ABMA+ : lightweight and efficient algorithm for HTTP adaptive streaming

A.Beben[1], P.Wiśniewski[1,2], J. Mongay Batalla[1,2], P. Krawiec[1,2]

[1] Warsaw University of Technology
Nowowiejska 15/19, Warsaw, Poland

[2] National Institute of Telecommunications
Szachowa 1, Warsaw, Poland

{a.beben, p.wisniewski, jordim, p.krawiec}@tele.pw.edu.pl

ABSTRACT

Currently exploited adaptive video streaming algorithms mainly focus on maximizing video representation while they usually not assess the risk of video freezing and not care about the influence of video representation switching. We propose a new Adaptation & Buffer Management Algorithm, called ABMA+, which selects video representation based on the predicted probability of video freezing. The algorithm continuously estimates segment download time characteristics and use the pre-computed playout buffer map to select the maximum video representation which guarantee smooth content playout. Our algorithm, thanks to the buffer map, avoids heavy online computation so it could be widely deployed on different terminals. The performed simulation and trial experiments confirm the efficiency of our approach.

CCS Concepts

•**Information systems → Multimedia streaming**

Keywords

Adaptive video streaming; MPEG DASH; modelling; performance evaluation.

1. INTRODUCTION

Adaptive video streaming has been widely accepted by content providers because it promises smooth content playout by adjusting video quality to present transfer conditions. An adaptive video player consecutively requests small portions of the HTTP-compliant video (called chunks or segments), each one with the appropriate media rate (called the representation rate). The adaptation algorithm selects the representation rate of successive segments adapting them to the current load conditions on the network and server sides. The commonly used adaptation algorithms select the representation, whose rate is slightly lower than available bandwidth [1]-[3] or pick the representation accordingly to playout buffer occupancy [4]-[7]; as shown in Section 2.

Unfortunately, currently exploited adaptation algorithms may degrade quality experienced by users because they primary focus on maximizing video representation, while they do not explicitly control the video freezing probability nor consider the negative impact of frequent representation switching. Both these factors were recently identified [8] as significant for user's satisfaction. In [9],[10], the authors proposed a new class of adaptation algorithms that adapts video representation based on the estimated probability of video rebuffering. Note that the rebuffering event occurs when the playout buffer becomes drained out so the video playout is frozen until the download of next video segments is finished. This probability is calculated on-line based on a queuing model of the playout buffer using as an input the estimated segment download time characteristics. The proposed adaptation method is more effective than currently exploited adaptation algorithms but it suffers significant computational overhead. This negative feature limits deployment of the re-buffering method only to the high end terminals.

In this paper, we propose new approach for Adaptation & Buffer Management Algorithm, called ABMA+. In principle, it follows the concept of adaptation algorithms based on predicted rebuffering probability. However our algorithm exploits a pre-computed buffer map to avoid heavy on-line computations. The buffer map determines the size of the playout buffer, which ensures a given rebuffering probability. As a consequence, the ABMA+ method allows to get the benefits of the rebuffering based approach even on the low computing video clients. Moreover, as the buffer map is calculated completely off-line, it could be derived using different queueing models, e.g. more accurate than the currently proposed, or it could be even collected from comprehensive simulations. Our contribution covers: i) new adaptation algorithm exploiting pre-computed buffer map, ii) improved model of adaptive streaming system, which reflects impact of the adaptation control logic by using state dependent description of the segment arrival process, iii) the prototype implementation of ABMA+ method, iv) the framework for performance evaluation and comparison of different adaptive streaming systems, and v) the performance evaluation of the ABMA+ method in comparison to the rate- and the buffer-based approaches.

The paper is organized as follows. In section II, we present analysis of the currently investigated classes of the adaptive video streaming methods. The proposed ABMA+ adaptation method is presented in section III. The framework for evaluation of adaptive streaming methods as well as the results of the experiments focused on the performance of ABMA+ method are provided in section IV. Finally, Section V summarizes the paper and gives an outline of further works.

2. SURVEY OF ADAPTIVE ALGORITHMS

The currently investigated HTTP-compliant adaptive stream-switching algorithms adjust video representation based on the download conditions observed by the video client. These conditions describe the combined available throughput of the streaming server and the network path. They may be assessed by

MMSys'16, May 10 - 13, 2016, Klagenfurt, Austria

ACM 978-1-4503-4297-1/16/05…$15.00
DOI: http://dx.doi.org/10.1145/2910017.2910596

the following factors: the estimated download rate [1]-[3], the observed occupancy of the playout buffer [4]-[7] and the estimated distribution of video segments download times [9]-[11]. Despite being strictly correlated, these factors have significant impact on the behavior of adaptation algorithms. As a consequence, we can roughly classify stream-switching algorithms into three basic classes: rate-based, buffer-based, and time-based. Although some approaches may exploit a combination of above factors, usually one of them is dominant.

In following sub-sections we outline the main principles behind identified classes and provide theirs representative examples. For the clarity of this paper, we assume that client sequentially requests the video segments, i.e. the next segment is requested only if the previous has been received.

2.1 Rate-based adaptive algorithms

The rate-based adaptive algorithms aim to pick the representation which rate best matches the available bandwidth on the path from the streaming server to the client. The bandwidth is most commonly assessed as TCP connection throughput measured by client. These algorithms select the representation rate of the requested video segments based on the estimated download rate of the previously downloaded segments. Generally, they select the video representation rate that is the highest from the set of available representations rates $\mathfrak{R}$ and not greater than the most recently estimated download rate $\widehat{drate}_k$. Formally, we can define it as in (1):

$$R_{k+1} = \sup\{R^i \in \mathfrak{R}: R^i \leq \widehat{drate}_k\} \quad (1)$$

where k denotes the number of segment most recently received, R_{k+1} denotes the representation rate of requested segment and $\sup\{\cdot\}$ denotes the supremum. The set of representation rates $\mathfrak{R}$ is known a'priori to client, i.e., it is received by client in manifest file.

Different rate-based adaptive algorithms differ in estimation techniques, i.e., the selected $\widehat{drate}_k$ estimator. The estimator may be based on the most recently measured segment rate (instant estimator) or on a vector of recently measured segment rates (smoothed/moving estimators) [1]-[3]. The instant estimator rapidly reacts to download rate changes. This may lead to unnecessary video quality oscillations as the bandwidth of the end-to-end path varies significantly in the short-term. The smoothed/moving estimators are more resistant to bandwidth variations but they suffer from late reaction in the case of significant deteriorations of available network bandwidth (depending on smoothing factor or on the number of accommodated rate probes). This late reaction should be compensated by bigger playout buffer. In this scope, authors of [1] proposed to adjust the smoothing factor on-the-fly following the measured deviations of download rate.

The rate-based adaptive algorithms focused solely on estimated rate are sensitive to variability of the segment size (video segments naturally vary in size due to commonly used VBR coding). This variability may cause buffer depletion even when download rate is stable. Rate-based adaptive algorithms are implemented in popular applications [2].

2.2 Buffer-based adaptive algorithms

The buffer-based adaptive algorithms analyze current buffer occupancy and the rate of buffer changes to select adequate video representation. They exploit a rate map that directly determines the representation based on the current buffer occupancy [4][5], or they introduce thresholds to keep the buffer occupancy in bounded region [7][12][13]. The second group often applies control loop feedback mechanisms, like proportional-integral-derivative controller (PID), to guide the adaptation [7][12].

