QoE-aware 3D Video Streaming via Deep Reinforcement Learning in Software Defined Networking Enabled Mobile Edge Computing

Pan Zhou, *Senior Member, IEEE,* Yulai Xie, *Member, IEEE,* Ben Niu, *Member, IEEE,* Lingjun Pu, *Member, IEEE,* Zichuan Xu, *Member, IEEE,* Hao Jiang, *Member, IEEE,* Huawei Huang, *Member, IEEE*

Abstract—With the advancements of wireless network transmission technology, 2D video is hard to satisfy people's requirement for multimedia. Therefore, the high-definition 3D video that can bring a whole new viewing experience is starting to enter people's vision. However, when a tremendously large number of users play 3D video, it puts enormous computational pressure on the cloud server, which incurs high transmission latency. To release the tension, in this paper we consider a promising computing and networking architecture by incorporating Mobile Edge Computing (MEC) and Software-defined Networking (SDN) and propose a novel resource allocation model (RAM) to allocate resources and reduce delay. At the same time, we introduce the Quality of Experience (QoE) Model (QoEM), which uses information collected during 3D video playback to adaptively allocate the rate of future tiles. The model addresses the problem of assigning the best transmission speed to the block in the case of time-varying characteristic factors during transmission. We propose an Actor-Critic-based deep reinforcement learning algorithm for viewport prediction and QoE optimization, called QoE-AC. For the differential transmission in the playback phase, we use the LSTM network for bandwidth and viewport prediction, while combining the historical information of the blocks into the Actor-Critic network as observations. The network can be adaptively assigned the best transmission speed for future tiles based on observations to maximize QoE. Finally, the experimental results show that the actual performance of the model is much better than other existing 3D video network models. Under different QoE targets, our proposed system can be adapted to all situations and has a 10%-15% performance improvement.

Index Terms—3D video, deep reinforcement learning, centrally controlled network, SDN, MEC, QoE

1 Introduction

VIDEO streaming consumes the most of Internet traffic today. With improving of transmission technique, researchers began to focus on 3D video. It can bring a lot of different experience from 2D video, i.e., immersive experience and stereoscopic vision. Unlike the fixed viewing direction of traditional video, when watching a 3D video, users can change the position of the viewport at will, thereby bringing an interactive experience. However, the high resolution and extremely large bit rate requirements of 3D video have prevented their wide spread on the Internet. To solve these problems, new technologies are starting to come into view, which push the intensive computing tasks to network edges, such as edge computing [1] and edge caching [2].

Pan Zhou and Yulai Xie are with Hubei Engineering Research Center on Big Data Security, School of Cyber Science and Engineering, Huazhong University of Science & Technology, Wuhan, 430074, China (e-mail: {panzhou, ylxie}@hust.edu.cn). (Corresponding Author: Yulai Xie)

Ben Niu is with Institute of Information Engineering, Chinese Academy of Sciences, 100864, China. (e-mail: fylxie@hust.edu.cn).

Lingjun Pu is with the College of Computer Science, Nankai University, Tianjin 300071, China. (e-mail: niuben@iie.ac.cn).

Zichuan Xu is with School of Software, Dalian University of Technology, Dalian, 116024, China. (e-mail: z.xu@dlut.edu.cn).

Hao Jiang, School of Electronic Information, Wuhan University, Wuhan, 430072, China. (e-mail: jh@whu.edu.cn).

Huawei Huang is with the School of Data and Computer Science, Sun Yat-Sen University, 510006 Guangzhou, China (e-mail: huanghw28@mail.sysu.edu.cn).

The playback process of 3D video is to encode the compressed block to transmit the video, and use the decoder to generate the video on the Head Mounted Display (HMD). Based on the high resolution and high bit rate of 3D video, there are two main problems with 3D video playback. First of all, the transmission of video and composite video will consume much computing resources. In order to improve the video quality during transmission, it is necessary to optimally allocate the bandwidth as well as computing resources during transmission and improve the utilization of these resources. Secondly, the main influence on the Quality of Experience (QoE) during video playback is the video quality. Therefore, to ensure the quality of the video streaming, we adopt a tile-based video streaming approach [3] in this study. As opposed to full-view transmission [4], this approach is characterized by assigning different transmission speeds to blocks, which ensures video quality while reducing bandwidth consumption. However, tile-based approach faces many challenges, such as the problem of optimizing multiple QoE objectives and the impact of time-varying characteristic factors during transmission.

Moreover, the propagation and synthesis of video consume a lot of communication resources, so some researchers have suggested putting the rendering process of the video to the cloud server [5], [6]. However, this approach has negative consequences for users, such as buffering and lower video quality. To meet the demand for low latency and caching resources for 3D video, we believe a distributed

paradigm would be more conducive to video playback, such as Mobile Edge Computing (MEC) [7]. MEC is a network architecture that implements computing functions at the edge of a mobile network. Compared to the traditional remote cloud servers, MEC can provide storage devices and computing resources in close proximity to mobile device, thus to offer faster response time for content delivery. On the other hand, the advantage of Soft-defined Network (SDN) [8] is that the control plane is separated from the data plane, which helps to optimize the server's resource allocation, such as computing resources or bandwidth resources. Therefore, it is promising to leverage the collaboration between MEC and SDN to address the bandwidth and computing resources allocation problems in the network.

To address the challenges of a tile-based approach, we need to design a streaming strategy that maps features to QoE targets and then construct an optimization model to find the optimal rate allocation scheme. In other words, the model chooses the optimal allocation rate to maximize QoE reward with limited bandwidth resources. The Markov Decision-making Process (MDP) is an appropriate theory to address this issue. As we know, if our problem is well described as an MDP, then Reinforcement Learning (RL) may be a good theoretical framework to find solutions. Especially, RL [9] is appropriate to adaptively optimize the long-term QoE reward. However, a tile-based video streaming approach leads to a significant increase in block state space, and RL cannot handle such problems effectively. By further embracing the any nonlinear expression capability of deep neural networks, we then find that Deep Reinforcement Learning (DRL) can solve decision problems that are difficult for RL to solve, i.e., the problem of high-dimensional state space and action space settings [10]. For example, DRL has achieved great success in video streaming [11], [12]. But their algorithm can only achieve low QoE due to low bandwidth utilization and frequent bit rate switching. Therefore, we propose an algorithm based on the Actor-Critic algorithm [13] to adaptively adjust the tile allocation speed. The biggest advantage of the Acotr-Critic algorithm is that it does not rely on the dataset, while assigning the bitrate to tile simply by observing the results of past decisions.

To tackle these problems, our proposed solution stems from two novel models, i.e., the Resource Allocation Model (RAM) and Quality of Experience Model (QoEM), which is named as RAM-QoEM. RAM primarily addresses the resource requirements for video playback. Due to the huge resource consumption of 3D video, we explore the characteristics of MEC to offload computing tasks from mobile users to the edge servers and it also can cache videos on the edge servers. However, this approach is not very efficient in utilizing resources in the edge servers. So to solve this problem, we have introduced the SDN technology. SDN can be used to allocate bandwidth and computing resources across the network, so it can further improve the overall resource utilization. QoEM mainly employs a tile rate allocation model based on the Actor-Critic algorithm. By predicting bandwidth and viewport information, the Actor-Critic algorithm is used to adaptively optimize the allocation rate of video tiles and thus improve QoE. Our contributions are summarized as follows:

- To address the stringent resource requirements of video playback for 3D video streaming, we propose a RAM model that takes advantage of MEC and SDN to resolve the resource allocation problem. The MEC servers can provide the resources of computing and caching while reducing the time to respond to user requests. SDN can optimize the resource allocation of MEC servers.
- We propose the LSTM-based Actor-Critic algorithm, called QoE-AC, to optimize various QoE goals that can adapt to dynamic environments, such as future changes in bandwidth and viewports.
- Our proposed QoEM has better performance in predicting future bandwidth and viewport location. Not only does it solve the problem of how to efficiently allocate block transmissions in dynamic networks, but it also addresses the optimization of multiple QoE targets.
- We set up comparative experiments with multiple QoE targets to evaluate multiple panoramic video streaming systems under various network conditions. The experimental results show that the performance of our proposed system is on average 10%-20% higher than other traditional algorithms.

The rest of the paper is organized as follows. Section 2 mainly introduces some related work of our paper. Section 3 explains the details of the model, including how the MEC in RAM connects to the SDN and how to determine the target of QoE in QoEM. Section 4 presents the problem formulation and transformation. Section 5 introduce our DRL algorithm and how its gradient is updated. Section 6 discusses our experimental results. Section 7 summarizes the conclusions of this paper.

2 RELATED WORK

In this seciton, we review and analyze the related works from two aspects, i.e., the resource allocation problem via MEC and SDN and the deep reinforcement learning based approaches for optimizing QoE and 3D video streaming.

2.1 Resource Allocation Problem via MEC and SDN

RAM uses the cooperation of MEC and SDN to solve the resource allocation problem. MEC technology can provide cloud services resources for users within the base station, and it allows operators or applications to deploy new applications at the edge, such as cloud video acceleration, augmented reality, etc. MEC technology not only has the advantages of low latency and high-speed transmission speed, but also can provide real-time radio network and context information. Therefore, MEC technology has a wide range of applications in enhancing video quality. In [14], Van *et al.* proposed an MEC-based method for improving video quality that supports seamless conversion in the presence of bandwidth fluctuations. In [15], Luo *et al.* proposed to utilize MEC to jointly consider novel mechanisms for buffer dynamics, video quality adaptation, edge caching, and video transmission. In [16], Bilal *et al.* presented the potential and prospects of MEC in the field of video streaming, while presenting the relevant upfront work and facing the challenges. Younis et al. [17] optimized the QoE of dynamic adaptive video streams by coordinating the Distortion Rate

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TNSE.2020.3038998, IEEE Transactions on Network Science and Engineering

(DR) characteristics of the video with the MEC server. Yao *et al.* [18] proposed a heuristic method to solve the problem of cost-effective edge base station placement. The approximate optimal location of the edge base station can be determined under conditions that satisfy users' constraints. A disadvantage of their research is that most of them focus on how to reduce the delay at the edge, but they do not consider optimizing the resource allocation at the edge. Due to the number of request tasks that each edge needs to handle varies, some nodes may have redundant resources, so resource consolidation optimization of edges can effectively avoid resource waste. Efficient configuration of resources allows the edge to handle more tasks and it can also reduce response time to demand.

The advantage of SDN is that it can use network programming to simplify network management and perform efficient network configuration, thereby improving network performance and monitoring of data. Therefore, due to the separation of data and control planes, SDN switches can provide fast data transmission functions. Lei *et al.* [19] proposed an efficient SDN-based caching mechanism that defines nodes in this network with different service requirements levels. However, these demand levels need to be set in advance, and it cannot effectively adapt to the network with dynamically changing resources.

