Viewport Prediction, Bitrate Selection, and Beamforming Design for THz-Enabled 360-Degree Video Streaming

Mehdi Setayesh, *Graduate Student Member*, *IEEE*, and Vincent W.S. Wong, *Fellow*, *IEEE*

***Abstract*—360° videos require significant bandwidth to provide an immersive viewing experience. Wireless systems using terahertz (THz) frequency band can meet this high data rate demand. However, self-blockage is a challenge in such systems. To ensure reliable transmission, this paper explores THz-enabled 360° video streaming through multiple multi-antenna access points (APs). Guaranteeing users' quality of experience (QoE) requires accurate viewport prediction to determine which video tiles to send, followed by asynchronous bitrate selection for those tiles and beamforming design at the APs. To address users' privacy and data heterogeneity, we propose a content-based viewport prediction framework, wherein users' head movement prediction models are trained using a personalized federated learning (PFL) algorithm. To address asynchronous decision-making for tile bitrates and dynamic THz link connections, we formulate the optimization of bitrate selection and beamforming as a macro-action decentralized partially observable Markov decision process (MacDec-POMDP) problem. To efficiently tackle this problem for multiple users, we develop two deep reinforcement learning (DRL) algorithms based on multi-agent actor-critic methods and propose a hierarchical learning framework to train the actor and critic networks. Experimental results show that our proposed approach provides a higher QoE when compared with three benchmark algorithms.**

***Index Terms*—Deep reinforcement learning (DRL), macro-action decentralized partially observable Markov decision process (MacDec-POMDP), personalized federated learning (PFL), quality of experience (QoE), terahertz (THz) communication, 360° video, viewport prediction.**

I. Introduction

In the realm of 360° video streaming, users delve into an immersive visual experience using a head-mounted display (HMD) for video playback. Compared with conventional video streaming, 360° video offers high-resolution 360° visual field across three degrees of freedom. The broader and higher-definition visual field entails that a larger number of pixels must be transmitted, thereby necessitating a significantly higher data rate [3]. The abundant bandwidth available in the terahertz (THz) frequency band can potentially overcome this challenge, especially for transmitting 360° video streams that require a high data rate in the range of gigabits per second (Gbps) and can provide a truly immersive user experience [4].

Wireless systems operating in THz frequency band encounter a number of challenges, including a limited communication range, channel impairment due to molecular absorption, and susceptibility to blockage by obstacles [5]. Moreover, when a user turns around to view another part of a 360° video with its HMD, the THz link may be blocked by the user's own body, which is known as *self-blockage* [4]. The availability of a line-of-sight (LoS) link is crucial for reliable THz communication. To improve the reliability in a THz-enabled 360° video streaming system, multiple access points (APs) can jointly transmit 360° videos to the users [6].

In a multi-user 360° video streaming system, delivering the entire 360° video with the highest quality to all users may exceed the available bandwidth. However, at any given time, a user is watching a 360° video only from one direction. The region of the video that a user is currently watching is called a *viewport* [7]. To efficiently utilize the network bandwidth, it is desirable that each user receives its viewport with the maximum possible quality, rather than the entire video frame [8]. Viewport prediction is a key enabler for streaming 360° videos over wireless systems.

Viewport prediction is categorized into content-independent and content-based approaches. The content-independent approach relies on users' historical head movements, while the content-based approach utilizes both the video content and users' historical head movements to predict future viewports. Thus, the content-based approach can achieve higher prediction accuracy [9]. In multi-user 360° video streaming systems, users' different viewing patterns lead to data heterogeneity. Additionally, users may be reluctant to share their historical data due to privacy concerns. To address these challenges, a personalized federated learning (PFL) algorithm can be employed to train the viewport prediction models [3].

For streaming 360° videos, each video is divided into chunks in the temporal domain, with each chunk containing a few seconds of video frames. In the spatial domain, each 360° video frame is divided into tiles [7]. Prefetching is used to prevent video stalling during playback. Specifically, a prefetching scheme is employed to decide when and how the tiles for the subsequent video chunks should be sent to

Manuscript received on Dec. 9, 2023; revised on Jul. 15, 2024 and Nov. 24, 2024; accepted on Nov. 27, 2024. This work was supported in part by Rogers Communications Canada Inc., Natural Sciences and Engineering Research Council of Canada, and the Digital Research Alliance of Canada. This paper has been accepted for publication in part in the *Proceedings of IEEE International Conference on Multimedia and Expo (ICME)*, Brisbane, Australia, July 2023 [1] and the *Proceedings of the IEEE International Conference on Communications (ICC)*, Denver, CO, June 2024 [2]. The editor coordinating the review of this paper and approving it for publication was Xuanyu Cao *(Corresponding author: Vincent W.S. Wong)*. M. Setayesh and V. W.S. Wong are with the Department of Electrical and Computer Engineering, The University of British Columbia, Vancouver, BC, V6T 1Z4, Canada (e-mail: {setayeshm, vincentw}@ece.ubc.ca). Color versions of one or more of the figures in this paper are available online at https://ieeexplore.ieee.org.