The baseline algorithm proposed in [5], simply called *Buffer-Based Algorithm* (BBA), assumes that: *(i)* if buffer occupancy B_k is lower than threshold T^{min}, then the lowest representation is picked; *(ii)* if it is higher than T^{max}, then the highest representation is picked; and *(iii)* if it is between these thresholds, then the representation is picked accordingly to linear function $f(B_k)$. Formally, we can define it as in (2).

$$R_{k+1} = \begin{cases} inf\{R^i \in \mathfrak{R}\} & if\ B_k \leq T^{min} \\ \lfloor f(B_k) \rfloor & if\ T^{min} < B_k \leq T^{max} \\ sup\{R^i \in \mathfrak{R}\} & if\ B_k > T^{max} \end{cases} \quad (2)$$

The $inf\{\cdot\}$ denotes the infimum, and the $\lfloor r \rfloor$ denotes the next lower video rate, i.e. $\lfloor r \rfloor = \sup\{R^i \in \mathfrak{R}: R^i \leq r\}$. The authors also proposed improved versions of BBA algorithm that use dynamic T^{min} calculation and segment map instead of rate map (BBA-1), and additionally exploited throughput in startup phase (BBA-2). The BBA-1 and BBA-2 behave better than the basic BBA but they need to know a'prori video segment sizes. Notice that this information is commonly not incorporated in manifest files.

The authors of [12] propose control loop feedback mechanism that uses buffered video time in order to guide the adaptation decisions, called *Smooth Video Adaptation Algorithm* (SVAA). The SVAA aims to pick the representation matching the estimated TCP throughput multiplied by buffer-based adjustment factor F_k, as shown in (3). The adjustment factor F_k is itself a product of 3 sub-factors: buffer size $F_k^{buff\ size}$, buffer trend $F_k^{buff\ trend}$, and video chunk size $F_k^{chunk\ size}$ adjustments. The $F_k^{buff\ size}$ and $F_k^{buff\ trend}$ are used to rectify the difference between the observed and the target (equilibrium) buffer occupancy and mimic the proportional and integral terms of PI controller, accordingly. The optional $F_k^{chunk\ size}$ sub-factor is used to compensate for the TCP's slow start phase in case of non-persistent HTTP connections.

$$\begin{cases} R^*_{k+1} = \sup\{R^i \in \mathfrak{R}: R^i \leq F_k * \widehat{drate}_k\} \\ F_k = F_k^{buff\ size} * F_k^{buff\ trend} * F_k^{chunk\ size} \end{cases} \quad (3)$$

The SVAA switches the representation down if the buffer occupancy drops below half of the target occupancy. The representation is switched up when the calculated rate R^*_{k+1} is higher than the current rate R_k for m consecutive segments. The parameter m is introduced to smooth the adaptation (in expense of responsiveness).

Although SVAA exploits both estimated download rate and buffer occupancy characteristics, we classify it as buffer-based since it intends to keep buffer occupancy in bounded region around the equilibrium point (target buffer occupancy).

2.3 Time-based adaptive algorithms

The time-based algorithms aim to pick the representation for which download times of video segments best synchronize with the segment playout time Ω. The Segment Download Time (SDT) is defined as the time that a segment takes to download (i.e. time interval from the instant of sending the request to the instant of receiving the last byte of the corresponding segment). Notice that SDT depends on both the representation rate of the segment and

the segment size. As a consequence, in order to estimate SDT for another representation, SDT is scaled by quotient of rates of the considered representations.

The authors in [11] propose to perform adaptation on the basis of the ratio between Ω and SDT of given segment k (SDT_k). They argue that this ratio should be kept between predefined thresholds. If the ratio is higher than upper threshold T^{up} than the next higher representation is selected, and if the ratio is lower than T^{down} threshold then best possible lower representation is picked. Such behavior aims at achieving conservative step-wise switch-up and aggressive switch down.

$$R_{k+1} = \begin{cases} inf\{R^i \in \Re: R^i > R_k\} & if\ \frac{\Omega}{SDT_k} > T^{up} \\ \sup\{R^i \in \Re: R^i < \frac{\Omega}{SDT_k} \times R_k\} & if\ \frac{\Omega}{SDT_k} < T^{down} \\ R_k & otherwise \end{cases} \quad (4)$$

Since the adaptation is based on instant estimator of segment download time (SDT_k), the video quality unnecessarily oscillates (the available bandwidth and video segment sizes are naturally time-varying). Therefore, the authors of [9] propose a novel approach for stream-switching adaptation based on the estimated SDT distribution and introduced analytical model of playout buffer. The SDT distribution is assumed to follow the folded normal distribution as each SDT is comprised of a number of packet download times. The proposed *Adaptation & Buffer Management Algorithm* (ABMA) aims at assuring probability of video rebuffering to sustain under given acceptable threshold while optimizing the video representation quality.

The analytical model of playout buffer is approximated based on the GI/D/1/K queuing system observed only in time instances when segments have finished their service (playout). This model is used to calculate rebuffering probabilities for available video representations. The ABMA selects the highest quality video representation which satisfies the assumed rebuffering probability threshold ε, as defined by (5).

Since the ABMA estimates SDT distribution based on a few of lastly measured probes (e.g. 50) it suffers from late reaction to changing network conditions. This estimation latency needs to be compensated by bigger playout buffer (similarly as in rate-based algorithms). Moreover, ABMA instantaneously adjust the buffer size to absorb the short-term download rate oscillations. As a consequence, the buffer size B_k, calculated after receiving k^{th} segment, ranges between the minimal buffer size needed to compensate for estimation latency C_k and the maximum allowed buffer size of M segments (see [9] for details). R_{k+1} and current buffer size computed by ABMA are formally defined in (5).

$$\begin{cases} R_{k+1} = sup\{R^i \in \Re: P_k^i(M - C_k) < \varepsilon\} \\ B_k = C_k + inf\{B^j = 1, \dots, M - C_k: P_k^{k+1}(B^j) < \varepsilon\} \end{cases} \quad (5)$$

where the $P_k^i(B^j)$ denotes the rebuffering probability for representation i (with rate R^i) calculated for a given buffer size B^j.

The key differences between ABMA and other adaptive algorithms are: *(i)* ABMA approximates segment arrival process using probability distribution values while other methods are based on a single value (e.g., rate estimator, value of current buffer occupancy, etc.), and *(ii)* ABMA uses an analytical model to estimate rebuffering probability while other methods aim at avoiding any rebuffering events by heuristics. On one hand, the SDT distribution introduced by ABMA allows better characterizing the real short-term-varying download conditions than single value but, on the other hand, it corresponds to significantly higher computation overhead.

3. ABMA+ ADAPTATION ALGORITHM

In this chapter we present proposed Adaptation & Buffer Management Algorithm (ABMA+). The main idea of ABMA+ is to predict the video playout rebuffering probabilities for the available representations and then select the maximum representation which satisfies the assumed probability threshold ε. The algorithm assesses available video representations based on the pre-computed buffer map using as an input the segment download time characteristics measured during the content download. The buffer map defines the capacity of the playout buffer which is required under a given segment download conditions to satisfy the assumed rebuffering threshold ε. Let us remark, that the buffer map approach is possible because the adaptive video streaming system state is expressed in the time domain, where the most important is the number of stored segments but not their actual size in bytes. As a consequence, the system state depends on the number of segments' arrivals observed during segment's service time. Therefore, the buffer map can be expressed in a standardised form, where the segment download time characteristics are normalised to the segment playout time, So, the ABMA+ algorithm may use just a single buffer map for different video representations.

In the following sections, we present the queuing model proposed for calculation of the buffer map and then we introduce the ABMA+ adaptation logic.

3.1 The model for calculating buffer map

In order to calculate the buffer map required by the ABMA+ adaptation algorithm, we model the adaptive video streaming system as a queuing system with K places, a single server and a finite queue storing video segments. We extend the approximate model proposed in [9],[10], by introducing the state depended description of an arrival process which accurately models the behaviour of the adaptation control logic.

In our model, the service process corresponds to the segments' playouts. Once the playout of a segment is finished, the system immediately takes the next segment from the buffer to assure smooth video playout. When the buffer is drained out, then the video playout is frozen at least until the next video segment becomes available. Note that some video players defer playout longer until the buffer reaches a assumed threshold. Most of adaptive streaming systems assume constant segment playout time although video segments have different size in bytes (especially segments belonging to different representations). Therefore, we model the segment playout by the deterministic process with the constant service time equal to the playout time, denoted by Ω.