Tootoonchian *et al.* [20] designed a centralized SDN control platform for Openflow, called HyperFlow. It hands the decision to a single controller, which minimizes the time that the control plane responds to data plane requests. Although they use a centralized SDN to reduce the response time to requirements, a single controller cannot handle complex requirements of tasks, such as caching resources and requests for computing resources during the playback of 3D video. Therefore, the response time needs to be adjusted in accordance with the usage of server resources.

2.2 Deep Reinforcement Learning for Optimizing QoE and 3D Video Streaming

The tile rate adaptation scheme based on Markov Decision Process (MDP) can effectively play a role in the traditional streaming system [21]. The general relationship between RL and MDP is that RL is a theoretical framework for solving problems that can be modeled as MDPs. By taking advantage of any nonlinear expression capability of deep neural networks, DRL agents can learn to make optimal decisions that adapt to changing environments. The work [12] proposed a RL-based Pensive model that selects bitrates for future video blocks based on observations collected by a client video player, and it does not rely on pre-programmed models or assumptions about the environment. However, the lack of predictions about bandwidth makes the network less adaptable to dynamic environments. Gadaleta [11] implemented a video streaming framework based on DQN, the framework combines feed-forward and recurrent deep learning networks with advanced policies to optimize QoE. Xie *et al.* [22] proposed a system based on cross-user learning (CLS), which uses the user's viewing interest area data to predict the position of the viewport and improve the accuracy of the viewport prediction. At the same time, a rate allocation scheme driven by QoE under bandwidth constraints is proposed. Yin *et al.* [23] proposed a novel model predictive control algorithm, which can best combine the throughput and buffer occupation information. In addition, they develop a control-theoretic model to calculate about various strategies in order to balance different QoE goals. The optimization effects of the above several algorithms have room for improvement, and they cannot effectively adapt to different QoE goals. At the same time, part of their algorithm requires the control of theoretical constraints or a high dependence on the data set, so it cannot adapt to dynamic networks.

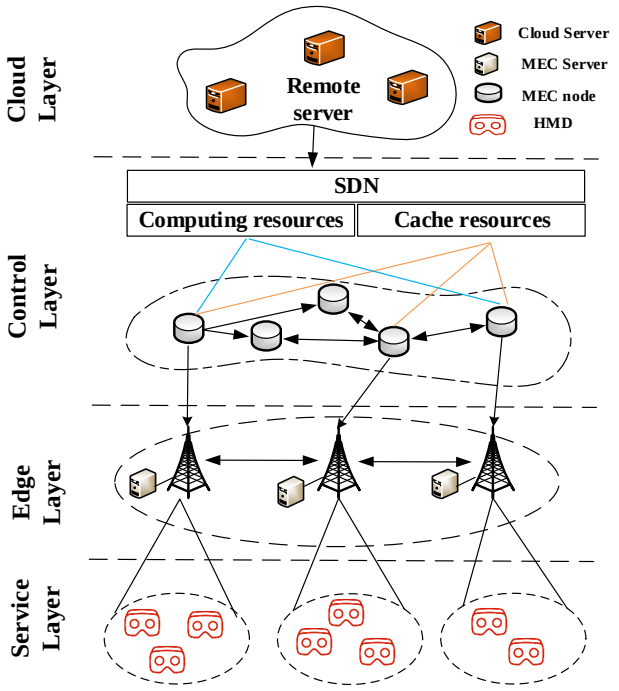

Fig. 1: Overall System Architecture.

3 SYSTEM MODEL

The system is shown in the Fig. 1, where the cloud server controls the computing and caching resources of each MEC node via SDN. The MEC server then allocates its own resources to the users in its covering scope, and the user side uses these resources for video caching, viewport prediction, and multiple QoE optimization. So this section describes the model structure of transmitting data and the structure of 3D video streaming. We will explain in detail which problems these two models solve. The main notations of the system model is described in Table 1.

3.1 Resource Allocation Model

Since 3D video consumes more computing resources and our study will involve making predictions about the video, we need to set up a buffer for the video. We take advantage of the fact that MEC can offload tasks and provide caching resources to solve resource allocation problems in conjunction with SDN. So the composition of the MEC in RAM is shown in Fig. 2. Each edge server has a certain coverage area and provides services to the users within that area. The

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TNSE.2020.3038998, IEEE Transactions on Network Science and Engineering

TABLE 1: System model's main notations

Notation	Description
K	The number of MEC servers
B_{user}	The user in MEC server coverage
N_{set}	The resource state set
D	The link communication cost of the MEC server
$Q(C)$, $Q(P)$	The caching and computing resources in the MEC server
α_i^K, β_i^K	The availability of caching and computing resources in the MEC server
$w(cu)$	The weight of computing resource
sa, sb	The number of composite view requirements and cache resource requirements
b	The bit error rate
s_p	The packet length
ρ_{SE_n}	The packet loss rate at the MEC server
T_d	The average queuing delay of a node
E	The number of tiles in the video
(x, y)	The position of a tile
ϖ	The candidate rate set
$g_{x,y}$	The size of the tile
$\nu(e)$	The total rate of the viewport
v_e	The download of the tile
B_e	The bandwidth of the tile
H_{buffer}	The remaining playing time of the video
μ	The weighting parameter
$B_{\max}$	The largest buffer area capacity
t_e	The time of e-th tile
H_e	The occupancy of the buffer area
Δt_e	The delay of video playback

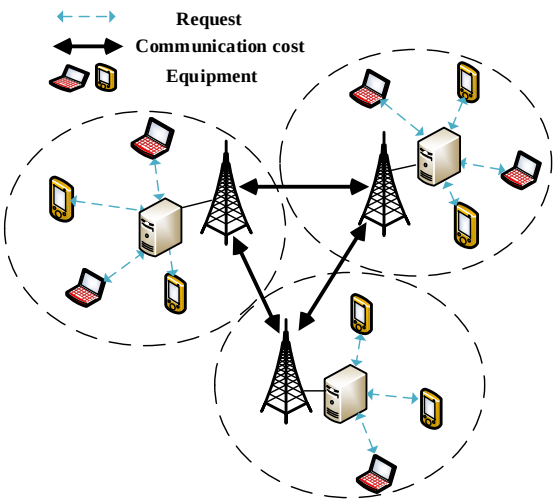

Fig. 2: The Architecture of RAM.

edge servers also communicate with each other, but this will generate additional communication consumption.

We assume that there are K MEC servers in the entire network, $\bigcup_{i=1}^{|A|} S_i \subseteq K$. S_i represents a set of original sources of view $a_i \in A$. And B_{user} denotes the users within the coverage of a MEC server. Therefore, the entire transmission network can be modeled as a weighted graph [24]:

$$G = \{N_{set}, D\}, \tag{1}$$

where N_{set} and D represent the resource state set as node set and link communication costs of the MEC servers as link set in the network. Due to limited caching and computing resources, the resources assigned to each server may not be the same. Therefore, the resources of each server need to be optimally allocated. First, the total cache resource is denoted as $Q(C)$ and the total computing resource is denoted as $Q(P)$. Therefore, the total cache size for MNO is denoted as $C = \sum_{i=1}^{K} C_i$, the total computing capacity is denoted as $P = \sum_{i=1}^{K} P_i$. So the total resources for cache and computation can be described as follows:

$$Q(C, P) = Q(C) + Q(P). \tag{2}$$

In the entire network, the resources on each MEC server are divisible, and these resources can be allocated to multiple requesters. Let $w(cu)$, $w(ca)$, $w(D)$, and $w(B)$ represent the weight of the MEC server's synthetic view computing resource weight, the caching resource weight, the communication cost weight, and the entire base station resource weight, respectively. The calculation cost weight and the buffer cost weight of the composite view can be set to determine the resources received by the MEC server. Let the total number of time slots is N. In each time slot, the resource demander is simply quantified into two resource modes: sa and sb represent the respective number of computing tasks and the number of cache tasks received by the MEC server, which is defined as follows.

$$w(ca) = \frac{\sum_{i=1}^{N} sa_i}{\sum_{i=1}^{N} (sa_i + sb_i)}, \tag{3}$$

$$w(cu) = \frac{\sum_{i=1}^{N} sb_i}{\sum_{i=1}^{N} (sa_i + sb_i)}. \tag{4}$$

Combining equ. (3), equ. (4) and equ. (2) can be rewritten as:

$$Q(C, P) = \sum_{i=1}^{K} \left(\sum_{j=1}^{N} w_j(ca) + w_j(cu) \right).$$

After $w(ca)$ and $w(cu)$ were obtained, $w(D)$ will be calculated next. For video transmission, the packet loss rate (PLR) is usually a typical indicator of communication quality. Because bit errors caused by data packet transmission on the link will cause application layer data packets to be missing. We assume that a path containing d segments is denoted as $SE_1, SE_2, ..., SE_n, ..., SE_d$. In order to capture the PLR introduced by the link error, under the premise of independent bit errors [25], we use the bit error rate (BER) b in physical layer and packet length s_p in routing layer to estimate the PLR on the MEC server as $\rho_{SE_n} = 1-(1-b)^{s_p}$. But in the real network, the bit error occurs from time to time [26]. For a packet of length s_p, since $s_p \times b$ is far less than 1, ρ_{SE_n} can be approximated as:

$$\rho_{SE_n} \approx s_p \times b.$$

Therefore, the packet loss probability caused by the link error of data packets on the entire path is calculated as:

$$\rho_{se} = 1 - \prod_{n=1}^{d} (1 - \rho_{SE_n}). \tag{5}$$

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TNSE.2020.3038998, IEEE Transactions on Network Science and Engineering

In addition to the PLR introduced by the error of the data packets on the link, the time delay caused by the data transmission speed or available bandwidth will also cause the PLR. We model the average queuing delay T_d of a node, which can be expressed as:

$$T_d = \frac{s_p}{R_b - R_s},$$

where R_b and R_s denote source coded bit rate assigned to the path b and the lowest available link capacity of all link segments in the path, respectively. The probability that the packet queuing delay exceeds the allowable delay threshold of the video can be written as [27] :

$$\rho_d = e^{-(R_b - R_s)\cdot T_{\max}/s_p},$$

where $T_{\max}$ denote the maximum latency of the queue.

