





Vabis: Video Adaptation Bitrate System for Time-Critical Live Streaming

Tongtong Feng , Haifeng Sun , Qi Qi , Member, IEEE, Jingyu Wang , Member, IEEE, and Jianxin Liao, Member, IEEE

Abstract—With the rise of time-critical and interactive scenarios, ultra-low latency has become the most urgent requirement. Adaptive bitrate (ABR) schemes have been widely used in reducing latency for live streaming services. However, the traditional solutions suffer from a key limitation: they only utilize coarse-grained chunk to solve the I-frame misalignment problem in different bitrate switching process at the cost of increasing latency. As a result, existing schemes are difficult to guarantee the timeliness and granularity of control in essence. In this paper, we use a frame-based approach to solve the I-frame misalignment problem and propose a video adaptation bitrate system (Vabis) in units of the frame for time-critical live streaming to obtain the optimal quality of experience (QoE). On the server-side, a Few-Wait ABR algorithm based on Reinforcement Learning (RL) is designed to adaptively select the bitrate of future frames by state information that can be observed, which can subtly solve the problem of I-frame misalignment. A rule-based ABR algorithm is designed to optimize the Vabis system for the weak network. On the client-side, three delay control mechanisms are designed to achieve frame-based fine-grained control. We construct a trace-driven simulator and the real live platform to evaluate the comprehensive live streaming performance. The results show that Vabis is significantly better than the existing methods with decreases in an average delay of 32%–77% and improvements in average QoE of 28–67%.

Index Terms—Bitrate adaptation, live streaming, ultra-low latency, reinforcement learning.

I. INTRODUCTION

WITH the proliferation of mobile devices such as smartphones and smart-watches, the proportion of live streaming traffic and the number of mobile live video streaming applications are also growing rapidly [1]. Many adaptation bitrate (ABR) algorithms have been widely adopted in live video streaming to improve Quality of Experience (QoE) by tradeoff

Manuscript received June 16, 2019; revised October 29, 2019; accepted December 19, 2019. Date of publication December 25, 2019; date of current version October 23, 2020. This work was supported in part by the National Key R&D Program of China 2018YFB1800502, in part by the National Natural Science Foundation of China under Grants 61671079 and 61771068, in part by the Beijing Municipal Natural Science Foundation under Grant 4182041, and in part by the Ministry of Education and China Mobile Joint Fund MCM20180101. The co-first author is H. Sun. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Mea Wang. (Corresponding authors: Jingyu Wang; Qi Qi.)

The authors are with the State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing 100876, China and with EBUPT.COM Beijing 100191, China (e-mail: ftt@bupt.edu.cn; hfsun@bupt.edu.cn; qiqi8266@bupt.edu.cn; wangjingyu@bupt.edu.cn; liaojx@bupt.edu.cn).

Color versions of one or more of the figures in this article are available online at <http://ieeexplore.ieee.org>.

Digital Object Identifier 10.1109/TMM.2019.2962313

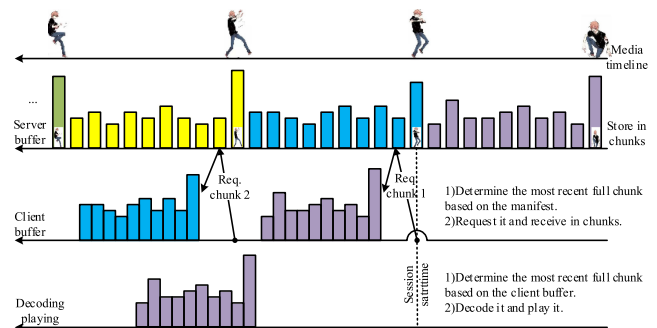


Fig. 1. Chunk-based streaming media transmission process. Video server can not start transmitting chunk 1 until the entire chunk is cached and the client can not start playing chunk 1 until the entire chunk is downloaded.

high quality and low rebuffering user experience [2], [3], such as buffer-based methods [4], rate-based methods [5], quality-based methods [6], and RL-based methods [7]. But with the advent of 5G, many time-critical and interactive applications are emerging in a blowout, such as a tip for the live show, the SMG football (lottery ticket for a football match) and virtual reality (VR)[8], [9]. This will result in ultra-low latency becoming the most urgent requirement in live streaming scenarios [10]. However, the existing live streaming mode is usually achieved by first buffering the video locally for 3–5 s and then playing the video through the ABR algorithm, and only utilizing simple and delay-insensitive chunk as the basic unit of decision making, which is not applicable for live streaming with extremely high latency requirements. How to achieve the goal of obtaining the optimal QoE [11], [12] on the basis of ensuring ultra-low latency [13], [14] has become an upcoming issue.

As shown in Fig. 1. Current excellent adaptive bitrate algorithms are implemented in a chunk-based streaming media framework. The feature is that the video server can not start transmitting a given chunk until the entire chunk is cached and the client can not start playing a given chunk until the entire chunk is downloaded. Due to that bitrate decision can only be made at the chunk boundary, the bitrate strategy cannot be adjusted until the current entire chunk is downloaded. This will cause the environmental state to not be quickly reflected in the bitrate execution and form *long-wait* ABR. As a result, it is difficult to guarantee the timeliness and granularity of control in essence.

The reason why the existing solutions do not use a frame-based approach that can achieve more fine-grained control and

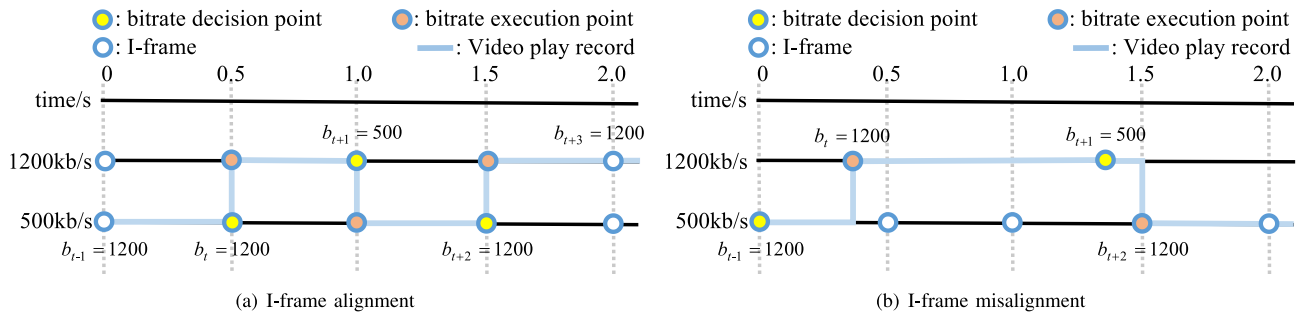


Fig. 2. Bitrate selection process. When the I-frame is aligned, the bitrate decision is synchronized with the bitrate execution. When the I-frame is not aligned, the bitrate decision is not synchronized with the bitrate execution and the I-frame position of the selected bitrate is unpredictable.

decision making is that the frame-based method can't solve the problem of I-frame misalignment, but the chunk-based method can.

The problem of I-frame misalignment is crucial for video transmission. As shown in Fig. 2, due to that video bitrate will only be switched at the I-frame position, the problem of I-frame misalignment will cause the bitrate decision is not synchronized with the bitrate execution. Its characteristic is that the location of the next I-frame of the selected bitrate video is unpredictable and the time difference between bitrate decision and bitrate execution is also unpredictable. Since the optimal bitrate decision is made based on the current state of the environment, this will result in the optimal bitrate at the time of the bitrate decision is not necessarily the optimal bitrate at the time of the bitrate execution. And then this will cause inaccurate bitrate decision and out-of-tolerance performance. More details can be found in Section 3.2.

The chunk-based methods, at the cost of increasing the latency, can solve the problem of I-frame misalignment between different bitrate videos through a transcoding server, which implements bitrate conversion, video buffering different bitrate and dividing the video into chunks according to I-frame position. The feature is that the client cannot perform frame downloading until at least a chunk is buffered on the server-side and I-frame alignment operation is implemented. This process greatly increases the waiting delay of the frame and increases the resource cost of the streaming media server. However, the characteristic of the frame-based method is that frame can be downloaded as long as there are frames in the server playback buffer and frame can be played as long as there are frames in the client playback buffer. So the frame-based method cannot achieve I-frame alignment by buffering.

ABR algorithm is a bitrate selection strategy. The existing outstanding ABR algorithm can be divided into the following two major categories. One is ABR algorithms based on fixed rules, such as buffer-based methods [4], rate-based methods [5], and quality-based methods [6]. These schemes require significant tuning and do not generalize to different network conditions and QoE objectives, which only build simple and inaccurate models in the current environment [7]. Therefore, these methods are inevitably unable to achieve optimal performance for various network conditions. The other is RL-based methods [15], which train a neural network model that adaptively

selects the bitrate of future segments through the observed state information. It can automatically adjust its parameters according to its input to obtain the optimal bitrate in complicated scenarios. Therefore, we adopt the RL-based approach to select the bitrate of future frames. However, due to the problem of I-frame misalignment, the RL-based method will inevitably cause mismatching between action and reward during training. To achieve the correct parameter update each time, how to achieve action and reward match is also a great challenge.

In this paper, we use a frame-based approach to solve the I-frame misalignment problem and propose a video adaptation bitrate system (Vabis) in units of the frame for time-critical live streaming to obtain the optimal QoE based on ensuring ultra-low latency. On the server-side, a Few-Wait ABR algorithm based on RL is designed to adaptively select the bitrate of future frames by state information that can be observed. It can skillfully solve the problem of *long-wait* between bitrate decision and bitrate execution and the problem that the action does not match the reward caused by the I-frame misalignment. Due to the weak network making the RL algorithm more conservative during training, as an optimization, we specifically build a rule-based ABR algorithm for the weak network. On the client-side, three delay control mechanisms are designed: slow playback, fast playback, and frame skipping to achieve frame-based fine-grained control.

