On delimiting video rebuffering for stream-switching adaptive applications

Piotr Wiśniewski, Andrzej Beben
{p.wisniewski, a.beben}@tele.pw.edu.pl
Warsaw University of Technology
Warsaw, Poland

Jordi Mongay Batalla, Piotr Krawiec
jordim@interfree.it, p.krawiec@itl.waw.pl
National Institute of Telecommunications
Warsaw, Poland

Abstract— **Most of the current adaptation algorithms assess download rate, or trace playout buffer occupancy to adapt video representation. We propose a novel approach for switching video representation based on the estimated probability of video rebuffering. This probability is calculated using an analytical model of playout buffer and measured segment download time characteristics. The results of simulation experiments and prototype trials, carried out in a controlled network environment and over the Internet, show that our method efficiently accommodates changing network conditions.**

Index Terms— **DASH, HTTP streaming, media adaptation, queueing models, stream switching**

I. Introduction

Video streaming constitutes a significant and fast growing part of Internet traffic. As a consequence, we observe a proliferation of adaptive streaming solutions designed for optimising user experience in time-varying conditions of the Internet. Currently, the most commonly investigated and commercially applied approach for adaptation is stream-switching. It is applicable to HTTP, allows to minimize server processing power and it is video codec agnostic [1]. This approach is exploited both in closed commercial solutions like *Adobe Dynamic Streaming* , Apple's *HTTP Adaptive Live Streaming* and Microsoft's *ISS Smooth Streaming*, and in open HTTP-based protocols like DASH (Dynamic Adaptive Streaming over HTTP) [2].

The main idea behind the stream-switching adaptation is to continuously select the highest possible video representation quality that ensures smooth playout in the current downloading conditions. This selection is performed on-the-fly during video playout from a pre-defined discrete set of available video rates and with a pre-defined granularity (accordingly to video segmentation). In general, the highest quality representation should be chosen, the rate of which is lower than the currently available download rate (with tolerance corresponding to playout buffer length).

The majority of stream-switching solutions select the video representation the rate of which best matches the estimated download bandwidth, e.g. [3], [4]. However this approach is not optimal as it may lead to an overly aggressive or an overly conservative adaptation when the download rate significantly varies [5]. We argue that in order to optimise user experience, we should select the best possible representation that assures a delimited and negligible probability of video freezing during playout (caused by video rebuffering).

In this paper, we propose a novel approach for stream-switching adaptation which aims at assuring probability of video rebuffering to sustain under given negligible threshold while simultaneously optimising the video representation quality. Our Adaptation & Buffer Management Algorithm (ABMA) is an example of this approach. Based on the proposed model of the client's buffer, ABMA estimates the probability of rebuffering for the available video representations and selects the highest quality video representation which satisfies the assumed threshold. Moreover, it adjusts the buffer size to compensate for the download rate oscillations. The input data for our adaptation algorithm are the segment download time characteristics (SDT) measured during playout.

The paper structure is the following. After the analysis of stream-switching solutions (section II), we present the model of the client's buffer jointly with its extensive validation (section III). Section IV presents the details of the proposed ABMA method. We have implemented the ABMA mechanism and integrated it with VLC DASH plug-in [6]. The results of experiments performed in controlled network conditions and over the Internet are described in Section V. Finally, Section VI shortly summarizes the paper and gives an outline of further works.

II. Related work

Handling variable network conditions is one of the most challenging issues of the research on stream-switching adaptive protocols. The application detects changes of network conditions by measuring the download rate or by observing the variations of the playout buffer occupancy. The latter approach is relatively new and rarely covered in literature. Only Huang, Johari and McKeown proposed this approach for introducing a new adaptation algorithm [5].

In contrast, there are many papers available about the bandwidth estimation for video adaptation. The approaches have evolved and refined successively. In [7], Seo and Zimmermann proposed to estimate the bandwidth based on the measurements taken in a short period of time (a few segments) instead of measurements of the whole streaming in order to reduce the influence of the network variation on the representation selection. Along the same line, the authors of [8] and [9] proposed history based and machine-learning-based approaches for TCP throughput prediction.

Other studies were focused on reducing the network bandwidth variation on the client's side. For example, some changes in TCP [10] and additional stability techniques [11], [3], [4] were proposed, respectively, to avoid bursts and to increase the stability in the available bandwidth. Moreover, the use of multiple connections was studied for improving the throughput and reducing the variation of the available bandwidth in the network [12-15].

The research undertaken in the last two years focuses on the design of robust adaptive video algorithms for multiple players competing at bottleneck links [16-20]. This competition may provoke a biased feedback loop effect causing high bandwidth variations [21]. Moreover, player selecting the higher quality representation observes the higher bandwidth [16].