Therefore, combined with the link error PLR calculated in equ. (5), the total PLR probability (i.e., weight of communication cost $w(D)$) of the transmission path can be expressed as

$$w(D) = \rho_{se} + (1 - \rho_{se}) \cdot \rho_d. \tag{6}$$

Combining equ. (3), equ. (4) and equ. (6), the total cost $w(B)$ of the base station can be obtained as:

$$w(B) = \{[w(ca), w(cu)], w(D)\}. \tag{7}$$

Combining equ. (1) and equ. (7), the expression of the weight graph can be converted into:

$$G = w(B). \tag{8}$$

3.2 Quality of Experience Model

3D video utilizes the MEC server's cache resources for video block transmission, while using the server's computing resources for video stitching. Each video block is composed of many consecutive frame images, and transmission is performed by compressing the video blocks. The entire video playback process is shown in the Fig. 3, which mainly consists of the server side and the application side. The server side is the MEC server, 3D video will use projection to project the video of the original sphere onto a flat surface, and then define every 1 second of video as a *video block*. The video block saves the spatial coordinates of this video. Therefore, it mainly converts 3D video to blocks and puts the information of the blocks into the block database. The database also keeps all candidate rate sets. The data is then put into a request execution program for the application side to download. After downloading the block data, the application side decodes and splices the blocks, and then pre-downloads the video to prepare it for playback. The player outputs observations to the QoE-AC model, which combines the values of observation and prediction to allocate a rate to the tiles and then the model places the tile information in the request queue.

Therefore, we assume that the video is divided into blocks of the same duration, denoted as T_s. These blocks are numbered from 1 to E, where E is the number of blocks in the video. Due to the limited viewport of HMD, it is not necessary to completely transmit high-quality video streams when playing on the device. Transmission of high-quality video streams requires more equipment resources.

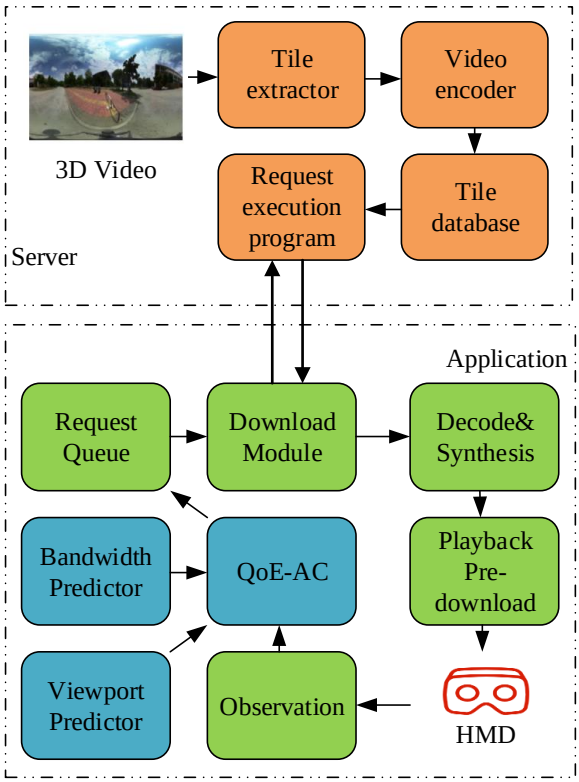

Fig. 3: The architecture of the QoEM: 3D video is pre-processed on the server and the history of the tiles is recorded after the request tile is downloaded on the application side, and QoEM uses the recorded information to allocate the transmission speed of the tiles.

The impact of buffering and video quality issues will greatly reduce the user's QoE, so we believe that the ideal system allocates a higher resolution in the HMD viewport and a lower resolution outside the viewport. The resolution of 3D video depends on the transmission speed of the tiles.

Therefore, we need to determine the speed of slicing, which can divide the entire viewport into multiple smaller network blocks. So we use (x, y) to represent the two-dimensional coordinates of the tile, and then the streaming system can formulate a strategy to allocate the rate of each slice by the video block transmission rate. Set a candidate rate set ϖ, and let $\varpi_{x,y}(e)$ denote the rate of the video block with the e-th block position (x, y). Therefore, after the system allocated the rate, it collects the corresponding video blocks and transmits them over the network to the HMD.

The limitation of video block rate selection is that the proposed system needs to follow the structure of DASH (Dynamic Adaptive Streaming over HTTP), and it can adaptively optimize QoE based on recorded playback statistics to effectively improve the user's QoE. The system needs to follow two rules when assigning the slice rate:

- The tiles distributed outside the viewport need to be fixed at a non-zero rate for transmission. In order to prevent the possible display to the user in the event of an opposite prediction, the user will not see an empty viewport even if

the prediction is wrong.

• The tiles in the viewport need to be allocated at the same rate, so that the viewport will not have a sense of screen fragmentation caused by different rates. And users cannot observe obvious tile boundaries between adjacent tiles.

The next is to consider the points that are important to the algorithm's learning strategy. When using HMD, users can only see the video in the viewport at the same time. For the viewpoint, $\varpi_{x,y}(e)$ was defined when the slice rate was allocated. We can determine the coverage of the user's viewport based on the hardware device of the HMD, then the (x, y) coordinates can be easily determined whether the tile is in the viewport. Let the matrix $f(e) \in \{0,1\}^{x \times y}$ indicate whether the tile is in the viewport. Use $f_{xy}(e)$ to indicate whether the coordinate (x, y) block is in the viewport. If $f_{xy}(e) = 1$, it means that (x, y) is in the viewport, otherwise $f_{xy}(e) = 0$.

Therefore, 3D video presents better viewing effects, and the video encoding is supported by different encoding methods. Different encoding methods will affect the download speed of tiles, such as DASH and HEVC. As mentioned earlier, the transmission is by compressing the video blocks, so that the size of the video block depends on the slice. The rate of the slice affects the resolution of the video. Changing the shape of the entire network, such as reducing the size of each block or dividing more blocks, increases the proportion of the viewport allocation [28]. But increasing the number of blocks in the entire viewport by dividing the blocks will increase the pressure on the grid, which may result in video buffering or video resolution incompatibility [29]. Therefore, we use $g_{x,y}(e, \varpi)$ to represent the size of the slice at (x, y) of the e-th block, where $\varpi_i = \varpi_{x,y}(e)$. For convenience, we use $g(e)$ to represent the size of the title, it can be written as

$$g(e) = \sum_{x=-X_{\max}}^{X_{\max}} \sum_{y=-Y_{\max}}^{Y_{\max}} g_{x,y}(e, \varpi_{x,y}(e)),$$

where $X_{\max}$ and $Y_{\max}$ denote the maximum horizontal coordinate and vertical coordinate of the current network structure, respectively. Therefore, the total rate of the viewport, denoted as $\nu(e)$, can be calculated as

$$\nu(e) = \sum_{x=-X_{\max}}^{X_{\max}} \sum_{y=-Y_{\max}}^{Y_{\max}} g_{x,y}(e, \varpi_{x,y}(e)) \cdot f_{xy}(e).$$

Therefore, the network bandwidth resources need to be considered after resolving the slice rate and slice size. We assume that the HMD sends the e-th tile at time t_e, the download speed of the tile is represented by v_e, and the bandwidth is represented by B_e. If the request for the same tile is continuous, the time required to download two tiles continuously is calculated as follows:

$$t_{e+1} = \begin{cases} t_e + \frac{g(e)}{v_e} + \Delta t_e, B_e \le B_{\max} \\ t_e + \frac{g(e)}{v_{\max}} + \Delta t_e, B_e > B_{\max} \end{cases},$$

where $v_{\max}$ denotes the maximum speed under the maximum bandwidth of the current network, Δt_e denotes the download delay of two consecutive blocks and $B_{\max}$ represents the largest buffer area capacity. Thus the current bandwidth of the network B_e is:

$$B_e = \frac{1}{t_{e+1} - t_e - \Delta t_e} \int_{t_e}^{t_{e+1} - \Delta t_e} v_e(t) dt.$$

So when watching the video, a buffer area needs to be set. In the HMD startup video stage, in order to prevent the buffer area from being blank, the buffer area starts to download tiles during the loading process. The cache area is set to multiple segments, each segment contains a video block. $H_{buffer}(t) \in [0, H_{\max}]$ represents the remaining playing time of the video in the cache area, and the occupancy rate of the cache area will continue to change during playback. The capacity will increase a segment every T seconds, and the video playback will consume a segment every T seconds. If the video downloads E_{buffer} tiles during the start-up phase, let $H_e = H(t_e)$ denote the occupancy of the buffer area when the e-th block request is sent. If $e < E_{buffer}$, the occupancy of the buffer area is $H_{e+1} = (E_{buffer} - e) \cdot T$, and if $E_{buffer} < e \le E$, then the occupancy rate will be jointly affected by the cache segments consumed by the download and playback of new tiles:

$$H_{e+1} = \max\left\{H_e - \frac{g(e)}{v_e}, 0\right\} + T - \Delta t_e.$$

In addition, when the video playback ends, the buffer area occupies 0 at this time. So there is at least one buffer segment in the buffer area before playing the next video, otherwise the video cannot be played. At the same time, it is necessary to reduce the block download rate outside the viewport and increase the block download rate inside the viewport. This can avoid unnecessary delays to the greatest extent and reduce the impact on the user's QoE. After the system completes the cache task in the viewport, it starts to increase the download speed of the tiles outside the viewport. When $H_e + 1 > H_{\max}$, the delay Δt_e to resume playback is given as:

$$\Delta t_e = \delta\left(\delta\left(H_e - \frac{g(e)}{v_e}\right) + T - H_{\max}\right),$$

where $\delta(\bullet) = \max(\bullet, 0)$ denotes the rectified linear function.

Finally, the goal of the proposed model is to optimize the QoE goal. In different situations, users have different requirements for video, resulting in different QoE standards. With these in mind, we are consistent with common 3D video systems, we consider the QoE indicator between block a and block b, focusing on the following goals:

3.2.1 Buffering Time

When the video has finished playing or the time for downloading the tiles is greater than the buffer capacity, the player will rebuffer until the new block is ready. The calculation formula for the buffering time is:

$$QoE_1 = T + \sum_{e=a}^{b} \delta\left(\frac{g(e)}{v_e} - H_e\right). \quad (9)$$

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TNSE.2020.3038998, IEEE Transactions on Network Science and Engineering

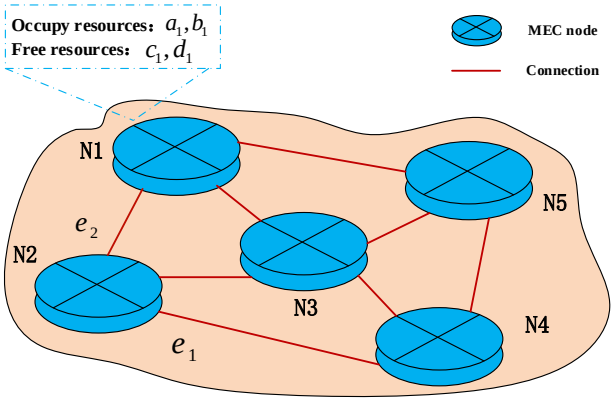

Fig. 4: Weight Graph for MEC nodes.