We compare Vabis with the existing excellent adaptive bitrate methods in different video and network scenarios. The results from the simulator show that Vabis is significantly better than the existing methods with decreases in an average delay of 32%–77% and improvements in average QoE of 28%–67%. Moreover, Vabis is better in the rapidly changing network environment and the network with a high proportion of weak network. In addition, in the real live scenarios, by comparing Vabis with the outstanding adaptive bitrate methods, the user can intuitively feel that Vabis has a lower delay and smoother playback than other methods.

In summary, this paper makes the following key contributions:

- Since the chunk-based approach will greatly increase the video transmission delay, we refine the video transmission unit to the frame level and construct a frame-based streaming media transmission system.
- Since the problem of I-frame misalignment solved by the transcoding server will greatly increase the video waiting delay and resource cost of the streaming server, a

Few-Wait ABR algorithm based on RL is designed to solve this problem.

- Vabis adds three delay control mechanisms: slow playback, fast playback, and frame skipping. Through the `target_buffer` parameter of the server feedback, the client and the server can jointly control to get ultra-low latency user experience.
- To the best of our knowledge, we are the first to apply rule-based and RL-based hybrid ABR algorithm to live streaming scenarios. Due to the weak network making the RL algorithm more conservative during training, as an optimization, we specifically build a rule-based ABR algorithm for the weak network.

II. SYSTEM DESIGN

A. Challenge

In this subsection, we introduce the challenges of building a frame-based streaming media transport framework to achieve optimal QoE on the basis of ensuring ultra-low latency.

- 1) Vabis need to balance a variety QoE goals, such as maximizing video quality (highest average bitrate), minimizing video rebuffering time (the duration time that the client buffer is empty), minimizing video bitrate switching times, minimizing video latency (the maximum time difference between when the video is captured and when the user sees it). However, many of these goals are inherently conflicting.
- 2) How to establish a frame-based streaming media transmission system and achieve fine-grained control based on the frame? Compared with chunk-based streaming media systems, frame-based systems not only need to handle more frequent state transmission but also need to balance the relationship between fine-grained control and control costs.
- 3) How to make the environment changes quickly reflected in the bitrate execution? We solve this problem of I-frame misalignment by the algorithm. Due to that the location of the next I-frame is unpredictable, the time difference between bitrate decision and bitrate execution is also unpredictable. This will result in that the decided bitrate is not optimal when the bitrate switching strategy is executed. So, how to achieve the few-wait between them is a problem to be solved.
- 4) We adopt an RL-based approach to select the bitrate of future frames. However, due to the problem of I-frame misalignment, the RL-based method will inevitably cause mismatching between action and reward during training. More details can be seen in section 4.2. In order to achieve the correct parameter update each time, how to achieve action and reward match during RL training is a great challenge.
- 5) How to optimize the performance of Vabis? Since the RL algorithm selects the bitrate of future frames by the strategy of maximizing the long-term cumulative reward, this will lead to that the higher the proportion of the weak network and the more conservative the training of the model

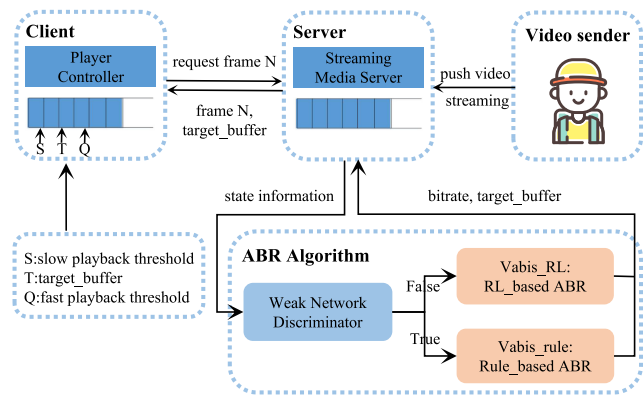


Fig. 3. Vabis's system architecture.

will be. More details can be found in Section IV. In order to optimize the performance of Vabis, how to design a rule-based ABR algorithm separately for the weak network is also a great challenge.

B. System Architecture

On the basis of the conventional chunk-based streaming media transmission framework, we refine the video transmission unit to the frame level and propose Vabis for time-critical live streaming, as shown in Fig. 3. On the client-side, Vabis designs three delay control mechanisms to achieve fine-grained control based on frames. On the server-side, a Few-Wait ABR algorithm based on RL is designed to adaptively select the bitrate of future frames, which can subtly solve the problem of I-frame misalignment. Specifically, to optimize the performance of Vabis, a rule-based ABR algorithm is proposed for the weak network.

Frame-based live streaming process: Fig. 3 demonstrates the end-to-end process of live streaming based on frame. First, the video sender pushes video streaming to the streaming media server. Second, the player integrated into the client requests the video frame N from the streaming media server frame by frame. Only in every decision point (such as 0.5 s), the player sends the client's state information to streaming media server. Where the client's state information is calculated in the client player, including the average of throughput, current buffer occupancy, throughput trends, and so on. At the same time, the ABR algorithm integrated into the streaming media server adaptively select the bitrate of the future frame and the `target_buffer` by client and server state information that can be observed. Third, the server responds to the client's request and sends the frame N with the latest bitrate and `target_buffer` to the player. Finally, the player requests the video frame one by one and plays it frame by frame according to three kinds of delay control mechanisms.

Its characteristic is that the frame can be downloaded as long as there are frames in the server playback buffer and frame can be played as long as there are frames in the client playback buffer. This can greatly reduce the waiting delay of the frame.

Delay control mechanisms: In Fig. 1, the `target_buffer` (T) can be constructed as a triple $[S, T, Q]$, where S represents the slow playback threshold ($S = \alpha T$), and Q represents the fast

playback threshold ($Q = \beta T$). The client's player controller performs the following three mechanisms according to the current playback buffer occupancy and triple: 1) When the occupancy of the playback buffer is greater than Q , the player performs fast playback that is the video of 1 s plays μs . 2) When the occupancy of the playback buffer is less than S , the player performs slow playback that is the video of 1 s plays νs . 3) The system default frame is downloaded in order. A frame skipping event occurs when the system end-to-end delay exceeds ϕs . That is the video frame requested to the server is no longer the next frame, but the latest frame with a delay of about φs .

ABR algorithm: ABR algorithm consists of three modules: weak network discriminator, RL-based ABR module (Vabis_RL) and rule-based ABR module (Vabis_rule). Weak network discriminator summarizes the unique characteristics of the weak network according to the current buffer information and historical network throughput records in client and server. Its goal is to identify the weak network. Vabis_RL is a Few-Wait ABR algorithm, which can train a neural network by network trace without weak network and realize adaptive bitrate selection for future frames in complex scenarios caused by variable network conditions and diverse video content. Because the bitrate selection of weak network is simpler and weak network will make RL-based ABR algorithm more conservative during training, we build Vabis_rule for the weak network, which is based on the preset rules of fine-tuning heuristic form to achieve bitrate selection.

Characteristics: Vabis is based on the HTTP adaptive streaming (HAS [16], [17]) framework and MPEG Common Media Application Format (CMAF [18]) standard for video delivery. The basic unit of HAS transmission is a chunk, which can significantly reduce the complexity of the operation. CMAF standard uses an HTTP chunked transfer encoding solution that long chunks can be generated and delivered in small pieces so that the encoder can output each small chunk for delivery immediately after encoding it. However, since the bitrate switching can only be performed at the chunk boundary, the bitrate switching period is at least one chunk length. Vabis combines the advantages of both, not only using frames as the basic unit of transmission but also solving the I-frame misalignment problem so that the period of bitrate switching is independent of the length of the chunk.

Vabis focus on server-side [19], [20] adaptation solutions. The reason is that the ABR algorithm in the live streaming scenarios not only needs to consider the state information of the client but also needs to consider the state information of the video sender. Deploying the ABR algorithm on the streaming media server can not only achieve the bitrate selection according to the global information but also facilitate the updating and optimization of the algorithm.

The latency considered by Vabis is the controllable delay between the client player and the streaming media server. The reason is that the network topology between the video sender and the server is intricate and its state information is difficult to predict and control. In order to optimize the user experience, we minimize the latency between the client player and the streaming media server.

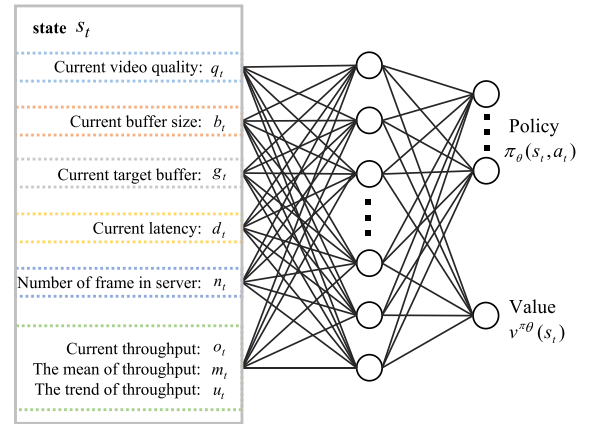


Fig. 4. Vabis_RL architecture overview.

