VAMP: A Predictive Approach to Audio/Video Bitrate Adaptation Over Wireless Networks

Weihe Li, Jiawei Huang, *Member, IEEE,* Wenjun Lyu, Jianxin Wang, *Senior Member, IEEE,* and Paul Patras, *Senior Member, IEEE*

***Abstract*—Adaptive bitrate (ABR) algorithms are employed for delivering media content across wireless networks. Current ABR schemes only focus on video bitrate adaptation, considering that audio content encoding has negligible impact on streaming quality, due to its smaller size. However, many commercial platforms use high-quality audio which is significant in size. Improper audio bitrate selection thus leads to imbalanced audio/video buffer levels and adversely affects video bitrate selection. To tackle this issue, we propose VAMP, an approach that drives Video and Audio adaptation using Multi-step Predictions. Experimental results demonstrate VAMP's vast superiority over SOTA solutions.**

***Index Terms*—audio/video, bitrate control, wireless networks**

I. Introduction

VIDEO traffic has seen a substantial rise in recent years, owing to the growing number of mobile devices and the increasingly pervasive wireless connectivity [1]. HTTP-based adaptive video streaming (DASH) is becoming the preferred choice of most video content providers. With DASH, the source media content is pre-encoded at different qualities and broken into segments/chunks with identical duration on the server-side. The client player implements ABR algorithms to dynamically request the bitrate at which each segment should be streamed, given the estimated network bandwidth and measured buffer occupancy. To provide users with good quality of experience (QoE) over changing wireless network environments, ABR algorithms need to maintain high video/audio quality, few playback rebuffering events, and stable video/audio bitrates.

Traditionally in media streaming, the majority of audio and video services were multiplexed together on the server-side. Over the years a shift to demultiplexed streaming has been observed, whereby the audio and video chunks are delivered separately, then put together and synchronized by the client player. Compared with the muxed mode, demuxed streaming brings a series of advantages, including more flexibility in accommodating different audio codec options for the same video content, saving costs by requiring less server storage, and increasing CDN caching efficiency [2].

Current players generally construct two separate buffers for audio/video and mainly focus on video rate adaptation, without considering audio bitrate selection. The general perception about audio content is that this is transmitted at a bitrate that is much lower than the video content bitrate. Consequently, the choice of audio bitrate should not impact on video rate adaption and quality. However, on mobile devices the audio bitrate is often as high as 384Kbps, which is significantly higher than the bitrates of low-quality video on commercial platforms such as YouTube [2]. In addition, by aiming to offer immersive listening experiences, more and more mobile devices support high-quality audio features, including stereophonic and surround (Dolby Atmos) sound [2]. Thus, blindly downloading audio content may require an important network bandwidth share and lead to unsuitable bitrate selections, resulting in imbalanced buffer levels, frequent re-buffering events, and consequently modest audio/video streaming quality.

To tackle this problem, in this letter we propose VAMP (Video and Audio adaptation using Multi-step Predictions), a novel ABR algorithm based on predictive control, which ensures synchronized download of audio/video chunks, circumventing undesirable playback stalls and bitrate fluctuations when streaming over wireless networks. In particular, VAMP utilizes an online bandwidth predictor to forecast the evolution of each buffer's occupancy and seeks to minimize an objective function of the A/V bitrates and buffer levels, over multiple future steps. With the coordination of video and audio bitrate selection, VAMP optimizes the overall QoE, reducing buffer imbalance by 52%–77% and buffering times by up to 99%.

II. Why Existing A/V Rate Adaptation Solutions are Inappropriate for Wireless Streaming

We start with a brief field analysis showing audio quality in media streaming is increasing and thus audio content can be larger in size than the accompanying video; we then demonstrate that existing ABR algorithms do not synchronize the download of audio and video content, increasing the risk of playback freezing and thus jeopardizing the viewing experience, which in turn motivates our design.

A. Audio/Video Bitrate Analysis

We select different types of videos available on YouTube to investigate the bitrate settings of video and audio contents. We find that YouTube commonly encodes video and audio contents into 6 and 3 discrete bitrate levels, respectively. Table I shows the declared encoding bitrates of audio tracks and the three lowest resolution video track options (144p, 240p, and 360p) for the videos selected. Note that such bitrate ladders are not only specific to a particular content, but widely applicable [2]. Since the audio content on YouTube is generally encoded with a constant bitrate (CBR), it has the

Weihe Li and Paul Patras are with the School of Informatics, the University of Edinburgh, United Kingdom. E-mail: {weihe.li,paul.patras}@ed.ac.uk.

