occupancy, indicating that average available bandwidth was slightly higher than selected representation, but not enough to increase video quality.

During the whole experiment the buffer occupancy was always above zero, what means that the downloaded video sequence was played out without any "freezes".

In conclusion, the obtained results illustrate that ABMA is able to perform appropriate adaptation of video quality and determine proper buffer size on highly variable, uncontrolled Internet environment.

TABLE VII
THE VALUES OF PERFORMANCE METRICS IN INTERNET EXPERIMENTS

Metric	Adaptation algorithm		
	ABMA	RBA	BBA
RSE [%]	73	80	91
RSR [%]	4	7	23
RSA [Mbps]	303	373	415
RER [%]	0.0	1.2	0.0
RED [s]	0.0	5.82	0.0

B. Comparison with Other Algorithms in the Internet

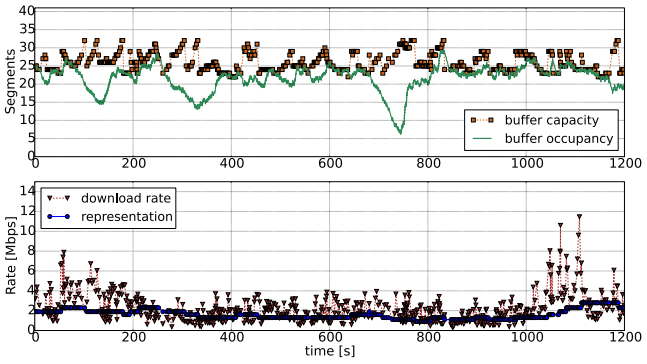
In the following experiment, we focus on comparison of the proposed ABMA with other adaptation methods in real network environment.

For this purpose, in our prototype we developed two additional classes which implement rate-based and buffer-based adaptation algorithms. Those classes were also integrated with the VLC DASH plug-in.

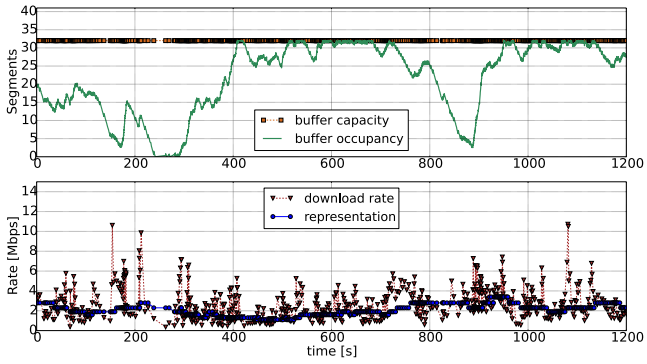
The experiment setup was similar to the presented in the previous subsection and assumes that a WiFi client downloads media content from the same, publicly available server through the Internet, using a multi-domain path. Moreover, two other clients, which are connected to the same WiFi network, launch in the background a couple of different streaming sessions from another server to reduce amount of available bandwidth. Both elements lead to high variability in download rate.

We repeated the test three times, for each of the investigated algorithm. The values of configuration parameters for each algorithm were the same as in Section V. In order to ensure the most similar conditions and repeatability in background streaming, the moment of starting the background sessions was identical for all three trials: during the first 200 seconds there were two parallel sessions in the background, then another two were launched, which finished after about 800 seconds. In this experiment we did not employ an uncontrolled YouTube service for background streaming but, instead, we used a dedicated server, which was placed in different domain than WiFi access point (easy to control). However, for the sake of uncontrolled Internet environment and external interferences occurred in WiFi network, we observed that the traffic conditions slightly vary between each trial. The DASH media content used for the tests was 20 minutes video sequence divided into 2-second long segments. The video was "Big Buck Bunny" [21] with constant resolution 360p, which is available with 14 quality rates. The representations span from 100 kbps up to 4500 kbps. Fig. 8 shows three sets of time plots gathered during the trials for each adaptation algorithm. Corresponding values of performance metrics are presented in Table VII.

