A Predictive Frame Transmission Scheme for Cloud Gaming in Mobile Edge Cloudlet Systems

Tianchu Zhao, Sheng Zhou, *Member, IEEE*,
Yuxuan Sun, *Member, IEEE*, and Zhisheng Niu, *Fellow, IEEE*

Abstract—Cloud gaming is promising yet poses big challenges to wireless communications, due to its stringent requirements for low response delay and high reliability. In this paper, we propose a predictive frame transmission scheme (PFT) in cloud gaming, to predict and pre-transmit future game frames to users. The PFT scheme takes full advantage of good network states to transmit the predicted frames, which consequently reduces the frame loss rate (FLR) against the network dynamics. We first model a FLR minimization problem in the single-user system with the PFT scheme, which allocates packets to carry the predicted frames. The upper and lower bounds of FLR are derived, respectively. Then, we study the system with Markovian property, and derive the optimal packet allocation policy via Markov Decision Process. A near-optimal policy is also proposed with low-complexity. The PFT scheme is further extended to the multi-server multi-user scenario, in which the users are adaptively scheduled to multiple servers based on their different requirements. Finally, we extend the policy to fit the scenario without direct knowledge of the network state by exploiting the packet loss rate estimation. We set up a practical testbed to evaluate the proposed PFT scheme, showing the capability of decreasing the mean FLR from 7% to 1%.

Index Terms—Mobile edge cloudlet, cloud gaming, frame prediction, Markov decision process

1 Introduction

With the proliferation of mobile devices, mobile game has gained great popularity in the past several years. According to the report by Newzoo [1], the global market of mobile game has maintained a growth rate of 10% since 2016, and it was worth US$77.2 billion in 2020 that accounted for 40% of the total game market. Mobile users expect to enjoy mobile game anywhere and anytime, with high-quality game graphics and good operating experiences. However, advanced games consume intensive CPU and GPU resources, which poses great challenges to resource-constrained mobile devices [2]. As a promising solution, cloud gaming is proposed to run games on cloud servers and stream video back to mobile devices. Because of its good potential, cloud gaming has attracted great attentions from both academia [3], [4], [5] and industry [6].

With cloud gaming, the computation resources of mobile devices are no longer bottlenecks, but the challenges migrate from computation to communication. According to the white paper [7] and surveys [2], [8], the real-time data transmission is of stringent requirements to realize the high Quality of Service (QoS) in cloud gaming, including:

- *High bandwidth consumption*: the download bandwidth should be at least 3Mb/s.
- *Low response delay*: the response delay of user interactions should be less than 50ms.
- *High FPS and small frame loss rate*: the FPS (Frames Per Second) of video should be at least 60, and the probability of video frame loss should be at most 1%.

With the development of 5G technologies, mobile devices will be rich in bandwidth resources, and the download speed can be as high as 100Mb/s [9]. But the response delay still remains a challenge for the existing commercialized cloud gaming platforms (e.g., Google Stadia), in which video is streamed from remote cloud servers to users via the Internet. Because of its long physical distance, the data transmission over the Internet results in large delay. According to the test on Google Stadia [10], the average response delay of cloud gaming varies from 100ms to 300ms, which is difficult to meet the 50ms delay requirements of gaming.

To decrease the delay of cloud gaming, novel architectures are proposed that set up gaming servers in mobile edge cloudlets (MEC) [8], [11], as shown in Fig. 1. In MEC, cloud servers are deployed at the edge of networks (e.g., base stations and access points), and connect to mobile devices via wireless networks directly. In each game frame, the user first uploads the instruction to the edge cloud server via the wireless network. Thereafter, game logic is run on the server and images are rendered accordingly. Finally, the images are streamed back to the mobile user by the wireless network. Because MEC utilizes the cloud servers near users,

- *The authors are with the Beijing National Research Center for Information Science and Technology, Department of Electronic Engineering, Tsinghua University, Beijing 100190, China. E-mail: zhaotc13@mails.tsinghua.edu.cn, {sheng.zhou, niuzhs}@tsinghua.edu.cn, sunyuxuan@mail.tsinghua.edu.cn.*

Manuscript received 22 February 2021; revised 6 October 2021; accepted 24 January 2022. Date of publication 7 February 2022; date of current version 5 June 2023.
This work was supported in part by the National Key R&D Program of China under Grant 2020YFB1806605, in part by the National Natural Science Foundation of China under Grants 62022049 and 61871254, and in part by China Postdoctoral Science Foundation under Grant 2020M680558, and Hitachi Ltd. (Corresponding author: Sheng Zhou.)
Digital Object Identifier no. 10.1109/TMC.2022.3149056

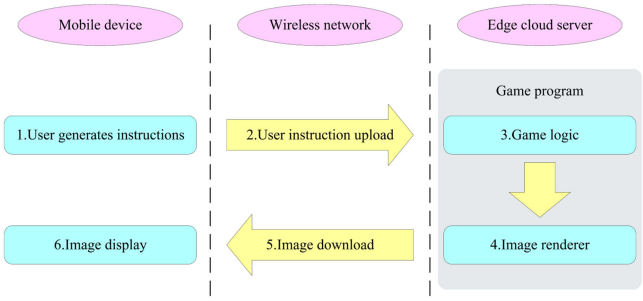


Fig. 1. The MEC-assisted cloud gaming system.

the delay can be notably reduced compared with remote cloud servers. In the experimental tests [8], MEC can reduce the delay of cloud gaming to 16ms.

In the MEC-assisted cloud gaming platforms, the dynamics of wireless networks become the major obstacles against high-quality gaming experiences. The video images are streamed to users by application-layer packets with high FPS, which is at least 60FPS for high-quality games [7]. But the packet loss occurs from time to time, due to the dynamic wireless channel and network congestions. Even if the system re-transmits packets after packet loss happens, the re-transmitted packets might still violate the delay requirement. As a result, frame loss happens when mobile devices do not receive packets *in time*, and the frame loss rate (FLR) can be larger than 5% in bad network states (e.g., the wireless channel is bad, or the network is congested) [12]. For mobile users, FLR is a very important metric in terms of the gaming experiences, and users can be annoying even if FLR is 1% [7].

1.1 Motivating Idea

In this paper, we aim at minimizing the FLR against the dynamic packet loss in networks. Our motivating idea is stated as follows.

In cloud gaming systems, the server deals with each game frame as a computational task. Considering the consistency of game logic, the sequential frames are highly correlated rather than independent tasks. In a new game frame, the server updates the game logic based on the previous frame as well as the user instruction. The updates of game logic have two cases. 1) If not a new instruction is generated from the user, the server will update the game logic according to the latest user instruction. 2) Otherwise, the server will update the instruction, and then update the game logic accordingly. If the user instruction is not updated or changed, the frame can be precisely predicted. Otherwise, the previously predicted frames should be updated accordingly.

Compared with the frame rate of gaming (e.g., 60FPS), the update of user instruction is of a far smaller frequency (e.g., 1Hz). Therefore, in most cases, the user instruction does not change, and the frame can be predicted correctly with a high probability.

Exploiting the frame *correlation*, we propose the predictive frame transmission scheme against the dynamics of networks. In each frame, based on the latest user instruction, the server processes the game logic and renders the image of the current frame, and it also predicts the game logic and images of future frames. Then, the predicted images will be pre-transmitted to the users depending on the network state. With the scheme, more images of future frames can be forwarded to the user when the network state is good, in case of the upcoming bad network states.

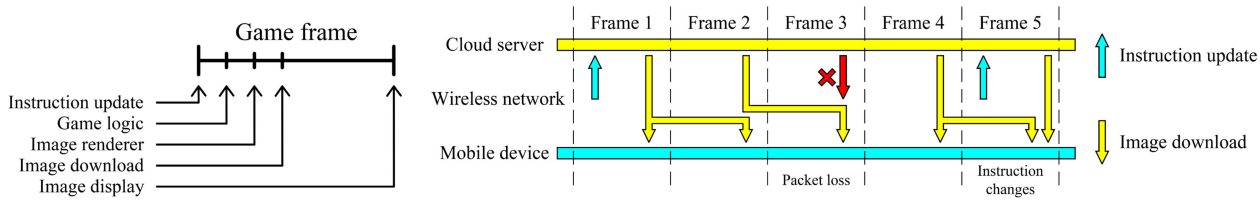
Many previous efforts [13], [14], [15], [16], [17], [18], [19] have been devoted to studying the data transmission of each game frame independently. Compared with these papers, we exploit the *correlation* between sequential frames against the network dynamics.

1.2 Main Contributions

We would like to highlight our contributions as follows.

- In the MEC-assisted cloud gaming systems, we propose the predictive frame transmission scheme, by which the system can predict and pre-transmit future frames to users via packet allocation. We formulate the FLR minimization problem with the proposed scheme in the single-user system. The upper and lower bounds of FLR are derived, respectively.
- By modeling the system as a Markov chain, we derive the FLR theoretically by solving the stationary distribution. Then, we derive the optimal packet allocation policy by Markov Decision Process (MDP), and analyze the structure of the optimal policy. To address the curse of dimensionality in MDP, we propose a near-optimal policy with low complexity, which allocates packets to the two images that will be displayed in the nearest future.
- We extend the proposed scheme to the multi-server multi-user scenario. An optimization problem is formulated to minimize the mean FLR of all users, by scheduling users to edge servers as well as allocating the computation and bandwidth resources. We propose a dynamic-programming-based algorithm to jointly derive the optimal user schedule and resource allocation.
- We further consider the scenario that the system does not have any prior information about the network states. In this case, based on the historical packet loss observations, we propose a sub-optimal packet allocation algorithm with packet loss rate estimation.
- We set up a testbed to realize the predictive frame transmission scheme in practical cloud gaming systems. It is shown that the maximum FLR can be decreased from 15% to 4% and the mean FLR can be decreased from 7% to 1% with the proposed scheme. We also carry out extensive simulations to evaluate the scheme.

The rest of the paper is organized as follows. In Section 2, we summarize the related work. Section 3 introduces the system model. In Section 4, we formulate the FLR minimization problem and analyze the bounds of FLR. In Sections 5 and 6, we study the optimal packet allocation policy in the system with Markov model and with general model, respectively. We also study the user schedule and resource allocation in the multi-server multi-user scenario. In Section 7, experiments and simulations are presented to evaluate the proposed scheme. The paper is concluded in Section 8.



(a) Structure of game frame. (b) An example of predictive frame transmission.

Fig. 2. The frame structure and an example of cloud gaming with predictive frame transmission.

2 Related Work

The timeliness of cloud gaming systems has lately been widely studied [13], [14], [15], [16], [17], [18], [19]. In [13], the authors study the age of information analytically in Markov model based cloud gaming systems. The age metric is then used to optimize the frame rate and lag synchronization. Ref. [14] studies the round trip delay of cloud gaming systems, which is optimized by controlling the data rate via the network congestion detection. Refs. [15], [16], [17] study the multi-objective optimization, which minimize the energy and resource cost of the system while satisfying the delay requirements of users. Ref. [15] is characterized by the resource virtualization in virtual machines, [16] is characterized by the system with geographically distributed data centers, and [17] focuses on the GPU resource cost of systems. Refs. [18] and [19] study a novel system that mobile devices can not only offload gaming tasks to cloud servers, but can also offload them to other mobile devices nearby. Their objectives are to minimize the energy consumption of mobile devices under the constraints of gaming delay. Ref. [18] considers that mobile devices can communicate with each other directly, while [19] studies an ad-hoc network. The papers [13], [14], [15], [16], [17], [18], [19] study the gaming delay in terms of each single game frame independently. In comparison, we study the sequential frames and exploit their correlations, so as to combat the dynamics of networks.