DASH [2] is an open standard which is gaining the principal place in the market of stream-switching protocols. The majority of DASH implementations use fixed buffer size (e.g., QTSamplePlayer, which uses libdash3.0 library [22], and VLC player with DASH plug-in [6]). The exception is dash.js [23], which sets the buffer size according to content duration and the Round Trip Delay. Nevertheless, the buffer size is not adjusted taking into account changes of download rate.

We designed ABMA to simultaneously adapt to both short and long-term network bandwidth variations. Short-term variations are diminished by adjusting the length of client buffer to current downloading conditions. At the same time, long-term network bandwidth changes are handled by switching video quality. This double adaptation is another novelty in our proposition.

III. Model of client's buffer

The content downloading process in stream-switching adaptive systems involves three interacting elements: end user's player application with stream-switching client, a content server and a network. The stream-switching client selects a video representation, requests new video segments and downloads them over TCP until there is space in the client's buffer. Simultaneously, the player application fetches data from the client's buffer into its decoding buffer at variable rate following decoding/playout process.

In order to dimension the client's buffer, two partially opposite requirements should be taken into account. On one hand long buffer inhibits correct adaptation and, on the other hand, too short buffer may provoke rebuffering during playout. Therefore, we aim to derive the minimum size of the client's buffer that will ensure that the probability of rebuffering during playout is below the assumed threshold ε.

A. The model of client's buffer

The system state can be rendered by the number of segments waiting in the client's buffer or played out with the number of segments being downloaded. The total number of segments in the system is determined by the client's buffer size increased by one segment being in service because the client requests a new segment from the server only if there is space available in the client's buffer. When the client's buffer becomes full, the content downloading process is deferred until the playout of the current segment is finished. After downloading has resumed, the next segment arrives to the client buffer at the Round Trip Delay (RTD) time. The RTD is the time elapsing between the moment of segment requesting and its receiving. Taking into account the fact that the system state does not change during deferring periods, we can model our system by the GI/D/1/K queuing system with a generic independent random arrival process, deterministic segment service time (Ω) and finite buffer space, as presented in Fig. 1.

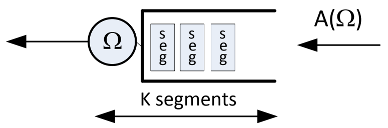


Fig. 1: The model of client buffer

We model the segment arrival process by a discrete random variable $A(\Omega)$, which describes the number of segment arrivals observed during the service time Ω. This random variable (r.v.) reflects the compound characteristics of the content download process by merging together the impact of the variable segment size, the varying segment transmission delay on the server and the fluctuating network conditions. In our model, we assume that the number of segments arriving in consecutive time slots Ω is independent. This assumption is satisfied for memoryless arrival processes, as e.g. Poissonian. For other types of processes, the number of arrivals in a given time slot Ω time is impacted by the residual Segment Download Time (SDT)[1] remaining from the previous time slot. The exact estimation of the residual time distribution seems unfeasible because r.v. $A(\Omega)$ is not known *a priori* in the system operating over the Internet. Fortunately, we may neglect the impact of the residual SDT because it would lead to underestimation of the arrival process (as long as SDT is a non-negative r.v.). As a consequence, we expect that the buffer size calculated by our model may be overestimated.

The segments stored in the client's buffer are served in constant time intervals Ω, which correspond to the segment playout time. The finite buffer space directly corresponds to the limited number of segments being in progress in the original stream-switching adaptive system.

In our approach, we observe the system immediately after a segment has finished its service (service time equal to Ω). In these time instants, the system state is expressed by the number of segments left behind just served customer. We denote the steady state probabilities of the system by $P^0, P^1, \ldots, P^K$. Note that the probability P^0 is directly related to the rebuffering events, because it corresponds to the situations when there is no segment for playing out. The probability P^K is equal to zero because we observe the system just after a

[1] SDT is the time that a segment takes to be received by the client (i.e. time interval in which the client receives all the data packets carrying a segment).

segment has finished service. Our analysis follows the imbedded Markov chain approach, where the probabilities of particular system states $P^0, P^1, \ldots, P^{K-1}$ can be formally written as the set of equations presented in (1). $Pr\{A(\Omega)=a\}$ is the distribution of r.v. $A(\Omega)$, i.e., the probability that during time Ω, exactly number a of new segments arrives to the client's buffer.

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

The first equation corresponds to the situation where no segment is left in the client's buffer just after the completion of the service of the previous segment. This happens when the buffer has been empty or has held just one segment and no segment has arrived during Ω service time. The second row describes a set of $K-2$ equations corresponding to n-th segments remaining in the client's buffer for $n=1, \ldots, K-2$. The last equation is a normalisation condition.