III. VABIS_RL FOR NORMAL NETWORK

In this section, our goal is that the Vabis_RL can automatically learn the video bitrate selection policy based on the state information that can be observed and can solve the problem of I-frame misalignment by the algorithm. So we used the RL-based method to construct the ABR algorithm (Vabis_RL), as shown in Fig. 4.

A. Vabis_RL Model

Vabis_RL trains a neural network, which can select the bitrate of future frames according to the observed state information in complex scenarios caused by variable network conditions and diverse video content. The design details of the Vabis_RL are shown below:

RL Framework: Many reinforcement Learning Frameworks can be used to train learning agents, such as A2C [21], TRPO [22], PPO [23], DQN [24], ACKTR [25], etc. Through comparison, we finally choose the ACKTR algorithm based on the AC framework as the algorithm framework of Vabis_RL. ACKTR guarantees the stability of the algorithm by TRPO and incorporates distributed Kronecker factorization that improves sample efficiency and scalability.

State: At each decision point t , the agent of Vabis_RL inputs the status information $s_t = (q_t, b_t, g_t, d_t, o_t, m_t, n_t, u_t)$ into the neural network, where q_t is the video quality at the current time; b_t represents the playback buffer occupancy of the player at the current time; g_t indicates the current target_buffer size; d_t means the estimated controllable latency between the client and the streaming media server at the current time, which can be calculated by the difference between the arrival time of the latest frame of the server and the current physical time; o_t indicates the estimated network throughput at the current time; m_t means the average value of the past 4 network throughput records; n_t represents the estimated number of frames in the server buffer, $n_t = (d_t - g_t) * frame_rate$, where $frame_rate$ represents the number of frames per second of video; u_t indicates the predicted probability of network throughput growth at the next moment. The calculation method is as follows: First, we define four modes for the trend of the adjacent four network throughput records:

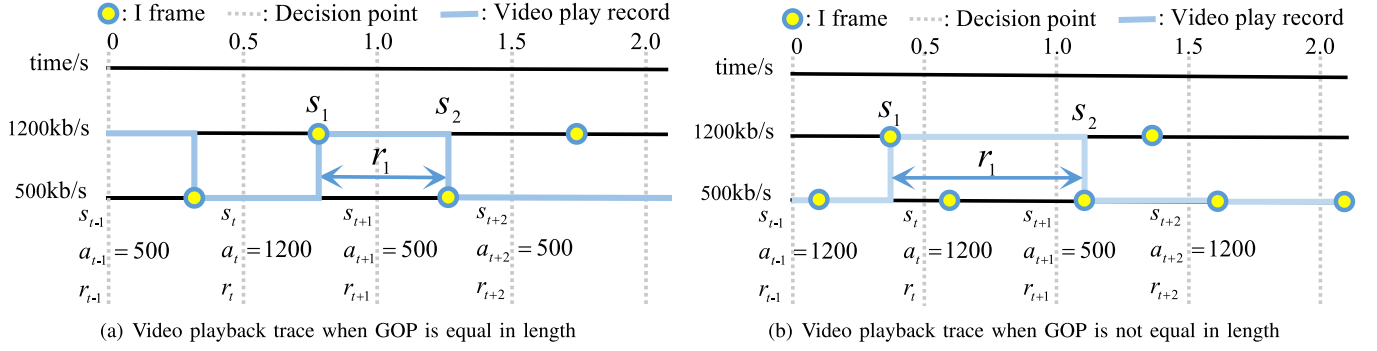


Fig. 5. Vabis_RL based Few-Wait ABR mechanism.

up_up_up, up_down_up, down_up_up, down_down_up. Then the probability of four modes appearing in the past 1000 historical network throughput records is calculated by a sliding window with the window size of 4. Finally, according to the trend of the past three network throughput records at the current time, predict the probability of network throughput rising at the next time.

Action: Action represents a decision made by an agent for future frames at each decision point based on the observed state information of the server and client. Unlike other ABR algorithms, the action of Vabis_RL algorithm is not a single video bitrate, but a combination of video bitrate and target_buffer.

Reward: There exists a significant variance in user preferences for video streaming quality of experience (QoE). In spite of various studies in the past few years have formed a variety of QoE metric [5]–[7], [26], they are all based on the chunk’s QoE metric. So building a QoE that adapts to the live streaming scenarios based on the frame will be crucial. The goal of Vabis is to achieve maximum QoE on the basis of ensuring ultra-low latency, including high quality, low rebuffering, high smoothness, and low latency. After rethinking the correspondence between structure based on chunk and structure based on the frame, according to the needs of the real live streaming scenarios, we finally construct the calculation formula of reward (QoE) based on QoE metric used by MPC [6] as:

$$QoE = QoE_1 + QoE_2 \quad (1)$$

$$QoE_1 = -\delta \sum_{n=1}^N |R_{n+1} - R_n| \quad (2)$$

$$QoE_2 = \sum_{n=1}^N \sum_{f=1}^N (\alpha R_{n,f} - \beta T_{n,f} - \gamma L_{n,f}) \quad (3)$$

Where R_n represents the bitrate of the n th decision-point; $R_{n,f}$ represents the bitrate of the f th frame between the n th decision points and the $(n + 1)$ th decision points. $T_{n,f}$ represents the rebuffering time that results from downloading the f th frame at bitrate $R_{n,f}$. If the client’s buffer is empty, then $T_{n,f}$ is equal to the download duration of the f th frame, otherwise $T_{n,f}$ is equal to 0. $L_{n,f}$ represents the latency that results from downloading the f th frame at bitrate $R_{n,f}$, which can be calculated by the difference between the arrival time of the latest frame of

the streaming media server and the current physical time. QoE_1 is used to penalize change in video quality to favor smoothness. Coefficient α represents the video length of each frame. Coefficient β , γ and δ represent the penalty weight of the rebuffering, latency and bitrate switching, respectively.

B. Vabis_RL Based Few-Wait ABR Mechanism

In order to achieve frame-based fine-grained control and ultra-low latency requirements, we do not use the transcoding server to solve the problem of I-frame misalignment but adopted a Few-Wait ABR mechanism to solve, where Few-Wait means that bitrate decision and bitrate execution can be synchronized as much as possible. The problems caused by I-frame misalignment are: 1) Bitrate decision and the bitrate execution are not synchronized; 2) The time difference between them is difficult to predict; 3) The decided bitrate is not optimal when the bitrate switching strategy is executed. For example, when the GOP (the distance between adjacent I-frame) is equal in length between different bitrate videos, as shown in Fig. 5(a), the agent selections an action a_t according to state s_t . Due to that the video bitrate will only switch at the I-frame position, the response made by the environment according to a_t is not $s_t \rightarrow s_{t+1}$ with r_{t+1} , but $s_1 \rightarrow s_2$ with r_1 . And RL performs parameter update based on $[s_1, a_t, r_1, s_2]$ instead of $[s_t, a_t, r_{t+1}, s_{t+1}]$. This will cause the problem of long-wait between action decision and action execution and the problem that the action does not match the reward during training. For example, when GOP is not equal in length between different bitrate videos, as shown in Fig. 5(b), at time t , the action may not act on the environment at all because it does not encounter an I-frame. This will cause the problem that the waiting time between action decision and action execution is unpredictable. Therefore, solving the I-frame misalignment problem will be a great challenge.

The traditional DRL model training process can be described as follows. At time t , first, the agent selects an action a_t based on the current state s_t of the environment. Then the environment responds according to the action and generates a new state s_{t+1} and reward r_{t+1} . Finally, model implements parameter updates to the neural network according to the quaternion $[s_t, a_t, r_{t+1}, s_{t+1}]$. During this time, action and reward are always synchronized. Few-Wait ABR mechanism designs a new method of decision

TABLE I
DECISION POINTS AND LEARNING POINTS DURING TRAINING

Logic	Timing of decision	Timing of learning
Logic_1	At each 0.5s boundary	At each 0.5s boundary
Logic_2	At each I-frame position	At each I-frame position
Logic_3	At each 0.5s boundary that I-frame within the previous 0.5s	I-frame position in the previous 0.5s
Logic_4	At each 0.5s boundary that I-frame within the next 0.5s	At each 0.5s boundary that I-frame within the next 0.5s

point selection and training logic structure to solve the I-frame misalignment problem. The details are as follows:

Decision point: Traditional chunk-based streaming media frameworks perform action selection at the GOP boundary. Since the GOP is a fixed length (4 s), the agent makes an action selection on average 4 s. To reduce the waiting time between bitrate decision and bitrate execution, we take action selection every N s. When N is equal to 0.5, it can achieve a great balance between fine-grained control and control costs.

Test logic: Perform an action selection every 0.5 s physical time. Only when the I-frame of the decided bitrate video in the next 0.5 s appears will the video bitrate be switched.

Training Logic: To achieve every parameter update of the Vabis_RL algorithm is correct and achieve the few-wait between action decision and action execution. We constructed four training logics in a heuristic way, as shown in Table I:

Logic_1: Although Logic_1 can ensure synchronization between training and testing stages, the biggest problem is that the reward acquired at the 0.5 s boundary is not necessarily the reward generated by the action made by the previous 0.5 s boundary. This will cause the Vabis_RL algorithm to perform multiple incorrect parameter updates, and thus will not be able to train the correct model at all.

Logic_2: Although Logic_2 can guarantee the $[s_t, a_t, r_{t+1}, s_{t+1}]$ matching of each parameter update in the training phase, the biggest problem is that the training and testing phases are not synchronized. This will result in poor performance of the trained model in the test phase.