The arrival to the system corresponds to the segment download process which is controlled by the adaptation logic. It requests new segments to fill up the playout buffer. If the buffer becomes full, the adaptation logic defers requesting new segments to avoid loses until the playout of the current segment is finished. The segment download time (SDT) depends on the segment size (in bytes), transport protocol behaviour and the available bandwidth in the network. Studies indicate that the TCP goodput [14],[15] as well as the video frame sizes [16],[17] may be described by a random variable of the log-normal distribution. As the segment download time is a quotient of the segment size and the network bandwidth, we argue that SDT could also be modelled by the log-normal distribution. The segment download process is controlled by the adaptation logic which may request

the number of segments equal to the number of free places in the system. As a consequence, the arrival process depends on the system state. So, we describe it by a set of r.v. $A^n(\Omega)$, which defines the number of segment arrivals occurring during service time Ω, where n=0,…, K, denotes the system state. The distribution of r.v. $A^n(\Omega)$ is derived following the same analysis as proposed in [10], assuming that the segment inter-arrival times are described by the log-normal distribution. So, let us consider that the probability density function of the SDT has the form (6),

$$F'(x) = \frac{1}{x\sigma\sqrt{2\pi}} e^{-\frac{(\log x-\mu)^2}{2\sigma^2}}, \quad x > 0, \sigma > 0. \tag{6}$$

It is well-known that a closed-form expression for the Laplace transform of the log-normal distribution does not exist. However, we may use one of the available approximations of this transform. For instance, in [18] the following approximation is studied:

$$f(s) \approx \frac{e^{-[W^2(se^{\mu}\sigma^2)+2W(se^{\mu}\sigma^2)]/(2\sigma^2)}}{\sqrt{1+W(se^{\mu}\sigma^2)}}, \tag{7}$$

where $W(x)$ is the Lambert function, i.e. the solution of the equation $W(x)e^{W(x)} = x$. As the average value of the log-normal distribution is (8),

$$m = e^{\mu+\frac{\sigma^2}{2}}, \tag{8}$$

we therefore obtain the steady state expressions (independent of the considered time slot) for the number of segment arrivals observed during the service time Ω as (9)

$$d_0(s) \approx \frac{e^{-\mu-\frac{\sigma^2}{2}}}{s^2}\left[\frac{e^{-[W^2(se^{\mu}\sigma^2)+2W(se^{\mu}\sigma^2)]/(2\sigma^2)}}{\sqrt{1+W(se^{\mu}\sigma^2)}} + e^{\mu+\frac{\sigma^2}{2}}s - 1\right],$$

$$d_n(s) \approx \frac{e^{-\mu-\frac{\sigma^2}{2}}}{s^2}\left[1 - \frac{e^{-\frac{[W^2(se^{\mu}\sigma^2)+2W(se^{\mu}\sigma^2)]}{2\sigma^2}}}{\sqrt{1+W(se^{\mu}\sigma^2)}}\right]^2 \times \tag{9}$$

$$\times \frac{e^{-\frac{(n-1)[W^2(se^{\mu}\sigma^2)+2W(se^{\mu}\sigma^2)]}{2\sigma^2}}}{\left(1+W(se^{\mu}\sigma^2)\right)^{\frac{n-1}{2}}}, \quad n \geq 1.$$

The r.v. D(Ω) is defined for the system with the infinite buffer, while in our case we are interested in r.v. $A^n(\Omega)$ which is defined for the finite buffer. Therefore, let us consider that as long as the number of arrived segments is lower than the number of free spaces in the system, the segment arrival process is exactly the same as in the case of infinite system. The main difference occurs when the arriving segment takes the last free space in the buffer. In the finite system no more arrivals may occur due to the deferring action taken by the adaptation logic, so the distribution of r.v. $A^n(\Omega)$ is given by (10),

$$\Pr\{A^n(\Omega) = a\} = \begin{cases} \Pr\{D(\Omega) = a\}, & a = 0, \dots, K-n-1 \\ 1 - \sum_{i=0}^{a-1} \Pr\{\Pr\{D(\Omega) = i\}\}, & a = K-n. \end{cases} \tag{10}$$

In order to derive the buffer map, we observe an adaptive streaming system just after a segment has finished its service. This approach simplifies system description because the residual service time equals zero in selected time instants. So the system state can be describe by a single r.v. N, which express the number of segments left behind just served segment. Moreover, the selected observation instances allow assessing the rebuffering probability as it directly corresponds to the events when departing segment left empty system. In the steady state conditions, the system can be described by the state equations (11).

$$\begin{cases} P^n = P^0 \times \Pr\{A^1(\Omega) = n\} + & n = 0, \dots, K-2 \\ \quad + \sum_{a=0}^{n} P^{n+1-a} \times \Pr\{A^{n+1-a}(\Omega) = a\}, & \\ \sum_{i=0}^{K-1} P^i = 1, & \end{cases} \tag{11}$$

where P^n denotes the steady state probability of n^{th} state. Note the value P^0 directly express the rebuffering probability.

We aim to derive the buffer map which is valid for different video representations, so we normalise SDT parameters to the segment playout time Ω by introducing two auxiliary variables *ov* and *cv* defined as (12)

$$\text{ov} = \frac{\Omega}{\text{E[SDT]}} - 1, \qquad cv = \frac{SD[SDT]}{E[SDT]}. \tag{12}$$

The over-rate factor (*ov*) normalizes the expected SDT value to the segment duration time Ω, while the *cv*, as a coefficient of variation, normalizes the standard deviation of SDT to the expected value of SDT. Note that the adaptation logic must kept *ov* value greater than zero otherwise the video rebuffering/freezing is inevitable regardless of the buffer size.

Then, we express parameters μ and σ of the SDT log-normal distribution by the *ov* and *cv* values as (13):

$$\mu = Log\left[\frac{\Omega}{(1+ov)\sqrt{1+cv^2}}\right], \qquad \sigma = Log\left[\sqrt{1+cv^2}\right]. \tag{13}$$

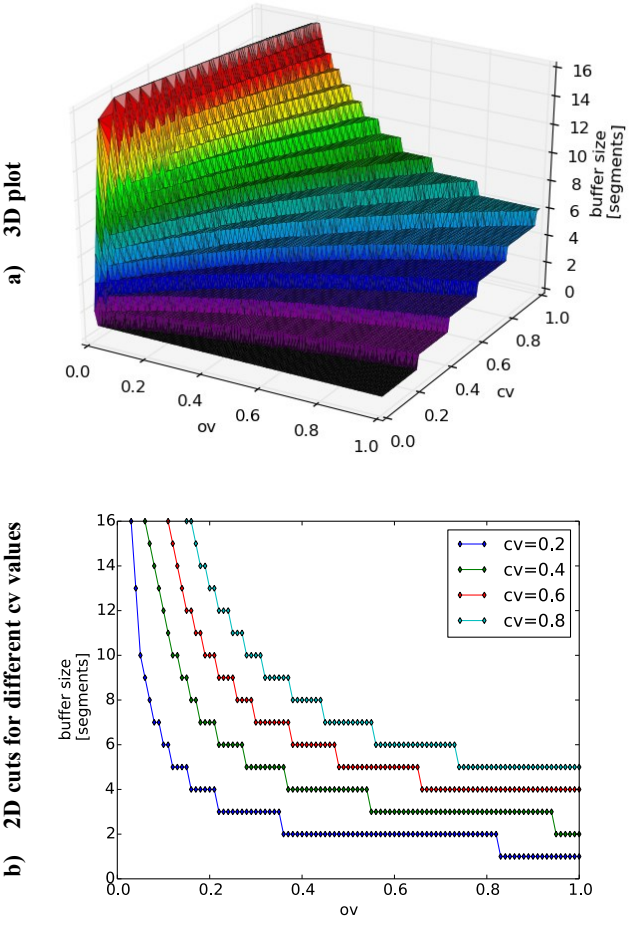

Figure 1. Exemplary buffer map for ABMA+.