The solution of the equations (1) allows to calculate the rebuffering probability for a given size of the client's buffer. However, our objective is to derive the minimum size of the client buffer that will ensure that the probability of rebuffering during playout is below the assumed threshold ε. Therefore, we apply a binary search procedure to find the required buffer size.

The value of K calculated in our model corresponds to an open loop system, while the original stream-switching adaptive system requested segments from the content server arrive after RTD. Therefore, we need to increase the buffer size by the number of segments required to compensate the latency introduced by the feedback control loop between the client and server. So, the final value of the buffer size B is given by (2):

$$B = K + \left\lceil \frac{RTD}{\Omega} \right\rceil. \tag{2}$$

B. Validation experiments

In order to validate the model of the client buffer, we simulate the content downloading system under stationary, controlled network conditions and compare the obtained rebuffering probability with the value calculated by our model (P_0). In both cases, we assume the same buffer size, which was fixed to guarantee the rebuffering probability at the level below 10^{-4} threshold. Moreover, we consider essentially different distributions of SDT to evaluate accuracy of the model. Please note that it is difficult to assess the actual distribution of SDT because it depends both on the video characteristics (related to different video coding and content dynamic) and on the network and server conditions. Therefore, we use two distinct distributions: exponential distribution, which is memoryless and characterises high variation; and folded normal distribution with coefficient of variation (CV) between 1.0 and 0.5, which reflects relatively smooth content downloading conditions. Moreover, we assume segment's playout time (Ω) equal to 2s. To achieve high reliability of results, each simulation run covered 5×10^8 segments and was repeated 20 times with an independent run-up. The details of the assumed SDT characteristics and buffer size as well as the obtained rebuffering probabilities are presented in Table 1.

Table 1: Validation of buffer model under different segment download time characteristics

SDT characteristics			**Buffer size [seg.]**	**Prob. of video rebuffering [x10^{-5}]**	
Dist.	**Mean [s]**	**CV**		**Calculation**	**Simulation**
Expo-nentia	1.33	1.0	10	5.66	5.69±0.04
	1.50		14	5.64	5.62±0.04
	1.80		33	8.71	8.74±0.13
Folded Normal	1.33	1.00	9	6.54	1.05±0.03
		0.75	7	5.65	1.72±0.03
		0.50	4	3.75	0.33±0.01
	1.5	1.00	14	5.52	0.83±0.21
		0.75	10	5.86	1.56±0.07
		0.50	6	3.26	1.71±0.03
	1.8	1.00	37	8.93	0.74±0.047
		0.75	24	9.79	3.38±0.066
		0.50	14	9.89	0.75±0.03

The results show that the proposed model yields the exact results for exponentially distributed SDT. For the folded normal distribution of SDT, P_0 values obtained from simulations are lower than calculated by our model. This effect stems from the fact that we neglect the impact of residual time in segment arrival process. As a consequence, the distribution of r.v. $A(\Omega)$ is underestimated and our model overestimates the required buffer size. Anyway, this feature is acceptable as it leads to conservative buffer dimensioning rules estimating the upper limit of K. Moreover, we can observe that the buffer size increases together with the increase in the value of CV. This effect results from the fact that the system needs a larger buffer space to compensate increasing SDT variations.

C. Evaluation of buffer management

The objective of this experiment is to evaluate the relationship between the SDT characteristics and the required buffer size computed by the buffer model. Therefore, for distinct distributions of SDT (exponential and normal), we use our model to calculate the buffer size ensuring that the probability of video rebuffering does not exceed 10^{-4}. In order to understand better the impact of the mean value of SDT, we use the over-rate factor (ov_{rate}), which normalizes the mean SDT value to the segment duration time Ω as indicated in (3):

$$\mathrm{ov}_{\mathrm{rate}} = \left(\frac{\Omega}{\mathrm{mean\ SDT}} - 1\right) \tag{3}$$

Fig. 2 presents the relation between the required buffer size and the SDT characteristics expressed by the over-rate factor. The results show that a large buffer size is required when the segment download rate is similar to the video playout rate and when the variance of SDT is high (high value of CV). The buffer size can be reduced by increasing ov_{rate} factor, i.e., by switching the stream to a lower quality representation. Moreover, we can observe that the system may become unstable when the ov_{rate} factor decreases below 15%. In these conditions even small fluctuations of ov_{rate} cause significant changes in the buffer size. Therefore, we have designed an adaptation logic directed to keep the stream-switching client in the overrate state (ov_{rate} between 25-75%) to make it robust to download rate changes and uncertainty of SDT estimation.