3.2.2 Viewport Quality Fluctuations

The rate change between two consecutive viewports will cause the viewport picture to be split, so the quality fluctuation of the viewport cannot be too large, which can be measured by the following formula:

$$QoE_2 = \frac{1}{b-a+1}\sum_{e=a}^{b}\left|\frac{g(e)}{\sum_{x=1}^{X}\sum_{y=1}^{Y} f_{xy}(e)} - \frac{g(e-1)}{\sum_{x=1}^{X}\sum_{y=1}^{Y} f_{xy}(e-1)}\right|. \tag{10}$$

3.2.3 Average Viewport Quality

The average quality of the viewport directly affects the quality of the video displayed in front of the user, and it is also the most important part of the user experience. The following formula is used to quantify the average quality of the video:

$$QoE_3 = \frac{1}{b-a+1}\sum_{e=a}^{b}\left(\frac{g\left(e\right)}{\sum_{x=1}^{X}\sum_{y=1}^{Y} f_{xy}\left(e\right)}\right). \tag{11}$$

Therefore, combining equ. (9), equ. (10) and equ. (11), the QoE of the entire video can be rewritten as

$$QoE' = \mu_1 QoE_1 + \mu_2 QoE_2 + \mu_3 QoE_3, \tag{12}$$

where μ_1, μ_2 and μ_3 correspond to weighting parameters.

4 Problem Formulation and Transformation

In this section, we first formulate the problem that we need to solve, and then tranfer the parameters of the problem to some basic components that will be input to a model in the next section.

4.1 Problem Formulation

After determining the required resources of each MEC server in the entire network, Fig. 4 shows the resources of the MEC nodes. The cache resources for each MEC node can be determined by equ. (8), i.e., $G = \{[w\left(ca\right), w\left(cu\right)], w\left(D\right)\}$. Set the information contained in each MEC node: $N_i = \{[a_i, b_i], [c_i, d_i]\}$, where $i \in K$. a_i and b_i denote the caching and computing resources owned by the MEC node, respectively. c_i and d_i represent the unused caching and computing resources of the MEC node, respectively. e represents the communication cost of the MEC node, $e = w\left(D\right)$. We then use SDN to control the distribution of user cache resources. Based on the Network Function Virtualization (NFV) technology, the functions of the devices can be integrated into the MEC server in the form of software to implement network devices of the function. Therefore, the caching resources of the MEC server can be allocated to different users by means of Virtual Machines (VMs). The VM placement algorithm places VMs to attain minimal data traffic by distributing virtual machine replica copies (VRCs) of applications to the edge network. We set α_i^K as a decision variable, which represents whether the caching resources on the MEC server are available. β_i^K indicates whether the computing resources on the MEC server are available. When the requested resource is available on the MEC server, $\alpha_i^K / \beta_i^K = 1$, otherwise it is 0. Let α_j^K be a decision variable, which indicates whether buffer resources are available on other MEC servers in the network. The decision variable β_j^K indicates whether the computing resources are available on other MEC servers in the network. If no resource is available in the adjacent MEC server, $\alpha_j^K \Big/ \beta_j^K \neq 1$, when the resource is requested. Otherwise, SDN will choose the smallest value around the current MEC server, i.e., $\min\{e\}$. And if $\min\{e\} \leq 0.03$, then the caching resources of the server can be used.

To solve the problem of how to allocate the block transmission speed, we design a model based on the Actor-Critic algorithm. So QoEM's algorithm structure is shown in Fig. 5. The environment will simulate the playback process, and the state at different moments will be recorded throughout the process. These parameters will be provided to the model, and the parameters will be updated in the subsequent training processes, and the strategy will be optimized to improve QoE. When the HMD needs to play a new video block, the environment will provide the previous bandwidth record and viewport record to the LSTM model, and then it inputs other information into the receiving LSTM. Therefore, the receiving LSTM outputs the estimated state value and sends it to the two FC layers. The model can receive new bandwidth and viewport records, and it performs these steps in a loop until the optimal strategy is reached. The next subsection describes the basic components and parameters of the model.

4.2 Basic Components and Parameters

4.2.1 State

Status information during playback includes tile e, viewport $Vi\left(e\right)$, bandwidth B_e, buffer occupancy H_e, and the size of the next tile $g_{x,y}\left(e\right)$. Let $B_e^{'}\left(t\right)$ denote the predicted band-

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TNSE.2020.3038998, IEEE Transactions on Network Science and Engineering

width at time t, and $Vi^{'}(e)$ denote the predicted viewport of e-th tile. So the observation o_e can be described as:

$$o_e = \{e+1, t_{e+1}, H_{e+1}, g(e), Vi'(e+1), B'_e(t_{e+2}+1), ..., B'_e(t_{e+2}+\tau)\}.$$

Let $Pi(\bullet)$ represent a collection of historical observations. The state (except the initial state s_0) is defined as a form of historical observation, which may help to transmit valuable information that was previously played:

$$s_e = Pi(o_1, o_2..., o_e).$$

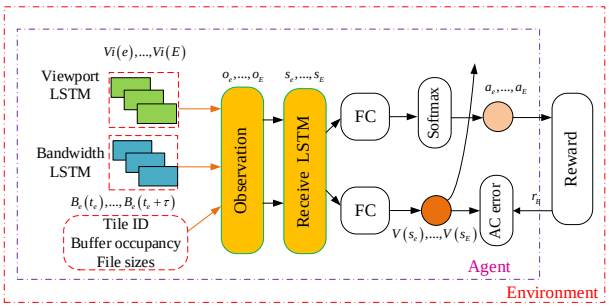

Fig. 5: Algorithm Structure.

4.2.2 Actions

The model uses the state s_e of the current tile e in the viewport to predict the rate of the $(e+1)$-th tile, which represents the action $a_e \in \varpi$.

4.2.3 Rewards

The playback application records the buffer time of each tile, the quality of the viewport and the fluctuation of the quality of the viewport. The goal of the model is to maximize QoE, so the value of the QoE goal is obtained in real time as the reward of the model. The reward for the e-th block r_e is calculated by $r_e = QoE'$.

5 Deep Reinforcement Learning Approach

In this section, we will introduce a model based on Actor-Critic algorithm to predict the bandwidth and viewport of future videos, and the model also reveals the relationship between features and QoE goals. The model uses Long Short-Term Memory (LSTM) networks and fully convolutional (FC) networks to improve the resolution accuracy of the model. Next, we will introduce the relevant details of the components of the model one by one.

5.1 Agent-Environment Interaction

DRL-based solutions have already intelligently solved problems from many aspects, such as resource allocation [30], adaptive streaming [31] and human-level game control [32], etc. These fully demonstrate the benefits of the algorithm's efficiency and cost control. Reinforcement learning settings include an agent that interacts with the environment. So at each title e, the agent accepts state s_e and performs action according to policy $\pi(s_e)$, which maps the states to actions or probability distributions. The goal of the agent is to get the reward r_e, predict the next state and calculate the return, i.e., $R_i = \sum_{i=e}^{E} \gamma^{i-e} r(s_i, a_i)$, where $r(s_i, a_i)$ denotes the reward function and $\gamma \in [0,1]$ is a discounting factor. Specifically, in the 3D video stream, the HMD playing the video can be regarded as an agent, and the content outside the agent can be regarded as the environment. Initially, the agent maintains state s_0, which includes the history and current information of the streaming system. In the eth-tile, the agent maintains a state s_e. When the tile is to be downloaded, the agent interacts with the environment and selects an action a_e, which is to allocate the rate to the tile $e+1$.

After the action a_e is performed, the agent will be affected by the QoE rewards r_{e+1}, and it updates its own statistical information to get the new state s_{e+1}. The algorithm learns favorable strategies through convolutional neural networks, and it continuously updates the best action value function. Therefore, it can obtain the maximum expected return after taking action a_e under the best state s_e.

$$Q^*_\pi(s_e, a_e) = \max_\pi \mathbb{E}[R_e | s_e, a_e, \pi]. \quad (12)$$

The key of this algorithm is to predict the next title. Therefore, the optimal way to calculate the action value: if the action a_{e+1} gets an optimized value $QoE(s_{e+1}, a_{e+1})$ in the state s_{e+1} of the next title $e+1$, we select it as the maximum expected value of $r_e + \gamma QoE(s_{e+1}, a_{e+1})$. The equ. (12) can then be rewritten be:

$$Q^*_\pi(s_e, a_e) = \mathbb{E}_{s_{e+1}}\left[r_e + \gamma \max_{a_{e+1}} Q^*_\pi(s_{e+1}, a_{e+1}) | s_e, a_e\right].$$

In the value-based method, the network will output discrete values for each operation, and then select a maximum operation as the next operation. Next, we will introduce the relevant parameter settings of the algorithm.

5.1.1 Policy gradient training

For QoE to be established, actions should be selected based on a random strategy to generate the probability distribution of all actions. The probability distribution of actions is defined as $\pi_\theta(s_e, a_e) \in [0,1]$. $\pi_\theta(s_e, a_e)$ is the probability of taking action a_e in the current state s_e. This strategy will obtain all the historical information of the playback process, and on this basis, generate the best strategy for allocating the tile rate. Therefore, equ. (11) can be written as:

$$\pi_\theta = \arg\max_{\pi_\theta} V^{\pi_\theta}(s_0),$$

where $V^{\pi_\theta}(s_0)$ is the cumulative reward expectation of the current state. The expression of this function is:

$$V^{\pi_\theta}(s_i) = \mathbb{E}_{\pi_\theta}\left[\sum_{i=e+1}^{E} \gamma^{i-e-1} r_i\right].$$

In the *Actor-Critic* algorithm, *Actor* will choose actions based on π_θ, *Critic* will return action values based on actions, and *Actor* will continue to modify the probability value of actions based on action values. π_θ can iterate the parameters according to θ, so the policy objective function is defined as:

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TNSE.2020.3038998, IEEE Transactions on Network Science and Engineering

$$J(\theta) = \mathbb{E}_{\pi_\theta}\left[\sum_{e=1}^{E} \gamma^{e-1} r_e \log \pi_\theta(s_e, a_e)\right].$$

Then, the formula for gradient update is:

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}\left[\sum_{e=1}^{E} \gamma^{e-1} r_e \nabla_\theta \log \pi_\theta(s_e, a_e)\right]. \quad (13)$$

However, this method must wait until the simulation playback is over before determining the QoE gradient. In this case, we introduce the AC error mechanism of the *Actor-Critic* algorithm, which only needs to wait until the next video block so that these methods are completely online and can be incremented. Define the AC error to evaluate the accuracy of the state value estimation:

$$\varphi_e = r_{e+1} + \gamma V(s_{e+1}, w) - V(s_e, w), \quad (14)$$

where $V(s_e, w)$ represents the weight vector of the state s_e. So the equ. (14) can be written as:

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}\left[\nabla_\theta \log \pi_\theta(s_e, a_e)\, \varphi_e\right].$$

Then set the update rule to:

$$\theta = \theta + \phi \nabla_\theta J(\theta),$$

where ϕ denotes the learning rate of the algorithm.