Logic_3: Vabis_RL selects an action at each 0.5 s boundary that I-frame within the previous 0.5 s, and learns the state information that can be observed at the I-frame position in the previous 0.5 s. The decision time of action will be exactly one GOP different from the execution time of action. The neural network can learn and update parameters according to $[s_t, a_{t-1}, r_{t+1}, s_{t+1}]$ at each decision point. This can greatly guarantee that the action matches the reward when each parameter is updated and reduce the number of times an action will never act on the environment. Because the sampling period of network throughput is 0.5 s, the state information of the I-frame position is similar to that of the 0.5 s boundary. This can also better ensure that the training phase matches the testing phase.

Logic_4: This logic structure can not only match the training phase with the test logic but also achieve a large degree of synchronization between action and reward. But the biggest problem is that it is difficult to predict whether there will be I-frame in the next 0.5 s. This will also lead to multiple wrong parameter updates of the neural network. For example, even if

we can know that there are I-frames in the next 0.5 s video trace in the training stage, I-frame may not appear in the next 0.5 s physical time due to video rebuffering.

Compared with the four training logics, Logic_3 can better solve the problem of long-wait between action decision and action execution, and the problem that the action does not match the reward during training. The Logic_3 achieves the best results through the experimental comparison in Section V. Therefore, we finally choose Logic_3 as our training logic structure.

IV. VABIS_RULE FOR WEAK NETWORK

Since the Vabis_RL selects the bitrate of future frames by the strategy of maximizing the long-term cumulative reward, this will lead to that the higher the proportion of the weak network and the more conservative the training of the model will be. Assuming that the network is poor for a long period, the long-term cumulative reward of the model calculation takes into account the role of the weak network. However, for the case of better network conditions in the next period, the model cannot instantaneously improve the decision bitrate. Instead, the decision bitrate is gradually improved. In order to improve the performance of the Vabis algorithm, we design a rule-based ABR algorithm separately for the weak network (The average network throughput is less than 1 Mb/s). Especially in the Wi-Fi environment, during the peak period of network usage, the optimization of weak networks will be more valuable.

Weak network discriminator: By summarizing and analyzing a large number of real traces of weak network, we obtain two unique characteristics of the weak network: 1) the average network throughput. We find that the average network throughput of the weak network is generally smaller than that of other networks. We can characterize the weak network by taking the average of the first m historical network throughput records. Through experiments verified that it is best when m is 4. 2) The changing trend of network throughput. We find that the throughput variance of the weak network is also generally smaller and most of the weak network throughput will show a concussion mode. As shown in Fig. 6, we can find that the three adjacent network throughput records increases or decreases continuously with a lower probability. Accordingly, we calculate the probability that the adjacent three network throughput records are continuously rising or continuously decreasing in the past n historical network throughput records. We verified through experiments that it is best to characterize the weak network when n is 12.

Vabis_rule algorithm: In the case of the weak network, the choice of bitrate is relatively simple and it is more inclined to

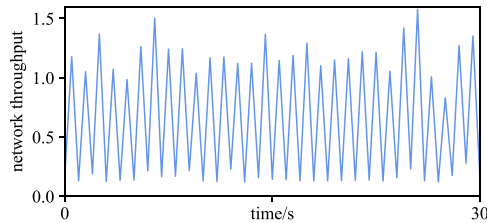


Fig. 6. Trace of network throughput change in the weak network.

Algorithm 1: Vabis_rule Algorithm**Input:** thr_record: historical network throughput records**Output:** T: target_buffer B: bitrate of future frames**if** get_tend(thr_record[-m:]) < α **then****if** mean(thr_record[-n:]) < β **then**

B=Rmin, T=0.2, return

end if**end if****if** get_tend(thr_record[-m:]) < γ **then****if** mean(thr_record[-n:]) < μ **then**

B=Rmin, T=0.2, return

end if**if** mean(thr_record[-n:]) < ν **then**

B=Rmin, T=0.35, return

end if**end if**

choose low bitrate. Therefore, high performance can be achieved by only using preset rules of fine-tuning heuristic form. The details of the Vabis_rule algorithm are as Algorithm 1. In Algorithm 1, thr_record is used to storage past network throughput records, and get_tend function is used to calculate the probability of continuous increase or decrease of network throughput in the past 12 historical network throughput records. The get_trend function is calculated in the same way as u_t in the state information of the Vabis_RL algorithm.

V. SYSTEM IMPLEMENTATION

In this section, we describe the specific implementation of the various components of the Vabis system.

First, we decide the best hyper-parameters of the Vabis_RL algorithm. Vabis_RL network structure contains two fully-connected layers, and the number of hidden cells per layer is 128. Results from these layers are then aggregated in a hidden layer that uses N (N is equal to the number of action) neurons to apply the softmax function. The critic network uses the same NN structure, but its final output is a linear neuron (without activation function). During training, we use the cumulative discount factor $\gamma = 0.99$, which realizes that the current action will be affected by the future four-time steps. The learning rate of the actor-network is 0.0001 and the learning rate of the critic network is 0.001. The number of actions is $n * m$, where n represents the number of bitrates and m represents the discrete number of target_buffer. Through experimental comparison, as

TABLE II
TARGET_BUFFER PERFORMANCE COMPARISON

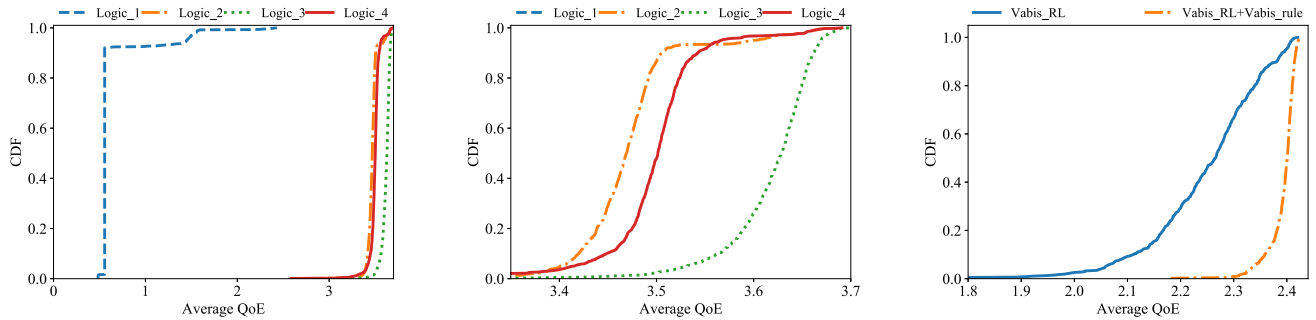
target_buffer	Average QoE
[0.55, 0.80, 1.2, 1.8, 2.5]	157.4916
[0.35, 0.80, 1.1, 1.6, 2.0]	161.9409
[0.00, 0.35, 0.8, 1.6, 2.0]	162.4379
[0.09, 0.35, 0.8, 1.6, 2.0]	165.1032

shown in II, we finally discretize the target_buffer to 5-tuple [0.09, 0.35, 0.8, 1.6, 2.0]. The reward metric has 4 hyperparameters, where α represents the video length of each frame, calculated as $\alpha = 1/frame_rate$. Because cooperative enterprise builds an outstanding set of evaluation indicators through years of user feedback, which can more accurately reflect the user's preference in the live streaming scenarios. Based on the evaluation indicators of cooperative enterprise in the live streaming scenarios, We set the hyper-parameter of QoE metric as $\beta = 1.5$, $\gamma = 0.005$, $\delta = 0.02$.

To generate Vabis_rule algorithm, all the best hyper-parameters are obtained according to the statistical characteristics of the real network traces. When $\alpha = 0.3$, $\beta = 0.38$, $\gamma = 0.15$, $\mu = 0.64$, $\nu = 0.80$, We have 95% confidence that the network trace of this segment is a weak network trace. Then, we decide the best hyper-parameters of the delay control mechanisms by grid search method. The results is $\alpha = 0.5$, $\beta = 2.0$, $\mu = 0.95$, $\nu = 1.05$, $\phi = 7$, $\varphi = 3$.

Vabis_RL vs. Vabis_RL + Vabis_rule: We perform a performance comparison test between the Vabis system with Vabis_rule algorithm and the Vabis system without Vabis_rule algorithm. We use a hybrid network trace where the ratio of the weak network is 10%. The experimental results are shown in Fig. 7(c). We can find that the Vabis using Vabis_rule algorithm is significantly higher than Vabis using only the Vabis_RL algorithm in terms of QoE. Especially in the Wi-Fi environment, during the peak period of network usage, Vabis_rule performance optimization will be more obvious. Therefore, we can confirm through experiments that the performance of the Vabis system can be improved by the Vabis_rule algorithm.

Training logic comparison: First, we compare the four training logic structures in Table I. The experimental results are shown in Fig. 7(a) and Fig. 7(b). We can clearly see that the training effect of Logic_1 is the worst, while the training effect of Logic_3 is the best. This is because Logic_1 completely violates the training mode of reinforcement learning. Most of the rewards do not match the action. The problem of Logic_2 is that the training and testing phases are not synchronized. This will result in poor performance of the trained model in the test phase. Because the location of the next I-frame is difficult to predict, Logic_4 also has many mismatches between action and reward. Logic_3 can better solve the problem of long-wait between action decision and action execution and the problem that the action does not match the reward during training. Through the experimental results, Logic_3 is an optimal Few-Wait ABR mechanism.



(a) Comparison of four training logic performance (b) Comparison of four training logic performance (c) Performance optimization of Vabis

Fig. 7. Vabis implementation.

