AMIS: Edge Computing Based Adaptive Mobile Video Streaming

Phil K. Mu[*], Jinkai Zheng[†], Tom H. Luan[†], Lina Zhu[*], Mianxiong Dong[‡], Zhou Su[§]
[*]School of Telecommunications Engineering, Xidian University, Shaanxi, China
[†]School of Cyber Engineering, Xidian University, Shaanxi, China
[‡]Department of Sciences and Informatics, Muroran Institute of Technology, Hokkaido, Japan
[§]School of Mechatronic Engineering and Automation, Shanghai University, Shanghai, China
Corresponding Author: Tom H. Luan, Email: tom.luan@xidian.edu.cn

Abstract—**This work proposes AMIS, an edge computing-based adaptive video streaming system. AMIS explores the power of edge computing in three aspects. First, with video contents pre-cached in the local buffer, AMIS is content-aware which adapts the video playout strategy based on the scene features of video contents and quality of experience (QoE) of users. Second, AMIS is channel-aware which measures the channel conditions in real-time and estimates the wireless bandwidth. Third, by integrating the content features and channel estimation, AMIS applies the deep reinforcement learning model to optimize the playout strategy towards the best QoE. Therefore, AMIS is an intelligent content- and channel-aware scheme which fully explores the intelligence of edge computing and adapts to general environments and QoE requirements. Using trace-driven simulations, we show that AMIS can succeed in improving the average QoE by 14%–46% as compared to the state-of-the-art adaptive bitrate algorithms.**

I. Introduction

Video streaming still represents the most popular Internet application. As reported in [1], videos have engrossed 60.6% of the overall Internet downstream traffic. Moreover, it is reported that 62% of online video views are now from mobile devices [2]. To accommodate the rising video bandwidth demand of mobile users is thus crucial to the scalable development of Internet.

The edge computing-based framework can well address the issue. As shown in Fig. 1, by pre-caching videos in a local buffer installed at the proximity of users, the edge computing can directly serve the cached video contents to users with low-cost one-hop wireless connection. As a result, not only users can acquire the desired high-rate services, but also the backbone bandwidth can be significantly saved.

In this work, we develop AMIS, an Adaptive Mobile vIdeo Streaming system which targets to fully explores the intelligence of edge computing. AMIS explores three features of the edge computing. First, *knowledge of channel*. As a self-running cloud-like device, the edge server can accurately sense the wireless channel and estimate the available channel bandwidth accordingly. Second, *knowledge of video contents*. Unlike WiFi routers which simply relay packets, with the entire video pre-cached in its own storage, edge servers have the full information of video contents, such as the structure of video frames, the feature of video content segments, and the impact of each segment on user's QoE. As a result, edge servers can adapt the strategy of segment presentation instead of simply forwarding video packets. To analyze and fully explore the knowledge of videos for adaptive playout represent the key contribution of AMIS. Three, *computing power*. With powerful computing capability onboard, edge servers can deploy complex AI algorithms to adapt the strategy of video playout by integrating all information to achieve the best perceived video experience of users.

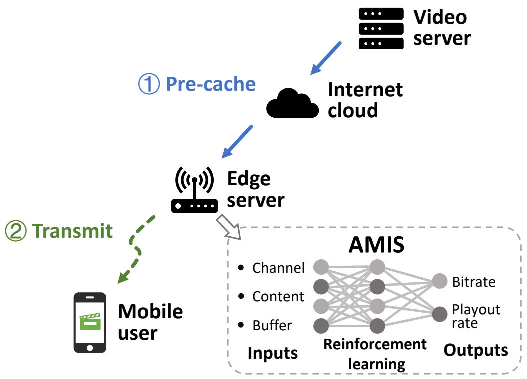


Fig. 1: Edge computing-based mobile video streaming. AMIS is a software installed at the edge server for adaptive video streaming using a neural network model.

To summarize, AMIS is a software framework installed on the edge server which automatically adapts the video streaming at the edge server side, as illustrated in Fig. 1. By estimating the wireless channel bandwidth and analyzing video files cached at the edge server, AMIS controls both the *image quality* and *rhythm* of video playout. For example, when facing varying and insufficient wireless bandwidth, AMIS can tune the video encoding bitrate and reduce the video quality from 720p to 480p to lower the bandwidth demand. Note that changing the video quality would result in severe impact on QoE and should be minimized, AMIS can also slow down the video playout rate to peacefully save the bandwidth. As shown in [3], slowing the rate of video playout up to 25% is often unnoticeable to users, but is quite beneficial for bandwidth saving. By coordinately tuning both bitrate and playout rate with a generic deep reinforming learning model, AMIS manipulates the video streaming strategy at the edge server.

The main contributions of this work are as follows:

- *Framework*: AMIS fully exploits the intelligence of edge computing by applying both bitrate and playout rate adaptations according to the network condition and user's player status to provide better viewing experience.
- *Algorithm Design*: Using content features and channel conditions as input, we exploit asynchronous advantage actor-critic (A3C), an advanced deep reinforcement learning algorithm, to generate both bitrate and rate adaptation strategy. The algorithm can adapt to different environments with modest computation consumption.
- *Validation*: We test AMIS on both real-world dataset and a simulated scenario. The result reveals that AMIS improves the average QoE by 14%–46% with the improvement in average video quality of 11%–18% and the decline in the negative effect caused by playback stall of 35%–40% compared with Pensieve [4], a state-of-the-art ABR algorithm.

The remainder of this paper is organized as follows: Section II makes a brief introduction to the related works. Section III presents the initial motivation and different modules of AMIS. Section IV elaborates the implementation of reinforcement learning and more design details. Section V states the evaluation method and analyses the performance of AMIS. Finally, Section VI closes the paper with the concluding remarks.

II. Related Work

Adaptive bitrate (ABR) streaming is one of the dominant approaches to improve users' quality of experience (QoE) and has long been researched. Early attempts at ABR algorithms are largely based on heuristic rules [5], stochastic methods [6], [7], [8], and control theory [9], [10]. Such algorithms include many parameters that need to be fine-tuned to work in different network conditions [11]. Mao et al. address the issue by introducing the deep reinforcement learning and propose a more generalized ABR algorithm Pensieve [4]. By training a neural network model with feedbacks from users, Pensieve is smart to adapt to different environments and QoE metrics. Inspired by Pensieve, AI-based ABR schemes have been developed in [12], [13], [14], [15], [16]. The above literature, however, only considers bitrate adaptation; while adapting the bitrate from 720p to 480p can significantly save the bandwidth, users would experience obvious changes in the video quality. To fine-tune the QoE, our work introduce the content-aware playout rate adjustment[1], in companion with bitrate adaptation, which provides a brand-new freedom of edge computing-based video adaptation.

Apart from ABR algorithms, edge computing-based video streaming systems with video pre-caching and transcoding have become an emerging trend of QoE optimization [17], [18], [19]. Video transcoding could provide more bitrate options for ABR systems. However, video transcoding is resource-consuming and unpractical for scalable deployment. As compared to above works, our solution is more economic in computing power and therefore more practical. It is also interesting to combine our work with existing works.

[1]Playout rate adaptation is attribute to Kalman et al. [3] which is implemented at the user end and does not benefit from AI algorithms.