Server selection and resource allocation for multi-user systems are also important in cloud gaming systems, which have been studied by many recent papers [20], [21], [22], [23], [24]. In [20], [21], [22], the authors study the system that multiple users are served by multiple servers, and users select the server for cloud gaming. Ref. [20] focuses on the minimum rental and bandwidth costs of the system. Ref. [21] proposes the reputation-based server selection scheme, which can dynamically update the information of servers. Ref. [22] proposes the server selection approaches which can use GPU resources efficiently. Refs. [23] and [24] deal with the resource allocation in cloud gaming systems. In [23], the authors use the cognitive information of the cloud gaming systems to allocate resources dynamically to users. Ref. [24] focuses on the complex relationship among system performances, and derives the resource allocation by machine-learning based approaches. Compared with the previous papers [20], [21], [22], [23], [24], our work studies the server selection and resource allocation problem based on the predictive frame transmission scheme.

A few papers rely on prediction to optimize the performances of cloud gaming systems [25], [26]. Ref. [25] proposes a user action prediction approach, by which the cloud server can predict and pre-transmit future images to users so as to reduce the response delay of cloud gaming. Our work has several differences, including: 1) Ref. [25] requires additional bandwidth resources to transmit the predicted images, while our work only exploits the bandwidth resources that are already allocated to the user. Besides, the data transmission in [25] is based on a fixed policy that does not consider the network states, while our proposed approach can adaptively allocate packets to the predicted images based on the current network state and buffer state. 2) In [25], once the misprediction occurs, the terminals consume GPU resources to execute error compensation which might be challenging for mobile devices. In comparison, in our work, mispredictions are corrected by buffer update and data transmission. Ref. [26] studies the multi-layered video coding in cloud gaming systems, by which an image can be partitioned into multiple layers. In each frame, the system can predict some layers of the image before the receiving of the updated user action. In [26], the prediction is only limited to one frame, but the images of future frames are not predicted. In comparison, our work focuses on the prediction and pre-transmission of multiple future frames, considering the dynamics of networks.

3 System Model

3.1 Predictive Frame Transmission Scheme

We propose a predictive frame transmission scheme in cloud gaming systems assisted by MEC. With the prediction of future frames, the server can pre-transmit them to users. The structure of a game frame is shown in Fig. 2a, explained as follow.

1) *Instruction update*: In each frame, if the user generates a new instruction, it will be sent to the server. Otherwise, the server will not receive any update, and it indicates that the user maintains the instruction in the previous frame.
2) *Game logic and image renderer*: Based on the latest instruction, the server processes the game logic and renders the image of the current frame, and it also predicts the game logic and images of the future frames. If the instruction does not change in the future frames, these predicted images will be used to display. Otherwise, these predicted images will be deleted.
3) *Image download*: The server transmits the images of the current frame and predicted frames to the user via the wireless network. Since the bandwidth of the network is limited, the server can only send some of the images. Also, the bad network state may lead to the loss of images during the transmission.

TABLE 1
Main Notations

Symbol	Definition
t	the index of game frames
N	the number of images rendered in one frame, including one current image and $N-1$ predicted images
i	the index of the frame for the rendered images, where $i=1$ denotes the current image and $i>1$ denotes the predicted images
L	the number of packets sent from the server in each frame
P^{FLR}	the average FLR
P_t^{PLR}	the PLR in the tth frame
M	the total number of PLR states
$P_j^{\mathrm{PLR-M}}$	the state space of PLR for Markov model based network, where $1 \le j \le M$
$Q_{j,j'}^{\mathrm{PLR-M}}$	the state transition probability from $P_j^{\mathrm{PLR-M}}$ to $P_{j'}^{\mathrm{PLR-M}}$
$P_t^{\mathrm{PLR-O}}$	the observation of PLR at the tth frame
$\boldsymbol{P}_{t_1:t_2}^{\mathrm{PLR-O}}$	the observations of PLR from the t_1th frame to the t_2th frame
$P_t^{\mathrm{PLR-E}}$	the estimated PLR in the tth frame
P^{I}	the probability that user instruction changes
B_i	a bool value indicates if the image of the $(i+t-1)$th frame is cached by the buffer at the tth frame
$\boldsymbol{B}_t$	$\boldsymbol{B}_t=(B_1,B_2,B_3,\ldots,B_N)$ that denotes the state of the buffer at the tth frame
S_t	the state of the system, which is $S_t=\{P_t^{\mathrm{PLR}},\boldsymbol{B}_t\}$
x_i	the number of the packets that is allocated to the ith image
$\boldsymbol{X}_t$	$\boldsymbol{X}_t=(x_1,x_2,x_3,\ldots,x_N)$ that denotes the packet allocation in the tth frame
π	the packet allocation policy, which denotes the mapping from S_t to $\boldsymbol{X}_t$

4) *Image display*: Upon the image reception, the mobile device caches them in the buffer. At the end of each frame, the mobile device detects if the image of the current frame is already cached. If so, the mobile device displays the image; otherwise, the current frame is skipped, and a frame loss occurs. For the images of predicted frames, the mobile device will cache and display them at their frames, as long as the user instruction does not change. But if the instruction changes, they will be deleted.

With predictive frame transmission, the system can take full advantage of good network states by transmitting the images of future frames. When bad network states come, although the images cannot be successfully transmitted, the images of current frames can be displayed because they have been cached in the user buffer already. In Fig. 2b, the example demonstrates the predictive frame transmission scheme. In Frames 1 and 2, the server predicts and pre-transmits images of future frames to the user. In Frame 3, the image cannot be downloaded due to the packet loss in the bad network state. But the mobile device can still display the image which has already been cached in the buffer. In Frame 5, the user instruction changes. The previously predicted images are deleted, and newly rendered images are re-transmitted to the mobile device.

Let t denote the index of game frames where $t=1,2,\ldots$. Table 1 summarizes the main notations used throughout this paper.

3.2 Game Logic and Image Renderer

The server consumes CPU resources to process game logic, and consumes GPU resources to render images. We assume that the CPU and GPU frequency of the server are F^{CPU} and F^{GPU} respectively, which are known to the server. To render one image, the consumption of CPU cycles and GPU cycles are denoted by C^{CPU} and C^{GPU} respectively. C^{CPU} and C^{GPU} are determined by the specific game program, and we assume that they can be tested by the server. Accordingly, the time consumed to process game logic and renderer for one image is

$$D_{\mathrm{I}}=\frac{C^{\mathrm{CPU}}}{F^{\mathrm{CPU}}}+\frac{C^{\mathrm{GPU}}}{F^{\mathrm{GPU}}}. \tag{1}$$

In one frame, constrained by the computation resources and computing time, the server can render at most N images including one image of the current frame and $N-1$ predicted images of future frames. We assume that the total time allocated to game logic and image renderer is D_{G} in each frame. Accordingly, the maximum number of images that can be rendered in one frame is

$$N=\left\lfloor\frac{D_{\mathrm{G}}}{D_{\mathrm{I}}}\right\rfloor, \tag{2}$$

where $\lfloor x\rfloor$ indicates the largest integer no larger than x.

Let i denote the index of the rendered images, where $1\le i\le N$. If $i=1$, the first image indicates the one of the current frame. If $i>1$, the ith image is of the predicted frame whose index is $t+i-1$ at the tth frame.

3.3 Data Transmission

We consider that one image is transmitted to the user in one application-layer packet. Constrained by the bandwidth resources of the network, the server can send at most L packets to the user in one frame. Assume that the total bandwidth is W, the data size of one packet is ΔL, and the time period of one frame is ΔT. Then, L is calculated by

$$L=\left\lfloor\frac{W\Delta T}{\Delta L}\right\rfloor. \tag{3}$$

The application-layer packet loss rate (PLR) denotes the probability that packets cannot be successfully received by the mobile device within the period of one frame. The PLR is affected by many factors, including 1) the wireless channel state in the physical layer, 2) the congestion in the network layer, 3) the timeout in the application layer, and so on.

In our work, we will not focus on the comprehensive model of PLR, which is not helpful to apply the predictive frame transmission to real systems. In contrary, we simply use the PLR to indicate the state of the wireless network. On one hand, PLR is dominated by the wireless channel state and network congestions, which reflects the state of the whole network directly. On the other hand, PLR is easy to be measured by the system, and it is a fundamental indicator that matters the image transmission and frame loss. We assume that the PLR remains constant during each frame, but it may change in different frames. Let P_t^{PLR} denote the PLR in the tth frame. We consider Markov model based network and general model based network, respectively.

- *Markov model based network*: In Markov model based networks, the dynamics of P_t^{PLR} is determined by a discrete state space Markov chain. Assume that the total number of states is M, and the state space of PLR is denoted by $P_j^{\text{PLR-M}}$ where $1 \leq j \leq M$. The state transition is denoted by $Q_{j',j}^{\text{PLR-M}}$, that is

$$Q_{j',j}^{\text{PLR-M}} = \text{Prob}\{P_{t+1}^{\text{PLR}} = P_j^{\text{PLR-M}} | P_t^{\text{PLR}} = P_{j'}^{\text{PLR-M}}\}. \tag{4}$$

  The transition between different states indicates the dynamics of the network state. If the network is of high state fluctuation, the state transition probability is of a larger value. Otherwise, the network is relatively constant, and the state transition probability becomes small. The number of states and the value of state transition probability can model networks with different dynamics.
- *General model based network*: In general model based networks, we consider that the dynamics of network states are not known to the server. The server can only get the PLR of previously transmitted packets. In this case, the server samples the PLR by historical frames, and estimates the PLR of the current frame. Let $\boldsymbol{P}_{t_1:t_2}^{\text{PLR-O}}$ denote the observations of PLR samples from the t_1th frame to the t_2th frame. Based on the observations $\boldsymbol{P}_{1:t-1}^{\text{PLR-O}}$, the server estimates the PLR $P_t^{\text{PLR-E}}$ at the tth frame.

Markov model describes the dynamic changes of networks over time, which has been widely used by a number of researchers [13], [27]. But when a user accesses a new server, the user may not have and prior information about the network states. In this case, we will also study the networks based on general models, so that our proposed scheme can be easily applied to real systems.

3.4 Operations on Mobile Devices

The functions of the mobile device are to upload user instruction and to display images. In each frame, the probability that user instruction has changed compared with the previous frame is denoted by P^{I}. At the start of one frame, if the mobile device detects the change of the instruction, it will upload the update to the server; otherwise, it will not send any data. We assume that the instruction update can always be received by the server in time, because its data size is very small, typically 10 bytes.

The downloaded images are cached in the buffer of the mobile device. Let $\boldsymbol{B}_t = (B_1, B_2, B_3, \ldots, B_N)$ denote the state of the buffer in the tth frame. B_i is a bool variable that $B_i \in \{0, 1\}$, where $i = 1, 2, \ldots, N$. If $B_i = 1$, it denotes that the ith image is cached in the buffer; otherwise, $B_i = 0$ and the ith image is not cached. According to the definition of i, B_1 indicates the image state of the current frame; $\forall i > 1$, B_i indicates the image state of a predicted frame where the index of its frame is $t + i - 1$ at the tth frame.

At the end of each frame, the image of the current frame is displayed if $B_1 = 1$; otherwise, the display of the image is skipped and a frame loss happens. $\forall i > 1$ and $i < N$, B_i is replaced by the image of B_{i+1}, because one frame has passed and the index of predicted frames is decreased by one. We assume that the user will always report its buffer state to the server. But the server cannot get the state as soon as each packet is received or lost because it takes time to transmit the report. Therefore, we assume that the server gets the synchronized buffer state at the end of each frame.