VI. SYSTEM EVALUATION

A. Simulator and Datasets

Simulator: Based on the Vabis system, we implement a public live simulator (Vabis simulator). It simulates the interaction between the client video player and the streaming media server for last-mile delivery. It not only implements the process of client player requesting frames and receiving in frames, but also implements the process of client decoder decoding and playing frames in the client buffer. Between the client and the streaming media server, it uses mahimahi's network emulation tools [27] to emulate many different link conditions (different Round-Trip Time and network bandwidth) by taking the real network trace as inputs. Vabis simulator takes the video trace and the output of Mahimahi as inputs to simulate the dynamics of video sources and the last-hop network bandwidth fluctuation. The bitrate and target_buffer are decided by the Vabis algorithm. Vabis simulator takes the outputs of the Vabis algorithm as the input to simulate server decisions. Vabis simulator takes the environmental status information of the client and the streaming media server as outputs to simulate environmental status changes per download of one frame.

Frame traces: The video dataset used in this paper has two main sources. Part one, the competition dataset (frame_trace_1) of the global intelligent network transmission competition [28]. frame_trace_1 contains video data of three application scenarios, namely: game live streaming, indoor live streaming, and sports live streaming. Each video dataset includes four types of bitrate video data, which are [500 kb/s, 850 kb/s, 1200 kb/s, 1850 kb/s]. Part two, dataset (frame_trace_2) is collected from streaming media servers deployed by the cooperative company in Japan. frame_trace_2 is acquired by collecting one hour a day for one month continuously. The dataset includes two types of bitrate video data, which are [500 kb/s, 1200 kb/s]. Each video trace contains three kinds of information: the time when the frame arrives at the streaming media server, the size of the frame, and whether the next frame is an I-frame.

Network traces: In order to verify that Vabis can obtain optimal QoE in a wide range of network environments, it is necessary to construct a large-scale network trajectory dataset containing different network conditions for training and testing. Therefore, we consider building the network trace dataset in the following three ways.

Construct a synthetic network trace dataset (network_trace_1). We generated 1000 weak network traces, 1000 medium network traces and 1000 strong network traces through the script provided by the global intelligent network transmission competition [28]. Then, different network traces are sliced in units of 5 s. Finally, three kinds of network traces are fused by random methods and 1000 mixed network traces are obtained. The collection unit of each network throughput record is 0.5 s.

Collect a real network trace dataset (network_trace_2). By using the real live streaming platform built by the cooperative company, we collected 1000 real network traces in WiFi and 4G scenarios. The collection unit of each network throughput record is 0.5 s.

Splice two public network trace datasets. We create a corpus of 500 network traces of 500 seconds by combining two public datasets, including the 3G/HSDPA mobile dataset collected from Norway [29] and the 4G/LTE mobile dataset collected from Belgium [30]. We create another corpus of 500 network traces of 500 seconds by Splicing FCC broadband dataset provided by the US Federal Communications Commission [31].

B. Baseline ABR Algorithms

We present performance comparisons between Vabis and five baseline algorithms, which are representative of the state-of-art in bitrate adaptation. To adapt to the frame-based streaming media transmission framework, the algorithms have been slightly modified.

- 1) BBA: Huang *et al.* proposed a buffer-based algorithm (BBA) [4], which uses a reservoir of 5 seconds and a cushion of 10 seconds. It attempts to select bitrates by linear regulation such that the buffer occupancy is always maintained above 5 seconds. The highest available bitrate is selected if the buffer occupancy exceeds 15 seconds.
- 2) Festive: Jiang *et al.* proposed a rate-based algorithm (Festive) [32], which predicts throughput using the harmonic mean of the throughput values obtained during the past 5 decision point. Then, it selects the highest available bitrate that is below the predicted throughput.
- 3) MPC: Yin *et al.* proposed a quality-based algorithm (MPC) [6], which selects bitrate to achieve the maximum

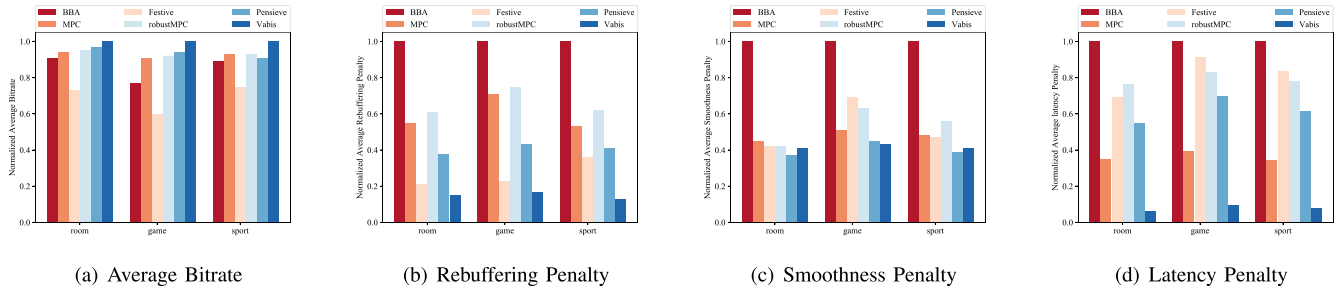


Fig. 8. Performance comparison between Vabis and existing ABR algorithms in different video scenarios in terms of average bitrate, rebuffering penalty, smoothness penalty and latency penalty by using `frame_trace_1` and `network_trace_1`.

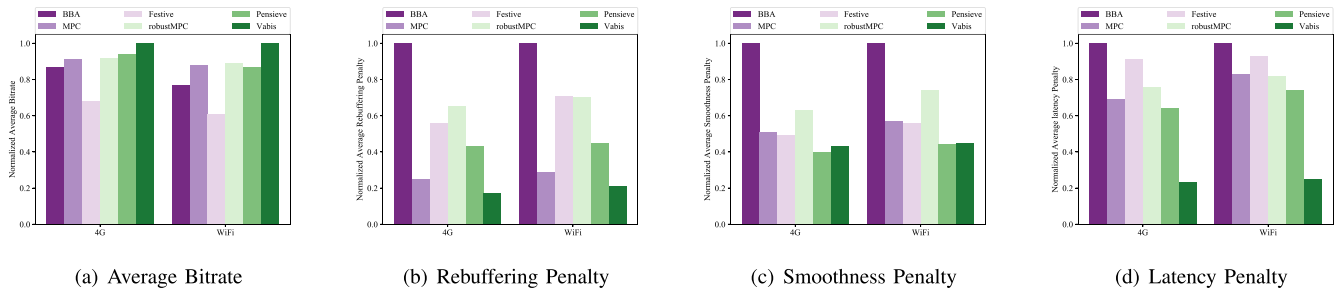


Fig. 9. Performance comparison between Vabis and existing ABR algorithms in different network scenarios in terms of average bitrate, rebuffering penalty, smoothness penalty and latency penalty by using `frame_trace_2` and `network_trace_2`.

of given QoE metric for future 5 horizons by buffer occupancy observations and throughput predictions (computed in the same way as Festive).

- 4) Robust-MPC: Using the same approach as MPC. It implements the optimization of the MPC algorithm by considering and accounting for errors observed between predicted and observed throughput values during the last 5 decision points.
- 5) Pensieve: Mao *et al.* proposed a RL-based algorithm (Pensieve) [7], which trains a neural network by optimizing the QoE metric and implements bitrate selection for future frames according to environmental state information that can be observed. Assume that the frame is continuously downloaded 2 s as a chunk. For the Vabis system, we retrain a neural network model by using the same configuration as the pensieve in the simulation environment.

C. Vabis Algorithm vs. Existing ABR Algorithms

In this subsection, in order to evaluate the Vabis algorithm, we compare it with existing state-of-the-art ABR algorithms in different video scenarios and network conditions in terms of each QoE metrics. Those outstanding ABR algorithms are listed in VI-B, which include BBA, Festive, MPC, Robust-MPC, and Pensieve. Those QoE metrics are listed in III-A, which include average QoE, bitrate utility, latency penalty, rebuffering penalty, and smoothness penalty. For comparison, we train all ABR algorithms in the Vabis system. In each experience, we continue to iterate the Vabis algorithm and Pensieve algorithm

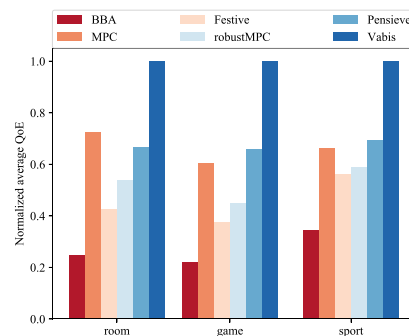


Fig. 10. Performance comparison between Vabis and existing ABR algorithms in different video scenarios in terms of QoE by using `frame_trace_1` and `network_trace_1`.

until we get the optimal model of the considered QoE metrics. For BBA, Festive, MPC, Robust-MPC algorithms, we adjust the parameters of each algorithm to get the optimal QoE metrics value. We use `frame_trace_1` and `network_trace_1` to compare the performance of various algorithms in different video scenarios. The experimental results are shown in Fig. 8 and Fig. 10. We use `frame_trace_2` and `network_trace_2` to compare the performance of various algorithms in different network scenarios. The experimental results are shown in Fig. 9 and Fig. 11.