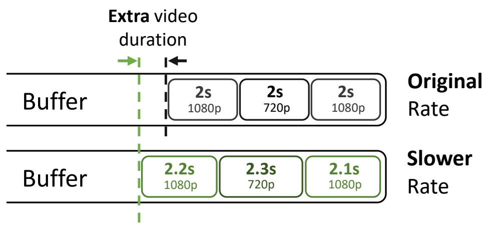


Fig. 2: Playing chunks with slower rates than normal elongates the duration of each video chunk resulting in less chance of the buffer being exhausted.

III. System Model

A. System Overview

We consider an edge computing system as shown in Fig. 1. The edge server pre-caches videos and serves mobile users without the assistance from the Internet. Videos are stored with Y encoding bitrates and each bitrate version is segmented into K *chunks* of the same duration d. Let x_k denote the kth chunk and $r(x_k)$ denote its encoding bitrate.

AMIS adapts the video presentation in two dimensions: *bitrate* and *playout rate*. With bitrate adaptation, AMIS determines $r(x_k)$ for each chunk. The playout rate adaptation is achieved by slowing down the video playout to reduce bandwidth demand. It is shown that people have low sensitivity on playout rate alterations. According to a study in [20], none of 100 viewers realize the rate alterations of a soccer match video up/down to 12% without being informed in advance, and the just noticeable difference of rate alternations is up to 18%. As such, AMIS secretly manipulates certain video chunks into a slower playout rate to reduce their equivalent bitrates under poor network conditions. As illustrated in Fig. 2, this is helpful to avoid the playout buffer being exhausted and reduce stall events. Let $s(x_k)$ denote the playout rate of chunk x_k. To avoid the negative effect of playout rate adaptation on QoE, the lower bound of $s(x_k)$ is limited according to the content feature of x_k, i.e., it is a content-based chunk-level playout rate adjustment.

The workflow of AMIS is plotted in Fig. 3 with details described in Algorithm 1. In Algorithm 1, AMIS analyses the video content cached before transmission and measures the real-time channel conditions to adapt video playout. It is worth noting that all computation jobs (i.e., content analysis, neural network computation, video rate processing) are performed by the edge server, whereas the user device only needs to work like a normal video player. The client is blind to the playback rate change since the rate adjustment is completed at the server.

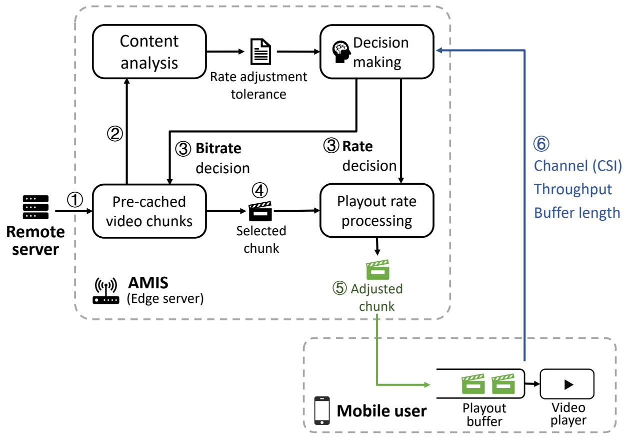


Fig. 3: Workflow of AMIS.

Algorithm 1 Workflow of AMIS (at the Edge Server)

```
Step 1: Pre-cache videos from a remote video server.
Step 2: Analyze the local pre-cached videos to calculate the tolerance of rate adjustment for each chunk, according to how fast the scene changes within that chunk. (Section III-C) // The tolerance of rate adjustment is designed to control the maximum rate alterations within the level where users cannot discern.
repeat
    Step 3: Make bitrate and playout rate decisions for the chunk to be transmitted according to user's playout buffer length, channel condition (throughput measurements and CSIs), and the pre-calculated tolerance of rate adjustment. (Section IV)
    Step 4: Configure the chunk with the corresponding bitrate according to the bitrate decision. Manipulate its playout rate according to the rate decision.
    Step 5: Send the chunk with adjusted rate to the client. // Once the client receives the chunk, it caches the chunk into its buffer to wait for playout.
    Step 6: Receive feedback (e.g., buffer occupancy and stall duration) from the client.
until All chunks are transmitted.
```

B. Model of User's Playout Buffer

The playout buffer at the user end caches received video chunks and submits to player for video presentation. The length of playout buffer is changing over time due to the variable download rate and video playback. Let $T(x_k)$ denote the average throughput during the download of chunk x_k, which can be expressed as

$$T(x_k) = \frac{1}{t_k^{end} - t_k} \int_{t_k}^{t_k^{end}} T(t)dt, \tag{1}$$

where t_k and t_k^{end} denote the start and end time of the download of chunk x_k, respectively. $T(t)$ is the throughput at time t. Except for the last chunk x_K, we consider that the download of x_{k+1} begins as soon as the download of x_k ends, i.e., $t_k^{end} = t_{k+1}, k < K$.

Let $B(t)$ denote the length of playout buffer at time t. $B(t)$ is defined in seconds which represents the time duration of the video cached in the playout buffer that can be played. Based on the above notations, $B(t)$ can be expressed as

$$B(t_{k+1}) = \max\left[B(t_k) - \frac{L(x_k)}{T(x_k)}, 0\right] + \frac{d}{s(x_k)}, \tag{2}$$

where $L(x_k)$ is the size of chunk x_k. $\frac{L(x_k)}{T(x_k)}$ in (2) is the download time of chunk x_k and if it is greater than the playout buffer length $B(t_k)$, a stall would occur due to the depletion of the playout buffer. $\frac{d}{s(x_k)}$ is the actual duration of x_k after adjustment.

C. Content-based Playout Rate Control Model

1) Tolerance of playout rate alternation: Research shows that human beings often have a low sensitivity to video playout rate alterations [20], [3]. In addition to that, we observe that, for different types of scenes, people show different sensitivities to playout rate alterations. Specifically, people are less sensitive to rate tuning for scenes with fast changing contents (e.g., battle scenes, scenes with complex special effects, etc.). This is because that most people do not have a clear idea about how fast the playout rate should be for such scenes so that we can slow down their rate more aggressively. On the contrary, people always share the common sense of proper rate of scenes where contents change normally (e.g., dialogue scenes). For such scenes, rate adjustment needs to be more conservative in order to avoid users' discomfort. Based on this observation, AMIS determines the tolerance of playout rate adjustment based on the degree of how fast the content changes in videos; the tolerance is then used to guide the neural network to make rate decisions and determine the final playout rate for each chunk. In what follows, we elaborate on the method of determining the tolerance of playout rate alternation based on the degree of content change.

2) Tolerance calculation based on SSIM: To evaluate the degree of content change, we first adopt the structural similarity index (SSIM), a widely accepted index measuring the similarity between two images, to analyse each of frames in a chunk [21]. Specifically, SSIM quantifies the perceptual difference between two images based on their luminance, contrast, and structure. SSIM is in the range of $[0, 1]$, where 1 means that the two images are exactly the same and 0 means totally different. Thus the average of all SSIMs between two adjacent frames within a chunk reflects the degree of content change of that chunk.