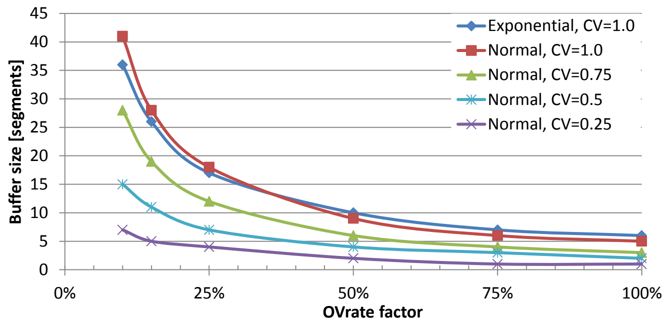


Fig. 2: Buffer size as a function of ov_{rate} factor under different values of CV

IV. Adaptation & Buffer Management Algorithm

We propose the Adaptation & Buffer Management Algorithm (ABMA) as an example to prove that adaptation based on rebuffering probability is feasible. ABMA adapts the video representation quality and buffer size to the current downloading conditions based on the queuing model presented in Section III.A. It exploits SDT and RTD characteristics measured by the DASH client. Each SDT value results from the concatenation of the downloaded segment's size with the current network and server loads. A sample of N recently measured SDT probes is used to estimate distribution of r.v. $A(\Omega)$. Such sampling introduces the estimation delay that needs to be compensated by DASH buffer in the case of a non-stationary downloading conditions (non-stationary time series of SDT values). The maximum value of this latency is N times higher than RTD (when N segments are sequentially requested and downloaded). Nevertheless, since downloading conditions rarely change rapidly in a non-stationary manner (with respect to N), usually a smaller compensation is enough. Consequently, equation (2) is superseded by eq. (4) in order to accommodate the number of segments required to compensate estimation latency.

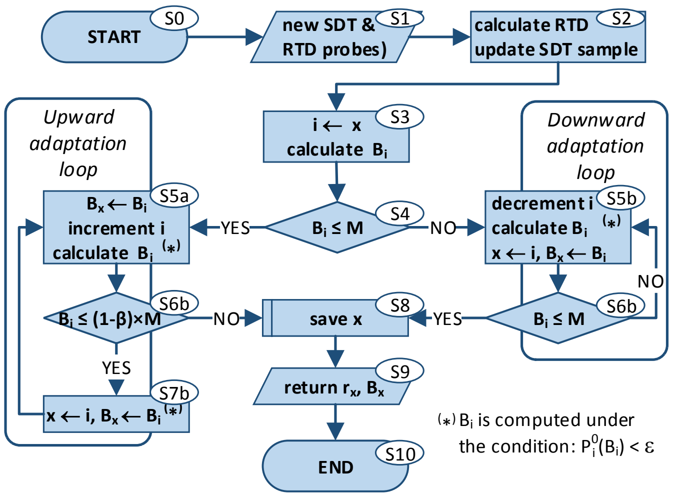
$$B = K + \left\lceil \gamma \times N \times \frac{RTD}{\Omega} \right\rceil, \qquad \gamma \in < \frac{1}{N}, \frac{N+1}{N} > \tag{4}$$

The parameter γ reflects nonstationarity of the segment arrival process (the lower and upper bounds of γ correspond respectively to the stationary and "highly" non-stationary conditions). Our preliminary studies showed that the value of 0.3 is sufficient for a wide variety of networks, including Wi-Fi IEEE 802.11 networks.

A. ABMA, assumptions and procedure

The aim of ABMA is to determine, from the set R of the available representation rates, the highest possible media representation rate r_x and the corresponding client buffer size B_x ensuring the video rebuffering probability $P_x^0(B_x)$ to be less than the given threshold ε. We assume that DASH buffer size is *a priori* limited to a user-defined value M (e.g. due to the terminal's memory capacity or the maximum acceptable adaptation delay). As a consequence, the aim of ABMA is to find the highest representation as in (5):

$$\max(r_x \in R): P_x^0(B_x) < \varepsilon \text{ and } B_x < M \tag{5}$$



START
S0
new SDT & RTD probes)
S1
calculate RTD update SDT sample
S2
i ← x calculate B_i
S3
$B_i \le M$
S4
Upward adaptation loop
$B_x \leftarrow B_i$ increment i calculate B_i (*)
S5a
YES
NO
Downward adaptation loop
decrement i calculate B_i (*) x ← i, $B_x \leftarrow B_i$
S5b
$B_i \le (1-\beta)\times M$
S6b
save x
S8
$B_i \le M$
S6b
YES
NO
x ← i, $B_x \leftarrow B_i$ (*)
S7b
return r_x, B_x
S9
END
S10
(*) B_i is computed under the condition: $P_i^0(B_i) < \varepsilon$

Fig. 3: Simplified flowchart of ABMA

Fig. 3 presents the main steps of the searching procedure. Having measured new SDT and RTD probes (step S1 in Fig. 3), ABMA updates measurement data (step S2) and calculates the buffer size B_i for the most recently determined representation (i←x), step S3. The B_i value is computed under the condition $P_i^0(B_i) < \varepsilon$ (this is described in the following paragraphs). Next in S4, ABMA checks if the determined value of the buffer size is legitimate. If not, the representation is sequentially decremented[2] until B_i gets legitimate, steps S5b, S6b. Otherwise, ABMA sequentially increments[2] the representation under the condition that the corresponding buffer size is lower than $(1-\beta) \times M$. The anti-oscillation parameter β is introduced in order to avoid representation swinging. It balances the frequency of representation switching against playout quality (its tuning is beyond the scope of this paper).