3.5 Packet Allocation Policy

The system state includes the network state and the buffer state of images. If the network state is based on Markov model, it is acquired directly; otherwise, the system will estimate the network state based on the historical observations. Let S_t denote the state of the system at the tth frame, where

$$S_t = \begin{cases} \{P_j^{\text{PLR-M}}, \boldsymbol{B}_t\}, \text{Markov model based network.} \\ \{P_t^{\text{PLR-E}}, \boldsymbol{B}_t\}, \text{General model based network.} \end{cases} \tag{5}$$

Let $\boldsymbol{X}_t = (x_1, x_2, x_3, \ldots, x_N)$ denote the packet allocation in the tth frame. x_i indicates the number of packets that is allocated to the ith image, which satisfies $x_i \in [0, L]$. Constrained by the total number of packets, the following inequality holds in any frame.

$$\sum_{i=1}^{N} x_i \leq L. \tag{6}$$

At the tth frame, given the current PLR P_t^{PLR} and the packet allocation $\boldsymbol{X}_t$, the transition of buffer state is given by

$$B_i = \begin{cases} B_{i+1}, & \text{The frame index is decreased by one,} \quad (7a) \\ 0, & \text{With prob. } P^{\text{I}}, \quad (7b) \\ 1, & \text{With prob. } 1 - (P_t^{\text{PLR}})^{x_i}, \quad (7c) \end{cases}$$

where the state transition should be sequentially executed from (7a) to (7c). Eq. (7a) denotes the index of predicted frames is decreased by one, because one frame has passed. Eq. (7b) denotes the user instruction has changed, and the previously predicted images are deleted. Eq. (7c) denotes that x_i packets are allocated to the ith frame, and the success transmission probability is derived accordingly.

The system determines the packet allocation based on the system state in each frame. The packet allocation policy π is defined as the mapping from the system state S_t to the packet allocation $\boldsymbol{X}_t$, which is

$$\pi : S_t \longrightarrow \boldsymbol{X}_t. \tag{8}$$

If the network state is based on Markov model, π is a Markov policy that relies on the system state in the tth frame only. Otherwise, if the network state is based on the general model, π is a history dependent policy.

4 Problem Formulation and Analysis

4.1 Frame Loss Rate Minimization

Our objective is to minimize the average FLR. In each frame, we assume that the user gets a reward R if the image of the current frame is displayed. The utility function is defined as the long-term reward

$$U = \lim_{T\to\infty} \frac{1}{T}\sum_{t=1}^{T}[R\mathbb{I}(B_1 = 1|\boldsymbol{B}_t)], \tag{9}$$

where

$$\mathbb{I}(x) = \begin{cases} 1, & \text{if the condition } x \text{ holds,} \\ 0, & \text{otherwise.} \end{cases} \tag{10}$$

To maximize the utility, the optimization problem is formulated as Problem P1.

$$\max_{\pi} U \tag{P1}$$

$$s.t. \sum_{i=1}^{N} x_i \le L, \forall \boldsymbol{X}_t \text{ where } t = 1, 2, \ldots, \tag{C1.1}$$

where (C1.1) indicates the constrained number of packet transmission in each frame.

Let P^{FLR} denote the FLR, which is the probability that frame loss occurs.

$$P^{\text{FLR}} = \text{Prob}\{B_1 = 0\}, \forall t. \tag{11}$$

According to the Law of Large Numbers, the utility is a function of FLR, where,

$$U = R(1 - P^{\text{FLR}}). \tag{12}$$

Thus, Problem P2 is an equivalent problem of Problem P1, which is

$$\min_{\pi} P^{\text{FLR}} \tag{P2}$$

$$s.t. \tag{C1.1.}$$

Because Problem P1 and Problem P2 are equivalent, the solution to either of them leads to the minimization of FLR, which denotes the optimal policy π.

4.2 Upper Bound of FLR

We consider a special case that the cloud gaming system does not predict any future frames. In this case, $N = 1$ holds in all frames. All packets are allocated to the unique rendered image in each frame, which is a subset of the policies for the condition of $N > 1$. Thus, the FLR of the system performs the upper bound.

In the system with Markov model based networks, let $\boldsymbol{P}_j^{\text{PLR-M-S}} = (P_1^{\text{PLR-M-S}}, \ldots, P_N^{\text{PLR-M-S}})$ denote the stationary distribution of all states, where $P_j^{\text{PLR-M-S}}$ indicates the stationary distribution for the PLR of $P_j^{\text{PLR-M}}$. Let $Q^{\text{PLR-M}}$ denote the probability transition matrix for different PLRs. The stationary distribution can be derived by solving the linear equations as follows,

$$\boldsymbol{P}_j^{\text{PLR-M-S}} = Q^{\text{PLR-M}} \boldsymbol{P}_j^{\text{PLR-M-S}}. \tag{13}$$

Accordingly, the upper bound of FLR is

$$P_{\text{Upper}}^{\text{FLR}} = \sum_{j=1}^{M} P_j^{\text{PLR-M-S}} (P_j^{\text{PLR-M}})^L. \tag{14}$$

In the system with general model based networks, the distribution of P_t^{PLR} can be derived by sampling enough PLRs. Based on the distribution, the upper bound of FLR is

$$P_{\text{Upper}}^{\text{FLR}} = \int_0^1 \text{Prob}\{P_t^{\text{PLR}} = x\} x^L \, dx. \tag{15}$$

4.3 Lower Bound of FLR

We consider an ideal case that the system gets an immediate feedback of packet transmission as soon as the transmission is finished, no matter whether the packet is received by the user or is lost. With the assumption, the system can allocate each packet before the transmission of the packet, based on the real-time buffer state of the mobile device. In comparison, without the assumption, all packets are allocated simultaneously before their transmission, because the buffer state is synchronized at the end of each frame. The FLR with the assumption performs the lower bound, because it gets the superset of information compared with the system without the assumption. We have the following proposition for the optimal packet allocation policy.

Proposition 1. *In the system that always gets the feedback of packet transmission immediately, the optimal packet allocation policy is as follow. At the beginning of packet transmission, the system finds the minimum i that $B_i = 0$, and it sends the ith image in this packet.*

Proof. See Appendix A, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TMC.2022.3149056. □

Based on the optimal policy, the images of smaller i will be transmitted earlier compared with images of larger i. Thus, $B_i \ge B_{i'}$ always holds where $i < i'$. Let $\widehat{\boldsymbol{B}}_k$ denote the state of the buffer, that is

$$\widehat{\boldsymbol{B}}_k = (B_1 = 1, \ldots, B_k = 1, B_{k+1} = 0, \ldots, B_N = 0), \tag{16}$$

where $B_i = 1$ holds for $i \le k$ and $B_i = 0$ holds for $i > k$.

In the tth frame, given the PLR P_t^{PLR}, the state transition probability from $\widehat{\boldsymbol{B}}_{k'}$ to $\widehat{\boldsymbol{B}}_k$ is given by

$$Q_{k',k}^{\text{Lower}}(P_t^{\text{PLR}}) = (1 - P^{\text{I}})\mathbb{I}(l > 0)\binom{l}{L}(P_t^{\text{PLR}})^{L-l}(1 - P_t^{\text{PLR}})^l \tag{17}$$

$$+P^{\text{I}}\binom{k}{L}(P_t^{\text{PLR}})^{L-k}(1 - P_t^{\text{PLR}})^k, \tag{18}$$

where $l = k - k' + 1$. Eq. (17) indicates the state transition probability that user instruction does not change, and (18) indicates the state transition probability that user instruction changes.

Considering the distribution of P_t^{PLR}, the total probability of state transition is

$$Q_{k',k}^{\text{Lower}} = \begin{cases} \sum_{j=1}^{M} P_j^{\text{PLR-M-S}} Q_{k',k}^{\text{Lower}}(P_j^{\text{PLR-M}}), & \text{(19a)} \\ \int_0^1 \text{Prob}\{P_t^{\text{PLR}} = x\} Q_{k',k}^{\text{Lower}}(x)\, dx, & \text{(19b)} \end{cases}$$

where (19a) indicates the system with Markov model based network, and (19b) indicates the system with general model based network.

As a result, the states of buffer state can be modelled as a Markov chain. Let $\boldsymbol{P}^{\text{Lower-S}} = (P_0^{\text{Lower-S}}, \ldots, P_N^{\text{Lower-S}})$ denote the stationary distribution of all states, where $P_k^{\text{Lower-S}}$ indicates the stationary distribution for the state $\widehat{\boldsymbol{B}}_k$. Let Q^{Lower} denote the matrix of transition probabilities for all states. The stationary distribution can be derived by solving the linear equations as

$$\boldsymbol{P}^{\text{Lower-S}} = Q^{\text{Lower}} \boldsymbol{P}^{\text{Lower-S}}. \tag{20}$$

Accordingly, the lower bound of FLR is

$$P_{\text{Lower}}^{\text{FLR}} = P_0^{\text{Lower-S}}, \tag{21}$$

which represents the probability that the mobile device does not cache any image at the end of each frame.

4.4 Cloud Gaming Metrics

As introduced in Section 1, the QoS of cloud gaming is based on many metrics, including FPS, FLR, the response delay, and the bandwidth usage. FPS and FLR jointly determine the smoothness of gaming [28], and therefore the high FPS and low FLR are directly required by users. In this part, we continue to show that the response delay and bandwidth usage are also highly related to FLR and PLR.

4.4.1 Mean Round Trip Delay

The round trip delay T_{RTT} is defined as the time duration from the update of a user instruction to the update of the corresponding image. Thus, T_{RTT} indicates the response delay of cloud gaming.

The time period of a game frame is ΔT. If one image is updated at the end of the frame that a new user instruction is received, $T_{\text{RTT}} = \Delta T$. Otherwise, T_{RTT} depends on the number of frame loss since the user instruction. Accordingly, the mean value of T_{RTT} is based on the distribution of FLR, which is given by

$$\text{E}(T_{\text{RTT}}) = \int_0^1 \text{Prob}\{P^{\text{FLR}} = x\} \left[\sum_{k=1}^{\infty} k(1-x)x^{(k-1)} \right] \Delta T\, dx. \tag{22}$$

4.4.2 Packet Overhead of Predicted Frames

The network consumes bandwidth to transmit the predicted frames. The packet overhead of the predicted frames depends on the packet allocation policy π. Given a policy π and the distribution of system states, the packet overhead is

$$O_\pi = \sum_{S_t} \text{Prob}\{S_t\} O_\pi(S_t), \tag{23}$$

where

$$O_\pi(S_t) = O_\pi(S_t, \boldsymbol{X}_t) = \sum_{i=2}^{N} x_i. \tag{24}$$

In Eq. (24), the packet allocation $\boldsymbol{X}_t$ is jointly determined by the system state S_t and the policy π. The number of packets allocated to the current frame is x_1, and other packets are allocated to the predicted frames. Thus, the packet overhead is based on the sum from x_2 to x_n.

5 Optimal Policy for Cloud Gaming Systems With Markov Model Based Network

In this section, we study the optimal packet allocation policy of the predictive frame transmission scheme, based on Markov model of the network. We first analyze the FLR. Then, we use MDP to derive the optimal policy. We also propose a near-optimal policy which greatly decreases the algorithm complexity. Finally, we extend the predictive frame transmission scheme to the multi-server multi-user scenario.

5.1 Theoretical Analysis on Frame Loss Rate

With the Markovian network states, the entire system can be modeled as a discrete-time and discrete-space Markov chain. In the tth frame, the system state S_t is determined by a $N+1$ dimension vector $(P_j^{\text{PLR-M}}, B_1, \ldots, B_N)$, where $P_j^{\text{PLR-M}}$ indicates the network state and B_i indicates the buffer state of the ith image. For simplicity, we define the state as

$$\boldsymbol{S}_{j,B_1,\ldots,B_N}^{\text{CM}} = (P_j^{\text{PLR-M}}, B_1, \ldots, B_N). \tag{25}$$

Because the system is Markov, the policy π determines the packet allocation in each frame, based on the current system state only.

$$\pi : \boldsymbol{S}_{j,B_1,\ldots,B_N}^{\text{CM}} \longrightarrow (x_1, \ldots, x_N). \tag{26}$$

For the policy π, the state transition probability from $\boldsymbol{S}_{j',B_1',\ldots,B_N'}^{\text{CM}}$ to $\boldsymbol{S}_{j,B_1,\ldots,B_N}^{\text{CM}}$ is

$$\begin{aligned} &Q_\pi^{\text{CM}}(\boldsymbol{S}_{j',B_1',\ldots,B_N'}^{\text{CM}}, \boldsymbol{S}_{j,B_1,\ldots,B_N}^{\text{CM}}) \\ &\quad = \text{Prob}\{\boldsymbol{S}_{j,B_1,\ldots,B_N}^{\text{CM}} | \boldsymbol{S}_{j',B_1',\ldots,B_N'}^{\text{CM}}, (x_1, \ldots, x_N)\} \\ &\quad = (1-P^{\text{I}}) Q_{j',j}^M \prod_{\text{C}_1^{\text{CM}}} [1 - (P_j^{\text{PLR-M}})^{x_i}] \prod_{\text{C}_2^{\text{CM}}} (P_j^{\text{PLR-M}})^{x_i} \\ &\qquad + P^{\text{I}} Q_{j',j}^M \prod_{\text{C}_3^{\text{CM}}} [1 - (P_j^{\text{PLR-M}})^{x_i}], \end{aligned} \tag{27}$$

where

$$\begin{aligned} &\text{C}_1^{\text{CM}}\text{: } \forall i, \text{that } B_i = 1 \text{ and } B_{i+1}^{'} = 0, \\ &\text{C}_2^{\text{CM}}\text{: } \forall i, \text{that } B_i = 0 \text{ and } x_i > 0, \\ &\text{C}_3^{\text{CM}}\text{: } \forall i, \text{that } B_i = 1. \end{aligned}$$

C_1^{CM} denotes the condition that the ith image is successfully received by the user; C_2^{CM} denotes the condition that the ith image is lost during the transmission; C_3^{CM} denotes the condition that the instruction has changed and the ith image is re-transmitted to the user.