Denote the SSIM between two images $\mathbf{x}$ and $\mathbf{y}$ by $v_{ssim}(\mathbf{x}, \mathbf{y})$. The average SSIM of chunk x_k is defined as

$$\overline{v}_{ssim}(x_k) = \frac{1}{n-1} \sum_{i=1}^{n-1} v_{ssim}(\mathbf{x}_i^k, \mathbf{x}_{i+1}^k), \tag{3}$$

where n is the total number of frames in chunk x_k, $\mathbf{x}_i^k$ and $\mathbf{x}_{i+1}^k$ are the ith and $(i+1)$th frames in chunk x_k. Then the

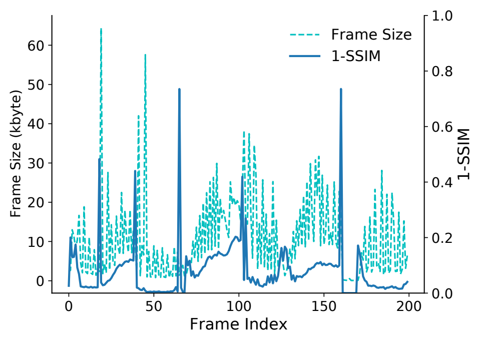
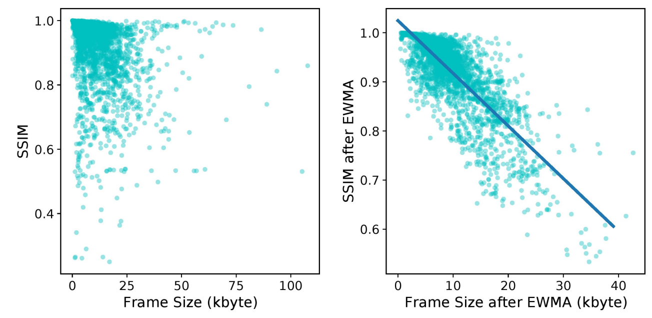


Fig. 4: Comparison between frame size and SSIM.



(a) Original

(b) after EMWA

Fig. 5: Correlation between frame size and SSIM. (a) shows the original data. The result of applying EWMA to both frame size and SSIM is shown in (b), in which the straight line is the result of linear regression.

tolerance of playout rate adjustment for chunk x_k is defined as

$$s_{tol}(x_k) = g(\overline{v}_{ssim}(x_k)), \tag{4}$$

where $g(\cdot) \in [0.1, 0.25]$ is a monotonically non-increasing function, i.e., the higher the average SSIMs (which means the content changes at a slower pace), the less tolerance of adjustment. $s_{tol}(x_k)$ determines the maximum rate alteration can be applied to chunk x_k. The upper bound of $g(\cdot)$ limits the slowest rate to 0.75x for any chunks suggested by [3] and its lower bound is set according to [20], indicating that 0.9x rate and higher is acceptable for all types of chunks. The exact mapping from average SSIM to tolerance is subjective and we set $g(\cdot)$ as a function that linearly maps the input to the range from 0.1 to 0.25 in our experiments.

However, leveraging SSIM to measure the degree of content change directly is time-consuming. Park et al. show that computing all the SSIMs of a whole HD video would take dozens of hours even on a well-equipped personal computer, which is unacceptable in practice [22]. Therefore, we exploit another easy-to-calculate video information, i.e., frame size, to estimate $\overline{v}_{ssim}(x_k)$.

3) Tolerance calculation based on estimated SSIM: In this part, first, we present our observation that frame sizes are related to SSIMs due to the popular variable bitrate (VBR) encoding scheme; then, an SSIM estimation method based on frame sizes is proposed to reduce the computation load as mentioned above.

Specifically, as a type of video encoding method, VBR allocates higher bitrates for complex scenes to make details clearer and lower bitrates for simple scenes to save space. The different compression algorithms for frames result in different frame types, i.e., I (intra-coded), P (predicted), and B (bidirectional predicted) frames. Generally speaking, the size of I frames are larger than that of P and B frames since P and B frames only encode the difference from their neighbor frame(s). Typically, the size of an I frame mainly reflects the content complexity in that frame; whereas the size of a P or B frame reflects the degree of content change between frames. Fig. 4 shows the frame sizes and the SSIMs of the same video. The trends of the two are quite similar, thus it is feasible to estimate SSIMs through frame sizes.

Fig. 5a illustrates the correlation between the original frame size and SSIM. Even though there is an obvious correlation as shown in Fig. 4, the exact relationship is still unclear due to the following reasons: (1) Different frame types result in a considerable frame size fluctuation thus the degree of content change is not the primary factor affecting frame sizes. (2) Not all the SSIMs of adjacent frames in a video are suitable to measure the degree of content change. As shown in Fig. 4, SSIM[2] spikes after a period of calm and such spikes are mainly caused by scene transitions. However, the number of scene transitions is not directly related to the degree of content change. For instance, in a dialogue scene where multiple people are discussing together, scene transitions would occur frequently since the camera needs to focus on the speaking person. It is possible that everyone is speaking at a normal pace and thus the degree of content change still remains low. Another case that makes certain SSIMs unreliable is that some videos contain many identical frames to maintain a high frame rate resulting in lots of meaningless SSIM spikes.

To address the first problem mentioned above, one possible solution is to remove all the I frames to alleviate frame size fluctuation caused by frame types. However, some P frames can still be large enough to cause such frame size fluctuation as well. This is because if a P frame is in the position of a scene transition, which means that the content in the reference frame (i.e., the previous frame) is totally different from that in the P frame, the size of this P frame could be very large because it encodes the difference from the reference frame. Actually, either the problem of frame size fluctuation or invalid SSIMs is a problem about outliers. If we could remove all the outliers from both frame sizes and SSIMs, their relationship would become clearer. In our work, we exploit Exponentially Weighted Moving Average (EWMA) filter to process original

[2]SSIM measures the degree of similarity between two frames; whereas 1-SSIM measures the degree of discrepancy between two frames. These two expressions essentially describe the same thing so that we do not distinguish them in this paper.

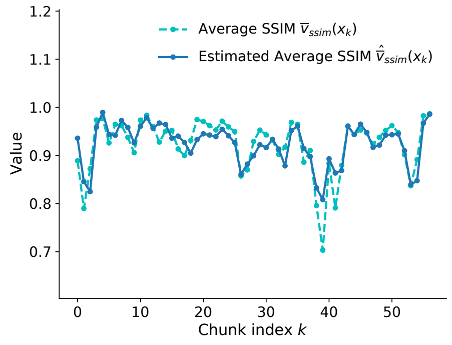


Fig. 6: Comparison between the true average SSIMs and the estimated average SSIMs.

frame size and SSIM data. EWMA is a kind of low-pass filter which filters out high-frequency components brought by outliers.

The ith frame size processed by EWMA is defined as

$$\begin{aligned} f^e(i) &= \alpha f(i) + (1-\alpha) f^e(i-1) \\ &= \sum_{n=0}^{i-1} \alpha(1-\alpha)^n f(i-n), \end{aligned} \tag{5}$$

where $f(i)$ is the ith frame size and $\alpha \in (0,1]$ is a constant that determines the update rate. Likewise, the SSIM between the ith and $(i+1)$th frame processed by EWMA is defined as

$$\begin{aligned} v^e_{ssim}(i) &= \alpha v_{ssim}(i) + (1-\alpha) v^e_{ssim}(i-1) \\ &= \sum_{n=0}^{i-1} \alpha(1-\alpha)^n v_{ssim}(i-n), \end{aligned} \tag{6}$$

where $v_{ssim}(i)$ is the SSIM between the ith and $(i+1)$th frame.