Finally, we calcuate the buffer map based on proposed model. Therefore, for each pair of <ov,cv> , ov∈ (0.01; 1.0) and cv∈ (0.01; 1.0), we derive the minimum playout buffer capacity assuring that the rebuffering probabiltiy is lower than the assumed threshold ε. The rebuffering probabiltiy is expresed by the varaible P^0, calculated by solving the equation (11). In our calculations, we assume the segment playout time Ω equal to 2s and the video rebuffering threshold ε equal to 10^{-4}. Moreover, the limit the maximum buffer size to 16 segments to avoid long buffering. The exemplary buffer map calcuated by the proposed method is presented on Fig. 1, where Fig. 1a presents the 3D plot, while Fig. 1b presents 2D cuts for different *cv* values. The value zero on the buffer map, observed for low *ov* and high *cv* values, means that there is impossible to satisfy the video rebuffering threshold under a given segment arrival conditions. From the shape of the buffer map presented on Fig.1, we may observe that the required buffer capacity under low values of the over rate factor (ov) is very sensitive to the segment download time variation (cv). In the case of the moderate over rate conditions, i.e. ov above 0.3, even a small buffer can accommodate relatively large segment download time variation. These results say that adaptation algorithm should keep the system in slightly over rated conditions to avoid instabilities.

3.2 Adaptation control logic

The objective of ABMA+ is to determine the highest representation for the next segment $k+1$: $R_{k+1} \in \Re$ and the required playout buffer size: $B_{k+1}<M$, which ensure the rebuffering probability $P^0(B_{k+1})$ at the level lower than the assumed threshold ε. Formally, these conditions are expressed by (5). The value *M,* counted in segments, is the maximum allowed buffer capacity limited by the terminal memory constraints or the conditions of the service [19]. We assume that the buffer contains a dedicated reservoir of length C segments which compensates the latency in SDT estimation. Moreover we assume that the size of the reservoir does not depend on the current downloading conditions, so $C_k = C$. The estimation latency derives directly from the fact that we assess current downloading conditions based on the past observations. Note that the buffer reservoir needs to counterweight the adaptation delay required to accommodate sample of X new SDT probes. The reservoir is considered into the ABMA+ by reducing the effective maximal buffer by C segments, see Fig. 2 for details.

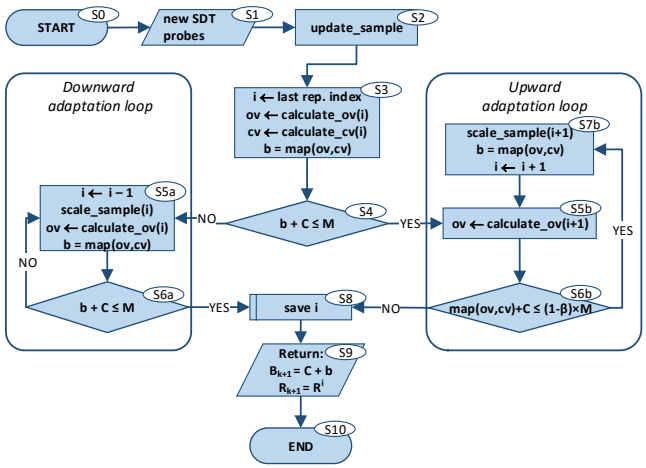

Figure 2. Flowchart of ABMA+.

The simplified flowchart of ABMA+ algorithm is presented in Fig. 2. The ABMA+ algorithm runs whenever a new segment has been downloaded and a new SDT value has been obtained: steps *S0, S1* in Fig. 2. Having updated SDT sample in *S2*, ABMA+ calculates *ov* and *cv* values and derives corresponding value *b* from buffer map, step *S3*. The <ov,cv> pair is computed for the most recently determined representation following equation (12). If the obtained buffer size of length $b+C$ (the buffer size needed to assure rebuffering probability under ε) is bigger than maximum allowed buffer size *M*, ABMA+ consecutively performs downward adaptation until the above condition is met, steps *S5a, S6a* (downward adaptation loop). Otherwise, ABMA+ successively checks if the buffer size estimated for the closest higher representation satisfies the $(1-\beta)\times M$ upper bound, and performs upward adaptation if it is so; steps *S5b, S6b, S7b* (upward adaptation loop). The $(1-\beta)\times M$ bound for upward adaptation is intended to create the anti-oscillation buffer margin of size $\beta \times M$. This margin is designed to reduce the representation swiching in the case when downloading conditions are just on the edge between two neighboring representations.

In order to estimate *ov* value for the i[th] representation, ABMA+ must calculate the first moment for this representation based on the SDT sample, as in (12). Please notice that sample, in general case, may consist of entries corresponding to different representation. As a consequence, if the current sample corresponds to representation *j* (SDT_k^i, k=1,..,*X*), then the sample for representation *i* (SDT_k^i, k=1,..,*X*) is estimated by multiplying SDT values by quotient of considered representations, i.e. $SDT_k^i = SDT_k^j \times R_k^i / R_k^j$ (k=1,..,*X*). ABMA+ performs sample scaling every time the representation is adjusted (steps S5a, S7b) resulting in whole sample corresponding to just one representation. Since the instantaneous video bitrate, for each representation, is varying throughout duration of the video content (due to variable bit rate coding), then the aforementioned sample scaling introduces an error, which may be considered negligible for a number of probes (*X*) large enough. Please note that *cv* value is independent on representation as it is itself an quotient of sample mean and standard deviations, see (12).

Finally, after determining the representation R_{k+1} and corresponding buffer size B_{k+1}, that are suit best to current downloading conditions, ABMA+ quits; steps *S8, S9, S10.*

4. PERFORMANCE EVALUATION

In this chapter we focus on the performance evaluation of the proposed ABMA+ adaptation algorithm. We aim to systematically evaluate the effectiveness of ABMA+ and compare its performance with other widely used adaptation methods, such as *(i) Rate-Based Algorithm* (RBA) which selects representation based on equation (1) and *(ii)* Buffer Based Algorithm (BBA), which selects representation based on equation (2).

In order to assure dependable comparison, we propose a new framework for performance evaluation of adaptive streaming systems (presented in section 4.1). It assumes that adaptation algorithms would be compared in the referenced test environment under the representative and repeatable traffic conditions. Moreover, we defined the set of reference performance metrics that express the effectiveness of adaptation algorithms from different criteria as: video freezing, adaptation efficiency, representation switching, etc. (see Section 4.2). Finally, in section 4.3 and 4.4 we present results obtained by the proposed framework and measured in the Internet trials.

4.1 Framework for performance evaluation

The proposed approach for analyzing performance of ABMA+ and other adaptive streaming algorithms follows the concept of the fluid flow system analysis. It assumes that the system is observed in specific time instants when its state can be strictly described by a set of variables. The evolution of the system state is defined by analytical equations that determine values of the system variables in consecutive time instants. These equations precisely model system features which have influence on the system variables. The main motivation behind using fluid flow approach is its ability to model different adaptation algorithms in an unified way. Moreover, we can easily compare effectiveness of different adaptation methods by calculating the values of performance metrics based on instantaneous values of system variables.

In our approach, we observe the adaptive video streaming system just after each video segment has been downloaded. We choose these time instants because adaptation algorithms have just updated information about downloaded segment to select the representation for the next video segments. In these time instants, we express the system state by three variables which describe: time instant (t_k) in seconds, buffer occupancy (B_k) in seconds and representation rate (R_k) in bps.