5.1.2 Prediction of bandwidth and viewport

The state of the target QoE includes bandwidth and viewport. For these two states, we use two layer LSTM networks to predict them separately. By setting the forget gate and input gate, we can determine what information can be discarded and what information is saved in the state s_e. ϑ_{LSTM} represent the parameters of the LSTM network, then we can use the LSTM model to predict the bandwidth and viewport, using the following formula:

$$B'_e(t+1) = \rho_{LSTM}\left(B_e(0), ..., B_e(t); \vartheta^{B_e}_{LSTM}\right).$$

The predicted value of the viewport can also be obtained:

$$Vi'(e+1) = \rho_{LSTM}\left(Vi(0), ..., Vi(e); \vartheta^{Vi}_{LSTM}\right),$$

where $\vartheta^{B_e}_{LSTM}$ and ϑ^{Vi}_{LSTM} represent the parameters of the predicted bandwidth and the parameters of the predicted viewport, respectively. $\rho_{LSTM}(\bullet)$ represents the historical LSTM information of the current state. The predicted values are cyclically input into the LSTM model, and the LSTM parameters are continuously updated to finally achieve the optimal effect.

5.2 Procedure of the QoE-AC Algorithm

The training method of the entire QoEM model named as QoE-AC is presented out in Alg. 1. Line 1-3 represent training the LSTM on the dataset of bandwidth and viewport to obtain the parameters $\vartheta^{B_e}_{LSTM}$, ϑ^{Vi}_{LSTM}. Line 4-14 represent the parameters obtained from one interaction between the algorithm and the environment, including predicted bandwidth information, predicted viewport information, and rate information for the next tile. Lines 15-17 represent the update process of the algorithmic strategy, using AC errors to update parameters w and learning rates θ. The final reward value is no longer changed and return the parameters $\vartheta^{B_e}_{LSTM}$, ϑ^{Vi}_{LSTM}, θ and w.

Algorithm 1 QoE-AC Algorithm

Input:
Discounting factor: γ
Learning rate of policy: ψ_θ
Learning rate of weight: ψ_w
Training dataset: D_{train}
Output:
Parameters of LSTM model: $\vartheta^{B_e}_{LSTM}$, ϑ^{Vi}_{LSTM}
policy θ; weight w

1: Initialize parameters $\vartheta^{B_e}_{LSTM}$, ϑ^{Vi}_{LSTM}, θ, w
2: **repeat**
3: Train LSTM model on D_{train}
$\vartheta^{B_e}_{LSTM} \leftarrow \vartheta'^{B_e}_{LSTM}$, $\vartheta^{Vi}_{LSTM} \leftarrow \vartheta'^{Vi}_{LSTM}$
4: **for** each dataset in D_{train} **do**
5: Initialize state s_0
6: **for** $e \in [0, ..., E]$ **do**
7: Perform a_e according to policy $\pi_\theta(s_e, a_e)$
8: Receive reward r_e and select learning rate by a_e
9: Predict $Vi'(e+2)$
10: Predict $B'_e(t_{e+2}+1), ..., B'_e(t_{e+2}+\tau)$
11: **if** $B'_e(t_{e+2}+1) > B_{\max}$ **then**
12: $B'_e(t_{e+2}+1) \leftarrow B'_e(t_{e+1}+1)$
13: **end if**
14: Receive Observation o_{e+1} and $s_e \leftarrow s_{e+1}$
15: AC error: $\varphi_e \leftarrow r_e + \gamma V(s_{e+1}, w) - V(s_e, w)$
16: Accumulate gradients w:
$w \leftarrow w + \phi^w \gamma^e \varphi_e \nabla_w V(s_e, w)$
17: Accumulate gradients θ:
$\theta \leftarrow \theta + \phi^\theta \gamma^e \varphi_e \nabla_\theta \pi_\theta(s_e, a_e)$
18: **end for**
19: **end for**
20: **until** Reward will not change anymore
21: **return** $\vartheta^{B_e}_{LSTM}$, ϑ^{Vi}_{LSTM}, θ, w

Before setting up the fully connected layer, join a receiving LSTM network, the network parameter is $\rho_{LSTM} = \vartheta^{\mathrm{Re}}_{LSTM}$, the task of learning strategy and estimating state value function share the receiving LSTM network parameters. The network takes the observations of the playback process as input and receives historical information from previous units to understand the historical playback conditions. Among the observations input to the receiving LSTM network are not only the output values of bandwidth LSTM and visual window LSTM, but also parameters such as time, block size, block count and buffer occupancy. Then the status of the receiving LSTM network in block e is written as:

$$s_e = \rho_{LSTM}\left(o_1, o_2, ..., o_e; \vartheta^{\mathrm{Re}}_{LSTM}\right).$$

When the receiving network receives the observation value from the LSTM network and outputs it to two fully connected layers,one of the fully connected layers is used to estimate the cumulative reward, and another fully connected layer uses the AC error to determine whether the current reward is favorable. If the parameter of the connected layer Fc_v is w', then $V(s_e, w)$ in equ. (14) is expressed as: $V(s_e, w) = Fc_v(s_e, w')$. Another role of the fully connected

layer is to classify and select the next reward. The parameter of the connected layer Fc_π is θ', which is represented by $\pi_\theta(s_e, a_e)$: $\pi_\theta(s_e, a_e) = Fc_\pi(s_e, \theta')$. Based on the learned strategy, the action with the largest classification value is selected, and the corresponding rate is assigned to the tiles in the predicted viewport. Within the environment, the state is transferred, and when the buffer has free space, the system can request the next video block.

6 Simulation Results and Discussion

In this section, we present the simulation results to estimate the performance of proposed scheme.

6.1 Setting the Conditions for the Experiment

Resource allocation: In order to determine the impact of resource allocation schemes on the MEC server, we assume that the maximum cache capacity is 1000GB and the maximum computing resource capacity of each MEC server is 2 GHz. The number of MEC servers in the entire network K is 5 and the number of users within each server range B_{user} is 10. Since the lowest resolution 3D video should be played normally during the test, the maximum bandwidth $B_{\max}$ is 20MHZ, the other network parameters are set in TABLE 2.

TABLE 2: Simulation parameters

Parameters	Values
Time, T	1s
Viewport LSTM, ϑ_{LSTM}^{Vi}	32
Bandwidth LSTM, $\vartheta_{LSTM}^{B_e}$	8
Forecast of future bandwidth, τ	10
Cache area, $H_{\max}$	3s
Video download, E_{buffer}	1
Maximum bandwidth, $B_{\max}$	20MHZ
Number of MEC servers, K	5
Number of users per server, B_{user}	10

Viewport traces: Sixteen 25-FPS 3D videos were collected from the data set [33], including fast-paced or slow-paced videos. And in this data set, 48 viewpoint trajectories were attached to each video in a 200ms time slot, which is a typcial scenario in practice. Two videos are randomly selected as the test set, and the remaining videos are used as the training set. We repeat the experiment ten times and report the average. Design the video to have five candidate rates, i.e., $\varpi = 5$. The entire HD video is encoded in X. 264 in Average Bitrate (ABR) mode, which is allocated by 1Mbps (360p), 5Mbps (720p), 8Mbps (1080p), 16Mbps (2K) and 35Mbps (4K). Then we divide the video into 1s blocks.

Bandwidth traces: We select 15 bandwidth trajectories of at least 300 seconds from the data set HSDPA [34] with various bandwidth modes. After that, randomly select three bandwidth trajectories in the test data. To ensure playback quality, the average bandwidth value should be between 2MHZ and 15MHZ.

Deep reinforcement learning algorithm: In order to evaluate our algorithm, the objects of the comparative experiment are the MPC [23], DDQN [35] and DQN [36] algorithms. Using the OpenAI gym environment with python, it can provide different game environments, which can be inserted into the code and tested for agents. The library is responsible for using the API to pass all the information needed by our agents, such as possible operations, current status and score. We just need to focus on the algorithm. Related parameters are listed in TABLE 3.

TABLE 3: Algorithm parameters

Parameters	Values
Number of Episodes	5000
Learning rate	0.25
Evaluation Interval	100
Bitch size	100
Discounting factor, γ	0.99
Reward gamma	0.99

3D video streaming systems: We compare the performance of the proposed 3D video optimization system with the four most advanced 3D video streaming systems. Naive-DASH [37] is a benchmark, which dynamically allocates the rate for the entire video, and it uses Linear Regression (LR) to predict bandwidth, and selects the maximum rate among the predicted bandwidth. FIXATION [38] relies on content-related functions to predict future visual windows, and uses the previous bandwidth instead of making predictions. CLS [22] manually groups the historical viewpoint trajectories through DBSCAN [39] and uses the clustering method to predict the visual window. It constructs a Knapsack model to determine the allocation rate of tiles. RUBIKS [40] is designed for video under scalable video encoding and uses historical head motion information to predict the future visual window.

QoE Objectives: We choose four sets of parameters of μ to indicate four QoE targets with different preferences, namely (1, 1, 1), (1, 0.25, 0.25), (1, 4, 1) and (1, 1, 4). For comparison of average QoE, we focus on port quality playback, average viewport quality and minimize the re-buffering time.

6.2 Performance of Resource Allocation

The Fig. 6 shows the effect of different cache capacities on the algorithm. In order to better show the comparison effect, we set three different scenarios to compare. Random cache capacity ("Radom") indicates that the model does not incorporate MEC and SDN technology, the same cache capacity ("Equal Resource") indicates that the model incorporates MEC technology, and resource allocation ("With Resource Allocation") indicates that the model incorporates MEC and SDN technology. So we can know that the third method has the best effect when the cache capacity is small. When the cache capacity is greater than 500 GB, the effect of the second and third methods is very close. This is because when the MEC server's cache resources are largely adequate for user needs, the effect of resource allocation between servers is reduced. When the server capacity reaches 1000 GB, the average delay of the three methods is the same, which also means that the MEC server can nearly cache all the requested contents.