Fig. 5b shows the results of applying EWMA to the original data. It is clear that frame size is linearly related to SSIM after processing. Then, we can estimate the value of SSIM with the corresponding frame size via linear regression, i.e.,

$$\hat{v}^e_{ssim}(i) = a f^e(i) + b, \tag{7}$$

where a and b are two parameters determined by linear regression. With the estimated SSIM $\hat{v}^e_{ssim}(i)$, the estimated average SSIM of chunk x_k is

$$\hat{\bar{v}}_{ssim}(x_k) = \frac{1}{n-1} \sum_{i=1}^{n-1} \hat{v}^e_{ssim}(i). \tag{8}$$

Fig. 6 compares the true average SSIM $\bar{v}_{ssim}(x_k)$ and the estimated average SSIM $\hat{\bar{v}}_{ssim}(x_k)$ (α is set to 0.8). The estimated result is quite in line with the true SSIM with the estimation accuracy of 97.6%. Noting that it is unnecessary to make the estimation accuracy as high as possible since, as discussed above, true SSIMs contain some meaningless outliers. Comparing to $\bar{v}_{ssim}(x_k)$, the calculation of $\hat{\bar{v}}_{ssim}(x_k)$ just needs frame size data, which can be easily obtained and

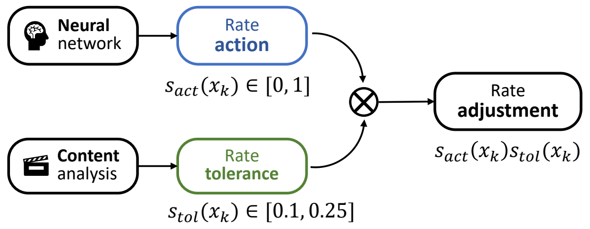


Fig. 7: The final playout rate of chunk x_k is determined by both the rate action made by the neural network and the rate tolerance calculated through content analysis.

do not need hours of time to calculate SSIMs.

Now, we can use the following equation to calculate the tolerance of rate adjustment for chunk x_k instead of (4):

$$s_{tol}(x_k) = g(\hat{\bar{v}}_{ssim}(x_k)). \tag{9}$$

4) Final Rate Adjustment: After tolerance $s_{tol}(x_k)$ is determined, the next step is to determine the exact playout rate of chunk x_k. The rate adjustment is determined by two factors: tolerance $s_{tol}(x_k)$ and rate decision (action) $s_{act}(x_k)$ made by the neural network, as illustrated in Fig. 7. The design of neural network and how the rate decision is made is introduced in Section IV. Here we just state the usage of $s_{act}(x_k)$. The playout rate of chunk x_k follows

$$s(x_k) = 1 - s_{act}(x_k)s_{tol}(x_k), \tag{10}$$

where $s_{act}(x_k) \in [0,1]$ describes the strength of rate adjustment. For instance, $s_{tol}(x_k) = 0.2$ means that the playout rate alteration of x_k should not be greater than 20%, and if $s_{act}(x_k) = 0.5$, the rate adjustment would be 10% indicating the final playout rate of x_k being 0.9x.

D. Channel Model

A key function of AMIS is to measure the real-time channel condition to adapt the video streaming. The existing works mostly use the packet throughput measured at the application layer for video adaptation [4], [12], [13]. AMIS uses the channel state information (CSI) measured at the physical layer, which is enabled by the edge computing that has the full access to the wireless network interface card. There are two advantages of using CSI as compared to the application layer throughput for channel estimation. First, CSI is a fine-grained parameter which is updated very frequently at the physical layer and therefore can accurately catch up with the changing wireless channel. Second, the application layer throughput measurements are available only after the packet transmission begins, which results in a blind period at the beginning of video streaming since no throughput measurement is available within that period. The CSI of the downlink channel is reported from the receiver periodically as long as the link is established. Therefore, CSI can instantaneously work in the video startup phase and persistently measure the channel.

We consider a multiple-input multiple-output (MIMO) communication system with N_{tx} transmit antennas and N_{rx} receive antennas, which is typical for IEEE 802.11n. Let

$\mathbf{x}_t \in \mathbb{C}^{N_{tx}}$ be the transmit signal at time slot t and then the received signal is

$$\mathbf{y}_t = \mathbf{H}_t \mathbf{x}_t + \mathbf{n}_t, \tag{11}$$

where $\mathbf{n}_t \in \mathbb{C}^{N_{rx}}$ is the additive noise distributed according to $\mathcal{CN}(0, \sigma_n^2 \mathbf{I}_{N_{rx}})$, and $\mathbf{H}_t \in \mathbb{C}^{N_{rx} \times N_{tx}}$ is the channel gain between the edge server and the user at time slot t. It follows

$$\mathbf{H}_t = \sqrt{g_t}\mathbf{G}_t, \tag{12}$$

where g_t is the large-scale path loss and shadowing, $\mathbf{G}_t \in \mathbb{C}^{N_{rx} \times N_{tx}}$ is the small-scale fading [23]. Here we adopt the block fading channel model which is general to indoor and outdoor environments, and leverage the first-order Gauss-Markov process to describe fading $\mathbf{G}_t$ [24]. The update of $\mathbf{G}_t$ follows

$$\mathbf{G}_t = \epsilon \mathbf{G}_{t-1} + \sqrt{1-\epsilon^2}\mathbf{\Lambda}_t, \tag{13}$$

where $\epsilon \in [0, 1)$ is the correlation coefficient, and $\mathbf{\Lambda}_t \backsim \mathcal{CN}(0, \mathbf{I}_{N_{rx}})$. According to the Jakes' statistical model, the correlation coefficient follows

$$\epsilon = J_0(2\pi f_D T), \tag{14}$$

where $J_0(\cdot)$ denotes the zeroth order Bessel function, f_D denotes the Doppler frequency, and T denotes the CSI feedback interval [25].

The achievable transmission rate at time slot t is given as

$$C_t = W log_2 det(\mathbf{I}_{N_{rx}} + P\mathbf{H}_t\mathbf{H}_t^H/\sigma_n^2), \tag{15}$$

where W is the bandwidth, P is the transmit power, and $\mathbf{H}_t^H$ is the Hermitian transpose of $\mathbf{H}_t$.

Videos are transmitted through the downlink channel to the user and then channel gain $\mathbf{H}_t$ is estimated there and reported periodically to the edge server. The usage of channel information is discussed in Section IV.

E. QoE Model

In our work, QoE is evaluated from three aspects: (1) video quality, (2) quality variation, and (3) playback stall [10]. The definitions of these three components are as follows:

First, define the quality of chunk x_k as

$$Q(x_k) = Q'(r(x_k)), \tag{16}$$

where $Q'(\cdot)$ is a monotonically non-decreasing function mapping bitrates to video qualities.