With the state transition probability, we can get the stationary distribution of the Markov chain. Let $P_{\pi}^{\text{CM-S}}$ denote the stationary distributions of the state space, which is a vector of elements $P_{\pi}^{\text{CM-S}}(\boldsymbol{S}_{j,B_1,\ldots,B_N}), \forall j, B_1, \ldots, B_N$. Let Q_{π}^{CM} denote the state transition matrix of the state space. The stationary distribution can be solved by

$$\boldsymbol{P}_{\pi}^{\text{CM-S}} = Q_{\pi}^{\text{CM}} \boldsymbol{P}_{\pi}^{\text{CM-S}}. \tag{28}$$

The FLR is the sum of the stationary state probability that $B_1 = 0$, which is

$$P_{\pi}^{\text{FLR}-\text{CM}} = \sum_{\forall j, B_2, \ldots, B_N, \text{and} B_1 = 0} P_{\pi}^{\text{CM-S}}(\boldsymbol{S}_{j,B_1,\ldots,B_N}). \tag{29}$$

Therefore, the FLR can be derived theoretically with policy π. Taking the FLR to the Problem P2, we can further optimize π so as to minimize the FLR, which is

$$\min_{\pi} P_{\pi}^{\text{FLR}-\text{CM}}. \tag{P3}$$

In the following part, we will use MDP to get the optimal solutions.

5.2 Optimal Policy by Markov Decision Process

Based on the Markov system, the FLR minimization problem evolves as a discrete-time and discrete-space MDP. We model the MDP problem with the following tuples.

- *State space*: $S_t = \boldsymbol{S}_{j,B_1,\ldots,B_N}^{\text{CM}}$.
- *Action space*: $\boldsymbol{X}_t = (x_1, \ldots, x_N)$.
- *State transition*:

$$Q_{\pi}^{\text{CM}}(S_t, S_{t+1}) = Q_{\pi}^{\text{CM}}\left(\boldsymbol{S}_{j',B_1',\ldots,B_N'}^{\text{CM}}, \boldsymbol{S}_{j,B_1,\ldots,B_N}^{\text{CM}}\right).$$

- *Reward*:

$$\text{R}_t(\boldsymbol{S}_{j,B_1,\ldots,B_N}^{\text{CM}}) = \begin{cases} R, \text{ If } B_1 = 1. \\ 0, \text{ If } B_1 = 0. \end{cases}$$

The value function of a state indicates the expected sum of the current and potential rewards. Let γ denote the discount factor of the potential reward, where $0 \leq \gamma < 1$. Thus, the value function of S_t based on the policy π is

$$V_{\pi}(S_t) = \text{E}_{\pi}[\text{R}_t(S_t) + \gamma V_{\pi}(S_{t+1}) | S_t]. \tag{30}$$

Taking the state space and state transition probability to (30), the Bellman equation of the value function is

$$V_{\pi}\left(\boldsymbol{S}_{j,B_1,\ldots,B_N}^{\text{CM}}\right) = \text{R}_t\left(\boldsymbol{S}_{j,B_1,\ldots,B_N}^{\text{CM}}\right) + \gamma \sum_{\forall j', B_1', \ldots, B_N'} Q^{\text{CM}}\left(\boldsymbol{S}_{j,B_1,\ldots,B_N}^{\text{CM}}, \boldsymbol{S}_{j',B_1',\ldots,B_N'}^{\text{CM}}\right) V_{\pi}\left(\boldsymbol{S}_{j',B_1',\ldots,B_N'}^{\text{CM}}\right). \tag{31}$$

The optimal policy π^* maximizes the value functions. Because the maximization of reward (Problem P1) and the minimization of FLR (Problem P2) are equivalent, π^* can actually minimize the FLR.

$$\pi^* = \arg\max_{\pi} V_{\pi}\left(\boldsymbol{S}_{j,B_1,\ldots,B_N}^{\text{CM}}\right). \tag{32}$$

The maximum value function based on π^* is defined as

$$V_*\left(\boldsymbol{S}_{j,B_1,\ldots,B_N}^{\text{CM}}\right) = V_{\pi^*}\left(\boldsymbol{S}_{j,B_1,\ldots,B_N}^{\text{CM}}\right). \tag{33}$$

We use value iteration to derive the optimal policy, as described in Algorithm 1.

Algorithm 1. Value Iteration Algorithm

1: **for** each $j, B_1, \ldots, B_N$ **do**
  $V_{\pi_0}(\boldsymbol{S}_{j,B_1,\ldots,B_N}^{\text{CM}}) = 0$
2: **end for**
3: **repeat**
4: **for** each $j, B_1, \ldots, B_N$ **do**
5: $V_{\pi_k}(\boldsymbol{S}_{j,B_1,\ldots,B_N}^{\text{CM}}) =$
6: $\max_{(x_1,\ldots,x_n)} \text{R}_t(\boldsymbol{S}_{j,B_1,\ldots,B_N}^{\text{CM}}) +$
7: $\gamma \sum_{\forall \boldsymbol{S}'} Q^{\text{CM}}(\boldsymbol{S}_{j,B_1,\ldots,B_N}^{\text{CM}}, \boldsymbol{S}') V_{\pi_{k-1}}(\boldsymbol{S}')$
8: **end for**
9: $\Delta \leftarrow |V_{\pi_k}(\boldsymbol{S}_{j,B_1,\ldots,B_N}^{\text{CM}}) - V_{\pi_{k-1}}(\boldsymbol{S}_{j,B_1,\ldots,B_N}^{\text{CM}})|$
10: **until** $\Delta \leq \alpha$
11: Return policy π^*

In Algorithm 1, the termination condition for the iteration is that the change of value function Δ is not larger the threshold α. We prove the convergence of Algorithm 1 as follow.

Proposition 2. *The Algorithm 1 converges to the optimal solution.*

Proof. See Appendix B, available in the online supplemental material. □

We further analyze the structure of the optimal policy. First, we define the packet-based value increment function, which denotes the marginal gain of value function by allocating one more packet to a specific image. Given the current state $S_t = \boldsymbol{S}_{j,B_1,\ldots,B_N}^{\text{CM}}$ and the packets that have already been allocated $\widehat{\boldsymbol{X}}_t = (\widehat{x}_1, \ldots, \widehat{x}_N)$, the packet-based value increment function of the ith image is defined as

$$\begin{aligned} &\Delta V_*^{\text{Packet}}(i | S_t, \widehat{\boldsymbol{X}}_t) \\ &= [(P_j^{\text{PLR-M}})^{\widehat{x}_i} - (P_j^{\text{PLR-M}})^{\widehat{x}_i + 1}] \\ &\quad [V_*(S_t | \widehat{\boldsymbol{X}}_t, B_i = 1) - V_*(S_t | \widehat{\boldsymbol{X}}_t, B_i = 0)], \end{aligned} \tag{34}$$

where $V_*(S_t | \widehat{\boldsymbol{X}}_t, B_i = b)$ denotes the expectation of value function that packets are allocated by $\widehat{\boldsymbol{X}}_t$ and $B_i = b$. Eq. (34) indicates that $\Delta V_*^{\text{Packet}}(i | S_t, \widehat{\boldsymbol{X}}_t)$ can be obtained by the value function, by allocating the next packet to the ith image. With more packets allocated to the same image, its packet-based value increment function becomes smaller.

With the packet-based value increment function, the unique structure of the optimal policy is stated as follow.

Proposition 3. *The optimal packet allocation policy π^* is a L-step greedy policy based on $\Delta V_*^{\text{Packet}}(i | S_t, \widehat{\boldsymbol{X}}_t)$. In each step, the policy allocates one packet to the ith image that maximizes $\Delta V_*^{\text{Packet}}(i | S_t, \widehat{\boldsymbol{X}}_t)$. Then, $\widehat{\boldsymbol{X}}_t$ is updated based on $\widehat{X}_i = \widehat{X}_i + 1$, and $\Delta V_*^{\text{Packet}}(i | S_t, \widehat{\boldsymbol{X}}_t)$ is updated based on the $\widehat{\boldsymbol{X}}_t$.*

Proof. See Appendix C, available in the online supplemental material. □

In Proposition 3, we demonstrate how the system allocates packets to all rendered images in each frame. The optimal policy is of L steps, because it allocates one packet in each step and totally L packets are to be allocated. In one step, the system greedily allocates the packet to the image that has the maximum marginal gain of value function, which is actually determined by $\Delta V_*^{\text{Packet}}(i|S_t, \widehat{\boldsymbol{X}}_t)$. As one packet is allocated, the value increment functions change, and the system should re-calculate them. After L steps, all packets are allocated, leading to the maximum value function.

5.3 Nearest-Future-Based Two-Image Transmission Policy

The optimal policy can be derived by MDP, but the size of the policy space is $\binom{L+N-1}{N-1}$, which increases as a factorial function with increasing N and L. This results in the curse of dimensionality in solving MDP problems [29]. In this part, we design a policy that has the near-optimal performance with low complexity.

First, we define the image-based value increment function $\Delta V_*^{\text{Image}}(i|S_t)$. Given the system state $S_t = \boldsymbol{S}^{\text{CM}}_{j,B_1,\ldots,B_N}$ that $B_i = 0$, $\Delta V_*^{\text{Image}}(i|S_t)$ denotes the increment of value function by making $B_i = 1$. Concretely,

$$\begin{aligned}&\Delta V_*^{\text{Image}}(i|S_t)\\ &= V_*(\boldsymbol{S}^{\text{CM}}_{j,B_1,\ldots,B_i=1,\ldots,B_N}) - V_*(\boldsymbol{S}^{\text{CM}}_{j,B_1,\ldots,B_i=0,\ldots,B_N}),\end{aligned} \tag{35}$$

which indicates the value obtained by successfully transmitting the ith image to the user. Because the optimal policy maximizes the value function, $\Delta V_*(i|S_t)$ reveals the priority for transmitting the ith image. If $\Delta V_*^{\text{Image}}(i|S_t)$ is larger, the ith image is of higher priority to be transmitted to the user.

Proposition 4. *For $S_t = \boldsymbol{S}^{\text{CM}}_{j,B_1,\ldots,B_N}$, $\forall j, B_1, \ldots, B_N$, if $i < i^{'}$, the following inequation holds:*

$$\Delta V_*^{\text{Image}}(i|S_t) > \Delta V_*^{\text{Image}}(i^{'}|S_t). \tag{36}$$

Proof. See Appendix D, available in the online supplemental material. □

Proposition 4 indicates that the value increment of an image becomes larger, if its frame index is smaller. In other words, the image to be displayed in a nearer future frame is of higher priority. This makes sense. Because $i < i^{'}$, the $i^{'}$th image will be displayed later compared with the ith image. Thus, the $i^{'}$th image has a larger probability to be deleted, due to the instruction change. Even if the instruction does not change, the $i^{'}$th image has more frames to wait for transmission compared with the ith image.