We can find that Vabis performance is optimal in terms of QoE, video quality, rebuffering duration, switching frequency and latency, compared with other outstanding ABR algorithms

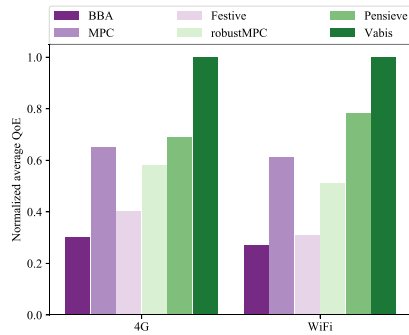


Fig. 11. Performance comparison between Vabis and existing ABR algorithms in different network scenarios in terms of QoE by using frame_trace_2 and network_trace_2.

whether in different video scenarios or different network scenarios. Especially in terms of latency and QoE, Vabis is significantly better than the existing methods with decreases in an average delay of 32%–77% and improvements in average QoE of 28%–67%. The main reasons are: 1) Vabis does not use chunk as the basic unit of decision making, but constructs a frame-based streaming media system to achieve frame-based fine-grained control. 2) Vabis adds three delay control mechanisms to achieve joint control between the client and the server. 3) Vabis adopts a rule-based and RL-based hybrid ABR algorithm for bitrate selection, which can optimize each other to achieve better performance. 4) Vabis does not solve the I-frame misalignment between different bitrate videos through the transcoding server but solves it through the Few-Wait ABR mechanism. In this way, Vabis can minimize the waiting delay of the frame and achieve the best QoE on the basis of ensuring ultra-low latency.

D. Generalization

In this subsection, to verify models learned by the Vabis algorithm has a strong generalization, we compare it with existing state-of-the-art ABR algorithms in different video scenarios and public network datasets in terms of average QoE. Those outstanding ABR algorithms are listed in VI-B, which include BBA, Festive, MPC, and Pensieve. We compare three types of video scenarios with the highest viewing frequency, which are room live streaming, game live streaming, and sports live streaming. We compared two types of commonly used network datasets, namely the 3G/4G wireless network [29], [30] and broadband network [31]. For comparison, we run all ABR algorithms in the Vabis system. The experimental results are provided in the form of full CDFs for all situations.

As shown in Fig. 12 and Fig. 13, Vabis algorithm can maintain high levels of performance for each video scenarios and network conditions considered. We can find that the average QoE performance of the Vabis algorithm in sports live streaming is optimal in different video scenarios. This is because, for large-scale sports events, the live streaming platform will allocate special lines and large bandwidth for video transmission, which will minimize the latency of video transmission and increase the quality of video transmission.

We can find that the average QoE performance of the Vabis algorithm in the broadband network is optimal in different network conditions. This is because the broadband network is relatively stable and the variance of network throughput variation is small. Therefore, network throughput prediction is more accurate. This will achieve more optimal bitrate decision making and get the best user experience. Conversely, in wireless networks with large changes in network throughput, network throughput prediction will be more inaccurate. To avoid rebuffering and decrease latency, the ABR algorithm often chooses a relatively conservative video bitrate instead of the optimal video bitrate.

E. Real World Evaluation

In this section, we establish a real live platform with the cooperative company to experimentally evaluate Vabis. First, the video sender pushes the live video streaming to the streaming media server through the wireless network and the client sends a video request to the streaming media server through the wireless network. Then the streaming media server calls Vabis to get the video bitrate and target_buffer based on the current state information. Finally, the streaming media server responds to the client's request and sends video streaming. To ensure the controllability of the following comparative experiments, we deploy a network loss meter between the streaming media server and the client to control network changes. We collect video traces on the server-side for simulator testing. First, we test the performance of Vabis and the outstanding ABR algorithms in real-world scenarios. Then we use the same video traces and network traces to test in the simulator.

In Fig. 14, We can see that the test results of Vabis and the outstanding ABR algorithms are slightly better in the simulator than in the real environment, but the overall difference is not big. This is because the state information of the real environment has a time difference in the process of data transmission. But overall, the overall performance of Vabis is optimal, both in the simulator and in the real world. And our algorithm is compared with the cooperative company's existing algorithms, the effect is the best from some evaluation indicators and the user's intuitive experience. Users can intuitively feel that Vabis has lower latency and smoother playback than existing methods.

VII. RELATED WORK

With the development of 5G, the volume of HTTP-based live streaming traffic will grow rapidly and live streaming applications will be developed in various fields [1]. Live streaming can now be watched at any time, anywhere, and under any network environment. Compared with the video transmission in the previous on-demand scenarios, live streaming on the basis of high-quality and low-rebuffering, adding new requirements for real-time interaction [33]. In order to improve the user watching experience [3], [12], [34], balance the relationship between high quality, low rebuffering, high smoothness, and low latency [35], many outstanding ABR algorithms and streaming media frameworks have been proposed to solve this problem.

ABR algorithm: The existing ABR algorithms can be mainly divided into the following four classes:

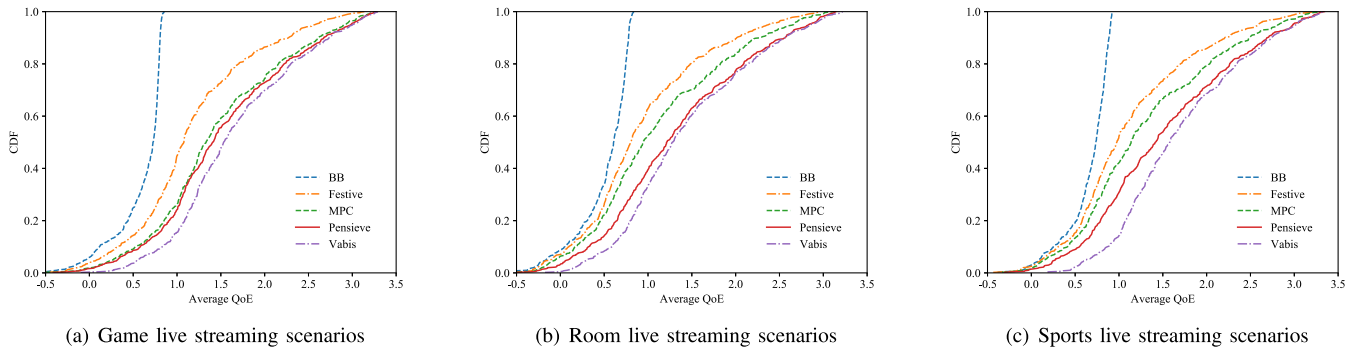


Fig. 12. Performance comparison between Vabis algorithm and existing outstanding ABR algorithms in different video scenarios in terms of QoE metrics. Result were collected on the 3G/HSDPA and 4G/LTE hybrid network datasets.

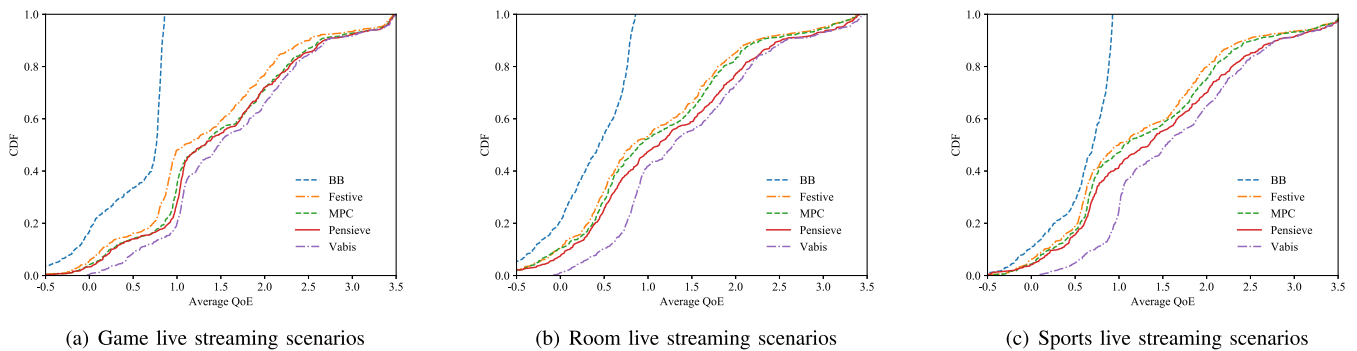


Fig. 13. Performance comparison between Vabis algorithm and existing outstanding ABR algorithms in different video scenarios in terms of QoE metrics. Result were collected on the FCC broadband network datasets.

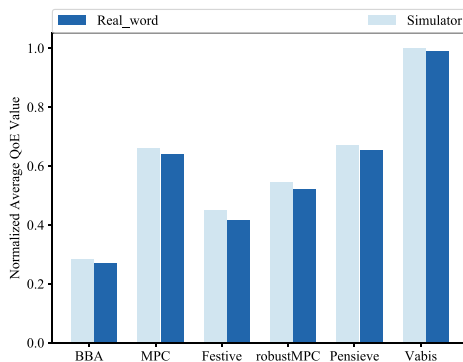


Fig. 14. Performance comparison between Vabis and existing ABR algorithms in real live platform and simulator in terms of QoE.

Buffer_based: The video bitrate requested by the client to the server at the next moment is determined only according to the current playback buffer occupancy. The goal of these algorithms is to keep the playback buffer occupancy within a suitable range that balances rebuffering and video quality. For example, the BBA algorithm [4] proposed by Huang *et al.* is a typical buffer-based ABR algorithm. It selects the bitrate by keeping the playback buffer occupancy above 5 s. And when the playback buffer occupancy exceeds 15 s, the highest bitrate is automatically selected. This type of algorithm also includes BOLA [26].

Rate_based: Estimate the network throughput available at the next moment according to the historical records of network throughput and determine the video bitrate that the client requests from the server at the next moment. For example, the FESTIVE algorithm [32] proposed by Jiang *et al.* The network throughput at the next moment is predicted by calculating the average network throughput during the past 5 chunks. The highest bitrate less than the predicted throughput is taken as the requesting bitrate of the client at the next moment. This type of algorithm includes CS2P [5], pandas [36], Squad [37], PSNR [38].