Wenjun Lyu is with the Department of Computer Science, Rutgers University, USA. E-mail: wenjun.lyu@rutgers.edu.

Jiawei Huang and Jianxin Wang are with the School of Computer Science and Engineering, Central South University, China. E-mail: {jiaweihuang, jxwang}@csu.edu.cn.

This work is supported by the National Natural Science Foundation of China (62132022, 61872387), Natural Science Foundation of Hunan Province, China (2021JJ30867).

Manuscript received xx xx, xxxx; revised xx xx, xxxx.

TABLE I: Example of audio and video bitrate settings on YouTube for 4 different videos.

Type	Science	Interview	Cartoon	Drama
Audio Bitrate (Kbps)	131	131	131	131
	195	195	195	195
	387	387	387	387
Video Bitrate (Kbps)	111	111	111	111
	245	246	244	245
	352	332	351	318

same average bitrate irrespective of content type, while video is encoded with variable bitrates (VBR), leading to different average bitrates across diverse videos.

Observe in Table I that the bitrate of the audio track with the lowest quality is 18% higher than that of a low-quality video track, while the highest audio bitrate is 9.9%-21.7% higher than that of video of highest quality in our dataset. This indicates that **the size of audio tracks is not significantly smaller than that of video tracks, but quite the opposite.**

In addition, to capture sound details accurately for a richer and more immersive experience, a series of high-quality audio features, such as stereo and Dobly Atmos, have been introduced. With these features, the audio bitrate can reach 1,481Kbps [3]. To compare the average bitrate of stereo audio and video, we take the Big Bunk Bunny (BBB) video with the highest resolution of 1080p and encode it into 6 discrete bitrates, splitting them into 2-second segments. The audio is also encoded into 6 levels, each with a duration of 2 seconds. The lowest three bitrates of the audio tracks follow the YouTube settings, while the other three stereo tracks are configured according to [3]. Table II reveals that the bitrate of stereo audio is only 32.08%-46.75% smaller than the video bitrate. Therefore, the download of audio chunks will also demand a considerable portion of the available network capacity, which questions the widely adopted assumption that audio downloads have a negligible impact on video streaming.

TABLE II: Comparison of stereo audio and video birates levels applied to the Big Buck Bunny video.

Bitrate Level	1	2	3	4	5	6
Audio Bitrate (Kbps)	131	195	387	525	713	1481
Video Bitrate (Kbps)	127	279	575	773	1339	2506

B. Improper Audio Bitrate Adaptation causes Playback Stalls

With DASH, client players frequently utilize the same ABR strategy for video and audio, and download the corresponding chunks without coordination. As audio content grows in size, improper bitrate decisions will lead to imbalanced audio and video buffer levels, resulting in frequent playback stalls [2]. To illustrate this problem, we conduct trace-driven streaming emulations with the state-of-the-art BOLA (buffer-based) [4] and RobustMPC (hybrid) [5] ABR algorithms (details of their operation, and video and network settings given in Sec. IV), recording the buffer dynamics over 200 seconds streaming across one randomly chosen network capacity trace. As shown in Fig. 1, both ABR algorithms exhibit severely imbalanced buffer levels, which leads to playback interruptions (as indicated by the shaded bar). BOLA selects video/audio bitrates only based on current V/A buffer occupancy. RobustMPC aims to maximize the QoE over the next few steps informed by current buffer occupancy and predicted capacity. However, the V/A capacity estimation (the predicted bandwidth for downloading the next video/audio chunk) is respectively based on the harmonic mean of the capacity for the past five V/A chunk downloads, which usually introduces large errors and thus leads to inaccurate bitrate selections [6]. Moreover, RobustMPC optimizes QoE without coordination, resulting in frequent imbalance between audio and video buffer levels.

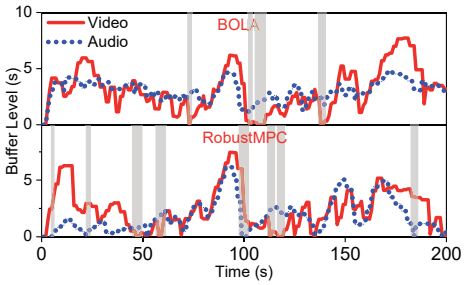


Fig. 1: Evolution of V/A buffer occupancy with BOLA (up) and RobustMPC (down) over a 200-second playback. Shaded areas indicate buffer underrun events, hence playback stalls.