Second, we define the penalty for quality variation as

$$P_{var} = \sum_{k=2}^{K} \left| Q(x_k) - Q(x_{k-1}) \right|. \tag{17}$$

Third, the penalty for playback stall is defined as

$$P_{stall} = \sum_{k \in \mathcal{P}} \left(p + \frac{L(x_k)}{T(x_k)} - B(t_k) \right), \tag{18}$$

where p is a constant and $\mathcal{P} = \left\{ k \mid B(t_k) < \frac{L(x_k)}{T(x_k)} \right\}$ denotes the set of the index of the chunks which suffer from stalls during downloading. The constant p is used to punish the number of stalls; whereas $\frac{L(x_k)}{T(x_k)} - B(t_k)$ is used to punish the stall duration.

Finally, QoE is defined as

$$QoE = \sum_{k=1}^{K} Q(x_k) - P_{var} - \eta P_{stall}, \tag{19}$$

where η is the weight of stall penalty.

IV. Learning-based Adaptive Video Streaming

In this section, we elaborate on the design of the decision-making agent targeting to maximize QoE under various network conditions.

A. CSI-based Throughput Prediction

CSI available at the edge server is used for throughput prediction. At each time slot, the channel gain $\mathbf{H}_t$ is a matrix that contains the channel information of $N_{rx} \times N_{tx}$ MIMO channels. In our work, we feed a series of consecutive channel gains to a 3-dimensional CNN to predict the average throughput in the next few seconds. A channel gain matrix reflects the spacial features among multiple MIMO channels and a 3-dimensional matrix composed of multiple consecutive channel gains contains the temporal correlation of each channel, which is critical for prediction.

The training of the throughput prediction module is independent of the training of the DRL agent who is responsible for decision making.

B. Agent Workflow

In our scenario, actions (i.e., bitrate and playout rate decisions) are taken before the transmission of the next chunk. Let s_k denote the state before the transmission of chunk x_k and a_k denote the bitrate and playout rate decisions for chunk x_k. After taking action a_k, the transmission of chunk x_k starts. When the transmission is complete, the agent would receive reward r_k for action a_k.

C. DRL Design

We design two versions of AMIS, i.e., AMIS-v1 and AMIS-v2 and the only difference between them is that AMIS-v2 contains the throughput prediction module (Section IV-A) whereas AMIS-v1 only has the access to measured throughput from the application layer. The neural network of AMIS-v2 is shown in Fig. 8.

State: The bitrate and playout rate decisions for chunk x_k are made based on state s_k. For AMIS-v1, s_k is

$$s_k = (\mathbf{l}_k, \mathbf{m}_k, \mathbf{n}_k, o_k, r(x_{k-1}), B(t_k), s_{tol}(x_k)), \tag{20}$$

where $\mathbf{l}_k \in \mathbb{R}_+^N$ is the throughput measurements for the past N chunks, $\mathbf{m}_k \in \mathbb{R}_+^N$ is the download time of the past N chunks, $\mathbf{n}_k \in \mathbb{R}_+^Y$ is the next chunk sizes of Y available bitrates, o_k is the number of left chunks, $r(x_{k-1})$ is the bitrate of the previous chunk, $B(t_k)$ is the current buffer length, and $s_{tol}(x_k)$ is the tolerance of rate adjustment for chunk x_k. For AMIS-v2, s_k is

$$s_k = (\mathbf{l}_k, \mathbf{m}_k, \mathbf{n}_k, o_k, r(x_{k-1}), B(t_k), s_{tol}(x_k), T_k^{pred}), \tag{21}$$

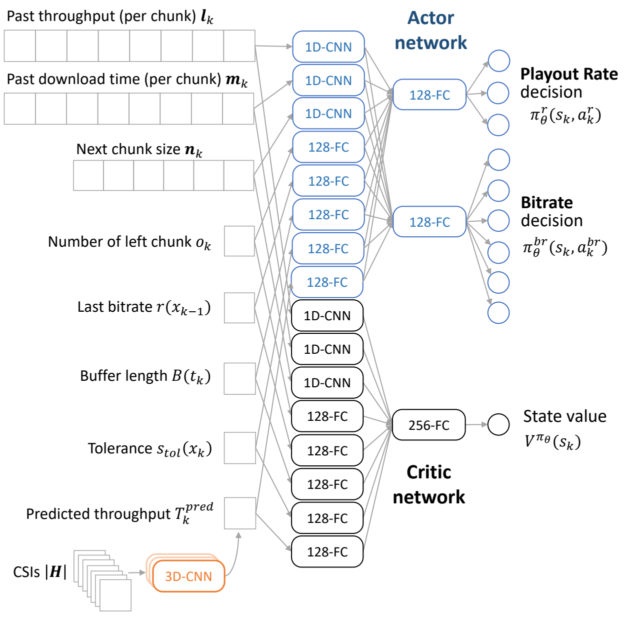

Past throughput (per chunk) l_k
Past download time (per chunk) m_k
Next chunk size n_k
Number of left chunk o_k
Last bitrate $r(x_{k-1})$
Buffer length $B(t_k)$
Tolerance $s_{tol}(x_k)$
Predicted throughput T_k^{pred}
CSIs $|H|$
3D-CNN
1D-CNN
128-FC
256-FC
Actor network
Critic network
Playout Rate decision $\pi_\theta^r(s_k, a_k^r)$
Bitrate decision $\pi_\theta^{br}(s_k, a_k^{br})$
State value $V^{\pi_\theta}(s_k)$

Fig. 8: Structure of the neural network of AMIS-v2.

where T_k^{pred} is the predicted average throughput of the next M seconds from the throughput prediction module.

Action: Action a_k is composed of bitrate decision a_k^{br} and playout rate decision a_k^r and the action spaces of both of them are designed to be discrete.

The number of units in the output layer of a_k^{br} equals the number of available bitrates. Denote the bitrate adaptation policy determined by parameter θ as $\pi_\theta^{br}(s_k, a_k^{br})$, which stands for the probability of taking action a_k^{br} at state s_k.

The outputs of the playout rate decision are the probability distribution of $s_{act}(x_k)$ standing for the strength of rate adjustment for chunk x_k. $s_{act}(x_k)$ is a real number in the range of 0 to 1, but here we quantize $s_{act}(x_k)$ into three levels, i.e., $s_{act}(x_k) \in \{0, 0.5, 1\}$, to decrease the action space. The three values of $s_{act}(x_k)$ stand for no adjustment, moderate adjustment, and aggressive adjustment, respectively. For the unity of expression, we use a_k^r instead of $s_{act}(x_k)$ to describe the rate decision made by the agent. Likewise, denote the rate adaptation policy determined by parameter θ as $\pi_\theta^r(s_k, a_k^r)$, which stands for the probability of taking action a_k^r at state s_k.