We set the size of the client's buffer according to the calculated value B_x to reduce the adaptation delay and resource usage comparing to buffer size fixed to M. ABMA determines the representation of the next segment to be requested and the current buffer size every time a new segment is downloaded from the server. At these moments, DASH management logic measures the current values of RTD and SDT. Nevertheless, if needed, the calculation may be performed more rarely to limit power consumption (this issue is not in the scope of our paper).

The calculation of the minimum buffer size B_i (steps S3, S5a/b in Fig. 3), for each representation i, is based on the estimated distribution r.v. $A_i(\Omega)$, and RTD. Since the equations in (1) and (4) allow to compute the rebuffering probability for arbitrary buffer size, ABMA exploits binary-search-like

[2] By incrementing/decrementing representation we mean selecting next representation with closest higher/lower representation rate.

algorithm in order to find its minimum value ensuring ε (i.e. $P_i^0(b) < \varepsilon$ for $b \geq B_i$, $P_i^0(b) \geq \varepsilon$ for $b < B_i$).

The RTD value is assessed by means of Exponential Weighted Moving Average (EWMA) algorithm based on the measured RTD probes (S2 in Fig. 3). The distribution of r.v. $A_i(\Omega)$, for each representation i, is estimated based on the measured SDT probes. Since each SDT probe corresponds to a representation of the downloaded segment, to estimate $A_i(\Omega)$ for other representations we scale the measured SDT values by quotient of rates of the considered representations ($sdt_j = r_j/r_i \times sdt_i$, sdt_i denotes SDT probe value representation i). Such estimation introduces negligible error as long as the set of probes is large enough to get averaging effect.

We may express the distribution $\Pr\{A(\Omega)=a\}$ by equation (6) if we assume that: 1) SDTs are described by an independent and identically distributed r.v. S, 2) $s > 0$ for $s \subset S$, and 3) the residual time from the previous Ω period is described by r.v. Y. The independency (of S) assumption is justified since SDTs are barely correlated being relatively long in comparison with the packet download times and, in addition, the segments have variable sizes and the size is not correlated between two different segments.

$$\Pr\{A(\Omega)=a\} = \begin{cases} \Pr\{Y > \Omega\}, & a = 0 \\ \int_0^{\Omega} \Pr\{Y + \underbrace{S + \cdots + S}_{a-1} = \tau\} \times \Pr\{S > \Omega - \tau\}\, d\tau, & \\ & a = 1, \dots, \infty \end{cases} \quad (6)$$

The first equation in (6) corresponds to the situation when the residual time is greater than Ω, so no segment arrives in time Ω. The following equations describe the situation when exactly a segments arrive in time Ω. Considering each Ω, the first segment arrives after the residual time defined by r.v. Y. The next a-1 segments must arrive within the current slot, while the last segment must arrive after the end of the slot. These segments arrive following r.v. S. After some algebra, we derive the simplified formula (7) for $Pr\{A(\Omega)=a\}$.

$$\Pr\{A(\Omega)=a\} = \begin{cases} 1 - F_Y(\Omega), & a = 0 \\ F_{Y+\underbrace{S+\ldots+S}_{a}}(\Omega) - F_{Y+\underbrace{S+\ldots+S}_{a+1}}(\Omega), & a = 1, \dots, \infty \end{cases} \quad (7)$$

where F_Y denotes the cumulative distribution function of Y.

In order to derive F_Y, let us consider an exemplary realization of segment arrival process presented in Fig. 4.