We now consider a special case that the system only allocates packets to two images. In this condition, we have the following proposition for the optimal packet allocation policy.

Proposition 5. *Consider any state that $S_t = \boldsymbol{S}^{\text{CM}}_{j,B_1,\ldots,B_N}$. If all packets are allocated to the ith image and $i^{'}$th image only, by relaxing x_i and $x_{i'}$ as continuous variables, the optimal packet allocation problem is convex. The optimal solution is based on the water-filling structure, which is*

$$\begin{cases} x_i^{\text{WL}} = \lambda^* - \log_{P_j^{\text{PLR-M}}}(V_2 - V_3), \\ x_{i'}^{\text{WL}} = \lambda^* - \log_{P_j^{\text{PLR-M}}}(V_1 - V_3), \end{cases} \tag{37}$$

where

$$\begin{cases} V_1 = \Delta V_*^{\text{Image}}(i|S_t), \\ V_2 = \Delta V_*^{\text{Image}}(i^{'}|S_t), \\ V_3 = \Delta V_*^{\text{Image}}(i, i^{'}|S_t), \\ \lambda^* = \frac{1}{2}[L + \log_{P_j^{\text{PLR-M}}}(V_1 - V_3)(V_2 - V_3)]. \end{cases} \tag{38}$$

Proof. See Appendix E, available in the online supplemental material. □

In Proposition 5, the optimal packet allocation can be derived in closed-form, if all packets are allocated to two images only. The optimal packet allocation is of the water-filling structure. Although the Proposition 5 relax x_i and $x_{i'}$ as continuous variables, it is easy to extend them to integer variables. Because the value function is convex with respect to x_i and $x_{i'}$, the optimal packet allocation is either the floor or ceil of x_i and $x_{i'}$.

Inspired by Proposition 4 and Proposition 5, we propose the nearest-future-based two-image transmission policy. In each frame, the policy only allocates packets to two images. The two images are the ones that are of the smallest frame index (of the nearest future), and are not yet received by the users. Thereafter, packets are allocated to the two images by the water-filling structure. The corresponding policy is shown in Algorithm 2.

Algorithm 2. Nearest-Future-Based Two-Image Transmission Algorithm

1: **for** each $j, B_1, \ldots, B_N$ **do**
  $V_{\pi_0}(\boldsymbol{S}^{\text{CM}}_{j,B_1,\ldots,B_N}) = 0$
2: **end for**
3: **repeat**
4: **for** each $j, B_1, \ldots, B_N$ **do**
  Get i and $i^{'}$ of the largest $\Delta V_*^{\text{Image}}(i|\boldsymbol{S}^{\text{CM}}_{j,B_1,\ldots,B_N})$
  Get x_i and $x_{i'}$ based on water-filling.
  Compare two policies $\pi_k^{\text{A}} = (\lceil x_i \rceil, \lfloor x_{i'} \rfloor)$ and $\pi_k^{\text{B}} = (\lfloor x_i \rfloor, \lceil x_{i'} \rceil)$, and
  $V_{\pi_k}(\boldsymbol{S}^{\text{CM}}_{j,B_1,\ldots,B_N}) = \max_{\{\pi_k^{\text{A}}, \pi_k^{\text{B}}\}} \text{R}_t(\boldsymbol{S}^{\text{CM}}_{j,B_1,\ldots,B_N}) +$
  $\gamma \sum_{\forall S'} Q^{\text{CM}}(\boldsymbol{S}^{\text{CM}}_{j,B_1,\ldots,B_N}, \boldsymbol{S}') V_{\pi_{k-1}}(\boldsymbol{S}')$
5: **end for**
6: $\Delta \leftarrow |V_{\pi_k}(\boldsymbol{S}^{\text{CM}}_{j,B_1,\ldots,B_N}) - V_{\pi_{k-1}}(\boldsymbol{S}^{\text{CM}}_{j,B_1,\ldots,B_N})|$
7: **until** $\Delta \leq \theta$
8: Return policy π^*

Algorithm 2 has good performance as well as low complexity. In Algorithm 1, all the different packet allocations are compared in each iteration, and the number of the action space is $\binom{L+N-1}{N-1}$. In comparison, Algorithm 2 can derive the optimal packet allocation by water-filling directly, which decreases the number of compared actions from $\binom{L+N-1}{N-1}$ to 2. Because Algorithm 2 emphasizes on the most important

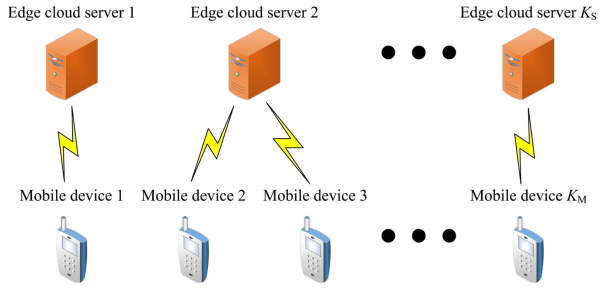


Fig. 3. The multi-server multi-user scenario.

two images for transmission, we will show that it has a near-optimal performance in Section 7.

Why two-images, yet not one-image? If all packets are allocated to one image, only the image of the current frame can be transmitted to the user in each frame. In this case, the system degrades to the one without predictive frame transmission. Why two-images, yet not three-images? If all packets are allocated to three images, the optimal packet allocation problem is non-convex, and the system should compare all the different packet allocations. As a result, the number of the action space becomes a factorial function to N and L.

5.4 Extension to the Multi-Server Multi-User Scenario

In the previous analysis, we mainly focus on the single-user system. Besides, the multi-user scenario is also important that multiple edge cloudlets can adaptively serve users in terms of load balancing, which has attracted great attentions in many papers [20], [21], [22]. Thus, in this part, we extend the proposed predictive frame transmission scheme to the multi-server multi-user scenario.

As shown in Fig. 3, totally K_{S} edge cloud servers can offload the computation from K_{M} mobile devices. Let k_{S} and k_{M} denote the index of the edge cloud server and the mobile device, respectively. Each mobile device connects to one edge server, and one server can serve multiple mobile devices simultaneously.

For the k_{M}th mobile device, the FLR is denoted by $P^{\mathrm{FLR}}_{k_{\mathrm{M}}}$. Let $P^{\mathrm{PLR}}_{k_{\mathrm{S}},k_{\mathrm{M}}}$ denote the network state between the k_{S}th server and the k_{M}th mobile device, which is assumed to be known to the system. Thus, $P^{\mathrm{FLR}}_{k_{\mathrm{M}}}$ is a function of π, $P^{\mathrm{PLR}}_{k_{\mathrm{S}},k_{\mathrm{M}}}$, N and L. Because π can be derived by Algorithm 1 and $P^{\mathrm{PLR}}_{k_{\mathrm{S}},k_{\mathrm{M}}}$ is known, $P^{\mathrm{FLR}}_{k_{\mathrm{M}}}$ is actually determined by N and L. N is a function of the CPU frequency F^{CPU} and the GPU frequency F^{GPU}, and L is a function of the bandwidth resources W. Accordingly, $P^{\mathrm{FLR}}_{k_{\mathrm{M}}}$ is a function based on F^{CPU}, F^{GPU} and W, that is

$$\begin{aligned} & P^{\mathrm{FLR}}_{k_{\mathrm{M}}} \\ &= \mathrm{F}^{\mathrm{FLR}}\left(N, L|\pi, P^{\mathrm{PLR}}_{k_{\mathrm{S}},k_{\mathrm{M}}}\right) \\ &= \mathrm{F}^{\mathrm{FLR}}\left(F^{\mathrm{CPU}}, F^{\mathrm{GPU}}, W\middle|\pi, P^{\mathrm{PLR}}_{k_{\mathrm{S}},k_{\mathrm{M}}}\right). \end{aligned} \tag{39}$$

To serve multiple users, the system should jointly determine: 1) *User schedule*: how to schedule users to different edge servers; 2) *Resource allocation*: how to allocate the CPU frequency, GPU frequency and bandwidth resources of each edge server to the multiple users. For the k_{S}th server, assume that the total CPU frequency is $F^{\mathrm{CPU-Max}}_{k_{\mathrm{S}}}$, the total GPU frequency is $F^{\mathrm{GPU-Max}}_{k_{\mathrm{S}}}$, and the total bandwidth resources are $W^{\mathrm{Max}}_{k_{\mathrm{S}}}$. Let $\boldsymbol{Y}_{k_{\mathrm{M}}} = [y_{1,k_{\mathrm{M}}}, \ldots, y_{k_{\mathrm{S}},\ k_{\mathrm{M}}}, \ldots, y_{K_{\mathrm{S}},k_{\mathrm{M}}}]$ indicate the user schedule of the k_{M}th mobile device, where $y_{k_{\mathrm{S}},k_{\mathrm{M}}} = 1$ denotes the schedule to the k_{S}th server and $y_{k_{\mathrm{S}},k_{\mathrm{M}}} = 0$ otherwise. Let $F^{\mathrm{CPU}}_{k_{\mathrm{S}},k_{\mathrm{M}}}$, $F^{\mathrm{GPU}}_{k_{\mathrm{S}},k_{\mathrm{M}}}$ and $W_{k_{\mathrm{S}},k_{\mathrm{M}}}$ indicate the CPU frequency, GPU frequency and bandwidth resources allocated to the k_{M}th mobile device in the k_{S}th server, respectively. To minimize the mean FLR of all users, the optimization problem is

$$\min \frac{1}{K_{\mathrm{M}}} \sum_{k_{\mathrm{M}}=1}^{K_{\mathrm{M}}} P^{\mathrm{FLR}}_{k_{\mathrm{M}}}. \tag{P4}$$

Based on the user schedule and resource allocation, Problem P4 can be transformed into Problem P5, which is presented as follow.

$$\min_{\boldsymbol{Y}_{k_{\mathrm{M}}}, F^{\mathrm{CPU}}_{k_{\mathrm{S}},k_{\mathrm{M}}}, F^{\mathrm{GPU}}_{k_{\mathrm{S}},k_{\mathrm{M}}}, W_{k_{\mathrm{S}},k_{\mathrm{M}}}}} \sum_{k_{\mathrm{S}}=1}^{K_{\mathrm{S}}} \sum_{k_{\mathrm{M}}=1}^{K_{\mathrm{M}}} \mathbb{I}(y_{k_{\mathrm{S}},k_{\mathrm{M}}} = 1)$$

$$\mathrm{F}^{\mathrm{FLR}}(F^{\mathrm{CPU}}_{k_{\mathrm{S}},k_{\mathrm{M}}}, F^{\mathrm{GPU}}_{k_{\mathrm{S}},k_{\mathrm{M}}}, W_{k_{\mathrm{S}},k_{\mathrm{M}}}|\pi^*, P^{\mathrm{PLR}}_{k_{\mathrm{S}},k_{\mathrm{M}}}) \tag{P5}$$

$$s.t. \sum_{k_{\mathrm{S}}=1}^{K_{\mathrm{S}}} y_{k_{\mathrm{S}},k_{\mathrm{M}}} = 1, \forall k_{\mathrm{M}}, \tag{C5.1}$$

$$\sum_{k_{\mathrm{M}}=1}^{K_{\mathrm{M}}} \mathbb{I}(y_{k_{\mathrm{S}},k_{\mathrm{M}}} = 1) F^{\mathrm{CPU}}_{k_{\mathrm{S}},k_{\mathrm{M}}} \le F^{\mathrm{CPU-Max}}_{k_{\mathrm{S}}}, \forall k_{\mathrm{S}}, \tag{C5.2}$$

$$\sum_{k_{\mathrm{M}}=1}^{K_{\mathrm{M}}} \mathbb{I}(y_{k_{\mathrm{S}},k_{\mathrm{M}}} = 1) F^{\mathrm{GPU}}_{k_{\mathrm{S}},k_{\mathrm{M}}} \le F^{\mathrm{GPU-Max}}_{k_{\mathrm{S}}}, \forall k_{\mathrm{S}}, \tag{C5.3}$$

$$\sum_{k_{\mathrm{M}}=1}^{K_{\mathrm{M}}} \mathbb{I}(y_{k_{\mathrm{S}},k_{\mathrm{M}}} = 1) W_{k_{\mathrm{S}},k_{\mathrm{M}}} \le W^{\mathrm{Max}}_{k_{\mathrm{S}}}, \forall k_{\mathrm{S}}, \tag{C5.4}$$

where (C5.1) denotes that one mobile device is scheduled to one edge server, (C5.2) and (C5.3) indicate the constrained CPU and GPU frequency of each edge server respectively, and (C5.4) indicates the constraint of bandwidth resources.