each user [10]. Each user asynchronously requests a new video chunk based on its buffer status. Transmitting a set of tiles for each 360° video chunk based on the predicted viewport of a user can reduce bandwidth consumption and enable a more flexible transmission mechanism through bitrate selection for the tiles [11]. Note that higher viewport prediction accuracy leads to better bitrate selection for tiles and an improved prefetching scheme, thereby resulting in higher bandwidth efficiency and better quality of experience (QoE) for the users. Additionally, since users' future head movements can be captured as an integral component of viewport prediction, proactive detection of self-blockage occurrences becomes possible in THz-enabled 360° video streaming systems.

In this paper, we propose a content-based viewport prediction framework that utilizes a PFL algorithm to train the head movement prediction model. This framework is an extension of the content-based approach we proposed in [1]. Our proposed framework incorporates a more practical saliency detection model that can be trained without requiring the saliency map of the video frames to be part of the training dataset. Furthermore, we describe how additional tiles that cover a marginal region of a viewport can be selected and sent to the users to account for prediction errors.

We study 360° video streaming in a multi-user THz wireless system with multiple multi-antenna APs. Users' requests for video chunks give rise to an optimization problem encompassing bitrate selection for the video tiles and beamforming design at the APs. Due to the asynchronous decision-making and hierarchical structure of this problem, we formulate it as a macro-action decentralized partially observable Markov decision process (MacDec-POMDP) [12], [13]. To solve this problem, we propose a hierarchical deep reinforcement learning (DRL) framework comprising two multi-agent deep deterministic policy gradient (DDPG) algorithms. The proposed DRL framework is an extension of the approach we proposed in [2]. In particular, we combine the viewport prediction framework with the bitrate selection and beamforming design algorithms in a 360° video streaming system, which is not trivial. Furthermore, we replace the weighted minimum mean square error (WMMSE) beamforming algorithm proposed in [2] with a multi-agent DDPG algorithm to obtain beamforming vectors in a computationally efficient manner. The main contributions of this paper are as follows:

- To support reliable transmission of 360° video streaming in a THz wireless system, multiple multi-antenna APs are used for video transmission. To improve the users' QoE, we propose a 360° video streaming approach that includes (a) a viewport prediction framework to determine which video tiles to transmit and (b) two multi-agent DDPG algorithms to determine the bitrate selection of the video tiles and beamforming vectors at the APs. Fig. 1 shows an illustration of our proposed 360° video streaming approach.
- In particular, we propose a content-based viewport prediction framework that decouples the viewport prediction into two models. The first model focuses on saliency detection. The second model is for head movement prediction and is trained with a PFL algorithm. Due to the decoupling of these two models, any saliency detection model can be incorporated into this viewport prediction framework. With the use of PFL, our framework can address users' privacy concerns and mitigate the issue of data heterogeneity. To predict the viewport for each user, we integrate the outputs of the aforementioned models, namely the video saliency map and the user's head orientation map, through a fusion technique.
- We formulate a MacDec-POMDP problem to determine the policies for bitrate selection and beamforming optimization. Taking into account the asynchronous requests from users for new video chunks, we propose a multi-agent DDPG algorithm using a macro-action-based independent actor with individual centralized critic (MacIAICC) approach. This algorithm can effectively obtain the policy for determining the bitrate of video tiles. Furthermore, we propose a multi-agent DDPG algorithm using a primitive-action-based centralized actor-critic (Prim-CAC) approach to obtain the policy for beamforming design at the APs. To accommodate the hierarchical structure of the problem, we develop a hierarchical learning framework that facilitates training of the actor and critic networks in these algorithms.
- We evaluate the performance of our proposed video streaming approach using a public 360° video dataset [14]. Simulation results show that our proposed video streaming approach outperforms several benchmark algorithms in terms of the average QoE. In particular, when there are twelve users, our proposed approach provides an average QoE which is 23.77%, 62.7%, and 116.57% higher than that of the video streaming with WMMSE beamforming algorithm, the combined field-of-view (FoV) tile-based adaptive streaming algorithm proposed in [8], and the video streaming with viewport prediction proposed in [15], respectively.