Reward: The agent receives reward r_k after the transmission of chunk x_k completes. To make the cumulative reward in line with the QoE, which is our optimization objective, the reward is defined as

$$r_k = \begin{cases} Q(x_k) - |Q(x_k) - Q(x_{k-1})| \\ \quad -\eta(p + t_{stall}) - \mu s_{act}(x_k)Q(x_k), & \text{if } k \in \mathcal{P} \\ Q(x_k) - |Q(x_k) - Q(x_{k-1})| \\ \quad -\mu s_{act}(x_k)Q(x_k), & \text{otherwise,} \end{cases} \tag{22}$$

where $t_{stall} = \frac{L(x_k)}{T(x_k)} - B(t_k)$ is the stall duration. To avoid unnecessary playout rate adjustments, $s_{act}(x_k)Q(x_k)$ in (22) is designed to penalize the agent for rate alteration with weight μ. The penalty is proportional to both adjustment strength $s_{act}(x_k)$ and video quality $Q(x_k)$. It is straightforward that the greater the adjustment strength, the greater the penalty is. The reason that we design the penalty proportional to quality $Q(x_k)$ is to encourage our system to slow down the playout rate under poor network conditions where lower bitrates are always selected and slower rate is preferred.

D. Training Method

A3C framework contains an actor network who is a decision-maker and a critic network who evaluates the performance of the actor network.

Critic network: Denote the parameters in the critic network as θ_v. Let the estimation of state value under policy π_θ be $V^{\pi_\theta}(s_k)$, which is the output of the critic network. $V^{\pi_\theta}(s_k)$ is an estimation of the expected cumulative discounted reward from state s_k by following policy π_θ.

Define n-step estimated return as

$$R_k^{(n)} = \sum_{t=0}^{n-1} \gamma^t r_{k+t} + \gamma^n V^{\pi_\theta}(s_{k+n}), \tag{23}$$

where γ is the discount factor. Then, the training of the critic network follows the n-step Temporal Difference (TD) method, that is

$$\theta_v \leftarrow \theta_v - \alpha' \sum_k \nabla_{\theta_v} (R_k^{(n)} - V^{\pi_\theta}(s_k))^2, \tag{24}$$

where α' is the learning rate of the critic network.

Actor network: The training of the actor network follows the Policy Gradient (PG) method, that is

$$\begin{aligned} \theta \leftarrow \theta + \alpha \sum_k (\nabla_\theta log \pi_\theta^{br}(s_k, a_k^{br}) \\ + \nabla_\theta log \pi_\theta^r(s_k, a_k^r)) A(s_k, (a_k^{br}, a_k^r)) \\ + \beta^{br} \sum_k \nabla_\theta H(\pi_\theta^{br}(s_k)) \\ + \beta^r \sum_k \nabla_\theta H(\pi_\theta^r(s_k)), \end{aligned} \tag{25}$$

where α is the learning rate of the actor network, $A(s_k, (a_k^{br}, a_k^r))$ is the advantage function, $H(\cdot)$ is the entropy, and β^{br} and β^r are regularization parameters. Given an action a_k, advantage function $A(s_k, (a_k^{br}, a_k^r))$ is used to evaluate how much better is taking action (a_k^{br}, a_k^r) at state s_k better than just following the action suggested by the current policy π_θ. In practice, the advantage function is calculated as

$$A(s_k, (a_k^{br}, a_k^r)) = R_k^{(n)} - V^{\pi_\theta}(s_k). \tag{26}$$

Parallel training: Parallel training is a critical feature of A3C. The whole framework is made up of a center model and multiple agents training simultaneously. We follow the asynchronous training method in [26].

Video demonstration of our method can be found at https://sites.google.com/view/amis-demo. The tolerance of rate adjustment is estimated based on the frame sizes of this movie clip. The reinforcement learning agent is trained in the way mentioned above. This demonstration shows the bitrate and playout rate adaptation result under a fluctuating network condition.

V. Evaluation

A. Experimental Setup

In this section, we validate the performance of AMIS using simulations in comparison with existing works.

1) Network trace: We use two types of network traces to evaluate the performance of AMIS, i.e., traces from HSDPA dataset and synthetic network traces. We use HSDPA dataset, which is the bandwidth measured on moving vehicles under 3G/HSDPA wireless network, to test the system performance in the real-measured network environment. Note that HSDPA dataset is mainly used to test AMIS-v1 since it does not contain channel information which is required for AMIS-v2. AMIS-v2 is evaluated through the synthetic network traces accompanied with the corresponding CSI. The channel follows the model described in Section III-D. In our experiment, we set $N_{rx} = N_{tx} = 3$, $f_D = 30Hz$, $T = 0.2s$, $W = 1MHz$, $\sigma_n^2 = 4\times10^{-10}W$, and $P = 4W$. For large-scale path loss and shadowing component g_t, we model g_t as a first-order Markov process with 5 states $\{122.2, 116.7, 110.9, 108.3, 101.7\}$dB. The self transition probability is set to the highest and other transition probabilities follow the decay rate of 20%. The state transition takes place every 2s.

2) Video dataset: We use the standard MPEG-DASH videos from ITEC-DASH dataset to train our agent [27]. The selected videos include Big Buck Bunny, Elephants Dream, and Of Forest and Men. Each video has the bitrates of $\{400, 600, 1000, 1500, 2100, 3100\}$kbps. To reduce the training complexity, we only use the first 60 chunks and each chunk contains 2s of video content.

3) AMIS settings: For the throughput prediction module (Section IV-A), the input layer consists of 25 past channel gains (i.e., the CSIs within the past 5s) with complex entries converted to their moduli. The training objective is set to predict the average throughput within the next 3s. The channel gains are fed into a 3D CNN with 256 filters of size $(15, 2, 2)$, stride 1, followed by 2 cascaded fully-connected layers with 512 and 256 units respectively. Finally, the output layer has 1 linear unit standing for the predicted throughput. We use Adam optimizer with the learning rate of 10^{-4}. The training set contains 10^5 pairs generated from the synthetic environment and the test set contains 2×10^4 pairs which are different from that in the training set.

For the reinforcement learning module (Section IV-C), we set η and μ to 4 and 0.2, respectively, and use linear quality mapping, i.e., $Q'(x) = x$. We set the number of past chunks N (i.e., the length of $\mathbf{l}_k$ and $\mathbf{m}_k$) to 8, and set the number of available bitrates Y to 6. The neural network architecture is illustrated in Fig. 8. For both AMIS-v1 and AMIS-v2, the first hidden layer consists of a series of fully-connected layers and 1D CNNs with 128 filters of size 4, stride 1. The output layer of rate decision contains 3 units corresponding to no adjustment, moderate adjustment, and aggressive adjustment,

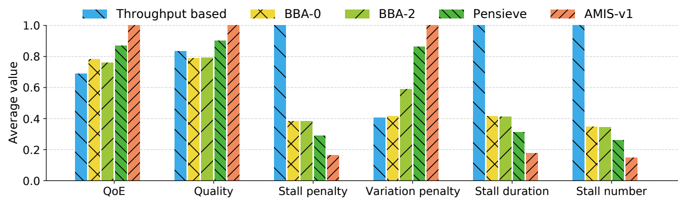


Fig. 9: Comparison of normalized QoE and its components between AMIS and other existing ABR algorithms on the HSDPA dataset.

respectively. The output layer of bitrate decision contains 6 units corresponding to the 6 available bitrates. We apply the softmax operator to both of them to output probabilities. For the critic network, the output layer contains only one linear unit representing the estimated state value. For both the actor network and critic network, units in the first and second hidden layer are activated by ReLu functions. RMSProp is selected as the optimizer with learning rates of 0.0001 and 0.001 for the actor network and the critic network respectively. The parallel training contains 16 agents in total. Discount factor γ is set to 0.99.