A straightforward way to tackle this issue would be to directly download the video/audio chunks with synchronization. However, current players like `dash.js` inherently download audio and video with relatively coarse granularity. Hence achieving optimally synchronized video/audio downloads would involve significant modifications. Moreover, handling video/audio segments jointly can decrease CDN cache hit ratio [2] and thus harm the efficacy of content distribution. Thus, these results motivate us to design a new ABR algorithm that coordinates the download of audio and video chunks to overcome the problems observed, as we explain next.

III. VAMP Design

A. Problem Formulation

1) Buffer Dynamics Model: We employ an approach similar to [6] to predict buffer occupancy, with the key observation that MSPC [6] does not coordinate the evolution of audio and video buffers, which leads to unnecessary downloads of A/V content and thus increases the risk of playback stalls, as our results in Sec. IV reveal. Instead, the solution we propose integrates audio and video rate adaptation, breaking the asynchronous download of A/V chunks, thereby significantly reducing playback interruptions and obtaining superior QoE.

The video/audio buffer occupancies describes the length of V/A playback (usually in seconds) stored by a player. Buffer occupancy depletes as the viewer watches content and fills with D seconds of video/audio when a chunk download completes. Since video/audio chunks are downloaded without synchronization, we use k and respectively m to denote indexes of chunks to be downloaded. Buffer occupancies evolve as

$$b_v(k+1) = b_v(k) - \frac{r_v(k)D}{c_v(k)} + D, \tag{1}$$

$$b_a(m+1) = b_a(m) - \frac{r_a(m)D}{c_a(m)} + D, \tag{2}$$

where $b_v(k) \in [D, B_{vmax}]$, $b_a(m) \in [D, B_{amax}]$. Note that when one of the buffers underruns, playback interruptions will occur, even if there exists content in the other buffer. $c_v(k)$ and

$c_a(m)$ denote the average banwidth when downloading k-th video chunk and m-th audio chunk, respectively.

To find the relationship between video/audio bitrate switches and the video/audio buffer occupancies, we look at $b_v(k+1)-b_v(k)$ and $b_a(m+1)-b_a(m)$. According to [5], [6], variations of wireless network bandwidth can be regarded as piecewise stationary, which means the network conditions (and thus the average bandwidth when downloading the k-th video chunk and m-th audio chunk) do not change dramatically during short time frames (several seconds), i.e. $c_v(k) \approx c_v(k-1)$, $c_a(m) \approx c_a(m-1)$. This allows us to express buffer occupancies as

$$b_v(k+1) \approx 2b_v(k) - b_v(k-1) - \frac{\Delta r_v(k) D}{c_v(k)}, \tag{3}$$

$$b_a(m+1) \approx 2b_a(m) - b_a(m-1) - \frac{\Delta r_a(m) D}{c_a(m)}, \tag{4}$$

where $\Delta r_v(k)$ and $\Delta r_a(m)$ indicate the bitrate variation between two consecutive video/audio chunks.

2) Optimization Goal: To guarantee a good QoE, the ABR algorithm should achieve a balance between potentially conflicting goals, including maximizing the video/audio quality, minimizing the number of rebuffering events, and reducing video/audio quality switching.

Video/audio segments with a higher quality encompass larger chunk sizes than those in the low-rungs, and thus downloading a higher quality video/audio is equivalent to depleting more buffer contents of lower quality. To mitigate the risk of playback interruptions, the download time of current V/A chunk should be smaller than the respective buffer occupancy [7]. Thus, maintaining high V/A quality and reducing the rebuffering time can be tackled jointly by pushing the V/A buffer occupancy, $b_v(k+1)$ and $b_a(m+1)$, to target levels B_v and B_a, respectively. To avoid imbalanced buffer levels, the difference between B_v and B_a should be as small as possible. To mitigate V/A quality fluctuations, the ABR algorithm should also minimize the video/audio bitrate variations, $\Delta r_v(k)$ and $\Delta r_a(m)$. With these considerations in mind, the ABR design needs to minimize the following objective function over N chunks:

$$\begin{aligned}\mathcal{F} := \mathbb{E}\left[\sum_{k=1}^{N}\left(b_v(k+1)-B_v\right)^2 + \sum_{k=1}^{N}\left(\eta_k \Delta r_v(k)\right)^2\right] + \\ + \mathbb{E}\left[\sum_{m=1}^{N}\left(b_a(m+1)-B_a\right)^2 + \sum_{m=1}^{N}\left(\mu_m \Delta r_a(m)\right)^2\right],\end{aligned} \tag{5}$$

where $\mathcal{F}$ is related to variables $\Delta r_v(k)$ and $\Delta r_a(m)$; $\mathbb{E}[\cdot]$ denotes the statistical expectation; η_k and μ_m reflect the penalty of bitrate switching on k-th video chunk and m-th audio chunk, respectively, and can be set according to a user's watching preferences. For instance, if a user is sensitive to bitrate switch, large η and μ can be adopted, meaning that more efforts will be made to obtain playback smoothness. On the contrary, η and μ can be set to relatively small values if a user is less concerned about bitrate oscillations. Since imbalanced buffer levels for audio and video are undesirable, we treat the audio and video terms with the same weight to constrain their buffer occupancy difference to a small range.

If information about the available network bandwidth can be obtained in advance, the optimal bitrates for each video/audio chunk can be determined by minimizing $\mathcal{F}$. Even though forecasting bandwidth is prone to errors, we can make reasonably good predictions over short horizons due to the piecewise stationary property mentioned above. Thus, bitrate selection for video/audio chunks can be modeled as a predictive control problem over future S steps. Besides, to avoid large bitrate oscillations, we control the bitrate occupanices $b_v(k+1)$ and $b_a(m+1)$ to reach the target levels B_v and B_a along trajectories $b_{vs}(k+1) = \alpha b_v(k) + (1-\alpha)B_v$ and $b_{as}(m+1) = \beta b_a(m) + (1-\beta)B_a$, where $\alpha \in [0,1), \beta \in [0,1)$. The optimization problem to solve can be thus written as:

$$\begin{aligned}\min_{r_v, r_a} F := \mathbb{E}\left[\sum_{i=1}^{S}\left[\left(\hat{b}_v(k+i|k) - b_{vs}(k+i)\right)^2 + \left(\eta_i \Delta r_v(k+i-1)\right)^2\right]\right] \\ + \mathbb{E}\left[\sum_{i=1}^{S}\left[\left(\hat{b}_a(m+i|m) - b_{as}(m+i)\right)^2 + \left(\mu_i \Delta r_a(m+i-1)\right)^2\right]\right],\end{aligned} \tag{6}$$

$$\text{s.t.} \begin{cases} b_{vs}(k+1) = \alpha b_v(k+1) + (1-\alpha)B_v, \\ b_{as}(m+1) = \beta b_a(m+1) + (1-\beta)B_a, \\ \eta_i = \eta(S-i+1), \mu_i = \mu(S-i+1), \\ r_v(k) \in \{1, ..., L\}, r_a(m) \in \{1, ..., G\}, \\ b_v(1) = D, b_a(1) = D, b_v(k) \in [L, B_{vmax}], \\ b_a(m) \in [L, B_{amax}], \alpha \in [0,1), \beta \in [0,1). \end{cases}$$

$\hat{b}_v(k+i|k)$ and $\hat{b}_a(m+i|m)$ are the i-th step video/audio buffer occupancy predictions. η_i and μ_i reflect the bitrate switching penalty at the i-th future step. Given that switching the video/audio bitrate in the near future has more influence on current QoE than a bitrate switch later on, η_i and μ_i are individually configured as $\eta(S-i+1)$ and $\mu(S-i+1)$, thus decreasing as the prediction horizon increases. $b_v(1) = D$ and $b_a(1) = D$ are the initial conditions when the video/audio download starts. We do not reset the buffer levels to D during each step of the optimization process.

B. Bitrate Selection

To make A/V bitrate selection decisions, VAMP firstly needs to predict the evolution of buffers' occupancy. The benefit of buffer occupancy prediction is that it indicates the relationship between the selected bitrate and the available bandwidth, while it correlates with the QoE metrics (playback bitrate and freezing time). Previous ABR algorithms that predict bandwidth directly mainly lack control of buffer occupancy, which makes the player's buffer levels fluctuate vastly and risks having insufficient buffer contents to counteract network outage/congestion, thereby increasing rebuffering times and impairing viewing experience [11]. VAMP maintains the audio/video buffer within a target range through buffer occupancy prediction, which guarantees players have enough buffered content when faced with network variations. In addition, we update the prediction value at each step after selecting the video/audio chunk bitrate, to preserve accuracy.