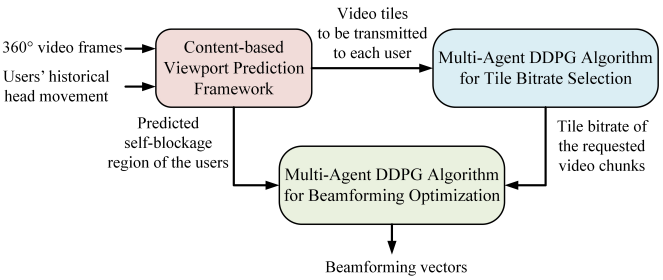


Fig. 1: Our proposed 360° video streaming approach.

This paper is organized as follows. The related work is discussed in Section II. The system model is presented in Section III. Section IV introduces the MacDec-POMDP problem formulation. Our proposed viewport prediction framework is presented in Section V. In Section VI, we present our proposed DRL algorithms. Simulation results are presented in Section VII. Conclusion is given in Section VIII.

Notations: In this paper, we represent vectors by boldface lowercase letters (e.g., $\boldsymbol{x}$), matrices by boldface uppercase letters (e.g., $\boldsymbol{X}$), and sets by calligraphic letters (e.g., $\mathcal{X}$). The cardinality of set $\mathcal{X}$ is denoted by $|\mathcal{X}|$. The symbol $(\cdot)^H$

denotes conjugate transpose operator. $\boldsymbol{I}_N$ denotes an identity matrix of size N. $\|\cdot\|$ denotes the norm of a vector. $\mathbb{1}(z \in \mathcal{Z})$ denotes the indicator function which is equal to 1 if $z \in \mathcal{Z}$, and is equal to zero otherwise. We define $[z]^+ = \max\{0, z\}$.

II. Related Work

A. Viewport Prediction

Recently, deep neural networks (DNNs) have been incorporated into viewport prediction models to improve their prediction accuracy. Chao *et al.* in [16] proposed a transformer-based architecture to predict users' viewports. Liu *et al.* in [17] proposed long short-term memory (LSTM) and gated recurrent unit (GRU) architectures to predict future viewports. DNNs require a large amount of data for training. Thus, each user may not be able to obtain a prediction model with high accuracy using only its local data. The prediction accuracy can be improved if all users collaboratively train a shared model. Federated learning (FL) facilitates distributed training while preserving users' privacy. FedAvg [18], which is a popular FL algorithm, has been used in [15] for viewport prediction. Furthermore, the issue of data heterogeneity among users' local data can be tackled using a PFL algorithm. In PFL, different users collaboratively train a shared global model. Then, each user utilizes its local data samples to fine-tune the global model and obtain a customized model [19]. Zhang *et al.* in [3] used a PFL algorithm based on meta-learning for viewport prediction. The aforementioned works fall into the category of content-independent viewport prediction approaches.

In content-based approaches, incorporating video content in viewport prediction can improve prediction accuracy by identifying the parts of the video frames that are more interesting for users to watch. Nguyen *et al.* in [20] proposed an LSTM architecture to predict the user's viewport using saliency maps of the past video frames and the user's historical head movements. Li *et al.* in [9] proposed a spherical convolution-empowered model, where the users' future viewports are predicted by combining the salient spatial-temporal features of video frames with the users' historical viewport information. Wu *et al.* in [21] proposed a preference-aware viewport prediction model that utilizes an attention mechanism to combine visual features from 360° video frames with users' viewing historical data. The aforementioned content-based viewport prediction models require centralized training. Specifically, since the server has access to the previous video frames, users must provide their historical head movements to the server for viewport prediction in those models. Decoupling the viewport prediction model into a saliency detection model and a head movement prediction model as proposed in this paper can offer some advantages. First, any state-of-the-art saliency detection model can be incorporated into the viewport prediction model. Second, users' privacy concerns and data heterogeneity issues can be addressed by using PFL for training the head movement prediction model.

B. 360° Video Streaming over Wireless systems

Recently, streaming 360° videos over wireless systems has received considerable attention. There are two main threads in related work. The first line of research aims to improve users' QoE by flexibly transmitting video tiles through adaptive bitrate selection mechanisms. The second line of research involves using new technologies, such as mobile edge computing (MEC), rate-splitting (RS), and millimeter wave (mmWave) band communication, in wireless system infrastructure for video streaming. For flexible video tile transmission, the main focus is on viewport prediction, selecting the tiles that should be sent to users, and the bitrate selection for those tiles. Kan *et al.* in [7] proposed a viewport identification method, a viewport-aware adaptive tiling scheme, and a DRL-based rate adaptation algorithm for 360° video streaming. Yaqoob *et al.* in [8] proposed a combined FoV prediction-assisted 360° video streaming algorithm and a priority-based bitrate adaptation algorithm. The capability of adaptive bitrate selection mechanisms remains limited in fulfilling the data rate requirements of high-resolution 360° videos. Thus, new technologies should be incorporated into wireless system infrastructure to enable the delivery of such 360° videos and new generation of virtual reality (VR) services [22].