Quality_based: Use the historical records of network throughput and the occupancy of the current playback buffer to jointly predict the video bitrate that the client needs to request from the server at the next moment. For example, the MPC algorithm [6] proposed by Yin *et al.* According to the current buffer occupancy and network throughput prediction, the bitrate selection at the next moment is performed to maximize the QoE metric. This type of algorithm includes BOLA-E [39], ELASTIC [40], ABMA + [41];

RL_based: Use the reinforcement learning algorithm [21], [25] to train a neural network according to the environment state that can be collected to realize the bitrate selection of future chunk in different network states. For example, the pensieve algorithm [7], [42] proposed by Mao *et al.* It can adjust the parameters adaptively according to different inputs to get the

optimal bitrate of future frames. This type of algorithm includes QARC [15].

Since the first three methods are adaptive bitrate algorithms based on fixed rules. They only build simple and inaccurate models based on fixed rules in the current environment. Therefore, these methods are inevitably unable to achieve optimal performance for complicated and various network conditions [43], [44]. The RL-based method can automatically adjust its parameters according to its input to obtain the optimal bitrate in complicated scenarios caused by variable network conditions [45]–[47] and diverse video content. However, because the existing RL-based approach does not consider latency requirements, it is not suitable for live streaming scenarios that are more complicated and require extremely low latency.

Streaming media framework: HTTP adaptive streaming (HAS) has become the common framework to offer viewers a great quality of experience, eg. MPEG's Dynamic Adaptive Streaming over HTTP (DASH)[16] and Apple's HTTP Live Streaming (HLS)[17], the feature of HAS is that the client can only request the most recent full chunk in the server buffer and can only decode the most recent full chunk in the client buffer. It can significantly reduce the complexity of the operation, but it will result in the end-to-end latency longer than two-chunks duration, which is intolerable in ultra-low-latency live video transmission.

For ultra-low latency live streaming scenarios, where the duration from the capture to rendering is expected to be five seconds or less, two improved transmission methods based on HAS framework have been proposed to reduce latency. One way is that a chunk of long duration is divided into chunks of short duration by inserting some Intra-coded frames (I-frames)[48], which can effectively reduce the video transmission latency by reducing the waiting time of the full chunk downloaded. Since I-frame tend to be larger than the non-I-frame, this approach would significantly reduce the efficiency of the encoding servers and increase transmission bandwidth overhead. Another way is to use MPEG Common Media Application Format (CMAF)[18] standard, an HTTP chunked transfer encoding solution [49], [50]. Long chunks can be generated and delivered in small pieces. The advantage of breaking up the chunk into pieces is that the encoder can output each piece for delivery immediately after encoding it. This will lead to a direct reduction in overall latency by the same amount.

However, since the bitrate switching can only be performed at the chunk boundary, the bitrate switching period is at least one chunk length. Vabis combines the advantages of both, not only using frames as the basic unit of transmission but also solving the I-frame misalignment problem so that the period of bitrate switching is independent of the length of the chunk.

VIII. CONCLUSION

In this paper, we propose Vabis, a video adaptation bitrate system in units of the frame for time-critical live streaming. Through delay control mechanisms, ABR algorithm based on RL and fixed rule hybrid, and Few-Wait ABR mechanism, Vabis

achieve the goal of obtaining optimal QoE on the basis of ensuring low latency. We compare Vabis with the existing outstanding adaptive bitrate methods. Vabis is significantly better than the existing methods with decreases in an average delay of 32%–77% and improvements in average QoE of 28%–67%. In the future work, we hope that this algorithm can be deployed in large-scale network topology, and adaptively construct a unique video adaptive bitrate system according to the environment state of different regions. And it is not uncommon that live streaming systems may encounter flash crowd issues [51]–[53]. We hope that our system can react to such challenging scenarios.

REFERENCES

- [1] Cisco, "Cisco visual networking index: Forecast and methodology, 2017–2022," *White Paper Cisco Systems Inc.*, 2018.
- [2] S. Akhshabi, A. C. Begen, and C. Dovrolis, "An experimental evaluation of rate-adaptation algorithms in adaptive streaming over http," in *Proc. ACM Conf. Multimedia Syst.*, 2011.
- [3] Q. He, J. Liu, C. Wang, and B. Li, "Coping with heterogeneous video contributors and viewers in crowdsourced live streaming: A cloud-based approach," *IEEE Trans. Multimedia*, vol. 18, no. 5, pp. 916–928, May 2016.
- [4] T.-Y. Huang, R. Johari, N. McKeown, M. Trunnell, and M. Watson, "A buffer-based approach to rate adaptation: Evidence from a large video streaming service," *SIGCOMM Comput. Commun. Rev.*, vol. 44, no. 4, pp. 187–198, Aug. 2014. [Online]. Available: <http://doi.acm.org/10.1145/2740070.2626296>
- [5] Y. Sun *et al.*, "Cs2p: Improving video bitrate selection and adaptation with data-driven throughput prediction," in *Proc. ACM SIGCOMM Conf.*, 2016, pp. 272–285. [Online]. Available: <http://doi.acm.org/10.1145/2934872.2934898>
- [6] X. Yin, A. Jindal, V. Sekar, and B. Sinopoli, "A control-theoretic approach for dynamic adaptive video streaming over http," *SIGCOMM Comput. Commun. Rev.*, vol. 45, no. 4, pp. 325–338, Aug. 2015. [Online]. Available: <http://doi.acm.org/10.1145/2829988.2787486>
- [7] H. Mao, R. Netravali, and M. Alizadeh, "Neural adaptive video streaming with pensieve," in *Proc. Conf. ACM Special Interest Group Data Commun.*, 2017, pp. 197–210. [Online]. Available: <http://doi.acm.org/10.1145/3098822.3098843>
- [8] Y. J. Park *et al.*, "Childar-bot: Educational playing projection-based arbot for children," in *Proc. 26th ACM Int. Conf. Multimedia*, 2018, pp. 1278–1282. [Online]. Available: <http://doi.acm.org/10.1145/3240508.3241362>
- [9] T. Kämäräinen, M. Siekinen, J. Eerikäinen, and A. Ylä-Jääski, "Cloudvr: Cloud accelerated interactive mobile virtual reality," in *Proc. 26th ACM Int. Conf. Multimedia*, 2018, pp. 1181–1189. [Online]. Available: <http://doi.acm.org/10.1145/3240508.3240620>
- [10] X. Ma, C. Zhang, J. Liu, R. Shea, and D. Fu, "Live broadcast with community interactions: Bottlenecks and optimizations," *IEEE Trans. Multimedia*, vol. 19, no. 6, pp. 1184–1194, Jun. 2017.
- [11] S. S. Krishnan and R. K. Sitaraman, "Video stream quality impacts viewer behavior: Inferring causality using quasi-experimental designs," *IEEE/ACM Trans. Netw.*, vol. 21, no. 6, pp. 2001–2014, Dec. 2013.
- [12] F. Dobrian *et al.*, "Understanding the impact of video quality on user engagement," *Commun. ACM*, vol. 56, no. 3, pp. 91–99, Mar. 2013. [Online]. Available: <http://doi.acm.org/10.1145/2428556.2428577>
- [13] W. Zhang, B. Han, and P. Hui, "Jaguar: Low latency mobile augmented reality with flexible tracking," in *Proc. 26th ACM Int. Conf. Multimedia*, 2018, pp. 355–363. [Online]. Available: <http://doi.acm.org/10.1145/3240508.3240561>
- [14] X. Wei, J. Zhu, S. Feng, and H. Su, "Video-to-video translation with global temporal consistency," in *Proc. 26th ACM Int. Conf. Multimedia*, 2018, pp. 18–25. [Online]. Available: <http://doi.acm.org/10.1145/3240508.3240708>
- [15] T. Huang, R.-X. Zhang, C. Zhou, and L. Sun, "Qarc: Video quality aware rate control for real-time video streaming based on deep reinforcement learning," in *Proc. 26th ACM Int. Conf. Multimedia*, 2018, pp. 1208–1216. [Online]. Available: <http://doi.acm.org/10.1145/3240508.3240545>
- [16] Akamai, "dash.js," 2016. [Online]. Available: <https://github.com/Dash-Industry-Forum/dash.js/>