The buffer predictions rely on the anticipated available bandwidth. Since the primary purpose of VAMP is not to enhance capacity prediction accuracy, we adopt a widely-used Kalman Filter-based capacity predictor to estimate future available bandwidth [6]. Then we leverage generalized predictive control (GPC) theory to find the extrema of the optimization model. GPC is robust to prediction errors, adapts anto environment changes [8], and can be easily implemented.

1) Buffer Occupancy Prediction: By (3)–(4), buffer occupancy in the first and i-th next steps can be expressed as:

$$\begin{cases} \hat{b}_v(k+1|k) = 2b_v(k) - b_v(k-1) - \frac{\Delta r_v(k)D}{c_v(k)}, \\ \hat{b}_v(k+i|k) = 2\hat{b}_v(k+i-1|k) - \\ \qquad -\hat{b}_v(k+i-2|k) - \frac{\Delta r_v(k+i-1)D}{c_v(k+i-1)}; \end{cases} \tag{7}$$

$$\begin{cases} \hat{b}_a(m+1|m) = 2b_a(m) - b_a(m-1) - \frac{\Delta r_a(m)D}{c_a(m)}, \\ \hat{b}_a(m+i|m) = 2\hat{b}_a(m+i-1|m) - \\ \qquad -\hat{b}_a(m+i-2|m) - \frac{\Delta r_a(m+i-1)D}{c_a(m+i-1)}. \end{cases} \tag{8}$$

2) Solution: For ease of explanation, we first define a set of vectors as follows:

$$\begin{cases} \hat{\mathbf{B}}_v = \left[\hat{b}_v(k+1), \hat{b}_v(k+2), ..., \hat{b}_v(k+S)\right]^T, \\ \hat{\mathbf{B}}_a = \left[\hat{b}_a(m+1), \hat{b}_a(m+2), ..., \hat{b}_a(m+S)\right]^T, \\ \mathbf{\Delta R}_v = [\Delta r_v(k), \Delta r_v(k+1), ..., \Delta r_v(k+S-1)]^T, \\ \mathbf{\Delta R}_a = [\Delta r_a(m), \Delta r_a(m+1), ..., \Delta r_a(m+S-1)]^T, \\ \mathbf{B}_v(k) = [b_v(k), b_v(k-1)]^T; \mathbf{B}_a(m) = [b_a(m), b_a(m-1)]^T, \\ \mathbf{B}_{vs} = [b_{vs}(k+1), b_{vs}(k+2), ..., b_{vs}(k+S)]^T, \\ \mathbf{B}_{as} = [b_{as}(m+1), b_{as}(m+2), ..., b_{as}(m+S)]^T, \\ \mathbf{\Lambda} = diag(\eta_1, \eta_2, ..., \eta_S); \mathbf{\Omega} = diag(\mu_1, \mu_2, ..., \mu_S). \end{cases}$$

The objective function in (6) can now be expressed as:

$$F = \mathbb{E}\left[\left[\hat{\mathbf{B}}_v - \mathbf{B}_{vs}\right]^T \left[\hat{\mathbf{B}}_v - \mathbf{B}_{vs}\right] + \mathbf{\Delta R}_v \mathbf{\Lambda} \mathbf{\Delta R}_v\right] + \\ + \mathbb{E}\left[\left[\hat{\mathbf{B}}_a - \mathbf{B}_{as}\right]^T \left[\hat{\mathbf{B}}_a - \mathbf{B}_{as}\right] + \mathbf{\Delta R}_a \mathbf{\Omega} \mathbf{\Delta R}_a\right]. \tag{9}$$

By employing (7) and (8) iteratively, the buffer occupancy predictions $\hat{b}_v(k+i|k)$ and $\hat{b}_a(m+i|m)$ can be obtain by the occupancies $\{b_v(k), b_v(k-1)\}$ and $\{b_a(m), b_a(m-1)\}$, along with the bitrate switch sequences $\{\Delta r_v(k), ..., \Delta r_v(k+i-1)\}$ and $\{\Delta r_a(m), ..., \Delta r_a(m+i-1)\}$ as follows:

$$\begin{cases} \hat{\mathbf{B}}_v = \mathbf{V}\mathbf{B}_v(k) + \mathbf{M}_v \mathbf{\Delta R}_v, \\ \hat{\mathbf{B}}_a = \mathbf{V}\mathbf{B}_a(m) + \mathbf{M}_a \mathbf{\Delta R}_a, \end{cases} \tag{10}$$

where

$$\mathbf{V}^T = \begin{bmatrix} 2 & 3 & \cdots & S+1 \\ -1 & -2 & \cdots & -S \end{bmatrix}; \mathbf{M}_v = \begin{bmatrix} -\frac{D}{\hat{c}_v(k)} & \cdots & 0 \\ -\frac{2D}{\hat{c}_v(k)} & \cdots & 0 \\ \vdots & \ddots & \vdots \\ -\frac{SD}{\hat{c}_v(k)} & \cdots & -\frac{D}{\hat{c}_v(k+S-1)} \end{bmatrix};$$

$$\mathbf{M}_a = \begin{bmatrix} -\frac{D}{\hat{c}_a(m)} & \cdots & 0 \\ -\frac{2D}{\hat{c}_a(m)} & \cdots & 0 \\ \vdots & \ddots & \vdots \\ -\frac{SD}{\hat{c}_a(m)} & \cdots & -\frac{D}{\hat{c}_a(m+S-1)} \end{bmatrix}.$$

From (10), we observe that the video/audio buffer occupancy predictions consist of two parts. The first part is related to current video/audio buffer lengths, and the second part depends on the future video/audio bitrate selections. Substituting (10) into (9), we see that the objective function F is only related to $\mathbf{\Delta R}_v$, $\mathbf{\Delta R}_a$, and other variables are constants. Therefore, the whole function F can be viewed as a quadratic equation of $\mathbf{\Delta R}_v$, $\mathbf{\Delta R}_a$, and the second derivative of F is greater than 0, indicating F is a convex function, and a global optimum exists. To minimize F, we need to solve $\frac{\partial F}{\partial \mathbf{\Delta R}_v} = 0$ and $\frac{\partial F}{\partial \mathbf{\Delta R}_a} = 0$, which yields:

$$\mathbf{\Delta R}_v = \left(\mathbf{M}_v^T \mathbf{M}_v + \mathbf{\Lambda}\right)^{-1} \mathbf{M}_v^T \left[\mathbf{B}_{vs} - \mathbf{V}\mathbf{B}_v(k)\right], \tag{11}$$

$$\mathbf{\Delta R}_a = \left(\mathbf{M_a}^T \mathbf{M}_a + \mathbf{\Omega}\right)^{-1} \mathbf{M}_a^T \left[\mathbf{B}_{as} - \mathbf{V}\mathbf{B}_a(m)\right]. \tag{12}$$

The first terms of $\mathbf{\Delta R}_v$ and $\mathbf{\Delta R}_a$ represent the bitrate variation at the k-th video chunk and respectively m-th audio chunk. Since the video/audio bitrates are discrete, the actual requested k-th video chunk and m-th audio chunk should be at bitrates that are $r_v(k-1) + \Delta r_v(k)$ and respectively $r_a(m-1) + \Delta r_a(m)$ less than the highest values.

In addition, when the download of each audio/video chunk finishes, the audio (video) buffer will monitor the occupancy of the current video (audio) buffer. To avoid exceptional cases where video/audio buffer occupancies are severely imbalanced (the difference between them exceeds the threshold Q), VAMP will pause the download of the following video or audio chunk guided by the amount of buffered contents. When the difference between the two buffers is less than the threshold, the following audio or video download process will be resumed.

IV. Evaluation

To evaluate VAMP and compare its performance against that of existing ABR algorithms, we employ a virtual player that faithfully simulates the server to client streaming process. The maximum video/audio buffers sizes are both set as 60 seconds [14]. We use BBB as the test video [3] with the same video/audio bitrate settings as in the analysis in Sec. II-A. To reproduce a realistic wireless network scenario, we use link bandwidth traces from an HSDPA dataset [9].