The main focus of the following works is on utilizing new technologies for video streaming over wireless systems. Zhao *et al.* in [23] proposed iterative algorithms to determine the beamforming vectors for maximizing the weighted sum rate in a multicast RS VR streaming system. Yang *et al.* in [24] proposed a DRL-based algorithm to improve the users' visual experience in a mmWave-enabled VR streaming system. Huang *et al.* in [25] proposed a DRL algorithm to optimize the intelligent reflecting surface (IRS) phase shifts, RS parameters, beamforming vectors, and bitrate selection of video tiles in an IRS-aided RS VR streaming system. The authors in [23]–[25] have considered that the users' viewports are known *a priori* (i.e., perfect viewport prediction). However, combining viewport prediction within a resource allocation optimization problem for 360° video streaming over wireless systems is not a trivial task. In particular, viewport prediction accuracy has an impact on bandwidth efficiency and users' QoE. Zhang *et al.* in [3] used a PFL-based viewport prediction and proposed a DRL algorithm for resource allocation in a multi-user MEC-enabled VR streaming system. The proposed approach in [3] is based on considering only two possible quality levels (i.e., high and low) for tiles. Perfecto *et al.* in [26] proposed a deep recurrent neural network architecture for viewport prediction and an algorithm based on matching theory for video frame scheduling using mmWave multicast transmission.

The aforementioned works consider either synchronized chunk requests or single-user video streaming. However, in practical multi-user video streaming systems, users can request and download new video chunks asynchronously based on their buffer status. Asynchronous video streaming can further improve bandwidth efficiency [27]. Moreover, it is envisioned that a data rate of $6.37-95.55$ Gbps is required for high-resolution 360° video streaming and ultimate VR, a new generation of VR services with better video quality and a multisensory experience [4]. Such a data rate is far beyond the maximum achievable data rate in the fifth generation (5G) wireless systems using mmWave. Migration towards higher frequency bands, e.g., THz bands, can address this challenge.

C. DRL for 360° Video Streaming

DRL can be applied in 360° video streaming over wireless systems to learn bitrate selection and resource allocation policies. Since DRL-based algorithms can adapt well to network dynamics and provide desired solutions in a timely manner, they have recently attracted great attention. To determine the bitrates of tiles in a 360° video streaming system, the authors in [7], [28], [29] proposed different DRL-based algorithms using asynchronous advantage actor-critic, LSTM-based actor-critic, and Rainbow, respectively. These DRL-based algorithms are not multi-agent DRL. Thus, the interaction among users in a multi-user 360° video streaming system has not been explored in these works. Moreover, the DRL-based algorithms proposed in these works are not used to obtain a resource allocation policy in the wireless system. Specifically, it is assumed that the allocated throughput or bandwidth to a user in the previous time slot is given to be considered as one of the system states in the current time slot. On the other hand, the authors in [3], [15], [24] proposed different DRL-based algorithms to determine the resource allocation policy in the wireless system. In particular, the proposed DRL-based algorithms in [3], [15], and [24] are used for resource block allocation, IRS reflection coefficient matrix optimization, and satisfying a predefined data rate threshold for users, respectively. These works do not use DRL to obtain the bitrate selection policy. The authors in [25] have shown that DRL can be used to jointly optimize the degrees of freedom provided by the wireless system infrastructure and the bitrate selection of video tiles, thereby improving users' QoE. However, the multi-agent DDPG algorithm proposed in [25] cannot be used when users asynchronously request video tiles.

Asynchronous video tile requests from users make multi-user 360° video streaming a challenging problem. First, the time slot that a decision should be made for bitrate selection of tiles requested by a user may not be aligned with other users. Second, each user cannot request a new video chunk in every time slot due to its buffer status, while the wireless system requires resource allocation in each time slot. To address the first challenge, we formulate this problem as a MacDec-POMDP, and define macro-actions and a shared extrinsic reward to efficiently capture the impact of asynchronous decisions made upon each user's chunk request. The second challenge is tackled by using a hierarchical learning framework, which can consider interactions among bitrate selection and resource allocation policies at different levels of temporal abstraction.