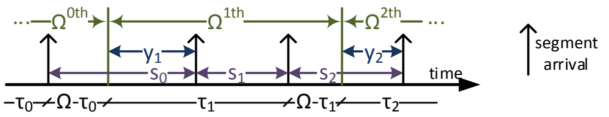


Fig. 4: Illustration of segment arrival process

Random variable Y equals y if and only if the sum of the residual time coming from the previous slot Ω and the SDTs of all segments downloaded in the current slot Ω, equal τ, whereas the SDT of the next segment will equal Ω-τ+y. Therefore, F_Y is the integral (over all possible values of τ and y) of the sum of probabilities of all events corresponding to segment arrivals starting from a single segment, up to an infinite number of segments, as presented in (8):

$$F_Y(y) = \int_0^y \int_0^{\Omega} \sum_{a=1}^{\infty} \Pr\{Y + \underbrace{S + \cdots + S}_{a-1} = \tau\} \times \Pr\{S = \Omega - \tau + t\}\, d\tau dt \quad (8)$$

In the case of r.v. S having memoryless property (i.e. exponential distribution), formulas (6) and (7) are easy to calculate. Nevertheless, since each SDT is composed of a number of packet download times, r.v. S tends to a normal distribution. As a consequence, we numerically calculate $Pr\{A(\Omega)=a\}$. Specifically, the management logic performs the following steps: 1) it computes sample mean μ and variance σ^2 of the measured SDTs[3], 2) it assumes that SDT is described by normal r.v. $X \sim N(\mu, \sigma^2)$ and generates a number[4] of random values according to S: $s_1,\dots,s_n$, 3) it calculates consecutive values of $A(\Omega)$ by counting numbers of $s_1,\dots,s_n$ falling into consecutive Ω slots, and 4) computes $Pr\{A(\Omega)=a\}$ by dividing the number of occurrences of $A(\Omega)=a$ by the total number of values of $A(\Omega)$.

V. Prototype and Experiments

We developed a prototype of ABMA to prove the feasibility of rebuffering-probability-based approach. The prototype was implemented in C++ as a piece of software that can be integrated into a variety of video applications or other solutions. Moreover we integrated it with 1) a developed simulator of stream-switching system in order to easily analyse the properties of ABMA, see section V.A, and 2) a popular VLC media player to test ABMA's behaviour in the real-life Internet environment, see section V.B. The *AdaptationManager* class implements ABMA procedure, presented in Fig. 3, and defines methods responsible for: i) addition of the measured SDT and RTD probes, ii) invocation of ABMA procedure, and finally iii) request for optimal representation rate and buffer size. The prototype's source code is publicly available on the web page: http://www.nit.eu/offer/research-projects-products/abma

For all experiments described in the following subsections we used the default setting of parameters, explicitly: target video rebuffering probability ε equal to 10^{-4}, SDT sample size N equal to 50 probes, nonstationarity factor γ equal to 0.3, and anti-oscillation factor β equal to 0.9. We performed the experimental studies to verify if ABMA: 1) selects optimal representation, 2) is able to compensate for the variability of SDT, 3) responds properly to changes of downloading conditions, and 4) prevents buffer depletion during content playout.

A. ABMA analysis in controlled simulated conditions

The objective of this simulation experiment is to analyse ABMA behaviour under a controlled network environment. Fig. 5 shows time plots of the buffer threshold calculated by ABMA vs. actual buffer occupancy (upper plot) as well as the video representation rate selected by ABMA vs. current download rate (lower plot).

[3] We compute μ and σ^2 from last 50 SDTs, since such a sample size provides good balance between results estimation stability and estimation duration.

[4] We generate 500,000 random values, as such a number corresponds to a negligible error of estimation of $Pr\{A(\Omega=a\}$.