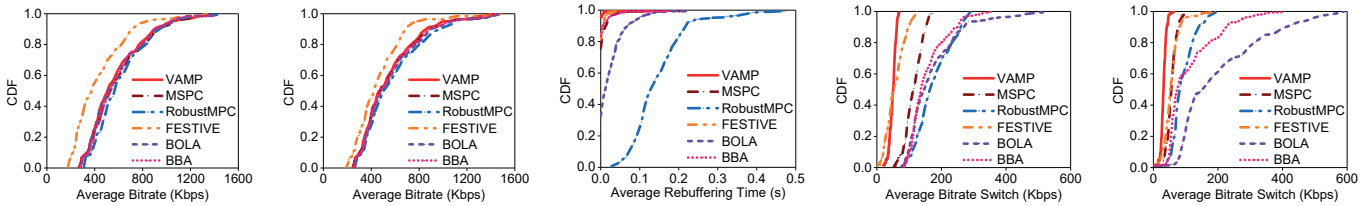
QoE Metric. The linear combination of three indicators (average bitrate, rebuffering time, and bitrate switching) is widely-used for QoE evaluation, since viewers can directly perceive these metrics while watching video content [5]. Give that buffer occupancy and buffer imbalance cannot impact QoE as long as the buffer of audio or video has not been exhausted, we do not incorporate these in the QoE model. As such, we modify the QoE metric also used previously by MPC [5], i.e.,

$$QoE = \sum_{k=1}^{N} r_v(k) + \sum_{m=1}^{N} r_a(m) - \lambda \sum_{k=1}^{N} T_k - \rho \sum_{m=1}^{N} T_m - \\ \sum_{k=1}^{N-1} |r_v(k) - r_v(k-1)| - \sum_{m=1}^{N-1} |r_a(m) - r_a(m-1)|,$$

where T_k and T_m represent the stall time when downloading the k-th video chunk and m-th audio chunk, respectively. λ and ρ are set to 2.5 and 1.5, as recommended by [5].

Parameter Settings. We set target buffer occupancies B_v and B_a to 4 chunks, as per [10]. The buffer difference threshold Q is configured as 8s. The prediction horizon is set to $S = 5$, while α and β are both set to 0.5. The bitrate switch penalties η and μ are configured as 10^{-4}, as suggested by [11].

Baselines. We compare VAMP to the following state-of-the-art ABR schemes: 1) FESTIVE [12]: an online rate-based scheme that chooses the highest available bitrate that is less than the predicted capacity; 2) BBA [13]: an online buffer-based algorithm that picks the bitrate according to a function of buffer occupancy with a reservoir of 5s and cushion of 10s; 3) BOLA [4]: a more sophisticated online buffer-based algorithm that utilizes Lyapunov optimization to pick the bitrate under



(a) Average video bitrate. (b) Average audio bitrate. (c) Average rebuffering time. (d) Average video rate switch. (e) Average audio rate switch.

Fig. 2: Performance comparison of VAMP vs existing ABR schemes over a wireless link.

buffer occupancy constraints; 4) RobustMPC [5]: a hybrid online algorithm that maximizes the QoE over the following five chunks based on buffer occupancy and throughput prediction; 5) MSPC [6]: an online prediction-based scheme that utilizes a Kalman-Filter to predict network capacity and buffer occupancy, to select the bitrate for the next video chunk. The parameters of MSPC are consistent with the original paper [6]; 6) Pensieve [14]: an offline approach that uses reinforcement learning to learn a policy to select bitrate for the next chunk. Note that for the existing solutions we use the same ABR strategy for both video/audio streams and conduct bitrate adaptation of each content types independently [2].

A. VAMP vs. Online ABR Schemes

Fig. 2 shows the performance of online ABR algorithms on *each QoE component considered (A/V bitrate, rebufferring time, bitrate switch)* in the form of Cumulative Distribution Function (CDF) to describe the probability that each metric takes on a value less than or equal to a specific target (shown on the X-axis) [15]. Observe that VAMP reduces significantly the rebuffering time and bitrate switching (measured as the difference between the bitrates of consecutive chunks), while maintaining as high V/A bitrates as the other schemes. Specifically, VAMP obtains playback without interruptions across 95% of the test network traces, better than the SOTA online MSPC (75.7% across test traces). The superior performance of VAMP in avoiding playback interruptions is due to buffer occupancy control, which prevents unsynchronized download of A/V chunks to a large degree and makes the system robust to bandwidth prediction errors. To demonstrate this, Fig. 3 shows the average difference between the audio and video buffer occupancies. Since VAMP controls V/A buffer levels around target values and pauses unnecessary audio or video downloads, it reduces such buffer occupancy differences by 52%–77%. Overall, VAMP reduces average rebuffering time by 18%–99%. VAMP's predictive control approach to bitrate selection leads to 13%–72% and 44%–84% fewer video/audio bitrate switch events. Further, VAMP achieves the highest QoE, attaining 2.94%–33.46% improvements (Table III).