III. System Model

Consider U users who are watching 360° videos in an indoor environment, using THz wireless links as shown in Fig. 2. We denote the set of users by $\mathcal{U} = \{1, \ldots, U\}$. The users are stationary. However, they can turn around to watch different parts of the video. Each user is equipped with a wireless HMD operating at THz frequency band. Let $\boldsymbol{l}_u = (x_u, y_u, h_u)$ denote the location of the HMD that is worn by user $u \in \mathcal{U}$, where x_u, y_u, and h_u denote the $x-$axis coordinate, the $y-$axis coordinate, and the height of user u's HMD from the ground, respectively. In order to mitigate self-blockage of THz links, multiple APs are used to transmit the

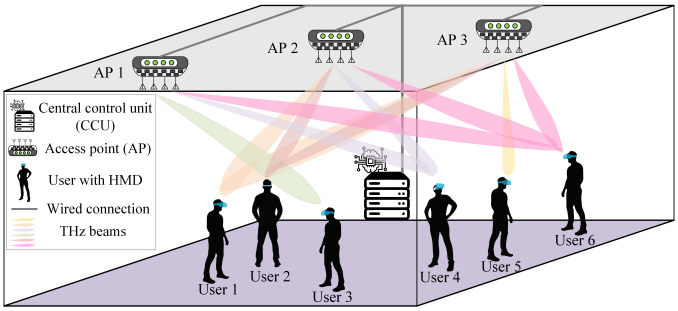


Fig. 2: A THz-enabled 360° video streaming system. There are six users and three APs in the environment. 360° video streams are sent to each user by the APs which are not in the user's self-blockage region. The APs are connected to a CCU via a wired connection.

360° video streams to the users. Let $\mathcal{A} = \{1, \ldots, N_{\text{AP}}\}$ denote the set of ceiling-mounted APs. We denote the location of AP $a \in \mathcal{A}$ by $\boldsymbol{l}_a = \left(x_a, y_a, h^{\text{AP}}\right)$, where x_a and y_a denote the coordinate of AP a on the $x-$ and $y-$axes, respectively, and h^{AP} is the height of the ceiling from the ground. Each AP and each user's HMD are equipped with a uniform linear array (ULA) of N_{t} and N_{r} antenna elements, respectively. Let f_{c} denote the carrier frequency of the transmitted signals by the APs' antennas. The spacing between adjacent antenna elements is chosen to be $d = \frac{\lambda_{\text{c}}}{2}$, where λ_{c} is the wavelength of carrier frequency f_{c}.

A. THz Channel and Downlink Transmission Model

THz band communications suffer from attenuation due to molecular absorption. Let $\kappa(f)$ (in m^{-1}) denote the molecular absorption coefficient at frequency f. $\kappa(f)$ can be predicted for a given transmission medium using the high resolution transmission molecular absorption database (HITRAN) [30]. We consider that $\kappa(f)$ remains relatively constant for the frequencies within the transmission bandwidth of the APs [5], [6]. Moreover, THz band communications are highly directional, especially when high gain antennas are used by the APs and users. Thus, we consider only the LoS path to obtain the THz channel gain between each AP and user [31]. Let $\gamma_{u,a}$ denote the LoS path gain between AP $a \in \mathcal{A}$ and user $u \in \mathcal{U}$. $\gamma_{u,a}$ is composed of the spreading and molecular absorption losses for the LoS path [6], [30]. We have $\gamma_{u,a} = \frac{c_0}{4\pi f_{\text{c}} \|\boldsymbol{l}_a - \boldsymbol{l}_u\|} e^{-\frac{1}{2}\kappa(f_{\text{c}})\|\boldsymbol{l}_a - \boldsymbol{l}_u\|}$, where c_0 is the speed of light. Let $\mathbf{a}_a(\psi_{u,a}^{\text{AoD}}) \in \mathbb{C}^{N_{\text{t}}}$ and $\mathbf{a}_u(\psi_{u,a}^{\text{AoA}}) \in \mathbb{C}^{N_{\text{r}}}$ denote the array steering vectors for ULA at AP a and user u, respectively. $\psi_{u,a}^{\text{AoD}}$ and $\psi_{u,a}^{\text{AoA}}$ represent the angle-of-departure (AoD) and the angle-of-arrival (AoA) of the THz beam transmitted from AP a to user u, respectively. We have $\mathbf{a}_a(\psi_{u,a}^{\text{AoD}}) = \left(1, e^{j\frac{2\pi d}{\lambda_{\text{c}}}\sin(\psi_{u,a}^{\text{AoD}})}, \ldots, e^{j\frac{2\pi d}{\lambda_{\text{c}}}(N_{\text{t}}-1)\sin(\psi_{u,a}^{\text{AoD}})}\right)$ and $\mathbf{a}_u(\psi_{u,a}^{\text{AoA}}) = \left(1, e^{j\frac{2\pi d}{\lambda_{\text{c}}}\sin(\psi_{u,a}^{\text{AoA}})}, \ldots, e^{j\frac{2\pi d}{\lambda_{\text{c}}}(N_{\text{r}}-1)\sin(\psi_{u,a}^{\text{AoA}})}\right)$. Let $\boldsymbol{G}_{u,a} \in \mathbb{C}^{N_{\text{t}} \times N_{\text{r}}}$ denote the channel gain matrix between AP a and user u. We have $\boldsymbol{G}_{u,a} = \sqrt{g_a g_u}\gamma_{u,a}\mathbf{a}_a(\psi_{u,a}^{\text{AoD}})\mathbf{a}_u^H(\psi_{u,a}^{\text{AoA}})$, where g_a and g_u are the antenna gains (in dBi) at AP a and user u, respectively.