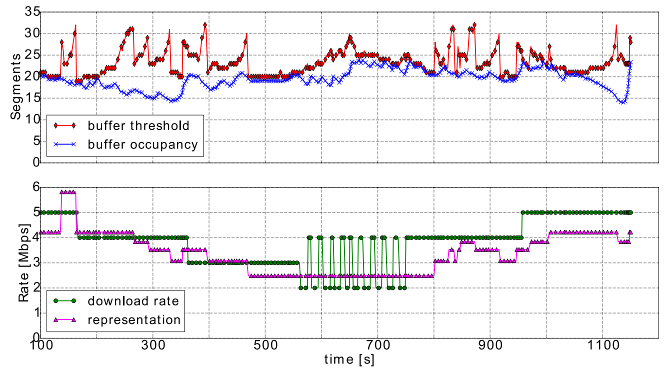


Fig. 5: Illustration of ABMA behaviour in controlled environment

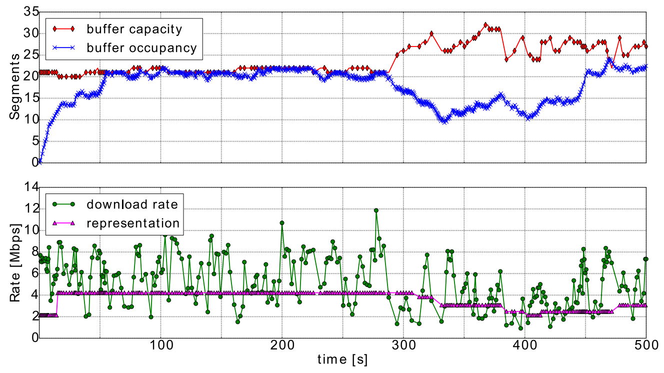


Fig. 6: Illustration of ABMA behaviour in the Internet - experiment A

We assumed that the download rate changed every 200s. At the beginning, it went down from 5Mbps until it reached 3Mbps. Then at 550s, we introduced a rate variation (alternating rate between 4 and 2 Mbps) with the average rate equal to 3Mbps. Next, the download rate started to increase until it reached again 5Mbps. In Fig. 5 we can observe that the representation selected by ABMA (solid line) follows the download rate changes (dashed line) with a slight shift to the right. This results from the latency introduced by estimation of segment arrival process. Moreover, we can see some points where ABMA selects a representation with a rate higher than the download rate (e.g. in time equal to 400s), which does not drain the client's buffer. This effect comes from the fact that ABMA exploits SDT which considers the actual size of downloaded segments which may be different from average size. Moreover, in 550th second we can observe that ABMA increased the buffer size to deal with increased download rate variation. In this case, ABMA did not change representation, because the average download rate was constant.

B. ABMA analysis in the Internet

In order to study ABMA behaviour in the real-live environment we have integrated it with the commonly-known open-source VLC media player due to its popularity and maturity. Specifically, we have incorporated *AdaptationManager* into the VLC DASH plug-in [5]. Furthermore we modified *HTTPConnectionManager* class of the plug-in to enable dynamic adaptation and buffer management. As a result, *HTTPConnectionManager* object calls *AdaptationManager* every time that a media segment is completely downloaded to determine the media representation and DASH buffer size.

We performed experiments in the uncontrolled Internet environment. The video was streamed from the server situated in Klagenfurt (Austria), through the Internet towards a client located in Warsaw (Poland). The client, running Linux Kubuntu 14.04, was connected to campus WiFi IEE 802.11g access network. Although we managed a number of terminals connected to the exploited access point, the client experienced high bandwidth oscillations due to the high density of WiFi networks in the area. The client downloaded DASH video movie "Big Buck Bunny" [24] with the 2-second segment length, in 1080p resolution and five quality levels: 2.1, 2.5, 3.1, 3.5, 3.8 and 4.2 Mbps.

In Fig. 6 and 7 we present the results from the two experiments in which the available client's bandwidth was in the range of "Big Buck Bunny" representation rates. The unique difference between both experiments is the network condition. We performed the tests twice for validating the implementation in different scenarios. In both test cases the "Big Buck Bunny" video was streamed simultaneously with two background 1080p YouTube videos that were played out on a separate terminal connected to the same access point. Then, in about 300th second in experiment A (Fig. 6) and 100th second in experiment B (Fig. 7), the competing terminal began downloading a 3rd YouTube stream. As a consequence, at this moment the available bandwidth started to deteriorate.

In experiment A, ABMA quickly determines the proper highest quality video representation and retains it until the rate degradation occurred in 300th second. During this time, we observe the high download rate oscillates between 1 and 12 Mbps. Nevertheless, ABMA keeps video rate constant and the buffer size almost steady because the available rate between 10th second and 300th second is slightly higher than the rate of the maximum video quality. In 300th second ABMA gradually starts to decrease the representation rate following the bandwidth deterioration caused by the download of the new stream. Simultaneously, the buffer size is increased to compensate higher rate variation. The occupancy of the buffer initially decreases (between 300th and 400th second) due to the latency of ABMA's rate estimation method. After the representation gets stabilised in about 410th second, the buffer occupancy increases.

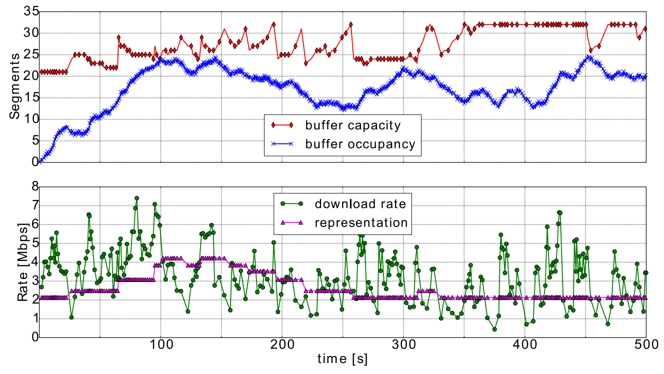


Fig. 7: Illustration of ABMA behaviour in the Internet - experiment B

In experiment B, the downloading conditions initially improve till 100th second. As a consequence, ABMA successively increases the video quality. After that moment, a new stream starts being downloaded causing the bandwidth starvation. ABMA adapts to these changes by reducing the video quality. Please note that during most of the experiment, except for the short period when the highest video representation is picked, the buffer is not fully occupied. This means that the representation rates picked by ABMA closely follow the available download rate values.