- [17] “Apple http live streaming.” 2014. [Online]. Available: <https://developer.apple.com/resources/http-streaming/>
- [18] *Information Technology–Multimedia Application Format (MPEG-A)—Part 19: Common Media Application Format (CMAF) for Segmented Media*, ISO/IEC Standard 23000-19:2018, 2017.
- [19] L. D. Cicco, S. Mascolo, and V. Palmisano, “Feedback control for adaptive live video streaming,” in *Proc. ACM Sigmconf. Multimedia Syst.*, 2011.
- [20] S. Akhshabi, L. Anantkrishnan, C. Dovrolis, and A. C. Begen, “Server-based traffic shaping for stabilizing oscillating adaptive streaming players,” in *Proc. Acm Workshop Netw. Operating Syst. Support Digit. Audio Video*, 2013.
- [21] V. Mnih, A. P. Badia, M. Mirza, A. Graves, and K. Kavukcuoglu, “Asynchronous methods for deep reinforcement learning,” 2016.
- [22] J. Schulman, S. Levine, P. Moritz, M. I. Jordan, and P. Abbeel, “Trust region policy optimization,” *Comput. Sci.*, pp. 1889–1897, 2015.
- [23] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, “Proximal policy optimization algorithms,” 2017.
- [24] M. Volodymyr *et al.*, “Human-level control through deep reinforcement learning,” *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.
- [25] Y. Wu, E. Mansimov, S. Liao, R. Grosse, and J. Ba, “Scalable trust-region method for deep reinforcement learning using kronecker-factored approximation,” 2017.
- [26] K. Spiteri, R. Uргаonkar, and R. K. Sitaraman, “BOLA: Near-optimal bitrate adaptation for online videos,” 2016, *arXiv:1601.06748*. [Online]. Available: <http://arxiv.org/abs/1601.06748>
- [27] R. Netravali *et al.*, “Mahimahi: Accurate record-and-replay for HTTP,” in *Proc. USENIX Annu. Tech. Conf.*, 2015, pp. 417–429.
- [28] T. University, “The global intelligent network transmission competition,” 2018. [Online]. Available: <https://www.aitrans.online/>
- [29] H. Riiser *et al.*, “Commute path bandwidth traces from 3G networks: Analysis and applications,” in *Proc. 4th ACM Multimedia Syst. Conf.*, 2013.
- [30] J. van der Hooft *et al.*, “HTTP/2-based adaptive streaming of HEVC video over 4G/LTE networks,” *IEEE Commun. Lett.*, vol. 20, no. 11, pp. 2177–2180, Nov. 2016.
- [31] F. C. Commission, “Raw data—Measuring broadband America,” 2016. [Online]. Available: <https://www.fcc.gov/reports-research/reports/>
- [32] J. Jiang, V. Sekar, and H. Zhang, “Improving fairness, efficiency, and stability in http-based adaptive video streaming with festive,” in *Proc. 8th Int. Conf. Emerg. Netw. Exp. Technol.*, 2012, pp. 97–108. [Online]. Available: <http://doi.acm.org/10.1145/2413176.2413189>
- [33] H. Pang *et al.*, “Optimizing personalized interaction experience in crowd-interactive livecast: A cloud-edge approach,” in *Proc. 26th ACM Int. Conf. Multimedia*, 2018, pp. 1217–1225. [Online]. Available: <http://doi.acm.org/10.1145/3240508.3240642>
- [34] K. Yamagishi and T. Hayashi, “Parametric quality-estimation model for adaptive-bitrate-streaming services,” *IEEE Trans. Multimedia*, vol. 19, no. 7, pp. 1545–1557, Jul. 2017.
- [35] J. De Praeter, G. Van Wallendael, J. Slowack, and P. Lambert, “Video encoder architecture for low-delay live-streaming events,” *IEEE Trans. Multimedia*, vol. 19, no. 10, pp. 2252–2266, Oct. 2017.
- [36] Z. Li *et al.*, “Probe and adapt: Rate adaptation for HTTP video streaming at scale,” 2013, *arXiv:1305.0510*. [Online]. Available: <http://arxiv.org/abs/1305.0510>
- [37] J. V. D. Hooft *et al.*, “HTTP/2-based adaptive streaming of HEVC video over 4G/LTE networks,” *IEEE Commun. Lett.*, vol. 20, no. 11, pp. 2177–2180, Nov. 2016.
- [38] S. Meng, J. Sun, Y. Duan, and Z. Guo, “Adaptive video streaming with optimized bitstream extraction and PID-based quality control,” *IEEE Trans. Multimedia*, vol. 18, no. 6, pp. 1124–1137, Jun. 2016.
- [39] K. Spiteri, R. Sitaraman, and D. Sparacio, “From theory to practice: Improving bitrate adaptation in the dash reference player,” in *Proc. 9th ACM Multimedia Syst. Conf.*, 2018, pp. 123–137. [Online]. Available: <http://doi.acm.org/10.1145/3204949.3204953>
- [40] L. D. Cicco, V. Calderaro, V. Palmisano, and S. Mascolo, “Elastic: A client-side controller for dynamic adaptive streaming over HTTP (DASH),” in *Proc. Packet Video Workshop*, 2013.
- [41] A. Beben, P. Wiśniewski, J. M. Batalla, and P. Krawiec, “Abma+: Lightweight and efficient algorithm for http adaptive streaming,” in *Proc. 7th Int. Conf. Multimedia Syst.*, 2016, pp. 2:1–2:11. [Online]. Available: <http://doi.acm.org/10.1145/2910017.2910596>
- [42] S. Sengupta, N. Ganguly, S. Chakraborty, and P. De, “Hotdash: Hotspot aware adaptive video streaming using deep reinforcement learning,” in *Proc. IEEE 26th Int. Conf. Netw. Protocols*, Sep. 2018, pp. 165–175.
- [43] T. Wu, W. Dou, Q. Ni, S. Yu, and G. Chen, “Mobile live video streaming optimization via crowdsourcing brokerage,” *IEEE Trans. Multimedia*, vol. 19, no. 10, pp. 2267–2281, Oct. 2017.
- [44] G. Gao *et al.*, “Optimizing quality of experience for adaptive bitrate streaming via viewer interest inference,” *IEEE Trans. Multimedia*, vol. 20, no. 12, pp. 3399–3413, Dec. 2018.
- [45] T.-Y. Huang, N. Handigol, B. Heller, N. McKeown, and R. Johari, “Confused, timid, and unstable: Picking a video streaming rate is hard,” in *Proc. Internet Meas. Conf.*, 2012, pp. 225–238. [Online]. Available: <http://doi.acm.org/10.1145/2398776.2398800>
- [46] K. Liu and J. Y. B. Lee, “Achieving high throughput and low delay in mobile data networks by accurately predicting queue lengths,” in *Proc. 12th ACM Int. Conf. Comput. Frontiers*, 2015, pp. 20:1–20:8. [Online]. Available: <http://doi.acm.org/10.1145/2742854.2742875>
- [47] Y. Zaki, T. Pötsch, J. Chen, L. Subramanian, and C. Görg, “Adaptive congestion control for unpredictable cellular networks,” *SIGCOMM Comput. Commun. Rev.*, vol. 45, no. 4, pp. 509–522, Aug. 2015. [Online]. Available: <http://doi.acm.org/10.1145/2829988.2787498>
- [48] J. Van Der Hooft *et al.*, “An HTTP/2 push-based approach for low-latency live streaming with super-short segments,” *J. Netw. Syst. Manag.*, vol. 26, no. 1, pp. 51–78, 2018.
- [49] A. El Essaili, T. Lohmar, and M. Ibrahim, “Realization and evaluation of an end-to-end low latency live dash system,” in *Proc. IEEE Int. Symp. Broadband Multimedia Syst. Broadcast.*, 2018, pp. 1–5.
- [50] A. Bentalib, C. Timmerer, A. C. Begen, and R. Zimmermann, “Bandwidth prediction in low-latency chunked streaming,” in *Proc. 29th ACM Workshop Netw. Operating Syst. Support Digit. Audio Video*, 2019, pp. 7–13.
- [51] F. Liu, B. Li, L. Zhong, B. Li, H. Jin, and X. Liao, “Flash crowd in P2P live streaming systems: Fundamental characteristics and design implications,” *IEEE Trans. Parallel Distrib. Syst.*, vol. 23, no. 7, pp. 1227–1239, 2011.
- [52] B. Li, G. Y. Keung, S. Xie, F. Liu, Y. Sun, and H. Yin, “An empirical study of flash crowd dynamics in a P2P-based live video streaming system,” in *Proc. IEEE Global Telecommun. Conf.*, 2008, pp. 1–5.
- [53] Y. Hu, D. Niu, and Z. Li, “A geometric approach to server selection for interactive video streaming,” *IEEE Trans. Multimedia*, vol. 18, no. 5, pp. 840–851, May 2016.



Tongtong Feng received the B.E. degree in communication engineering from Nanjing University of Posts and Telecommunications, Nanjing, China, in 2017. He is currently working toward the Ph.D. degree in computer science and technology with the Institute of Network Technology, Beijing University of Posts and Telecommunications, Beijing, China. His current research interests include video transmission optimization, adaptive multimedia streaming, and traffic engineering.



Haifeng Sun received the Ph.D. degree from the Beijing University of Posts and Telecommunications, Beijing, China, in 2017. He is currently a Lecturer with the State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications. His research interests include broad aspects of AI, NLP, big data analysis, object detection, deep learning, and deep reinforcement learning.



Qi Qi received the Ph.D. degree from the Beijing University of Posts and Telecommunications, Beijing, China, in 2010. She is currently an Associate Professor with the State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications. She has authored or coauthored more than 30 papers in international journal, and is the recipient of two National Natural Science Foundations of China. Her research interests include edge computing, cloud computing, Internet of Things, ubiquitous services, deep learning, and deep reinforcement learning.



path transmission, overlay networks, and traffic engineering.

Jingyu Wang received the Ph.D. degree from the Beijing University of Posts and Telecommunications, Beijing, China, in 2008. He is currently a Professor with the State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications. He has authored or coauthored more than 50 papers in international journals and famous conferences, including the *IEEE Communication Magazine*, *IEEE SYSTEMS JOURNAL*, *CVPR*, *AAAI*, *EMNLP*, etc. His research interests include broad aspects of SDN, multimedia services, multi-



National Natural Science Foundation of China in 2005, and the specially invited Professor of the Yangtze River Scholar Award Program by the Ministry of Education in 2009. His main research interests include cloud computing, mobile intelligent networks, service network intelligent, networking architectures and protocols, and multimedia communication.

Jianxin Liao received the Ph.D. degree from the University of Electronics Science and Technology of China, Chengdu, China, in 1996. He is currently the Dean of the Network Intelligence Research Center and a Full Professor with the State Key laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications. He has authored or coauthored hundreds of research papers and several books. He has won a number of prizes in China for his research achievements, which include the Premiers Award of Distinguished Young Scientists from