Fig. 7 shows the impact of different computing resource capacities on the different scenarios. When the computing resource capacity is less than 1 GHz, the time delay of the algorithm with resource allocation is increased by 25%, indicating that resource allocation plays a stronger role. In the future, the role of resource allocation decreases, but

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TNSE.2020.3038998, IEEE Transactions on Network Science and Engineering

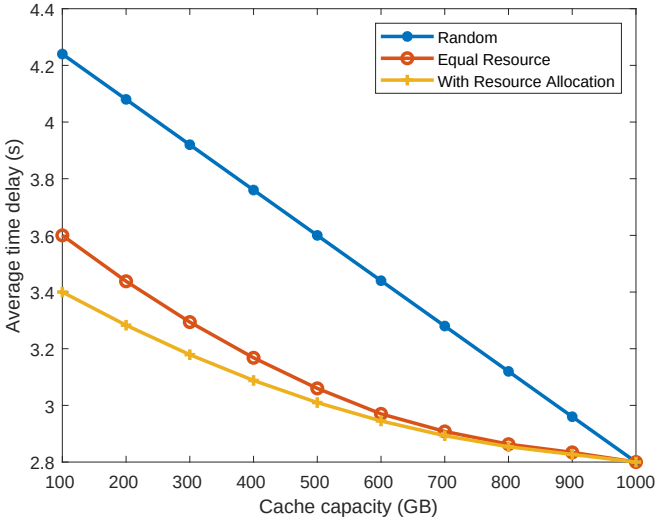

Fig. 6: Average time delay for different cache capacities

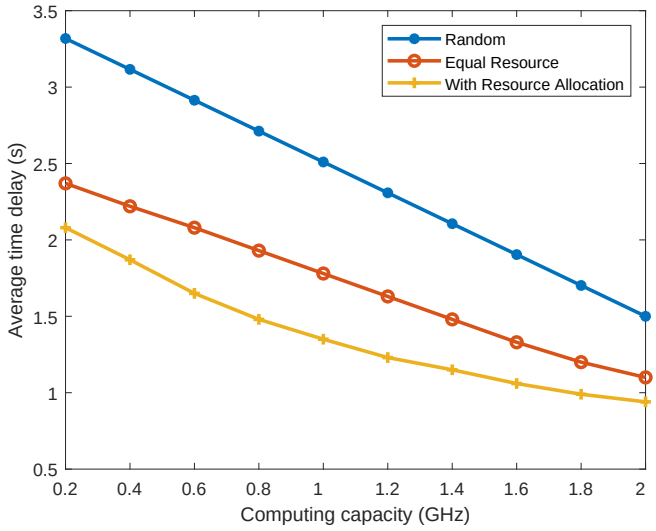

Fig. 7: Average time delay for different computing capacities

when the computing resource capacity reaches 2 GHz, the three methods do not achieve a uniform delay. Compared with Fig. 6, it is found that 3D video requires greater computing resource capacity because of the synthesis of tiles and rendering requires more computing resources.

Table 4 mainly shows the QoE values obtained from different model configurations in different environments. "Equal" indicates the model with MEC technology, and "Control" indicates the model with MEC and SDN technology. From the data in the table, we can see that under four different scenarios, "Equal" has a 13% performance improvement and "Control" has a 11.5% performance improvement, which shows that the addition of MEC and SDN has a significant impact on QoE.

6.3 Algorithm Performance Comparison

In order to compare the performance of the algorithm on the optimization strategy, the balanced game of Cartpole-v0

TABLE 4: QoE Values in Different Configurations

Environment	Random	Equal	Enhance	Control	Enhance
(-1, -1, 1)	0.69	0.78	13%	0.87	11.5%
(-0.25, -0.25, 1)	0.77	0.87	13%	0.98	11.3%
(-1, -1, 1)	0.66	0.74	12.1%	0.82	10.8%
(-1, -4, 1)	0.49	0.56	14.2%	0.63	12.5%

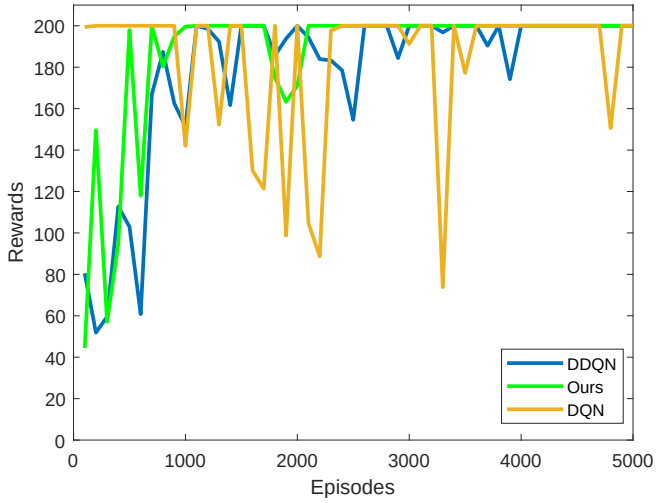

Fig. 8: Rewards of different algorithms

in OpenAI Gym [41] is used. The game is set to a maximum reward value of 200. It is generally considered that episodes with an average award of 195 can be considered to achieve the optimal strategy. The Fig. 8 shows the reward performance of the algorithm. The number of episodes between evaluations is 100. It can be seen from the figure that when the number of episodes is 2100, the algorithm we proposed reaches the optimal, DDQN achieves the optimal when the number of episodes is 3000, and the DQN algorithm reaches the maximum when the number of episodes is 3600. After the number of episodes of 3800 to the end of our observation 5000, the DQN algorithm reaches the optimal and is quite stable. The result shows that the algorithm we proposed has the best effect, with a 40% improvement, which can quickly reach the optimal policy.

6.4 LSTM Model Prediction Effect

In order to evaluate the effect of the two-layer LSTM network, we compare several different neural networks, we set the maximum buffer space to 3 seconds, and use CDF to complete the evaluation of the experimental results. The Fig. 9 shows the results using Mean Absolute Error (MAE) for the prediction of bandwidth. Baseline uses the most recent bandwidth to estimate the future bandwidth. LR uses linear regression to predict bandwidth. Therefore, from the results, it can be seen that the MAE values of the baseline range from 0.18 to 0.20, the MAE values of the LR range from 0.16 to 0.2, and the MAE values of the LSTM range from 0.14 to 0.2, so the effect of the LSTM network can be improved by 10%.

The Fig. 10 uses a precision indicator for the prediction of the visual window, which determines the gap between the predicted visual window and reality. From the experimental results, although the remaining algorithms are better

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TNSE.2020.3038998, IEEE Transactions on Network Science and Engineering

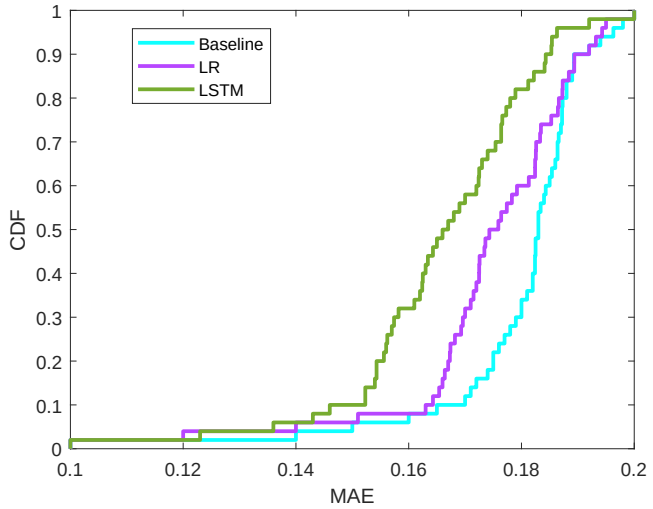

Fig. 9: Bandwidth prediction of different models

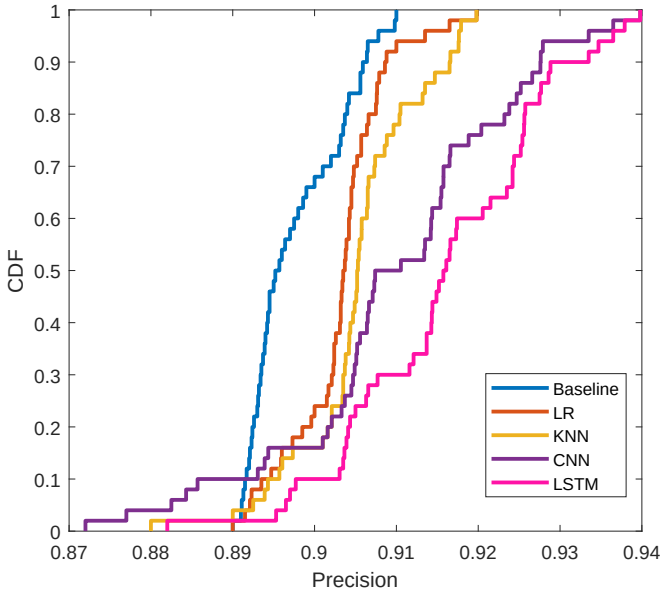

Fig. 10: Viewport prediction of different models

than the Baseline model, our LSTM network is significantly better than other networks, and there is a 5% performance improvement.

The Pensive model [12] also uses the Actor-Critic algorithm to learn the optimization strategy, and our proposed model joins the LSTM network on the basis of Actor-Critic. Therefore, in order to compare the performance difference between the QoE-AC model and the Pensive model, a comparison experiment is set up, and $\mu = (-1, -1, 1)$. There are six types of videos to watch, and the parameter settings of the LSTM network are shown in TABLE 2. Among them, No. 1-2 are viewed in the 3G network environment, No. 3-4 are viewed in the WiFi network environment, and No. 5-6 are viewed in the 4G-LTE network environment. From TABLE 5, we can see that our proposed model is 2.6 times more effective than the Pensive model in a 3G network environment. This indicates that the QoE-AC model with the prediction information of bandwidth and viewport has a better viewing experience when bandwidth is scarce. The QoE-AC model outperforms the Pensive model by 15% in both the WiFi and 4G-LTE environments. Although the difference is not as pronounced as in the 3G network environment, the QoE-AC model still outperforms the Pensive model, and it means that the improvement is more pronounced with the addition of the LSTM network.