B. ABR Algorithms

In the experiments, we compare the performance of our system with the following existing ABR algorithms:

- **Throughput-based algorithm:** This algorithm predicts the future throughput with the harmonic mean of 5 past throughput measurements and chooses the highest bitrate below the predicted throughput for the next chunk.
- **BBA-0:** This is a buffer-based ABR algorithm which linearly maps the buffer length to the bitrate decisions. [5]. In our experiments, BBA-0 selects the lowest bitrate when the buffer length is lower than 4s and selects the highest bitrate when the buffer length is greater than 24s.
- **BBA-2:** BBA-2 is an updated version of BBA-0 [5]. The main difference is that BBA-2 linearly maps the next chunk sizes to the bitrate decisions. Besides that, in the start-up phase, BBA-2 increases bitrates more aggressively to try out the upper bound of the network throughput to adapt to the network quicker.
- **Pensieve:** Pensieve is a DRL-based ABR algorithm that leverages A3C as well [4]. Its experiment result shows that Pensieve is the state-of-the-art ABR algorithm and robust in various network environments. We retrain Pensieve in our scenario where chunk length is 2s.

C. Experiments and Results

1) Evaluation on HSDPA dataset: We evaluate the performance of AMIS-v1 on the HSDPA dataset. The comparison result in terms of normalized QoE and its components is illustrated in Fig. 9. AMIS-v1 outperforms other ABR algorithms in terms of QoE with improvements of 14%–41% achieved by higher video quality, shorter stall duration, and less stall number. Compared with the state-of-the-art ABR algorithm

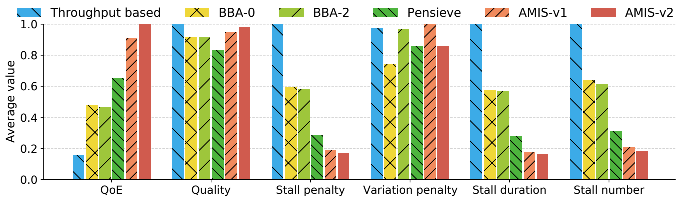
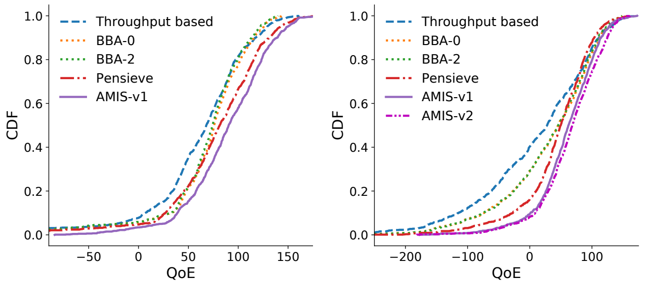


Fig. 10: Comparison of normalized QoE and its components between AMIS and other existing ABR algorithms on the synthetic dataset.



(a) HSDPA dataset. (b) Synthetic dataset.

Fig. 11: Comparison of QoE distribution on different dataset.

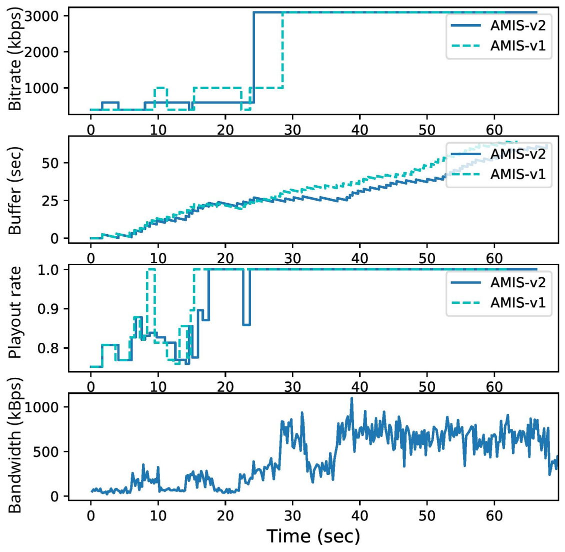


Fig. 12: Comparison of decision process between AMIS-v1 and AMIS-v2.

Pensieve, AMIS-v1 improves the average video quality by 11% and decreases the average stall penalty by 40% where the stall duration is 42% shorter and the stall number is 37% less. Rate adjustment allows AMIS to slow down the playout rate instead of just switching to lower bitrates when necessary. However, the sacrifice is quality variation. From Fig. 9, we observe that the average variation penalty of AMIS-v1 is the highest among all the algorithms indicating that the bitrate switch of AMIS-v1 is more frequent.

The distribution of QoE is illustrated in Fig. 11a. There is an evident gap between Pensieve and AMIS-v1 indicating that AMIS-v1 performs well from poor to good network conditions. As the network condition gets better, the curves of AMIS-v1 and Pensieve converge together since the sufficient network bandwidth allows the agent to choose the highest bitrate all the time so that the rate adjustment is unnecessary.

2) Evaluation on synthetic dataset: The throughput trace from the synthetic data is accompanied by the corresponding channel information which is used for throughput prediction in AMIS-v2. The comparison result is illustrated in Fig. 10.

First, we focus on the performance of AMIS-v1. From Fig. 10, we observe that AMIS-v1 achieves the highest QoE which is a 35% increase compared with Pensieve. In addition, AMIS-v1 reduces the stall duration and stall number by 34% and 36%, respectively, compared with Pensieve. The video quality achieved by AMIS-v1 is slightly lower than that of the throughput-based algorithm whereas the stall duration and stall number of the throughput-based algorithm are around 5 times greater than that of AMIS-v1. As AMIS-v1 does on the HSDPA dataset, AMIS-v1 has the highest variation penalty among all. Fig. 11b depicts the distribution of QoE, where the performance of AMIS is similar to that on the HSDPA dataset.

Second, by introducing the channel information from the physical layer, AMIS-v2 achieves a further improvement compared with AMIS-v1. The training result shows that the average throughput prediction error is around 120kBps. Compared with AMIS-v1, AMIS-v2 improves the average QoE by 8%. Specifically, the average video quality of AMIS-v2 slightly exceeds AMIS-v1 by 3% and the stall duration and number are reduced by 2% and 11% respectively. The most significant improvement of AMIS-v2 is that it reduces the variation penalty by 17% compared with AMIS-v1. In other words, the channel information helps AMIS-v2 reduce the unnecessary bitrate switches.

To have a better understanding of the performance difference between AMIS-v1 and AMIS-v2, Fig. 12 profiles the behavior of both algorithms in the same environment. The network starts with insufficient bandwidth and gradually gets better in this trial. From Fig. 12, we can observe that both algorithms adapt to the changing bandwidth well with the lowest bitrate at the beginning and the highest bitrate in the end. The difference is that the time when AMIS-v2 switches to the highest bitrate is earlier than that of AMIS-v1 by several chunks. In other words, AMIS-v2 reacts faster to bandwidth changes resulting in a higher bandwidth utility. Besides that, the bitrate decisions of AMIS-v2 are more stable when the bandwidth is insufficient (within the range of 0s to 25s) which is consistent with the above statistic result.