The APs are connected to a central control unit (CCU) via a wired connection. Given the THz LoS link availability between the APs and users, as well as the channel gain matrices, the beamforming vectors can be determined by the CCU in each

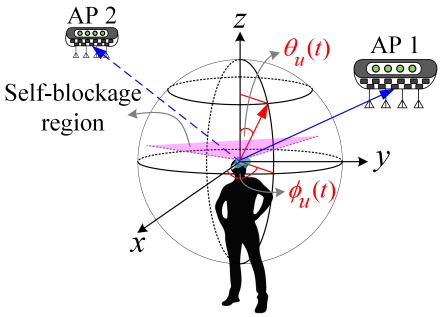


Fig. 3: Illustration of the latitude and longitude angles of a user's head, as well as its self-blockage region. AP 2 is located within the self-blockage region, whereas AP 1 is not.

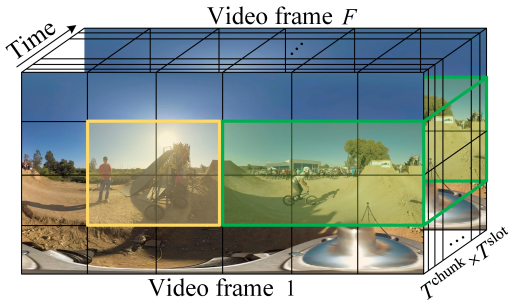


Fig. 4: Illustration of a video chunk. There are F video frames within each chunk. Each video frame is divided into 24 tiles using 6×4 tiling pattern. The tiles in the viewport are shown in green, while those in the marginal region are shown in yellow. Other tiles are considered in the invisible region.

time slot. Let $\mathcal{T} = \{1, 2, \ldots\}$ denote the set of time slots. Each time slot has a duration of T^{slot} (in millisecond (ms) [26]). Let $\boldsymbol{b}_{u,a}(t) \in \mathbb{C}^{N_{\text{t}}}$ denote the beamforming vector from AP $a \in \mathcal{A}$ to user $u \in \mathcal{U}$ in time slot $t \in \mathcal{T}$. Considering $P^{\max}$ as the maximum transmit power of an AP, we have

$$\sum_{u \in \mathcal{U}} \|\boldsymbol{b}_{u,a}(t)\|^2 \leq P^{\max}, \ a \in \mathcal{A}, \ t \in \mathcal{T}. \tag{1}$$

An HMD has internal sensors (e.g., gyroscope, accelerometer) that enable it to track the head movement of its user [32]. For each user $u \in \mathcal{U}$, let $\theta_u(t)$ and $\phi_u(t)$, respectively, denote the latitude and longitude angles of its head orientation in time slot $t \in \mathcal{T}$ in the local spherical coordinate system[1]. The latitude angle $\theta_u(t)$, where $0 \leq \theta_u(t) \leq \pi$, is the angle measured from the $z-$axis. The longitude angle $\phi_u(t)$, where $0 \leq \phi_u(t) \leq 2\pi$, is the angle measured from the $x-$axis after projection onto the $x-y$ plane. We define a self-blockage angle ϕ^{blocked} to characterize the self-blockage region of the users with respect to the locations of the APs [4], [5]. Fig. 3 shows an illustration of a user's self-blockage region. Let $\mathcal{A}_u^{\text{nb}}(t)$ denote the set of APs which are not in the self-blockage region of user $u \in \mathcal{U}$ in time slot $t \in \mathcal{T}$. We have