Problem P5 is a mixed integer nonlinear program (MINLP) problem. We propose Algorithm 3 to jointly derive the optimal user schedule and resource allocation. The algorithm compares all different user schedules under the constraint (C5.1). For a specific user schedule, each edge server allocates resources to the served mobile devices. Let $\boldsymbol{Y}_{k_{\mathrm{S}}} = [y_{k_{\mathrm{S}},k_{\mathrm{M}}(1)}, \ldots, y_{k_{\mathrm{S}},k_{\mathrm{M}}(G_{k_{\mathrm{S}}})}]$ denote the users scheduled to the k_{S}th server, where totally $G_{k_{\mathrm{S}}}$ users are served by the server.

Under a user schedule, the optimal resource allocation can be derived by dynamic programming. The algorithm decomposes the resource allocation of each server into $G_{k_{\mathrm{S}}}$ stages. In each stage, the algorithm will consider serving a new user, and all served users are jointly considered after $G_{k_{\mathrm{S}}}$ stages. Let $g_{k_{\mathrm{S}}}$ denotes the index of each stage. The value function of the dynamic programming is defined as $V_{\mathrm{DP}}(g_{k_{\mathrm{S}}}, f^{\mathrm{CPU}}, f^{\mathrm{GPU}}, w)$. When the edge server is of the CPU frequency f^{CPU}, GPU frequency f^{GPU} and bandwidth resources w, $V_{\mathrm{DP}}(g_{k_{\mathrm{S}}}, f^{\mathrm{CPU}}, f^{\mathrm{GPU}}, w)$ denotes the minimum sum of FLR from the first user to the $g_{k_{\mathrm{S}}}$th user. The corresponding resource allocation is defined as $R_{k_{\mathrm{S}}}(g_{k_{\mathrm{S}}}, f^{\mathrm{CPU}}, f^{\mathrm{GPU}}, w)$.

In the g_{k_S}th stage, the $k_M(g_{k_S})$th user will be taken into consideration. The algorithm compares all different possible resource allocations for the user, and $V_{DP}(g_{k_S}, f^{CPU}, f^{GPU}, W)$ is updated by the minimum one. The transition function of the dynamic programming is given by

$$\begin{aligned} &V_{DP}(g_{k_S}, f^{CPU}, f^{GPU}, w) \\ &= \min_{\hat{f}^{CPU}, \hat{f}^{GPU}, \hat{w}} V_{DP}(g_{k_S}-1, f^{CPU}-\hat{f}^{CPU}, f^{GPU}-\hat{f}^{GPU}, w-\hat{w}) \\ &\quad + F^{FLR}(\hat{f}^{CPU}, \hat{f}^{GPU}, \hat{w}|\pi^*, P^{PLR}_{k_S,k_M(g_{k_S})}). \end{aligned} \tag{40}$$

In each stage, $\forall f^{CPU}, f^{GPU}, w$ that meets the constraints $f^{CPU} \le F^{CPU-Max}_{k_S}$, $f^{GPU} \le F^{GPU-Max}_{k_S}$, and $w \le W^{Max}_{k_S}$ (the constraints (C5.2)-(C5.4)), $V_{DP}(g_{k_S}, f^{CPU}, f^{GPU}, w)$ and $R_{k_S}(g_{k_S}, f^{CPU}, f^{GPU}, w)$ will be updated by the transition function. This indicates the minimum FLR for any given amount of resources at the stage.

After G_{k_S} stages, all served users are considered, and $V_{DP}(G_{k_S}, F^{CPU-Max}_{k_S}, F^{GPU-Max}_{k_S}, W^{Max}_{k_S})$ indicates the minimum FLR with the total resources. The corresponding optimal resource allocation is $R_{k_S}(G_{k_S}, F^{CPU-Max}_{k_S}, F^{GPU-Max}_{k_S}, W^{Max}_{k_S})$. Indeed, the minimum FLR and the resource allocation are based on the specific user schedule. By comparing all of the different user schedules, Algorithm 3 can derive the optimal user schedule and resource allocation jointly.

Algorithm 3. Dynamic-Programming-Based Algorithm for the Optimal User Schedule and Resource Allocation

```
for each {Y_1, ..., Y_{K_M}} under (C5.1) do
  for each k_S do
    for g_{k_S} = 1, ..., G_{k_S} do
      for each f^CPU, f^GPU, w under (C5.2)-(C5.4) do
        Update V_DP(g_{k_S}, f^CPU, f^GPU, w) by (40)
        Update R_{k_S}(g_{k_S}, f^CPU, f^GPU, w)
      end for
    end for
  end for
end for
```

In the multi-server multi-user scenario, the system can benefit from load balancing, and serve different users in terms of their different resource requirements and network states. For users with high computation requirements, the system will allocate more CPU and GPU frequency. For users with bad network states, the system will allocate more bandwidth resources. Even if the users are of relatively high mobility, the system can dynamically schedule these users to different edge servers based on the network state changes. In Section 7, we will show that the optimal user schedule and resource allocation can efficiently reduce the mean FLR of multiple users.

6 Optimal Policy for Cloud Gaming Systems With General Model Based Network

In this section, we study the packet allocation policy for the system in which the network state is based on general models. Because the model does not rely on any prior information of network states, the policy can be easily applied to practical cloud gaming systems.

6.1 Packet Loss Rate Estimation

The packet allocation policy depends on the network state and the buffer state jointly. The buffer state is reported by the user in each frame. But with the general model based networks, the PLR is not known to the system directly. Therefore, the system gets the historical PLR samples, and estimates the PLR of the current frame based on the historical observations. Each sample of PLR is defined as the PLR over a period of time. In this paper, for simplicity, we assume that a sample denotes the PLR in one frame, which is derived from the L transmitted packets in the frame. To extend our method to practical systems, a sample can be derived from several frames rather than one, so that it can be more precise when L is relatively small.

There are two possibilities for PLR in each frame. First of all, the PLR remains constant as the previous frame. Second of all, the PLR has changed compared with the previous frame. Thus, the estimation of PLR is actually the detection of the change of PLR, which is an online change point detection problem [30]. We use Bayesian approaches to solve the PLR estimation problem.

In the tth frame, the observations of PLRs are from the 1st frame to the $(t-1)$th frame, which are denoted by $\boldsymbol{P}^{PLR\text{-}O}_{1:t-1}$. Let r_t denote the number of frames since the last change of PLR. Let $P(r_t|\boldsymbol{P}^{PLR\text{-}O}_{1:t-1})$ denote the distribution of r_t given the observations $\boldsymbol{P}^{PLR\text{-}O}_{1:t-1}$. Thus, the estimation of PLF is derived by full probability formula, which is

$$P^{PLR-E}_t = \sum_{r_t} \left[P(r_t|\boldsymbol{P}^{PLR\text{-}O}_{1:t-1}) \left(\frac{1}{r_t} \sum_{n=t-r_t}^{t-1} P^{PLR\text{-}O}_n \right) \right]. \tag{41}$$

The distribution of $P(r_t|\boldsymbol{P}^{PLR\text{-}O}_{1:t-1})$ can be calculated recursively as follow.

$$P(r_t|\boldsymbol{P}^{PLR\text{-}O}_{1:t-1}) = \sum_{r_{t-1}} P(r_{t-1}|\boldsymbol{P}^{PLR\text{-}O}_{1:t-2}) P(r_t|r_{t-1}, \boldsymbol{P}^{PLR\text{-}O}_{1:t-1}). \tag{42}$$

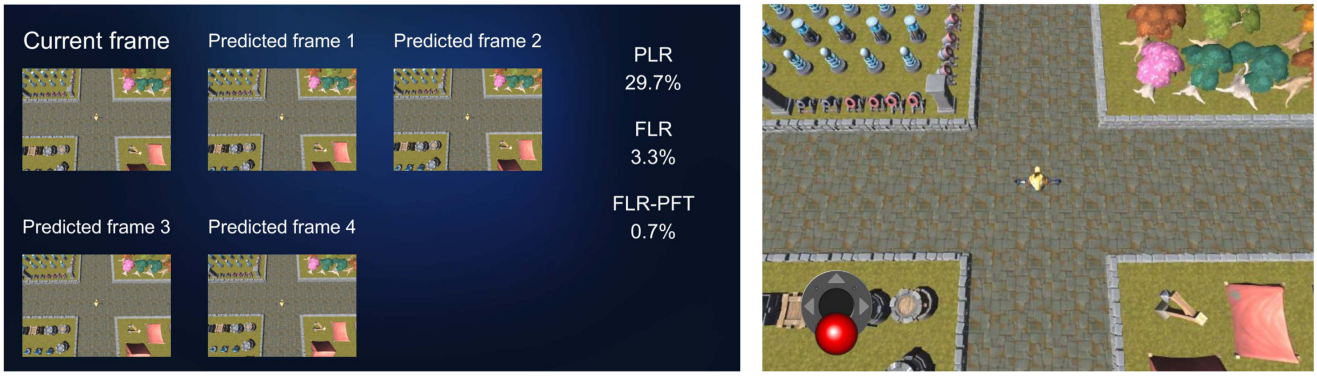
To summarize, at the end of the $(t-1)$th frame, the historical observations $\boldsymbol{P}^{PLR\text{-}O}_{1:t-1}$ is updated by the newest sampled PLR. Then the distribution of r_t is updated by (42), which denotes the number of frames since the last change of PLR. Finally, the system can derive the estimated PLR at the tth frame based on (41).

6.2 PLR-Estimation-Based Packet Allocation Policy

In each frame, the server allocates packets based on the buffer state and the estimated PLR. The whole procedure is shown in Algorithm 4. After PLR estimation, Algorithm 4 is used to derive the packet allocation, where the total number of network states is set $M = 1$ and PLR is P^{PLR-E}_t.

Algorithm 4. PLR-Estimation-Based Packet Allocation Algorithm

```
In the t th frame:
Get the observation P^{PLR-O}_{t-1}
Update r_t based on P^{PLR-O}_{1:t-1}
Estimate P^{PLR-E}_t based on r_t and P^{PLR-O}_{1:t-1}
M ← 1 and P^{PLR-M}_1 ← P^{PLR-E}_t
Get the policy π by Algorithm 1
Allocates packets based on π and B_t
```



(a) Cloud gaming program on the edge cloud server. (b) Cloud gaming program on the smart phone.

Fig. 4. Cloud gaming programs with the predictive frame transmission.

Algorithm 4 can be applied to any system that does not have the prior information about network states. Although the estimated PLR is not as precise as the system that knows PLR directly, the PLR-estimation-based packet allocation policy performs well as validated by our experiments and simulations.

7 Performance Evaluation

In this section, we first set up a practical cloud gaming testbed, and test the experimental performances. Then, we present more results from extensive simulations.

7.1 Testbed Setup

The cloud gaming testbed consists of a personal computer (i7-8700K CPU and GTX1080Ti GPU) as the edge cloud server, a TPlink WIFI router (3x3 MIMO with the rate of 450Mbps) as the access point, and a smart phone for the user. The FPS of cloud gaming is 60, and the frequency of user instruction change is 1Hz. In each frame, the server renders 5 JPEG images and sends 3 packets.