The variable t_k describes time instant, when downloading of k^{th} segment has been finished. It is defined by equation (14).

$$t_k = t_{k-1} + SDT_k + \max(B_{k-1} + \Omega - B; 0) \quad (14)$$

where SDT_k denotes download time of k[th] segment, function *max(.)* determines the duration of download deferring periods occurring when the playout buffer was full, B is the maximum buffer size (expressed in seconds), and Ω is the segment playout time. Ω is constant for a given video and it is independent from the representation rate. The segment download time SDT_k depends on the segment size in bits and the download rate as expressed by equation (15).

$$SDT_k = \frac{SegSize_k(Fadapt(.))}{\widehat{drate}_k} \quad (15)$$

The segment size depends on the video representation selected by the adaptation algorithm following *Fadapt(.)* function. This function models the behavior of adaptation algorithm, therefore it uses algorithm-specific arguments, for example download rate estimator, actual buffer occupancy or segment download time characteristics. The $\widehat{drate}_k$ denotes an average download rate experienced by k[th] segment.

The second variable B_k describes the buffer occupancy observed at the moment just after the k[th] video segment has been downloaded. B_k expressed by equation (16) defines the total playout time of video frames stored in the playout buffer.

$$B_k = \max[B_{k-1} - SDT_k - \max(B_{k-1} + \Omega - B; 0); 0] + \Omega \quad (16)$$

where the first function *max(.)* is used for modelling the buffer occupancy after re-buffering events, while the second function *max(.)* is similar as in (14). Once the buffer becomes empty, the video remains frozen until the download of current segment will be finished.

The last variable R_k describes the video representation selected for next downloaded segment, R_{k+1}. The video representation is selected by adaptation specific function (17).

$$R_{k+1} = F_{adapt}(\widehat{drate}_k,\ B_k\ , \overrightarrow{SDT}\ , \dots) \quad (17)$$

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

The equations presented above describe the evolution of the system state. These equations can be solved in a recursive way or numerically. For performance evaluation of ABMA+ and other adaptive algorithms, we design a tool which numerically evaluates equations in a segment by segment manner. This tool is easily extendable for the analysis of other algorithms due to the isolation of adaptation algorithm operation into a separate module (*AdaptationManager*). For this purpose, the analyzed adaptation algorithm should be implemented as an object that inherits functions of *AdaptationManager* class and provides required adaptation logic. These functions cover: (i) *adddata()* which provides adaptation algorithm information about current system state, downloaded segment size and rate, (ii) *adaptation()* which initiates adaptation process and (iii) *getRepresentation()* which returns representation selected by adaptation algorithm. More information about the tool and its source code is available at http://wp2.tele.pw.edu.pl/disedan/software/abma-plus.

4.2 Performance metrics

The performance metrics proposed in the literature generally try to highlight some characteristics of a given algorithm, often leaving out many other characteristics. Therefore, we try to give a complete picture of the potential metrics, expressed in an universal way, which (we believe) may be a help for future research on this field. Note that different authors present slightly different names and formulas for the metrics presented in this section. We selected the names and formulas that, in our opinion, are more general and are accepted by a larger number of research papers. Moreover, we formally defined some of the parameters, which were not defined (formally) in the literature.

The competition of resources component is analyzed by metrics as (un)fairness, (in)efficiency and (in)stability for comparing n=1,..,N clients, each one downloading k_n=1,..,K_n segments [3][20]. In the following formulas, it is considered that each segment k of client n has representation rate equal to R_{kn} and is downloaded with rate $\widehat{drate}_{kn}$. The unfairness is the unequal repartition of the total bottleneck bandwidth BW_T, which is the sum of the bandwidth allocated to each client BW_n, so that BW_T=Σ$_n$$BW_n$. In case of real network conditions, the bottleneck bandwidth is variable in time and BW_n is calculated as the mean value of the bandwidth observed by client n during the downloading of all the segments. The unfairness is calculated by the Unfair Bandwidth Repartition (UBR) parameter as indicated in (18). UBR is related with the Jain index [21]. Mueller, Lederer and Timmerer proposed to compare values of average bandwidth available to each user for an scenario of two clients and one intermediate proxy [22]. Note that the metric proposed by these authors is a reduction of formula (18).

$$UBR = \sqrt{1 - \frac{{BW_T}^2}{N \times \sum_{n=1}^{N} \sum_{kn=1}^{K_n} \widehat{drate}_{kn} / K_n}} \quad (18)$$

The efficiency indicates the responsiveness of the adaptation algorithm (within the clients) in matching the available bandwidth choosing the adequate representation. It is calculated by the Representation Selection Efficiency (RSE), which is the relation between the representations selected during the download and the minimum of two parameters: bottleneck, BW_n, and highest representation of client n, $max_j R_{jn}$, [23], see (19).

$$RSE = \sum_{n=1}^{N}\left(\frac{\frac{1}{K_n}\times\sum_{k=1}^{K_n} R_{kn}}{\min\{\max_j R_{jn}, BW_n\}}\right) \quad (19)$$

At last, instability is calculated as the probability of representation switches during the streaming (per user). This parameter is called Representation Switch Ratio (RSR) [24][25], see (20).

$$RSR = \frac{1}{N}\times\sum_{n=1}^{N}\left(\frac{1}{K_n-1}\times\sum_{k_n=2}^{K_n}\begin{cases}1, \text{if } R_{k_n}\neq R_{k_n-1}\\ 0, \text{if } R_{k_n}=R_{k_n-1}\end{cases}\right) \quad (20)$$

The second component that delineates the problem of adaptive streaming is the adaptation component. The tests analyzing this component study the behavior of one unique adaptive client in standalone scenario with delimited resources and compare it with other clients running in the same scenario. The metrics considered in the tests are based on the quality of the user's experience of Media events (QoE).

The main difficulties of calculating the QoE of video at the streaming protocol level is that the protocol only has information about the service at a level between the network and the application (note that the streaming protocols are, in general, codec-agnostic). Many studies tried to map quality of service metrics at the adaptive client level (rebuffering rate, switching rate) and the metrics of quality of media event experience at the playout [26][27]. These parameters could be then used in no-reference methods [28] to define the features of the adaptive client in given scenario.

The main idea of no-reference methods is to define the quality based on objective parameters. This is the same idea under the 3GPP TS 26.247 initiative [29], which proposes a set of parameters taken at the underlying level (TCP and adaptive streaming protocol) that could be used to optimize resource allocation and adaptation [24].

The general model of quality of experience of the Media event can be modeled as in (21), so the QoE is the average quality minus its temporal variability [28][30][31]. The temporal variability of the quality is the sum of factors that introduce changes in the playout video rate: rebuffering time duration, rebuffering rate, representation switches, etc. All these factors influence negatively into the general quality experienced by the end user. For example, rebuffering produces playout glitches and may increase the annoyance of the users [32] and representation switches causes flickers in video playout [33].

$$Quality = Mean\_Quality - Quality\_variability \quad (21)$$

The list of objective QoE parameters measured at the client level is long if we take into account the vast number of publications on this issue. The most outstanding parameters having influence in the quality of experience are described below.

RSE as specified in (19) (with N=1 client), indicates the efficiency of the algorithm.

The smoothness of the playout is measured by the rebuffering events at the client buffer during the download [25][27]. Generally, rebuffering events are monitored by observing the state of the client buffer, $B(t)$, when a new segment arrives to the client buffer or is retrieved from the client buffer. If, just after retrieving a segment from the client buffer, the buffer is empty ($B(t)$=0), then we consider a rebuffering event. The duration of the rebuffering event is calculated as the period of time between retrieval of last segment and arrival of a new segment to the client buffer. Let us remark that rebuffering event does not necessarily means that the playout suffers a "glitch", but these parameters are highly correlated. Two parameters measure smoothness: Rebuffering Event Ratio (RER), and Rebuffering Event Average Duration (RED).

RER ϵ [0,1] is calculated as the relation of the number of rebuffering events and the number of segments K of the single client analyzed, as indicated in (22).

$$RER = \frac{1}{K}\times|B(t)=0\wedge B(t^-)\neq 0| \quad (22)$$

where |function| indicates the cardinality of function.