$$\mathcal{A}_u^{\text{nb}}(t) = \left\{ a \,\middle|\, a \in \mathcal{A}, \, |\phi_u(t) - \phi_{u,a} - \pi| \geq \frac{\phi^{\text{blocked}}}{2} \right\}, \tag{2}$$

where $\phi_{u,a} = \text{mod}(\text{arctan2}(y_a - y_u, x_a - x_u), 2\pi)$ denotes the longitude angle of the LoS link between user u and AP a. $\text{arctan2}(\cdot, \cdot)$ returns an angle, ranging from $-\pi$ to π, that represents the angle between the positive $x-$axis and a given vector. $\text{mod}(\cdot, 2\pi)$ is the modulo operator, which is used to obtain a value between 0 and 2π for $\phi_{u,a}$.

Let $r_u(t)$ denote the data rate of user $u \in \mathcal{U}$ in time slot $t \in \mathcal{T}$. For the sake of brevity, we define vector $\boldsymbol{d}_{u,u'}(t) = \sum_{a \in \mathcal{A}_{u'}^{\text{nb}}(t) \cap \mathcal{A}_u^{\text{nb}}(t)} \boldsymbol{G}_{u,a}^H \boldsymbol{b}_{u',a}(t)$. We have [33]:

$$r_u(t) = B \log_2 \left(1 + \boldsymbol{d}_{u,u}^H(t) \boldsymbol{\Gamma}_u^{-1}(t) \boldsymbol{d}_{u,u}(t)\right), \tag{3}$$

where B is the transmission bandwidth and $\boldsymbol{\Gamma}_u(t)$ is the interference-plus-noise covariance matrix at user u. We have $\boldsymbol{\Gamma}_u(t) = \sum_{u' \in \mathcal{U} \setminus \{u\}} \boldsymbol{d}_{u,u'}(t) \boldsymbol{d}_{u,u'}^H(t) + \sigma^2 \boldsymbol{I}_{N_{\text{r}}}$, where σ^2 is the variance of the additive white Gaussian noise.

[1]The local spherical coordinate system of each user $u \in \mathcal{U}$ has its origin at point $\boldsymbol{l}_u$ and its axes are aligned with the three-dimensional (3D) Cartesian coordinate system.

B. Tile Request and Buffer Model

To enhance bandwidth efficiency in streaming 360° videos, we leverage a tile-based approach. Let $\mathcal{V} = \{1, \ldots, V\}$ denote the set of available 360° videos. In the time domain, each video $v \in \mathcal{V}$ is segmented into C_v chunks. Let $\mathcal{C}_v = \{1, \ldots, C_v\}$ denote the chunk indices for video v. As shown in Fig. 4, we consider that each video chunk has a fixed duration of T^{chunk} (in time slot) and contains F video frames. Let $\mathcal{F} = \{1, \ldots, F\}$ denote the set of indices of the video frames within a chunk. In the spatial domain, each video frame is divided into N tiles. Let $\mathcal{N} = \{1, \ldots, N\}$ denote the set of indices corresponding to the tiles of each video frame.

360° videos are streamed to users as chunks. We consider a prefetching scheme in which each user downloads one chunk at a time and subsequently requests the next chunk based on its buffer status [34]. When a user requests a video chunk, a viewport prediction framework, to be presented in Section V, is utilized to predict the tiles that may be viewed by the user for that chunk. The viewport prediction framework facilitates dividing each video frame into three regions: viewport, marginal, and invisible regions. To save wireless system bandwidth, the tiles corresponding to the invisible region are not transmitted to the users. To provide a high QoE, the tiles covering both the viewport and marginal regions are transmitted to the users. For user $u \in \mathcal{U}$ requesting chunk $c \in \mathcal{C}_v$ of video $v \in \mathcal{V}$, let $\mathcal{N}_{u,f,c,v}^{\text{view}}$ and $\mathcal{N}_{u,f,c,v}^{\text{marg}} \subset \mathcal{N}$ denote the set of tile indices corresponding to the viewport and marginal regions of video frame $f \in \mathcal{F}$, respectively. Let $\mathcal{N}_{u,c,v}^{\text{pred}}$ denote the set of tile indices that are predicted to be transmitted to user u upon its request for chunk c of video v. We have $\mathcal{N}_{u,c,v}^{\text{pred}} = \cup_{f \in \mathcal{F}} \left(\mathcal{N}_{u,f,c,v}^{\text{view}} \cup \mathcal{N}_{u,f,c,v}^{\text{marg}}\right)$.