VI. Conclusion

We argue that the edge computing based video streaming is fundamental to the scalable development of Internet and will become increasingly important to serve advanced multimedia applications. This work proposes AMIS, an AI-enabled content- and channel-aware adaptive video streaming system.

AMIS fully exploits the capability of edge computing by optimizing both the bitrate and playout rate adjustment based on the analysis of content features and real-time CSI. By applying the advanced deep reinforcement learning algorithm with all above ingredients, AMIS is general to different deployment environments. Using extensive experimental result, we show that AMIS succeeds in improving the average QoE by 14–46% compared with the best state-of-the-art ABR algorithm. In the future work, we intend to deploy and test AMIS in the real-world environment.

ACKNOWLEDGMENT

The work of Tom H. Luan was partially supported by the national Key R&D Program of China under the Grant No. 2018YFB1700200. The work of Mianxiong Dong was partially supported by JSPS KATENHI under the Grant No. JP20F20080. The work of Zhou Su was partially supported by NSFC under the Grant No. U1808207.

REFERENCES

[1] “2019 global internet report,” https://www.sandvine.com/global-internet-phenomena-report-2019.

[2] “Mobile share of global digital video plays from 3rd quarter 2013 to 2nd quarter 2018,” https://www.statista.com/statistics/444318/mobile-device-video-views-share/.

[3] M. Kalman, S. Eckehard, and B. Girod, “Adaptive playout for real-time media streaming,” in *Proceedings of IEEE ISCAS*, 2002.

[4] H. Mao, R. Netravali, and M. Alizadeh, “Neural adaptive video streaming with Pensieve,” in *Proceedings of ACM SIGCOMM*, 2017.

[5] T.-Y. Huang, R. Johari, N. McKeown, M. Trunnell, and M. Watson, “A buffer-based approach to rate adaptation: Evidence from a large video streaming service,” in *Proceedings of ACM SIGCOMM*, 2014.

[6] K. Spiteri, R. Urgaonkar, and R. K. Sitaraman, “BOLA: Near-optimal bitrate adaptation for online videos,” in *Proceedings of IEEE INFOCOM*, 2016.

[7] Y. Sun, X. Yin, J. Jiang, V. Sekar, F. Lin, N. Wang, T. Liu, and B. Sinopoli, “CS2P: Improving video bitrate selection and adaptation with data-driven throughput prediction,” in *Proceedings of ACM SIGCOMM*, 2016.

[8] T. H. Luan, L. X. Cai, and X. Shen, “Impact of network dynamics on user’s video quality: Analytical framework and QoS provision,” *IEEE Transactions on Multimedia*, vol. 12, no. 1, pp. 64–78, 2009.

[9] Y. Qin, R. Jin, S. Hao, K. R. Pattipati, F. Qian, S. Sen, B. Wang, and C. Yue, “A control theoretic approach to ABR video streaming: A fresh look at PID-based rate adaptation,” in *Proceedings of IEEE INFOCOM*, 2017.

[10] X. Yin, A. Jindal, V. Sekar, and B. Sinopoli, “A control-theoretic approach for dynamic adaptive video streaming over HTTP,” in *Proceedings of ACM SIGCOMM*, 2015.

[11] Z. Akhtar, Y. S. Nam, R. Govindan, S. Rao, J. Chen, E. Katz-Bassett, B. Ribeiro, J. Zhan, and H. Zhang, “Oboe: Auto-tuning video ABR algorithms to network conditions,” in *Proceedings of ACM SIGCOMM*, 2018.

[12] X. Xiao, W. Wang, T. Chen, Y. Cao, T. Jiang, and Q. Zhang, “Sensor-augmented neural adaptive bitrate video streaming on UAVs,” *IEEE Transactions on Multimedia*, vol. 22, no. 6, pp. 1567–1576, 2020.

[13] S. Sengupta, N. Ganguly, S. Chakraborty, and P. De, “HotDASH: Hotspot aware adaptive video streaming using deep reinforcement learning,” in *Proceedings of IEEE ICNP*, 2018.

[14] T. Huang, X. Yao, C. Wu, R. Zhang, Z. Pang, and L. Sun, “Tiyuntsong: A self-play reinforcement learning approach for ABR video streaming,” in *Proceedings of IEEE ICME*, 2019.

[15] T. Huang, C. Zhou, R.-X. Zhang, C. Wu, X. Yao, and L. Sun, “Stick: A harmonious fusion of buffer-based and learning-based approach for adaptive streaming,” in *Proceedings of IEEE INFOCOM*, 2020.

[16] Y. Zhang, Y. Zhang, Y. Wu, Y. Tao, K. Bian, P. Zhou, L. Song, and H. Tuo, “Improving quality of experience by adaptive video streaming with super-resolution,” in *Proceedings of IEEE INFOCOM*, 2020.

[17] Y. Guo, F. R. Yu, J. An, K. Yang, C. Yu, and V. C. M. Leung, “Adaptive bitrate streaming in wireless networks with transcoding at network edge using deep reinforcement learning,” *IEEE Transactions on Vehicular Technology*, vol. 69, no. 4, pp. 3879–3892, 2020.

[18] J. Luo, F. R. Yu, Q. Chen, and L. Tang, “Adaptive video streaming with edge caching and video transcoding over software-defined mobile networks: A deep reinforcement learning approach,” *IEEE Transactions on Wireless Communications*, vol. 19, no. 3, pp. 1577–1592, 2020.

[19] F. Fu, Y. Kang, Z. Zhang, and F. R. Yu, “Transcoding for live streaming-based on vehicular fog computing: An actor-critic DRL approach,” in *Proceedings of IEEE INFOCOM WKSHPS*, 2020.

[20] C. de’Sperati and G. B. Bovy, “Low perceptual sensitivity to altered video speed in viewing a soccer match,” *Scientific reports*, vol. 7, no. 1, pp. 1–7, 2017.

[21] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, “Image quality assessment: from error visibility to structural similarity,” *IEEE Transactions on Image Processing*, vol. 13, no. 4, pp. 600–612, 2004.

[22] K. Park and M. Kim, “EVSO: Environment-aware video streaming optimization of power consumption,” in *Proceedings of IEEE INFOCOM*, 2019.

[23] L. Liang, J. Kim, S. C. Jha, K. Sivanesan, and G. Y. Li, “Spectrum and power allocation for vehicular communications with delayed CSI feedback,” *IEEE Wireless Communications Letters*, vol. 6, no. 4, pp. 458–461, 2017.

[24] T. Kim, D. J. Love, and B. Clerckx, “Does frequent low resolution feedback outperform infrequent high resolution feedback for multiple antenna beamforming systems?” *IEEE Transactions on Signal Processing*, vol. 59, no. 4, pp. 1654–1669, 2011.

[25] J. Proakis, “Digital communications, new york: Mcgrawhill,” 2001.

[26] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, “Asynchronous methods for deep reinforcement learning,” in *Proceedings of ICML*, 2016.

[27] S. Lederer, C. Müller, and C. Timmerer, “Dynamic adaptive streaming over HTTP dataset,” in *Proceedings of ACM MMSys*, 2012.