The presented experiments results show that the selected representation rate and the buffer occupancy are noticeably correlated proving that ABMA properly responds to changes in the downloading conditions. Please note that in both experiments the buffer occupancy has never reached zero, resulting in the video being played out smoothly without rebuffering. As a conclusion, we can state that ABMA properly adapts video representation and adjust the buffer size in variable, uncontrolled network conditions.

VI. Summary

The paper presents a novel approach for stream-switching adaptation that selects the video representation based on the estimated probability of playout rebuffering. The proposed Adaptation & Buffer Management Algorithm (ABMA) adjusts playout buffer size and switches the video representation in order to download the content with the highest representation rate and ensure delimited negligible probability of video rebuffering under dynamic server and network conditions. Short-term bandwidth variations are diminished by adjustment of client buffer size, whereas long-term network bandwidth changes are handled by switching video quality. ABMA method uses a derived analytical model of the playout buffer to calculate the rebuffering probability for the available video representations. The input data for our adaptation algorithm are segment download time characteristics that merge together the impact of the variable bit rate of the content, the varying server load and the fluctuating network conditions.

To the best of our knowledge, the presented proposition is the first solution for optimization of video playout quality that assures delimited rebuffering probability. Moreover, we propose simultaneous adaptation of buffer size and video quality and we present an analytical model of the stream-switching client's buffer.

The behaviour of ABMA method has been evaluated by both simulation and prototype experiments. The simulation experiments validate the client's buffer model on a wide range of network conditions, and evaluate ABMA properties under controlled network environment. The experiments with a developed prototype (implemented as a VLC plug-in) have confirmed the effectiveness of ABMA and pointed out to its slight conservative behaviour.

Our further work will focus on the enhancement of SDT estimation and on the evaluation of ABMA responsiveness in dynamically changing network conditions. Moreover, we plan to provide a detailed comparison of performance of ABMA and other adaptation approaches.

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] L De Cicco, S Mascolo, V Palmisano "Feedback control for adaptive live video streaming", ACM MMSys, 2011

[2] ISO/IEC 23009-2, "Information technology — Dynamic adaptive streaming over HTTP (DASH)". 2013

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

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

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

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

[7] W. C. B. Seo and R. Zimmermann, "Efficient video uploading from mobile devices in support of http streaming", ACM MMSys, 2012

[8] Q. He, C. Dovrolis, and M. Ammar, "On the predictability of large transfer tcp throughput", ACM SIGCOMM, 2005

[9] M. Mirza, J. Sommers, P. Barford, and X. Zhu, "A machine learning approach to tcp throughput prediction", ACM SIGMETRICS, 2007

[10] M. Ghobadi, Y. Cheng, A. Jain, and M. Mathis. Trickle: Rate Limiting YouTube Video Streaming. In Proc. USENIX ATC, 2012

[11] R. K. P. Mok, X. Luo, E. W. W. Chan, and R. K. C. Chang. "QDASH: A QoE-aware DASH system", MMSys, 2012

[12] S. Gouache, G. Bichot, A. Bsila, and C. Howson. Distributed and Adaptive HTTP Streaming. In Proc. ICME, 2011

[13] D. Havey, R. Chertov, and K. Almeroth. Receiver driven rate adaptation for wireless multimedia applications. In Proc. MMSys, 2012

[14] R. Kuschnig, et al., "Evaluation of http-based request-response streams for internet video streaming". Multimedia Systems, 2011

[15] C. Liu, I. Bouazizi, and M. Gabbouj. "Parallel Adaptive HTTP Media Streaming", ICCCN, 2011

[16] J. Jiang, V. Sekar and H. Zhang, "Improving Fairness, Efficiency, and Stability in HTTP-based Adaptive Video Streaming with FESTINE". CoNEXT'12, 2012

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

[18] S. Akhshabi, A. Begen, and C. Dovrolis, "An Experimental Evaluation of Rate Adaptation Algorithms in Adaptive Streaming over HTTP", MMSys, 2011

[19] J. Esteban, et al., "Interactions Between HTTP Adaptive Streaming and TCP", NOSSDAV, 2012

[20] R. Houdaille and S. Gouache. Shaping http adaptive streams for a better user experience. In Proc. MMSys, 2012

[21] T.-Y. Huang, et al., "Confused, Timid, and Unstable: Picking a Video Streaming Rate is Hard", IMC, 2012

[22] C. Mueller, S. Lederer, J. Poecher, and C. Timmerer, "libdash – An Open Source Software Library for the MPEG-DASH Standard", IEEE International Conference on Multimedia and Expo 2013, USA, 2013

[23] DASH Industry Forum – a reference client implementation: https://github.com/Dash-Industry-Forum/dash.js

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