RED is calculated as in (23):

$$RED = \frac{1}{K\times RER}\times\sum_{j=1}^{K\times RER}(t_2-t_1)\,|_j \text{ , where}$$

$$B(t_2)\neq 0\wedge B(t_2^-)=0\wedge B(t_1)=0\wedge B(t_1^-)\neq 0 \quad (23)$$

The parameters describing the (in)stability of the process are the RSR, see (20) (with N=1 client) and the Representation Switch Amplitude (RSA) [34][27][35].

RSA is the average increase/decrease gap value of representation rate during the switches $|R_j\text{-}R_{j-1}|$ of the single client analyzed, as indicated in (24).

$$RSA = \frac{1}{K\times RSR}\times\sum_{j=2}^{K}\left|R_j - R_{j-1}\right| \quad (24)$$

At last, two more parameters indicate the responsiveness speed of the clients. On the one hand, the Time Adaptation Speed (TAS) is calculated as the number of segments necessary for increasing the requested Representation from the lowest representation up to the bottleneck bandwidth [11]. TAS indicates how fast the adaptation algorithm gets the current situation of the network:

$$TAS = i\in K|\{R_i = \lfloor BW_T\rfloor\wedge R_j\neq\lfloor BW_T\rfloor, \forall j<i\} \quad (25)$$

where $R_i = \lfloor BW_T\rfloor$ is the highest representation that is lower or equal to BW_T.

On the other hand, Initial Playout Delay (IPD) (also called Join time [31]), is the time from when the adaptive client calls for the first segment and the moment when media content is retrieved from the client buffer [24]. Let us remark that IPD does not consider the time of manifest file retrieval.

4.3 The simulation experiments

Adaptive streaming methods significantly differ in the inherent mechanisms used for representation adaptation. Our objective is to evaluate and understand the main features of the proposed ABMA+ method and compare its effectiveness to other adaptive algorithms, i.e. rate-based (RBA) and buffer-based (BBA). Therefore, we analyze the behavior of adaptation algorithms under controlled and repeatable conditions using the framework proposed in Section 4.1. In simulation experiments, we follow three steps presented below:

1. Step 1: Initialize the system state. In this step, we initialize the system variables. In particular, 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{R}.

2. Step 2: Calculate the values of system variables. In this step, we calculate the values of the system variables for consecutively downloaded segments. It is repeated until the last video segment is downloaded. So, starting from segment k=1, we first calculate the SDT of segment k, SDT_k, based on the values of *segment_size* and $drate_k$ corresponding to k[th] segment which are provided as an input data for our analysis, see formula (15). Next, we calculate

the time instant when the kth segment is downloaded by applying equation (14). We use the buffer occupancy after downloading the previous segment *k-1*. At last, we calculate the buffer occupancy observed after downloading kth segment by applying equation (16) and the representation for the next segment based on equation (17). The last equation models behavior of adaptation algorithm and therefore it is specific for each class of algorithms.

3. Step 3: Calculate the values of performance metrics. In this step, we calculate the values of performance metrics based on the values of system variables derived in the step 2. In our experiments, we calculated a subset of metrics that covers RSE, RER, RED, RSR and RSA.

In our basic experiments performed by simulations or Internet trials, we consider a single video client which downloads "Big Buck Bunny" cartoon encoded in different representations ranging from 45kbps up to 15Mbps as defined in the manifest file. The segment playout duration was fixed to 2s (Ω=2s). In the video client, we set the playout buffer size to 32 segments, which allows for around 1 min of continuous video playout. This value is typically used in commercial clients. For analyzed adaptation algorithms, we use the default values of parameters that were recommended by their authors. In particular, for RBA, we used the moving average estimator with the last 50 probes. In case of BBA, we use T^{min} threshold (reservoir) of 5 segments, the defined rate map by linear function between 5 and 30 segments. For ABMA+, we assume the rebuffering probability threshold ε equal to 10^{-4}, the SDT sample size X equals 50 probes (the same value as used for the rate estimator in RBA), the buffer reservoir C equal to 5 segments (the same value as used for BBA), and the anti-oscillation factor β equal to 0.1. We evaluated performance of ABMA+, RBA and BBA algorithms using traces collected from downloading the video content by a client connected by WiFi network (the network was being shared with other users) from the server located in Klagenfurt (Austria). The maximum download rate was intentionally limited to 4 Mbps to create the bottleneck at the WiFi network. We took measurements of download rate every two seconds during more than 6 hours (HTTP direct download of a long content). Since the available measurements of video download rate in the Internet are scarce, we decided to make available our measurements for researchers. They may be found in http://wp2.tele.pw.edu.pl/disedan/software/traces. These measurements were used in our simulation framework to obtain the SDT_k, t_k and R_k values.

Fig. 3 shows the time plots of the actual buffer occupancy vs. buffer capacity (upper plot) as well as the rate of video representation selected by the analyzed adaptation algorithm vs. current download rate (lower plot) that were collected for ABMA+, RBA and BBA algorithms. In the case of the RBA algorithm (Fig. 3a), we observe the rebuffering events and significant variation of buffer occupancy (see situation about 850th second) which point out that the average rate estimator does not properly reflect the high variability of the download conditions. Moreover, we can observe the increasing trend in the buffer occupancy which suggests that the RBA algorithm selects lower video representation than it would be possible. In the case of the BBA algorithm (Fig. 3b), we observe frequent and significant representation switching that happens even in the relatively smooth download conditions (see situation after 1000th second). This effect comes from the buffer occupancy oscillations that must happen whenever download rate is not exactly the same as the representation rate. Finally, we observe on Fig. 3c that the ABMA+ algorithm avoids rebuffering events and minimizes the

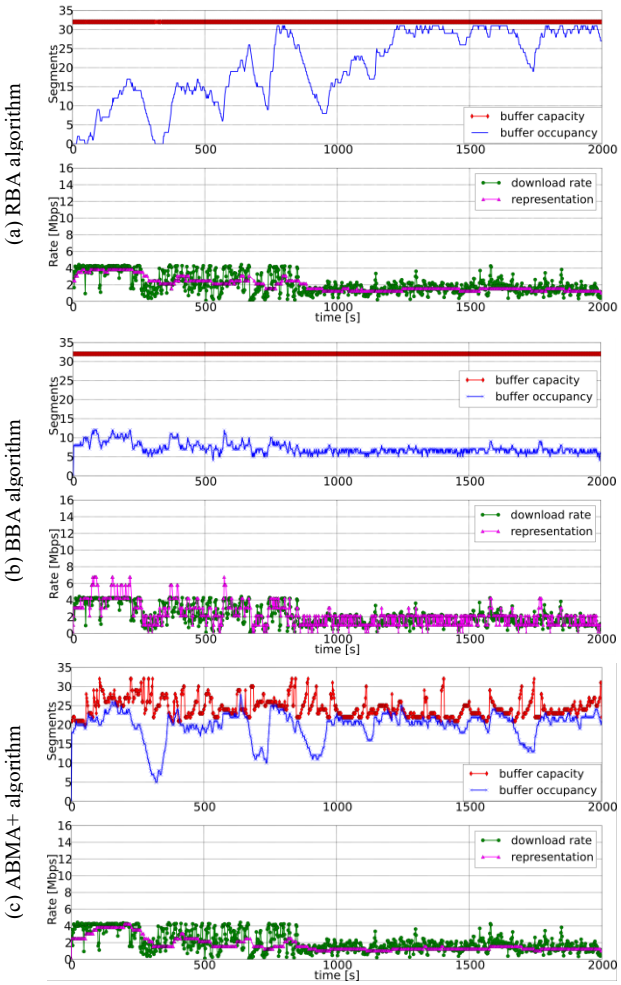

Figure 3. Comparison of adaptation algorithms in simulation experiment, where (a) RBA, (b) BBA and (c) ABMA+.

representation changes. This effect comes from the better estimation of SDT which provides much more knowledge about the segment arrival process than used by other algorithms. Moreover, the ABMA+ method adjusts buffer size to current download conditions in the range of the minimum value up to the physical buffer capacity in order to avoid unnecessary buffering delay. From the queuing theory viewpoint, the required buffer size strictly depends on *ov* and *cv* values as it is shown by the buffer map presented on Fig. 1. So, ABMA+ wants to reflect this feature.