TABLE 5: Comparison of QoE-AC and Pensive in different network environments

ID	QoE Values / Environment	Pensive	QoE-AC
1	3G	0.08	0.21
2	3G	0.15	0.38
3	WiFi	0.62	0.76
4	WiFi	0.64	0.75
5	4G-LTE	0.81	0.92
6	4G-LTE	0.83	0.93

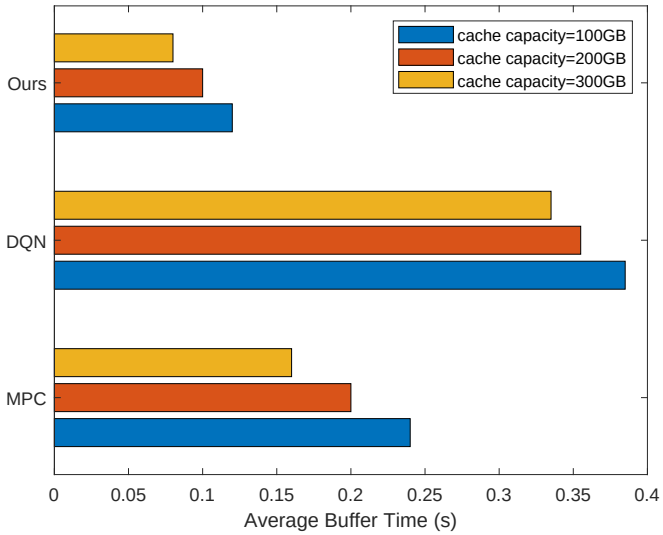

Fig. 11: Average rebuffer time of different algorithms

6.5 QoE Performance

In order to evaluate the advantages of the algorithm in optimizing the QoE target, we use the DQN and MPC algorithms to compare the impact of four different preference QoE targets. The reason for the negative number is that fluctuations in both cache time and visual window quality can negatively affect users. The Fig. 11 shows the impact of the algorithm on the rebuffer time, which we experimented based on three different sets of cache capacities. With a cache capacity of 100 GB, the average rebuffer time of the MPC algorithm is 0.24 seconds, the average rebuffer time of the DQN algorithm is 0.38 seconds, and the time of our algorithm is 0.12 seconds. Compared with the MPC algorithm, the effect is doubled, so our algorithm performs the best in terms of average rebuffer time.

It can be seen from Fig. 12 that our algorithm has achieved good results in the four QoE targets. By comparing the first group and the third group, we found that adjusting the buffer time ratio from 1 to 4 does not have a great impact on QoE, which shows that users have a higher tolerance

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TNSE.2020.3038998, IEEE Transactions on Network Science and Engineering

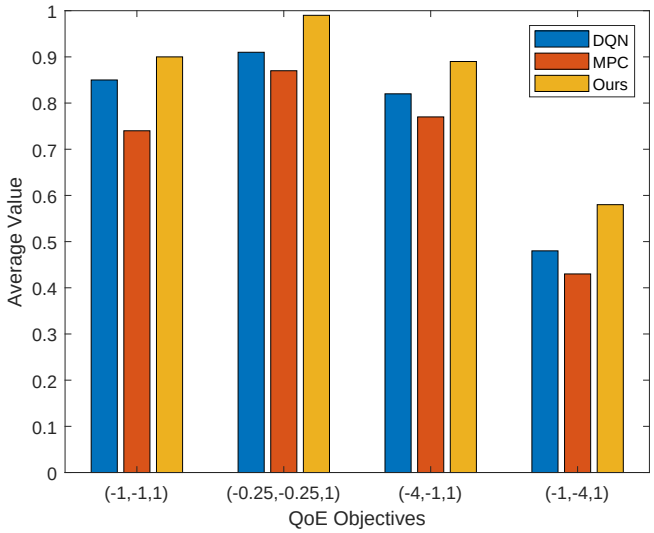

Fig. 12: The Effect of different QoE

for video buffering. Comparing the experiments of the first group and the fourth group, after adjusting the proportion of video quality fluctuation from 1 to 4, it was found that the value of QoE decreased significantly. Therefore it shows that users are more sensitive to fluctuations in video quality, because fluctuations will not only reduce the viewing experience, but may also cause adverse physiological reactions such as dizziness and nausea.

6.6 Video Streaming System Performance

The Fig. 13 shows the performance of different video streaming systems on QoE targets. Our model improves the average visual window quality by 10% in experiments that seek to maximize the average quality of the QoE goal (i.e., (1, 0.25, 0.25)). And when the QoE target has a larger buffer time preference, i.e., (-4, -1, 1), the effect of our model is not reduced much. On the QoE target (-1, -4, 1) with large fluctuations in video quality, the 3D video optimization model can still maintain an average value of about 0.6. Further analysis reveals a 13% performance improvement of our model compared to FIXATION and CLS. This is due to the fact that FIXATION and CLS tend to focus more on the average rate of the viewport and ignore the impact of changes in the surrounding environment on the QoE. There is a 10% performance improvement in our model relative to the RUBIKS model. The reason for this is that RUBIKS is based on the control theory, which attempts to conservatively improve the QoE goal in several subsequent blocks value. This method can help the transport stream system to perform equally on various QoE targets, but it is difficult to achieve the best results. Therefore, different QoE goals require different strategies, and optimization strategies can be learned adaptively using predicted results. So our system can keep the system always at a high level.

To further compare the models, we set up nine volunteers to watch the videos, in groups of three. They then watched the videos in 3G, WiFi, and 4G-LTE network environments, giving scores for buffering time, viewport quality fluctuations, and viewport quality after viewing. The QoE

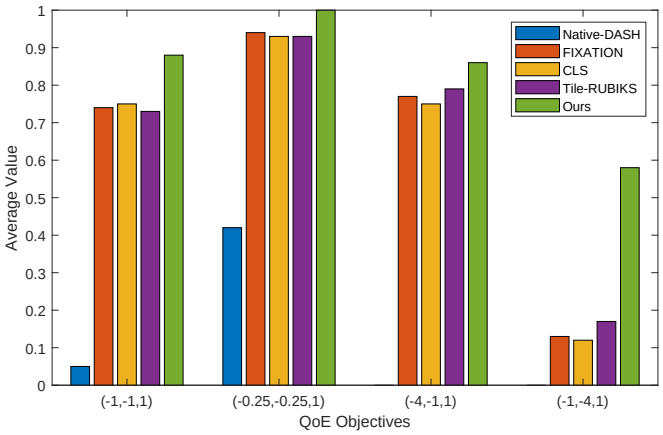

Fig. 13: QoE of different transport stream systems

weight is set to (-1, -1, 1), so as shown in Table 6, our model's viewing effect improves the most in the 3G network environment, indicating that our model can still give a better viewing experience in the absence of bandwidth resources. And it should be noted that the viewport trajectories of the volunteers in the real world will be more random than in the dataset, which will lead to a decrease in the prediction accuracy of the model, so that the improvement in the performance of the model can be more intuitively felt under such experimental settings.

TABLE 6: Comparison of Models in Different Network Environments

QoE Values / Environment	CLS	Tile-RUBIKS	Ours
3G	0.06	0.01	0.16
	0.37	0.09	0.40
	0.10	0.01	0.27
WIFI	0.61	0.52	0.71
	0.73	0.61	0.78
	0.70	0.65	0.78
4G-LTE	0.99	0.97	0.98
	0.94	0.98	0.97
	0.96	0.93	0.96

7 Conclusion

This paper focuses on assigning the optimal rate to the 3D video streaming in a dynamic communications network environment and improving the QoE of the user. The transmission of 3D video requires extremely high bandwidth resources, and at the same time has extremely high requirements on the network latency. To address these challenges, we propose to use MEC technology to offload the computational tasks of 3D video from the cloud to the MEC server, then we use the SDN technology to allocate caching and computing resources to the MEC server. The experimental evaluation shows a 10% improvement in the delay metric of our proposed model. Next, we use the resource allocation provided by RAM as the state space for the QoE-AC algorithm and combine it with the prediction information from LSTM to assign the optimal rate to the tiles. The experimental results show that our model consistently works well in different scenarios, with a 15% performance improvement over the traditional model.

ACKNOWLEDGMENT

The work of Professor Pan Zhou was supported in part by National Natural Science Foundation of China (NSFC) under Grant 61972448. The work of Professor Yulai Xie was supported in part by NSFC under Grant 61972449. The work of Professor Lingjun Pu was suported by National Natural Science Foundation of China (No.61702287); Young Elite Scientist Sponsorship Program by the Tianjin Association for Science and Technology (No. TJSQNTJ-2018-19). The work by Zichuan Xu was partially supported by the National Natural Science Foundation of China (Grant No. 61802048) and the "Xinghai Scholar Program" in Dalian University of Technology, China. The work of Professor Hao Jiang was supported in part by the National Natural Science Foundation of China Enterprise Innovation Development Key Project under Grant U19B2004, the Equipment Project of the Equipment Development Department under Grant 41412010702, the Major R&D Platform Project of New R&D Institutions in Zhongshan City under Grant 2017F1FC0001. The work of Professor Huawei Huang was supported in part by NSFC (No. 61902445), and Guangdong Basic and Applied Basic Research Foundation (No. 2019A1515011798).

REFERENCES

[1] J. Xu, B. Palanisamy, H. Ludwig, and Q. Wang, "Zenith: Utility-aware resource allocation for edge computing," in *2017 IEEE international conference on edge computing (EDGE)*. IEEE, 2017, pp. 47–54.

[2] J. Kwak, Y. Kim, L. B. Le, and S. Chong, "Hybrid content caching in 5g wireless networks: Cloud versus edge caching," *IEEE Transactions on Wireless Communications*, vol. 17, no. 5, pp. 3030–3045, 2018.

[3] M. Xiao, C. Zhou, Y. Liu, and S. Chen, "Optile: Toward optimal tiling in 360-degree video streaming," in *Proceedings of the 25th ACM international conference on Multimedia*, 2017, pp. 708–716.

[4] A. Zare, K. K. Sreedhar, V. K. M. Vadakital, A. Aminlou, M. M. Hannuksela, and M. Gabbouj, "Hevc-compliant viewport-adaptive streaming of stereoscopic panoramic video," in *2016 Picture Coding Symposium (PCS)*. IEEE, 2016, pp. 1–5.

[5] T. Kämäräinen, M. Siekkinen, J. Eerikäinen, and A. Ylä-Jääski, "Cloudvr: Cloud accelerated interactive mobile virtual reality," in *Proceedings of the 26th ACM international conference on Multimedia*, 2018, pp. 1181–1189.

[6] W. Huang, L. Ding, G. Zhai, X. Min, J.-N. Hwang, Y. Xu, and W. Zhang, "Utility-oriented resource allocation for 360-degree video transmission over heterogeneous networks," *Digital Signal Processing*, vol. 84, pp. 1–14, 2019.

[7] T. X. Tran, A. Hajisami, P. Pandey, and D. Pompili, "Collaborative mobile edge computing in 5g networks: New paradigms, scenarios, and challenges," *IEEE Communications Magazine*, vol. 55, no. 4, pp. 54–61, 2017.