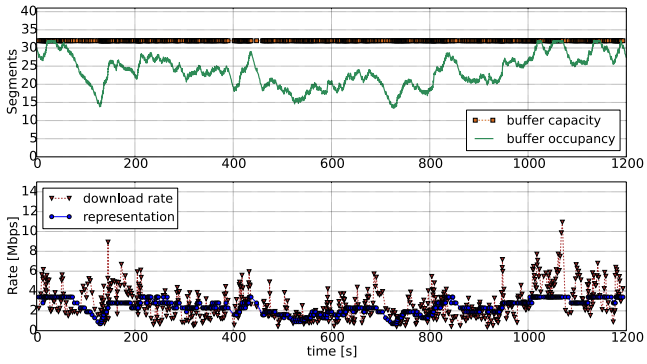
The obtained results confirm aforementioned features of the investigated algorithms. As depicted in Fig. 8a, ABMA



(a) ABMA



(b) RBA



(c) BBA

Fig. 8. Performance of adaptation algorithms observed during trials over Internet.

allows for smooth media playout, as it compensates fluctuations in traffic conditions by modifying the buffer capacity and changing quality rate, if necessary. It avoids extensive representation switching (low value of RSR metric - see Table VII), which is beneficial for the overall quality perceived by the user. However, the behaviour of the proposed algorithm is slightly conservative, and therefore its efficiency is below other algorithms.

On the other hand, RBA, which bases solely on the rate estimation, suffers due to rebuffering events. In turn, BBA algorithm, similarly to ABMA, avoids buffer depletion and achieves the best efficiency (see RSR value in Table VII), but it distorts user's QoE by frequent changes of media representations (value of RSR metric is much higher in case of BBA in comparison to the other algorithms).

VII. Conclusion

The paper discusses the role of adaptive applications in improving the quality of multimedia transmission and, concretely, the capability of assuring users' QoE. We proposed a new class of adaptation algorithms that focuses on assuring maximum (low) rebuffering probability, which is a crucial factor for the satisfaction of the users downloading content through the Internet.

The Adaptation and Buffer Management Algorithm (ABMA) manages both the maximum buffer size and the selection of adequate representation rate for absorbing short and long-term variations in the network and video files.

The results of the tests performed on the ABMA show the efficiency of the algorithm in fulfilling the objective of controlled low rebuffering probability. Moreover, the straightforward solution of the proposed model assures that the ABMA may be installed in devices with medium computing capacity.

The comparison with other classes of algorithms showed the features of the different adaptive logics: Rate-based algorithms focus on correct estimation of download rate but are not able to control rebuffering. On the contrary, Buffer-based algorithms reach also low levels of rebuffering probability, however they react to small changes in the download process, which causes continuous representation switches. At last, ABMA reaches the desired objectives while improving other parameters related to QoE.

ABMA seems to be appropriate in Future Internet scenarios, which are characterised by high variability due to the increasing number of mechanisms in all the levels.

Acknowledgment

This work is part of the DISEDAN project within the European CHIST-ERA Program. We want to thank the other project partners for their support and contribution to the ideas presented here.

References

[1] M. K. Mukerjee, D. Naylor, J. Jiang, D. Han, S. Seshan, and H. Zhang, "Practical, real-time centralized control for cdn-based live video delivery," *SIGCOMM Comput. Commun. Rev.*, vol. 45, no. 4, pp. 311–324, Aug. 2015.

[2] P. Wisniewski, A. Beben, J. Batalla, and P. Krawiec, "On delimiting video rebuffering for stream-switching adaptive applications," in *Communications (ICC), 2015 IEEE International Conference on*, June 2015, pp. 6867–6873.

[3] S. Akhshabi, L. Anantakrishnan, A. C. Begen, and C. Dovrolis, "What happens when http adaptive streaming players compete for bandwidth?" in *Proceedings of the 22Nd International Workshop on Network and Operating System Support for Digital Audio and Video*, ser. NOSSDAV '12. New York, NY, USA: ACM, 2012, pp. 9–14.