Table 1. Performance evaluation of adaptation algorithms.

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

Table 1 presents the values of the performance metrics collected in our test. We can observe that the RBA algorithm suffers from rebuffering events, while both BBA and ABMA+ allow to

guarantee smooth content playout. On the other hand, the BBA algorithm suffers high representation switching (RSR) at 44%. So, it changes representation almost every two segments! Moreover, we can observe that ABMA+ is slightly conservative. This effect comes from the worst case assumptions exploited in proposed model, e.g. lack of correlation in the arrival process. The results for ABMA+ say that there is still a room for algorithm improvements.

In the next experiment, we have analyzed the effectiveness of ABMA+ for three videos that differ in the fast-moving scenes. We use: 1) "Of Forest And Men" documentary film, 2) "Big Buck Bunny" cartoon (the same as the previous experiment), and 3) "The Swiss Account" sport video. Similar as in the previous experiment, all videos were encoded in different representations ranging from 45kbps up to 15 Mbps, and the segment playout duration was fixed to 2s (Ω=2s). The obtained results are presented in Table 2.

Table 2. Performance of ABMA+ for different videos.

Metric	Type of video		
	Of Forest and Man (documentary)	Big Bunny (cartoon)	The Swiss Account (sport)
RSE [%]	77.54	77.36	75.69
RSR [%]	4.00	3.67	2.93
RSA [Mbps]	0.378	0.38	0.48
RER [%]	0.0	0.00	0.0
RED [s]	0.0	0.00	0.0

We can observe that adaptation efficiency becomes slightly lower, i.e. about 2%, for the sport video. This effect comes from the fast-moving scenes that makes adaptation harder. On the other hand, we can observe that the representation switching rate is slightly lower but its amplitude is slightly higher. Anyway, the impact of different videos on the ABMA+ efficiency is rather week, so we can conclude that the efficiency of ABMA+ is almost not affected by the type of the played video.

4.4 Trials over the Internet

The objective of the trials is to evaluate the performance of ABMA+ approach and compare its effectiveness with other adaptation algorithms in network environment. Specifically, we performed measurements of the three algorithms over the Internet and checked whether the aforementioned properties of the algorithms are preserved in the real network.

We implemented the ABMA+, RBA and BBA adaptation algorithms (same parameters as in the simulation test) on the top of VLC player with DASH plug-in [36]. The algorithms are provided as separate modules, that are called by the *HTTPConnectionManager* object of the DASH plug-in just after downloading each segment. The source code is available http://wp2.tele.pw.edu.pl/disedan/software/abma-plus.

The experiments were performed assuming the network shown in Fig. 4 which consists of three clients sharing the same WiFi access network and two servers with media content available through the Internet. On terminal labeled as Client#1 we launched VLC player with implemented adaptive mechanisms. This player downloaded and played "Big Buck Bunny" cartoon with constant resolution (480x360). The segment duration equaled 2 seconds

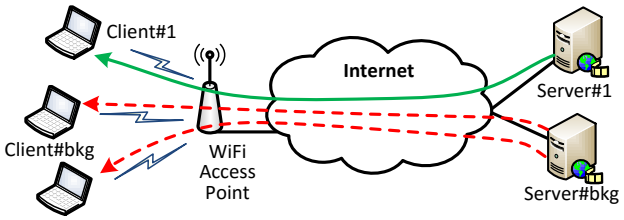

Figure 4. Network scenario for trials.

and the representations covered bitrates from 100 to 4500 Kbps. The cartoon was streamed by Server#1 located in Klagenfurt (Austria) [37]. Two other terminals (Client#bkg) run, at different moments, a number of background streaming sessions from Server#bkg situated in Warsaw (Poland), to disrupt Clinet#1's streaming process.

Time plots obtained during three one-hour experiments are presented in Fig. 5. Although the background traffic scheme (how and when the contents are claimed by the clients) was the same in all experiments, the traffic conditions from one experiment to another may slightly differ due to uncontrolled interference in wireless access network as well as varying load of paths between access point and servers

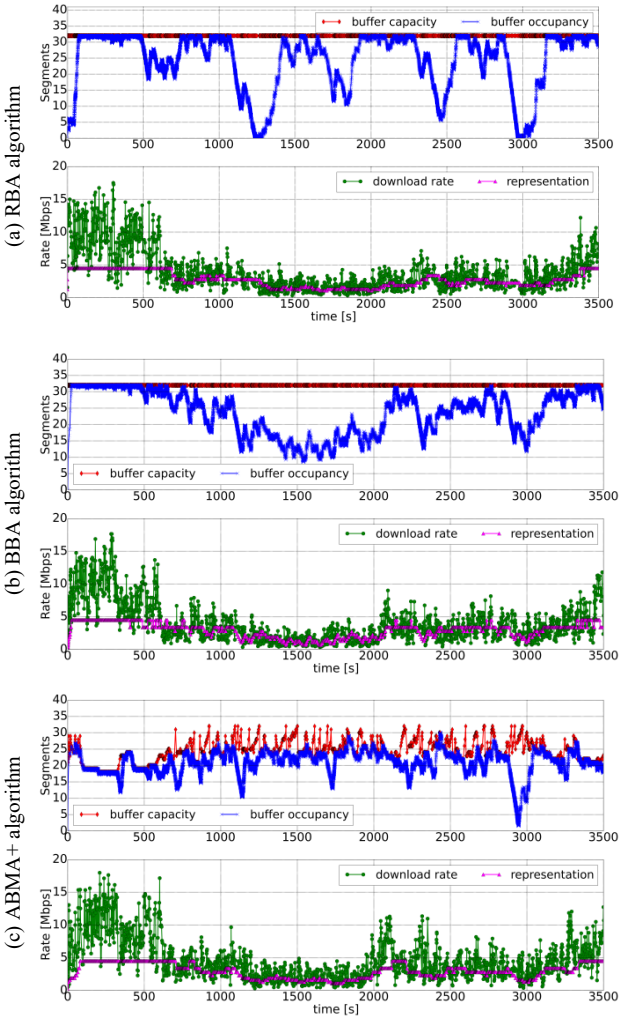

Figure 5. Comparison of adaptation algorithms in the trials over Internet, where (a) RBA, (b) BBA and (c) ABMA+.

The presented plots and the values of performance metrics shown in Table 3, confirm the conclusions derived from the simulation experiment. In particular, the RBA leads to rebuffering events due to inadequate modelling of the segment arrival process. The BBA allows for the best efficiency (highest RSE value) but it suffers from significant frequency of representation switching (the value of RSR metric is four times higher than for other algorithms).

Table 3. The values of performance metrics in Internet trials.

Metric	Adaptation algorithm		
	RBA	**BBA**	**ABMA+**
RSE [%]	90	94	86
RSR [%]	4	16	4
RSA [Mbps]	0.490	0.487	0.509
RER [%]	1.7	0.0	0.0
RED [s]	2.19	0.0	0.0

In contrary to results achieved by simulations (see Section 4.3), value of RSA metric for the BBA is on a similar level as for other adaptations. The reason can be found in the fact that a significant number of representation switches was carried out between representations of low bitrates (see Fig. 5b, period between 1100 and 2100 s), which decisively decreased RSA metric. In turn, the ABMA+ avoids rebuffering events at the cost of slightly lower efficiency. Moreover, it allows to avoid frequent representation switching which is the main drawback of the BBA approach (and, in general, the buffer-based algorithms).