[8] M. Liyanage, A. Gurtov, and M. Ylianttila, *Software Defined Mobile Networks (SDMN): Beyond LTE Network Architecture*. John Wiley & Sons, 2015.

[9] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[10] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath, "Deep reinforcement learning: A brief survey," *IEEE Signal Processing Magazine*, vol. 34, no. 6, pp. 26–38, 2017.

[11] M. Gadaleta, F. Chiariotti, M. Rossi, and A. Zanella, "D-dash: A deep q-learning framework for dash video streaming," *IEEE Transactions on Cognitive Communications and Networking*, vol. 3, no. 4, pp. 703–718, 2017.

[12] H. Mao, R. Netravali, and M. Alizadeh, "Neural adaptive video streaming with pensieve," in *Proceedings of the Conference of the ACM Special Interest Group on Data Communication*, 2017, pp. 197–210.

[13] V. R. Konda and J. N. Tsitsiklis, "Actor-critic algorithms," in *Advances in neural information processing systems*, 2000, pp. 1008–1014.

[14] L. Van Ma, V. Q. Nguyen, J. Park, and J. Kim, "Nfv-based mobile edge computing for lowering latency of 4k video streaming," in *2018 Tenth International Conference on Ubiquitous and Future Networks (ICUFN)*. IEEE, 2018, pp. 670–673.

[15] 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, 2019.

[16] K. Bilal and A. Erbad, "Edge computing for interactive media and video streaming," in *2017 Second International Conference on Fog and Mobile Edge Computing (FMEC)*. IEEE, 2017, pp. 68–73.

[17] A. Younis, T. X. Tran, and D. Pompili, "On-demand video-streaming quality of experience maximization in mobile edge computing," in *2019 IEEE 20th International Symposium on" A World of Wireless, Mobile and Multimedia Networks"(WoWMoM)*. IEEE, 2019, pp. 1–9.

[18] J. Li, X. Li, Y. Gao, Y. Gao, and R. Zhang, "Dynamic cloudlet-assisted energy-saving routing mechanism for mobile ad hoc networks," *IEEE Access*, vol. 5, pp. 20 908–20 920, 2017.

[19] F. Lei, J. CAI, J. Luo, Q.-y. DAI, and H.-m. ZHAO, "Efficient caching mechanism based on soft defined information-centric networks," *Computer Science*, vol. 43, no. 8, pp. 74–78, 2016.

[20] A. Tootoonchian and Y. Ganjali, "Hyperflow: A distributed control plane for openflow," in *Proceedings of the 2010 internet network management conference on Research on enterprise networking*, vol. 3, 2010.

[21] C. Zhou, C.-W. Lin, and Z. Guo, "mdash: A markov decision-based rate adaptation approach for dynamic http streaming," *IEEE Transactions on Multimedia*, vol. 18, no. 4, pp. 738–751, 2016.

[22] L. Xie, X. Zhang, and Z. Guo, "Cls: A cross-user learning based system for improving qoe in 360-degree video adaptive streaming," in *Proceedings of the 26th ACM international conference on Multimedia*, 2018, pp. 564–572.

[23] X. Yin, A. Jindal, V. Sekar, and B. Sinopoli, "A control-theoretic approach for dynamic adaptive video streaming over http," in *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication*, 2015, pp. 325–338.

[24] J. Kuo, D. Yang, W. Li, and W. Chen, "Efficient multi-view 3d video multicast with mobile edge computing," in *2018 IEEE Global Communications Conference (GLOBECOM)*, 2018, pp. 1–7.

[25] Y. Andreopoulos, N. Mastronarde, and M. van der Schaar, "Cross-layer optimized video streaming over wireless multihop mesh networks," *IEEE Journal on Selected Areas in Communications*, vol. 24, no. 11, pp. 2104–2115, 2006.

[26] Y. Liu, J. Liu, A. Argyriou, and S. Ci, "3d qoe-oriented and energy-efficient 2d plus depth based 3d video streaming over centrally controlled networks," *IEEE Transactions on Multimedia*, vol. 20, no. 9, pp. 2439–2453, 2018.

[27] X. Zhu, E. Setton, and B. Girod, "Congestion–distortion optimized video transmission over ad hoc networks," *Signal Processing Image Communication*, vol. 20, no. 8, pp. 773–783, 2005.

[28] C. Zhou, M. Xiao, and Y. Liu, "Clustile: Toward minimizing bandwidth in 360-degree video streaming," in *IEEE INFOCOM 2018 - IEEE Conference on Computer Communications*, 2018, pp. 962–970.

[29] Z. Li, G. Fei, J. Geng, L. Dan, and U. Zafar, "Dante: Enabling fov-aware adaptive fec coding for 360-degree video streaming," in *the 2nd Asia-Pacific Workshop*, 2018.

[30] H. Mao, M. Alizadeh, I. Menache†, and S. Kandula†, "Resource management with deep reinforcement learning," in *Proceedings of the 15th ACM Workshop on Hot Topics in Networks*, 2016, pp. 50–56.

[31] H. Mao, R. Netravali, and M. Alizadeh, "[acm press the conference of the acm special interest group - los angeles, ca, usa (2017.08.21-2017.08.25)] proceedings of the conference of the acm special interest group on data communication - sigcomm ï7 - neural adaptive video streaming with pensiev," pp. 197–210.

[32] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[33] C. Wu, Z. Tan, Z. Wang, and S. Yang, "A dataset for exploring user behaviors in vr spherical video streaming," in *Proceedings of the 8th ACM on Multimedia Systems Conference*, 2017, pp. 193–198.

[34] H. Riiser, P. Vigmostad, C. Griwodz, and P. Halvorsen, "Commute path bandwidth traces from 3g networks: analysis and applica-

tions," in *Proceedings of the 4th ACM Multimedia Systems Conference*, 2013, pp. 114–118.
[35] H. Van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double q-learning," in *Thirtieth AAAI conference on artificial intelligence*, 2016, pp. 2094–2100.
[36] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing atari with deep reinforcement learning," *arXiv preprint arXiv:1312.5602*, 2013.
[37] T. Stockhammer, "Dynamic adaptive streaming over http– standards and design principles," in *Proceedings of the second annual ACM conference on Multimedia systems*, 2011, pp. 133–144.
[38] C.-L. Fan, J. Lee, W.-C. Lo, C.-Y. Huang, K.-T. Chen, and C.-H. Hsu, "Fixation prediction for 360 video streaming in head-mounted virtual reality," in *Proceedings of the 27th Workshop on Network and Operating Systems Support for Digital Audio and Video*, 2017, pp. 67–72.
[39] M. Ester, H.-P. Kriegel, J. Sander, X. Xu *et al.*, "A density-based algorithm for discovering clusters in large spatial databases with noise." in *Kdd*, vol. 96, no. 34, 1996, pp. 226–231.
[40] J. He, M. A. Qureshi, L. Qiu, J. Li, F. Li, and L. Han, "Rubiks: Practical 360-degree streaming for smartphones," in *Proceedings of the 16th Annual International Conference on Mobile Systems, Applications, and Services*, 2018, pp. 482–494.
[41] A. Nandy and M. Biswas, "Reinforcement learning with keras, tensorflow, and chainerrl," in *Reinforcement Learning*. Springer, 2018, pp. 129–153.

Pan Zhou (S'07–M'14-SM'20) is currently an associate professor and PhD advisor with Hubei Engineering Research Center on Big Data Security, School of Cyber Science and Engineering, Huazhong University of Science and Technology (HUST), Wuhan, P.R. China. He received his Ph.D. in the School of Electrical and Computer Engineering at the Georgia Institute of Technology (Georgia Tech) in 2011, Atlanta, USA. He received his B.S. degree in the Advanced Class of HUST, and a M.S. degree in the Department of Electronics and Information Engineering from HUST, Wuhan, China, in 2006 and 2008, respectively. He held honorary degree in his bachelor and merit research award of HUST in his master study. He was a senior technical member at Oracle Inc., America, during 2011 to 2013, and worked on Hadoop and distributed storage system for big data analytics at Oracle Cloud Platform. He received the "Rising Star in Science and Technology of HUST" in 2017. He is currently an associate editor of IEEE Transactions on Network Science and Engineering. His current research interest includes: security and privacy, big data analytics and machine learning, and information networks.

Yulai Xie received the B.E. and Ph.D. degrees in computer science from Huazhong University of Science and Technology (HUST), China, in 2007 and 2013, respectively. He was a visiting scholar at the University of California, Santa Cruz in 2010 and a visiting scholar at the Chinese University of Hong Kong in 2015. He is now an associate professor in School of Cyber Science and Engineering, in HUST, China. His research interests mainly include cloud storage and virtualization, digital provenance, intrusion detection, machine learning and computer architecture.

Ben Niu received the B.S. degree in information security and M.S. and Ph.D. degrees in cryptography from Xidian University, Xi'an, China, in 2006, 2010, and 2014, respectively. He is currently a Research Assistant with the State Key Laboratory of Information Security, Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China. He was a Visiting Scholar with Pennsylvania State University, State College, PA, USA, from 2011 to 2013. His current research interests include wireless network security and privacy computing.

Lingjun Pu received the Ph.D. degree from Nankai University, Tianjin, China, in 2016. He was a joint Ph.D. student with the University of Göttingen, Göttingen, Germany, from 2013 to 2015. He is currently an Associate Professor with the College of Computer Science, Nankai University. His current research interests include mobile cloud/edge computing, edge caching, Internet of Things, 5G C-RAN, SDN/NFV, and opportunistic routing.

Zichuan Xu received the BSc and ME degrees from the Dalian University of Technology, China, in 2008 and 2011, respectively, and the PhD degree from the Australian National University, in 2016, all in computer science. He was a research associate with the University College London. He currently is an associate professor in the School of Software, Dalian University of Technology. His research interests include cloud computing, software-defined networking, wireless sensor networks, algorithmic game theory, and optimization problems.

Hao Jiang (M'08) received the B.S. and Ph.D. degrees in communication networks from Wuhan University, Wuhan, China, in 1998 and 2004, respectively. He is currently a Professor with the School of Electronic Information, Wuhan University. His current research interests include wireless LANs, wireless ad hoc networks, and vehicle ad hoc networks.

Huawei Huang received the Ph.D degree in computer science and engineering from the University of Aizu, Japan. He is currently an Associate Professor at School of Data and Computer Science, Sun Yat-Sun University, China. His research interests mainly include SDN/NFV, edge computing, and blockchain system optimization. He received the Best Paper Award of TrustCom-2016. He used to be a Visiting Scholar at the Hong Kong Polytechnic University (2017–2018), a Postdoctoral Research Fellow of JSPS (2016–2018), anAssistant Professor atKyoto University, Japan (2018–2019).