[4] M. Mirza, J. Sommers, P. Barford, and X. Zhu, "A machine learning approach to tcp throughput prediction," *Networking, IEEE/ACM Transactions on*, vol. 18, no. 4, pp. 1026–1039, Aug 2010.

[5] B. Seo, W. Cui, and R. Zimmermann, "Efficient video uploading from mobile devices in support of http streaming," in *Proceedings of the ACM Multimedia Systems Conference*, ser. ACM MMSys 2012, 2012.

[6] K. E. Psannis and Y. Ishibashi, "Enhanced h.264/avc stream switching over varying bandwidth networks." *IEICE Electronic Express*, vol. 5, no. 19, pp. 827–832, 2008.

[7] M. Grafl, C. Timmerer, H. Hellwagner, G. Xilouris, G. Gardikis, D. Renzi, S. Battista, E. Borcoci, and D. Negru, "Scalable media coding enabling content-aware networking," *MultiMedia, IEEE*, vol. 20, no. 2, pp. 30–41, April 2013.

[8] T. C. Thang, Q.-D. Ho, J. W. Kang, and A. Pham, "Adaptive streaming of audiovisual content using mpeg dash," *Consumer Electronics, IEEE Transactions on*, vol. 58, no. 1, pp. 78–85, February 2012.

[9] J. Jiang, V. Sekar, and H. Zhang, "Improving fairness, efficiency, and stability in http-based adaptive video streaming with festive," in *Proceedings of the 8th International Conference on Emerging Networking Experiments and Technologies*, ser. CoNEXT '12. New York, NY, USA: ACM, 2012, pp. 97–108.

[10] T.-Y. Huang, R. Johari, N. McKeown, M. Trunnell, and M. Watson, "A buffer-based approach to rate adaptation: Evidence from a large video streaming service," *SIGCOMM Comput. Commun. Rev.*, vol. 44, no. 4, pp. 187–198, Aug. 2014.

[11] C. Liu, I. Bouazizi, and M. Gabbouj, "Rate adaptation for adaptive http streaming," in *Proceedings of the Second Annual ACM Conference on Multimedia Systems*, ser. MMSys '11. New York, NY, USA: ACM, 2011, pp. 169–174.

[12] A. Finamore, M. Mellia, M. M. Munafò, R. Torres, and S. G. Rao, "Youtube everywhere: Impact of device and infrastructure synergies on user experience," in *Proceedings of the 2011 ACM SIGCOMM Conference on Internet Measurement Conference*, ser. IMC '11. New York, NY, USA: ACM, 2011, pp. 345–360.

[13] W. Feller, *An introduction to probability theory and its applications. Volume II*, ser. Wiley series in probability and mathematical statistics. New York, London, Sydney: J. Wiley, 1971.

[14] J. Abate, G. L. Choudhury, and W. Whitt, *Computational Probability*, W. K. Grassmann, Ed. Boston, MA: Springer US, 2000.

[15] K. E. Psannis and Y. Ishibashi, "Impact of video coding on delay and jitter in 3g wireless video multicast services," *EURASIP J. Wirel. Commun. Netw.*, vol. 2006, no. 2, pp. 51–51, Apr. 2006.

[16] S. Akhshabi, A. C. Begen, and C. Dovrolis, "An experimental evaluation of rate-adaptation algorithms in adaptive streaming over http," in *Proceedings of the Second Annual ACM Conference on Multimedia Systems*, ser. MMSys '11. New York, NY, USA: ACM, 2011, pp. 157–168.

[17] O. Oyman and S. Singh, "Quality of experience for http adaptive streaming services," *IEEE Communications Magazine*, vol. 50, no. 4, pp. 20–27, April 2012.