5. SUMMARY & CONCLUSIONS

The paper proposes new algorithm for HTTP adaptive streaming, called ABMA+, that adapts video representation based on the estimated probability of video rebuffering. The algorithm continuously estimates segment download time characteristics and exploits pre-computed playout buffer map to select the maximum video representation, which satisfy the assumed rebuffering threshold. Thanks to exploiting the pre-computed buffer map, the ABMA+ algorithm avoids heavy on-line calculations allowing to get benefits of rebuffering based approach even on thin video clients. The paper contribution covered: i) proposal for new adaptation algorithm exploiting pre-computed buffer map, called ABMA+, ii) improved model of the adaptive streaming system used for calculating buffer maps, which reflects impact of the adaptation control logic by using state dependent description of the arrival process, iii) prototype implementation of ABMA+ method as DASH plug-in to VLC, iv) a framework for performance evaluation of different adaptive video streaming algorithms and v) performance evaluation of ABMA+ method in comparison to the rate- and buffer-based approaches.

The obtained numerical results confirmed that the ABMA+ method efficiently adjusts video representation to the variable network conditions allowing to minimize the risk of video freezing and preventing frequent representation switching. Moreover, the pre-computed buffer map eliminates heavy on-line computations making our approach feasible for widely deployment. Therefore, we believe that proposed ABMA+ method constitutes an interesting alternative for currently exploited rate- and buffer-based adaptation algorithms.

6. ACKNOWLEDGMENTS

This work is partially founded by the DISEDAN project within the European CHIST-ERA Program. We want to thank all project partners for their support and contribution to the ideas presented here.

7. REFERENCES

[1] T. C. Thang, et al. Adaptive Streaming of Audiovisual Content using MPEG DASH. IEEE Transactions on Consumer Electronics. Vol. 58, No. 1, pp. 78-85, Feb. 2012.

[2] S. Akhsabi, A. Begen and C. Dovrolis. An Experimental Evaluation of Rate-Adaptation Algorithms in Adaptive Streaming over HTTP. ACM MMSys, New York, 2011.

[3] J. Jiang, V. Sekar and H. Zhang. Improving Fairness, Efficiency, and Stability in HTTP-based Adaptive Video Streaming with FESTIVE. ACM CoNEXT, 2012.

[4] T. Huang, R. Johari and N. McKeown. Downton Abbey Without the Hippcus: Buffer-Based Rate Adaptation for HTTP Video Streaming. ACM FhMN Workshop. SIGCOMM, Hong Kong, 2013.

[5] Te-Yuan Huang et al. A Buffer-Based Approach to Rate Adaptation: Evidence from a Large Video Streaming Service. ACM SIGCOMM, 2014.

[6] Hung T. Le et al. Buffer-based Bitrate Adaptation for Adaptive HTTP Streaming. International Conference on Advanced Technologies for Communications (ATC), 2013.

[7] L. D. Cicco et al. ELASTIC: a Client-side Controller for Dynamic Adaptive Streaming over HTTP (DASH). IEEE Packet Video Workshop, San Jose, USA, 2013.

[8] M. Seufert, et al., A Survery on Quality of Experience of HTTP Adaptive Streaming. IEEE Communications Surveys & Tutorials, 2015.

[9] P. Wiśniewski, A. Beben, J. M. Batalla, P. Krawiec. On delimiting video rebuffering for stream switching adaptive applications. IEEE ICC, 2015

[10] J. M. Batalla, et al., Adaptive video streaming: rate and buffer on the track of minimum re-buffering“, Journal of Selected Areas of Communications, 2016 (accepted for publication)

[11] C. Liu, I. Bouazizi, and M. Gabbouj. Rate adaptation for adaptive http streaming. ACM MMSys, New York, 2011.

[12] G. Tian, Y. Liu. Towards agile and smooth video adaptation in dynamic HTTP streaming. ACM CoNEXT, 2012.

[13] K. Miller, et al. Adaptation Algorithm for Adaptive Streaming over HTTP, Packet Video Workshop, 2012.

[14] H. Balakrishnan, S. Seshan, M. Stemm, and R. H. Katz. Analyzing Stability in Wide-Area Network Performance. In ACM SIGMETRICS, June 1997.

[15] Y. Zhang, L. Breslau, V. Paxson, and S. Shenker. On the Characteristics and Origins of Internet flow rates. In ACM SIGCOMM, 2002

[16] K. Kim and H. A. Latchman. Statistical traffic modeling of MPEG frame size experiments and analysis. 2009.

[17] K. Salah, F. Al-Haidari, M. H. Omar, and A. Chaudhry. Statistical analysis of H.264 video frame size distribution. Communications, IET , vol. 5, no. 14, 2011

[18] S. Asmussen, J.L. Jensen, L. Rojas-Nandayapa. On the Laplace transform of the Lognormal distribution.

Methodology and Computing in Applied Probability, Springer, DOI 10.1007/s11009-014-9430-7, pp. 1-18, December (2014)

[19] A. Finamore, M. Mellia, M.M. Munafò, R. Torres and S.G. Rao, YouTube everywhere: impact of device and infrastructure synergies on user experience. ACM SIGCOMM. 2011.

[20] S. Akhshabi, et al. What Happens when HTTP Adaptive Streaming Players Compete for Bandwidth?. NOSSDA. 2012.

[21] R. Jain, A. Durresi, and G. Babic. Throughput fairness index: An explanation. The Ohio State University, Department of CIS, Columbus, OH, 1999.

[22] C. Mueller, S. Lederer and C. Timmerer. A proxy effect analysis and fair adaptation algorithm for multiple competing dynamic adaptive streaming over HTTP clients. IEEE Visual Communications and Image Processing (VCIP), 2012.

[23] L. D. Cicco, S. Mascolo and V. Palmisano. Feedback Control for Adaptive Live Video Streaming. ACM MMSys, 2011.

[24] O. Oyman and S. Singh. Quality of Experience for HTTP Adaptive Streaming Services. IEEE Communications Magazine, vol. 50, issue. 4, pp. 20-27, April 2012.

[25] C. Muller, S. Lederer and C. Timmerer. An Evaluation of Dynamic Adaptive Streaming over HTTP in Vehicular Environments. 4th Workshop on Mobile Video, 2012.

[26] M. Fiedlet, T. Hossfeld and P. Tran-Gia. A generic quantitative relationship between Quality of Experience and Quality of Service. IEEE Network, vol. 24, issue 2, 2010.

[27] C. Alberti et al. Automated QoE evaluation of Dynamic Adaptive Streaming over HTTP. Fifth Inter. Workshop on Quality of Multimedia Experience (QoMEX), 2013.

[28] X. Deng and G. Han. QoE Evaluation for HTTP Adaptive Streaming. Informational Draft: draft-deng-tsvwg-qoe-evaluation-has-00.txt, June 2014.

[29] 3GPP TS 26.247 version 11.1.0 Release 11.

[30] V. Joseph, De Veciana G. NOVA. QoE-driven Optimization of DASH-based Video Delivery in Networks. IEEE INFOCOM, Canada 2014.

[31] F. Dobrian et al. Understanding the impact of video quality on user engagement. ACM SIGCOMM 2011.

[32] P. Le Callet, S. Moller, A. Perkis. Qualinet White Paper on Definitions of Quality of Experience. European Network on Quality of Experience in Multimedia Systems and Services, Laussane, Switzerland, June 2012.

[33] P. Ni, R. Eg, A. Eichborn, C. Griwodz, P. Halvorsen. Spatial flicker effect in video scaling. International Workshop on Quality of Multimedia Experience (QoMEX), 2011.

[34] R. Mok, X. Luo, E. Chan, and R. Chang. QDASH: A QoE-aware DASH system. ACM MMSys, 2012.

[35] X. Yin, V. Sekar and B. Sinopoli. Toward a Principled Framework to Design Dynamic Adaptive Streaming. Algorithms over HTTP", 13th ACM Workshop on Hot Topics in Networks, 2014.

[36] C. Müller, C. Timmerer. A VLC Media Player Plugin enabling Dynamic Adaptive Streaming over HTTP. ACM Multimedia, 2011.

[37] Stefan Lederer, Christopher Müller and Christian Timmerer. Dynamic Adaptive Streaming over HTTP Dataset. ACM MMSys 2012.