TABLE III: Average QoE for Online Algorithms.

Scheme	VAMP	MSPC	Robust MPC	FESTIVE	BOLA	BBA
Aver. QoE	1.053	1.022	0.954	0.789	0.882	0.936

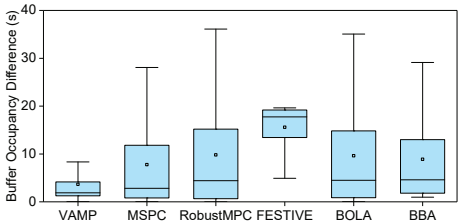


Fig. 3: V/A buffer occupancy difference.

B. VAMP vs. Offline ABR Scheme

In addition to the online ABR schemes, a series of *offline* schemes that utilize machine learning (ML) have been proposed recently [14], [16]. We compare the performance of VAMP with one representative scheme, namely Pensieve [14], employing the pre-trained model provided by the authors. Given that Pensieve is a reinforcement learning-based *offline* ABR scheme, a comparison with the *online* algorithms would not be fair. Hence we decided not to include Pensieve in Table III. The results obtained reveal Pensieve achieves an average QoE of 1.017, and remark that VAMP's optimality holds. Since Pensieve is trained with a fixed video and QoE function, it finds it difficult to adapt to new environment settings. Benefiting from the predictive buffer control and bitrate switching, VAMP improves the average QoE of Pensive by 3.54%, without requiring data intensive model training.

V. Conclusions

This letter introduces VAMP, a predictive approach to synchronizing the download of audio/video chunks. By leveraging predictive multi-step control, VAMP effectively mitigates video/audio buffer imblaance and guarantees smooth high bitrate playback. Experimental results demonstrate that VAMP attains higher QoE than previously proposed ABR solutions, reducing rebuffering time by up to 99%.

References

[1] "Ericsson Mobility Report", June 2021.
[2] Y. Qin, *et al.*, "ABR Streaming with Separate Audio and Video Tracks: Measurements and Best Practices," ACM CoNEXT, 2019.
[3] "Xiph.org Video Test Media," https://media.xiph.org/video/derf/.
[4] K. Spiteri, *et al.*, "BOLA:Near-Optimal Bitrate Adaptation for Online Videos," IEEE INFOCOM, 2016.
[5] X. Yin, *et al.*, "A Control-Theoretic Approach for Dynamic Adaptive Video Streaming over HTTP," ACM SIGCOMM, 2015.
[6] B. Wang, *et al.*, "Improving the Performance of Online Bitrate Adaptation with Multi-Step Prediction Over Cellular Networks," IEEE Transactions on Mobile Computing, 2021.
[7] H. Yuan, *et al.*, "Spatial and temporal consistency-aware dynamic adaptive streaming for 360-degree videos," IEEE J. Sel. Sig. Proc. 2020.
[8] B. Wang, *et al.*, "Towards Forwad-looking Online Bitrate adaptation for DASH," in Proceedings of ACM MM, 2017.
[9] "HSDPA dataset," http://home.ifi.uio.no/paalh/dataset/hsdpa-tcp-logs.
[10] T. Zhang, *et al.*, "Modeling and Analyzing the Influence of Chunk Size Variation on Bitrate Adaptation in DASH," IEEE INFOCOM, 2017.
[11] B. Wang, *et al.*, "Improving Robustness of DASH Against Unpredictable Network Variations," IEEE Transactions on Multimedia, 2021.
[12] J. Jiang, *et al.*, "Improving Fairness, Efficiency, and Stability in HTTP-based Adaptive Video Streaming with FESTIVE," ACM CoNEXT, 2012.
[13] T.Y. Huang, *et al.*, "A Buffer-based Approach to Rate Adaptation: Evidence from a Large Video Streaming Service," SIGCOMM, 2014.
[14] H. Mao, *et al.*, "Neural Adaptive Video Streaming with Pensieve," ACM SIGCOMM, 2017.
[15] J.S. Arora, "Additional Topics on Optimum Design," in Introduction to Optimum Design (Third Edition), 2012.
[16] H. Yuan, *et al.*, "An Ensemble Rate Adaptation Framework for Dynamic Adaptive Streaming over HTTP," IEEE Trans. Broadcasting, 2020.