[18] C. Müller, S. Lederer, and C. Timmerer, "An evaluation of dynamic adaptive streaming over http in vehicular environments," in *Proceedings of the 4th Workshop on Mobile Video*, ser. MoVid '12. New York, NY, USA: ACM, 2012, pp. 37–42.

[19] A. Balachandran, V. Sekar, A. Akella, S. Seshan, I. Stoica, and H. Zhang, "Developing a predictive model of quality of experience for internet video," *SIGCOMM Comput. Commun. Rev.*, vol. 43, no. 4, pp. 339–350, Aug. 2013.

[20] C. Müller and C. Timmerer, "A vlc media player plugin enabling dynamic adaptive streaming over http," in *Proceedings of the 19th ACM International Conference on Multimedia*, ser. MM '11. New York, NY, USA: ACM, 2011, pp. 723–726.

[21] S. Lederer, C. Müller, and C. Timmerer, "Dynamic adaptive streaming over http dataset," in *Proceedings of the 3rd Multimedia Systems Conference*, ser. MMSys '12. New York, NY, USA: ACM, 2012, pp. 89–94.

Jordi Mongay Batalla He received his M.Sc. degree from Universitat Politecnica de Valencia (2000) and Ph.D. degree from Warsaw University of Technology (2009), where he still works as an assistant professor. In the past he worked at Telcordia Poland (Ericsson R&D), and he is now with the National Institute of Telecommunications, where, since 2010, he is head of the Internet Architectures and Applications Department. His research interest focuses mainly on Quality of Service/Quality of Experience for multimedia adaptive streaming, Future Internet architectures (Content Aware Networks, Information Centric Networks, architectures for NFV) as well as applications for Future Internet (Internet of Things, Smart Cities, IPTV). He is author or co-author of more than 100 papers published in books, and international and national journals and conference proceedings and editor/TPC member of several journals and conferences.

Piotr Krawiec He received his M.Sc. (2005) and Ph.D. (2011) degrees in telecommunications from Warsaw University of Technology. Since 2012 he is an Assistant Professor at the Department of Internet Architectures and Applications, National Institute of Telecommunications, and Institute of Telecommunications, Warsaw University of Technology. His research areas include IP networks (fixed and wireless), Future Internet architectures and applications, prototyping and testbeds.

Andrzej Beben He received his M.Sc. and Ph.D. in telecommunications from Warsaw University of Technology (WUT), Poland, in 1998 and 2001, respectively. Since 2001 he has been Assistant Professor at WUT, where he is a member of the Internet Architectures and Applications research group. His research areas cover Future Internet, IP networks, Information Centric Networks, adaptive video streaming, network virtualisation, traffic engineering, multi-criteria decision theory, simulation techniques, measurement methods, and testbeds.

Piotr Wisniewski He is a Ph.D. candidate at the Institute of Telecommunications at the Warsaw University of Technology, where he received his M.Sc. (2010) and B.SC. (2009) degrees in Telecommunications. He works as a specialist at the National Institute of Telecommunications in Warsaw (Poland). His research interests include: adaptive media streaming, Quality of Service, Information Centric Networks, network virtualization and Software Defined Networks.

Andrzej Chydzinski He received his M.Sc. degree (with honours) in mathematics in 1997, and his Ph.D. and D.Sc. degrees in informatics in 2002 and 2008, respectively. In 2015 he received the Professor title from President of the Republic of Poland. Currently he is a professor at the Institute of Informatics and the head of Division of Computer Networks and Systems. His scientific interests are in computer networking, particularly in mathematical modelling and performance evaluation of computer networks, Future Internet design, queueing models, discrete event network simulators and active queue management in Internet routers. He has participated is several research projects, in four of them as the project leader. He authored and coauthored four books and about ninety journal and conference papers. His works are widely cited (about 800 times according to Google Scholar). He serves as a reviewer for several high-quality journals, such as Telecommunication Systems, Performance Evaluation, Queueing Systems, Mathematical Problems in Engineering, Applied Mathematical Modelling, Annals of Operations Research.