We use the Unity gaming engine to implement the programs on the server and smart phone, as shown in Fig. 4. The user operates the joystick on the smartphone, to control the movement of the object. In each frame, the server renders one image of the current frame and 4 images of predicted frames, based on the latest user instruction. In the figure, as the joystick instruction is to move down, the images of predicted frames are forward in the downward direction, compared with the image of the current frame.

The server transmits the images to the smart phone by our proposed predictive frame transmission scheme, and records the packet transmission results. The average PLR is updated every 60 frames. The server derives the FLR *with* and *without* the predictive frame transmission, as shown by the FLR-PFT and FLR in Fig. 4.

7.2 Experiment Results

The data sizes of images depend on the resolution, as summarized in Table 2.

TABLE 2
Data Size of Images

Resolution	360P	480P	720P	1080P
Image size	30KB	55KB	111KB	264KB

Table 3 presents the experiment results. In the experiment, PLR and FLR are tested with different image resolutions, wireless network environments, and network workloads. Each test runs the game for more than 60s. When playing game, the user holds the joystick to one dedicated direction to ensure that the game scene keeps changing. The user changes the direction of the joystick at a frequency of 1Hz, which indicates the frequency of user instruction update. In the table, the column of environment indicates the distance and obstacles between the smart phone and the WIFI router. The network user's column indicates the number of users in the network, and all users are playing cloud gaming simultaneously. The column of FLR and FLR-PFT denote the FLR *without* and *with* predictive frame transmission.

Based on the results in Table 3, it is shown that the PLR is highly related to the packet size, the wireless environment, and the network traffic load. When the network state is bad, the PLR can be higher than 40%, and the FLR is larger than 10%. With predictive frame transmission scheme, the FLR can be decreased greatly. For example, when the PLR is 47%, our proposed scheme decreases the FLR from 12% to 3%.

In Fig. 5, we compare the FLR between the cloud gaming systems with and without predictive frame transmission. In the experiment setup, the image resolution is 480P, the FPS is 60, and the frequency of instruction change is 1Hz. The user takes the mobile device and walk around, so that the wireless channel condition is continuously varying. As a result, the PLR and FLR fluctuate over time, producing a dynamic network environment. We trace the packet transmission samples over a period of 300s. Based on the samples, we apply different schemes to the samples, and test the FLR accordingly. When the network state is bad, the predictive frame transmission can decrease the maximum FLR from 15% to 4%, and decrease the mean FLR from 7% to 1%.

7.3 Simulation Results

In this part, we present more simulation results. We adopt various settings so as to simulate the cloud gaming system in different cases. The PLR varies from 1% to 50%. In each frame, the server renders 1 to 5 images, and the transmitted packets varies from 2 to 5.

TABLE 3
Experiment Results

Image resolution	Environment	Network users	PLR	FLR	FLR-PFT
360P	5m distance	5	2%	0%	0%
480P	5m distance	5	5%	1%	0%
720P	5m distance	5	19%	2%	1%
1080P	5m distance	5	57%	21%	7%
480P	5m distance	1	2%	0%	0%
480P	7m distance and a door	1	18%	2%	1%
480P	10m distance and a wall	1	70%	39%	23%
480P	5m distance	1	1%	0%	0%
480P	5m distance	5	5%	1%	0%
480P	5m distance	10	47%	12%	3%
480P	5m distance	25	65%	31%	13%

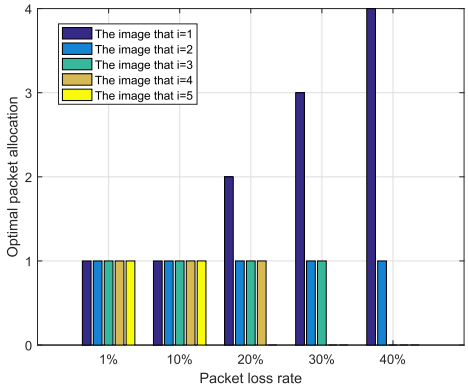


Fig. 6. The optimal packet allocation policy with different packet loss rates.

In Fig. 6, the optimal packet allocation policies with different PLRs are shown. In the simulation, the number of rendered image in each frame is $N = 5$, and the packets transmitted in each frame is $L = 5$. When the PLR is not larger than 10%, the policy allocates one packet to each image. With the increasing PLR, the policy allocates more packets to the image of the current frame. When the PLR is 40%, 4 packets are allocated to the image of the current frame, and only 1 packet is allocated to the predicted images. To this end, the optimal packet allocation is highly related to the network state. The optimal policy takes full advantage of good network states by transmitting more images of future frames to the user, and it focuses on the image of the current frame when the network state is bad. Fig. 6 also shows that the predicted images of nearer future are of higher priorities to be transmitted.

In Fig. 7, the packet overhead to transmit the predicted frames are compared with different PLRs. In the simulation, the packets transmitted in each frame is $L = 3$. When PLR is relatively small, the overhead is very close to 3. In this case, the predicted images are always sent by the server and displayed in the mobile devices. Accordingly, most of the packets are allocated to the predicted frames and the packet overhead is large. When PLR becomes larger, the packet allocation policy is conservative, and more packets are allocated to the current frame. In this case, the packet overhead of predicted frames becomes smaller.

In Fig. 8, the FLR with different PLRs are compared between different systems and policies. The upper bound of FLR is derived without the predictive frame transmission, and the lower bound of FLR is derived with the predictive frame transmission and with the immediate feedback of packet transmission. It is shown that the upper and lower bounds of FLR are 14% and 0.6% respectively, when the PLR is 45%. Although the optimal policy by MDP does not have the immediate feedback of packet transmission, it can also decrease FLR notably, which is 1.9% when the PLR is 45%. The nearest-future-based two-image transmission policy has the near-optimal performance compared with the optimal policy by MDP. It can decrease FLR to 2.2% when the PLR is 45%. The fixed packet allocation policy is compared as a baseline from Ref. [25], in which packet allocation is fixed regardless of the system state and network state. It is shown that the FLR of the fixed packet allocation policy is relatively high. Our proposed approach that allocates packets based on the system state can decrease the FLR from 13.2% to 1.9%, compared with the fixed packet allocation policy.

Fig. 9 shows how the number of the rendered images in each frame influences the FLR. When the number of rendered images N is 1, the system does not predict any future images. In this case, the FLR is 11%. By predicting 1 future image that $N = 2$, the FLR can be decreased to 3%. By predicting 2 or more future image, the FLR is smaller than 1%. Thus, if the server has enough CPU and GPU resources, it can decrease the FLR greatly by rendering enough predicted images. Even if the server only predicts 1 image, the FLR is decreased substantially. Fig. 9 also shows that the

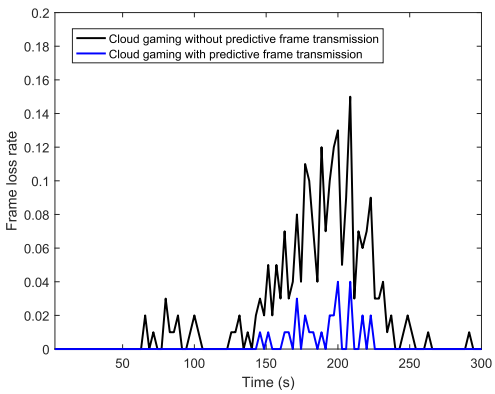


Fig. 5. Frame loss rate over a period of 300s.

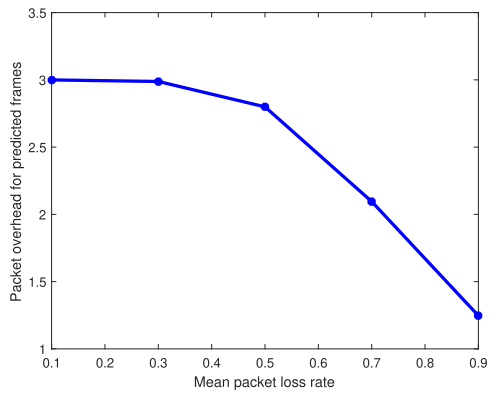


Fig. 7. The packet overhead of predicted frames.

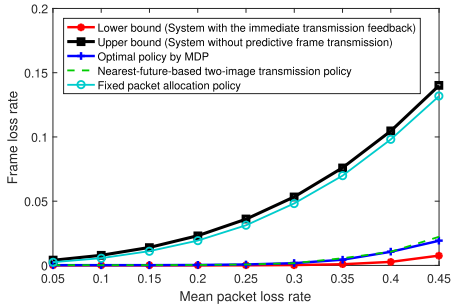


Fig. 8. Frame loss rate versus mean packet loss rate.

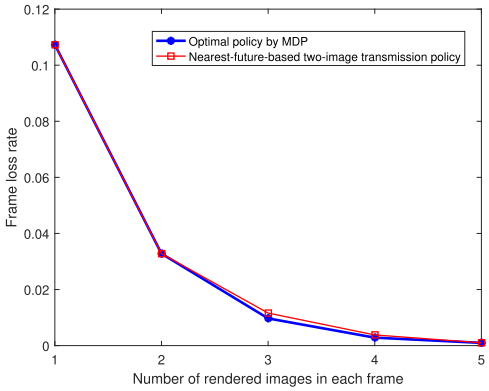


Fig. 9. Frame loss rate versus number of rendered images in each frame.

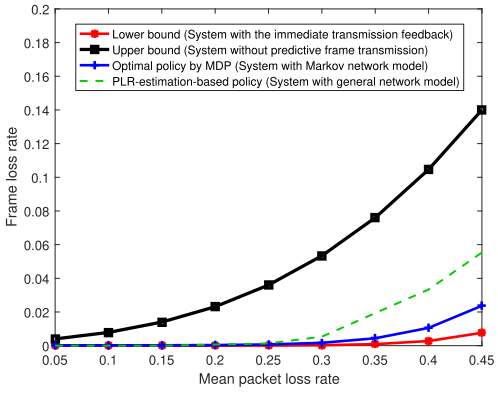


Fig. 10. Frame loss rate versus mean packet loss rate.

nearest-future-based two-image policy has the near-optimal performances.

Fig. 10 presents the performances of Algorithm 4 in the cloud gaming systems with general model based networks. Although the system estimates the PLR, the system performs well. When the PLR is 45%, the upper bound of FLR is 14%, the PLR-estimation-based packet allocation policy can decrease the FLR to 5.5%.

Fig. 11 shows the mean round trip delay of cloud gaming systems. As shown by Eq. (22), the round trip delay is a function of FLR. Because the predictive frame transmission scheme can decrease the FLR, it also leads to a smaller delay. When the PLR is 45%, the round trip delay of the system without the predictive frame transmission is 18.3ms, and the predictive frame transmission scheme can decrease the delay to 16.8ms.

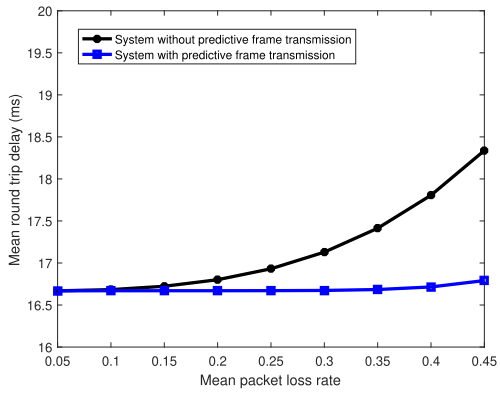


Fig. 11. Mean round trip delay versus mean packet loss rate.

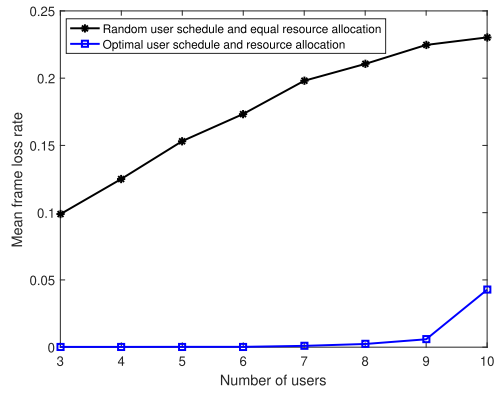


Fig. 12. Mean FLR of multiple users versus number of users.

Fig. 12 presents the performance of the optimal user schedule and resource allocation in the multi-server multi-user scenario. In the simulation, totally 3 edge servers are available where N and L vary from 3 to 5. The optimal user schedule and resource allocation are jointly derived by Algorithm 3. In comparison, the random user schedule and equal resource allocation just schedule users to the edge servers randomly, and allocate the computation and bandwidth resources equally. Numerical results show that the optimal user schedule and resource allocation can efficiently reduce the mean FLR of multiple users, which is decreased from 23% to 4.2% when the number of users is 10. Thus, Algorithm 3 can adaptively serve users based on their different requirements and network states, and realize load balancing for the multi-server multi-user system.

8 Conclusions

In this paper, we study the cloud gaming systems assisted by mobile edge cloudlets, and propose predictive frame transmission schemes against the dynamic network states. The key idea is to predict and pre-transmit images of future frames to users in good network states, so as to avoid the frame losses under bad network states. We model and formulate the packet allocation problem for the minimization of FLR, and derive the upper and lower bounds of FLR, respectively. In the system with Markovian property, we use MDP to derive the optimal packet allocation policy, and we also analyze the structure of the policy.

To decrease the complexity, we further propose the nearest-future-based two-image transmission policy that allocates packets to two images in each frame, which shows near-optimal performances. We extend the predictive frame transmission scheme to the multi-server multi-user scenario. A dynamic-programming-based algorithm is proposed to jointly derive the optimal user schedule and resource allocation, which can efficiently decrease the mean FLR of all users. For more general network models, we propose to estimate the PLR by change point detection, and then propose the packet allocation policy with the PLR estimation. By setting up a practical cloud gaming testbed, it is validated that the predictive frame transmission scheme can decrease the maximum FLR from 15% to 4% and decrease the mean FLR from 7% to 1%.

REFERENCES

[1] Newzoo, "Global games market report," 2020. [Online]. Available: https://newzoo.com/insights/trend-reports/newzoo-global-games-market-report-2020-light-version/

[2] Y. Xu, Q. Shen, X. Li, and Z. Ma, "A cost-efficient cloud gaming system at scale," *IEEE Netw.*, vol. 32, no. 1, pp. 42–47, Jan./Feb. 2018.

[3] C. Huang, K. Chen, D. Chen, H. Hsu, and C. Hsu, "GamingAnywhere: The first open source cloud gaming system," *ACM Trans. Multimedia Comput. Commun. Appl.*, vol. 10, pp. 1–25, 2014.

[4] W. Cai *et al.*, "A survey on cloud gaming: Future of computer games," *IEEE Access*, vol. 4, pp. 7605–7620, 2016.

[5] W. Cai, R. Shea, C. Y. Huang, K. T. Chen, and C. H. Hsu, "The future of cloud gaming," *Proc. IEEE*, vol. 104, no. 4, pp. 687–691, Apr. 2016.

[6] R. Payne, "NVIDIA GeForce NOW shows explosive growth for cloud-based gaming in 2020," 2020. [Online]. Available: https://chromeunboxed.com/nvidia-geforce-now-explosive-growth-cloud-gaming-2020/

[7] Huawei, "Cloud gaming experience model (Cloud gMOS) white paper," 2019. [Online]. Available: https://www.huawei.com/en/technology-insights/industry-insights/outlook/mobile-broadband/xlabs/insights-whitepapers/cloud-gmos-white-paper

[8] X. Zhang *et al.*, "Improving cloud gaming experience through mobile edge computing," *IEEE Wireless Commun.*, vol. 26, no. 4, pp. 178–183, Aug. 2019.

[9] J. Rogerson and S. Kavanagh, "How fast is 5G?," 2022. [Online]. Available: https://5g.co.uk/guides/how-fast-is-5g

[10] GamersNexus, "Stadia latency testing," 2020. [Online]. Available: https://www.youtube.com/watch?v=hVTsj66g9bA

[11] L. Lin, X. Liao, H. Jin, and P. Li, "Computation offloading toward edge computing," *Proc. IEEE*, vol. 107, no. 8, pp. 1584–1607, Aug. 2019.

[12] D. Murray, T. Koziniec, M. Dixon, and K. Lee, "Measuring the reliability of 802.11 WiFi networks," in *Proc. Internet Technol. Appl.*, 2015, pp. 233–238.

[13] R. D. Yates, M. Tavan, Y. Hu, and D. Raychaudhuri, "Timely cloud gaming," in *Proc. IEEE Conf. Comput. Commun.*, 2017, pp. 1–9.

[14] A. Als, F. Morn, P. Carballeira, D. Berjn, and N. Garca, "Congestion control for cloud gaming over UDP based on round-trip video latency," *IEEE Access*, vol. 7, pp. 78 882–78 897, 2019.

[15] M. Basiri and A. Rasoolzadegan, "Delay-aware resource provisioning for cost-efficient cloud gaming," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 28, no. 4, pp. 972–983, Apr. 2018.

[16] Y. Gao, L. Wang, and J. Zhou, "Cost-efficient and quality of experience-aware provisioning of virtual machines for multiplayer cloud gaming in geographically distributed data centers," *IEEE Access*, vol. 7, pp. 142 574–142 585, 2019.

[17] H. Chen *et al.*, "T-Gaming: A cost-efficient cloud gaming system at scale," *IEEE Trans. Parallel Distrib. Syst.*, vol. 30, no. 12, pp. 2849–2865, Dec. 2019.

[18] T. Zhang, C. F. Chiasserini, and P. Giaccone, "TAME: An efficient task allocation algorithm for integrated mobile gaming," *IEEE Syst. J.*, vol. 13, no. 2, pp. 1546–1557, Jun. 2019.

[19] F. Chi, X. Wang, W. Cai, and V. C. M. Leung, "Ad-Hoc cloudlet based cooperative cloud gaming," *IEEE Trans. Cloud Comput.*, vol. 6, no. 3, pp. 625–639, July–Sep. 2018.

[20] Y. Deng, Y. Li, R. Seet, X. Tang, and W. Cai, "The server allocation problem for session-based multiplayer cloud gaming," *IEEE Trans. Multimedia*, vol. 20, no. 5, pp. 1233–1245, May 2018.

[21] Y. Lin and H. Shen, "CloudFog: Leveraging fog to extend cloud gaming for thin-client MMOG with high quality of service," *IEEE Trans. Parallel Distrib. Syst.*, vol. 28, no. 2, pp. 431–445, Feb. 2017.

[22] H. E. Dinaki, S. Shirmohammadi, and M. R. Hashemi, "Boosted metaheuristic algorithms for QoE-aware server selection in multiplayer cloud gaming," *IEEE Access*, vol. 8, pp. 60 468–60 483, 2020.

[23] W. Cai, Y. Chi, C. Zhou, C. Zhu, and V. C. M. Leung, "UBCGAming: Ubiquitous cloud gaming system," *IEEE Syst. J.*, vol. 12, no. 3, pp. 2483–2494, Sep. 2018.

[24] Y. Li *et al.*, "Towards minimizing resource usage with QoS guarantee in cloud gaming," *IEEE Trans. Parallel Distrib. Syst.*, vol. 32, no. 2, pp. 426–440, Feb. 2021.

[25] K. Lee *et al.*, "Outatime: Using speculation to enable low-latency continuous interaction for mobile cloud gaming," in *Proc. 13th Annu. Int. Conf. Mobile Syst. Appl. Serv.*, 2015, pp. 151–165.

[26] I. H. Hou, N. Z. Naghsh, S. Paul, Y. C. Hu, and A. Eryilmaz, "Predictive scheduling for virtual reality," in *Proc. IEEE Conf. Comput. Commun.*, 2020, pp. 1349–1358.

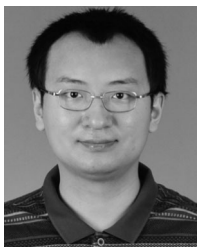
[27] W. Zhang, Y. Wen, K. Guan, D. Kilper, H. Luo, and D. O. Wu, "Energy-optimal mobile cloud computing under stochastic wireless channel," *IEEE Trans. Wireless Commun.*, vol. 12, no. 9, pp. 4569–4581, Sep. 2013.

[28] A. A. Laghari, H. He, K. A. Memon, R. A. Laghari, I. A. Halepoto, and A. Khan, "Quality of experience (QoE) in cloud gaming models: A review," *Multiagent Grid Syst.*, vol. 15, no. 3, pp. 289–304, 2019.

[29] R. E. Bellman, *Dynamic Programming*. Mineola, NY, USA: Courier Dover Publications, 2003.

[30] R. Adams and D. Mackay, "Bayesian online changepoint detection," 2007, *arXiv:0710.3742*.

[31] D. Bertsekas, *Dynamic Programming and Optimal Control*. Belmont, MA, USA: Athena Sci., 2007.



Tianchu Zhao received the BE and PhD degrees in electronic engineering from Tsinghua University, Beijing, China, in 2013 and 2021, respectively. From October 2016 to March 2017, he was a visiting student with the Electrical and Computer Engineering Department, University of California, Los Angeles. His research interests include mobile edge computing, cloud computing, and cloud gaming.

Sheng Zhou (Member, IEEE) received the BE and PhD degrees in electronic engineering from Tsinghua University, Beijing, China, in 2005 and 2011, respectively. In 2010, he was a visiting student with the Wireless System Lab, Department of Electrical Engineering, Stanford University, Stanford, CA, USA. From 2014 to 2015, he was a visiting researcher with Central Research Lab, Hitachi Ltd., Japan. He is currently an associate professor with the Department of Electronic Engineering, Tsinghua University. His research interests include cross-layer design for multiple antenna systems, mobile edge computing, vehicular networks, and green wireless communications. He was the recipient of IEEE ComSoc Asia–Pacific Board Outstanding Young Researcher Award in 2017.

Yuxuan Sun (Member, IEEE) received the BS degree in telecommunications engineering from Tianjin University, Tianjin, China, in 2015, and the PhD degree in electronic engineering from Tsinghua University, Beijing, China, in 2020. From October 2018 to April 2019, she was a visiting student with the Department of Electrical and Electronic Engineering, Imperial College London, U.K. She is currently a postdoctoral researcher with the Department of Electronic Engineering, Tsinghua University, and a visiting researcher with Imperial College London. Her research interests include edge computing and edge learning. Since August 2020, she has been an assistant to the editor-in-chief of *IEEE Transactions on Green Communications and Networking*.

Zhisheng Niu (Fellow, IEEE) graduated from Beijing Jiaotong University, China, in 1985, and received the ME and DE degrees from the Toyohashi University of Technology, Japan, in 1989 and 1992, respectively. During 1992–1994, he was with Fujitsu Laboratories Ltd., Japan. In 1994, he joined Tsinghua University, Beijing, China, where he is currently a professor with the Department of Electronic Engineering. His main research interests include queueing theory, traffic engineering, mobile Internet, radio resource management of wireless networks, and green communication and networks. Since 2000, he has been serving the IEEE Communications Society. He was the chair of Beijing Chapter during 2000–2008, director of Asia-Pacific Board during 2008–2009, director of Conference Publications during 2010–2011, chair of Emerging Technologies Committee during 2014–2015, director of Online Contents during 2018–2019, and is currently the editor-in-chief of *IEEE Transactions on Green Communications and Networking*. He was the recipient of Best Paper Award of Asia-Pacific Board in 2013, Distinguished Technical Achievement Recognition Award of Green Communications and Computing Technical Committee in 2018, and Harold Sobol Award for Exemplary Service to Meetings and Conferences in 2019, all from IEEE Communication Society. He was selected as a distinguished lecturer of IEEE Communication Society during 2012–2015 and IEEE Vehicular Technologies Society during 2014–201. He is a fellow of IEICE.