In our proposed tile-based 360° video streaming approach, after receiving the chunk request from a user, the CCU needs to determine the quality level of the tiles. When a tile is encoded at a better quality level, it requires a higher bitrate for transmission accordingly. Let $\mathcal{M} = \{1, \ldots, M\}$ denote the set of quality levels, where the lowest quality is represented by 1. We consider that the tiles with the same quality level have identical bitrate [25], [35]. Let ν_m (in bits/s) denote the bitrate required to encode a tile at quality level $m \in \mathcal{M}$. We use the binary decision variable $\beta_{u,n,m}$ to indicate whether quality level $m \in \mathcal{M}$ is selected for tile $n \in \mathcal{N}_{u,c,v}^{\text{pred}}$ when user $u \in \mathcal{U}$ requests chunk $c \in \mathcal{C}_v$ of video $v \in \mathcal{V}$ ($\beta_{u,n,m} = 1$) or

not ($\beta_{u,n,m}=0$). We have

$$\sum_{m\in\mathcal{M}}\beta_{u,n,m}\leq 1,\ u\in\mathcal{U},\ n\in\mathcal{N}^{\text{pred}}_{u,c,v},\ c\in\mathcal{C}_v,\ v\in\mathcal{V}. \quad (4)$$

The time when a user requests for a new video chunk depends on its buffer status (i.e., the playback time of the previously downloaded video chunks in its buffer). Let $\tau^{\text{REQ}}_{u,c,v}$ (in time slot) denote the time when the tiles of video chunk $c\in\mathcal{C}_v$ are requested by user $u\in\mathcal{U}$ who is watching video $v\in\mathcal{V}$. Since the chance of playback stalling depends on the video delivery time, we employ a time slot-based definition for the buffer status of the users. Let $B_u\left(\tau^{\text{REQ}}_{u,c,v}\right)$ (in time slot) denote the buffer status of user u in time slot $\tau^{\text{REQ}}_{u,c,v}$. To prevent buffer overflow, each user only requests a new video chunk when its buffer status is below a certain threshold [7]. Let B^{THR}_u (in time slot) denote the buffer size threshold for user u. When the buffer status of user u is above B^{THR}_u, the user will wait for a period of time before requesting the next video chunk to avoid buffer overflow. Let $\tau^{\text{WT}}_{u,c,v}$ (in time slot) denote the waiting time for user u after receiving chunk c of video v. We have

$$\tau^{\text{WT}}_{u,c,v}=\left[\left[B_u\left(\tau^{\text{REQ}}_{u,c,v}\right)-\tau^{\text{TD}}_{u,c,v}\right]^+ +T^{\text{chunk}}-B^{\text{THR}}_u\right]^+, \quad (5)$$

where $\tau^{\text{TD}}_{u,c,v}$ (in time slot) denotes the time it takes for chunk c of video v to be transmitted from the APs to user u. It is given by $\tau^{\text{TD}}_{u,c,v}=\min\left\{t'\in\mathcal{T}\,\middle|\,\sum_{t=\tau^{\text{REQ}}_{u,c,v}}^{t'}r_u(t)\geq T^{\text{chunk}}\sum_{n\in\mathcal{N}^{\text{pred}}_{u,c,v}}\sum_{m\in\mathcal{M}}\beta_{u,n,m}\nu_m\right\}-\tau^{\text{REQ}}_{u,c,v}+1$.

Considering the waiting time $\tau^{\text{WT}}_{u,c,v}$, the next chunk of video $v\in\mathcal{V}$ is requested by user $u\in\mathcal{U}$ in the following time slot:

$$\tau^{\text{REQ}}_{u,c+1,v}=\tau^{\text{REQ}}_{u,c,v}+\tau^{\text{TD}}_{u,c,v}+\tau^{\text{WT}}_{u,c,v},\quad u\in\mathcal{U},\ c\in\mathcal{C}_v\backslash\{C_v\},\ v\in\mathcal{V}. \quad (6)$$

The buffer status of user u at time slot $\tau^{\text{REQ}}_{u,c+1,v}$ is given by $B_u(\tau^{\text{REQ}}_{u,c+1,v})=\left[[B_u(\tau^{\text{REQ}}_{u,c,v})-\tau^{\text{TD}}_{u,c,v}]^+ +T^{\text{chunk}}-\tau^{\text{WT}}_{u,c,v}\right]^+$.

C. Quality of Experience Model

The CCU aims to maximize the users' QoE. Let $\Upsilon_{u,c,v}$ denote the QoE of user $u\in\mathcal{U}$ for chunk $c\in\mathcal{C}_v$ of video $v\in\mathcal{V}$. We consider that $\Upsilon_{u,c,v}$ depends on four factors: the average quality of the tiles in the viewport $\bar{\ell}^{\text{view}}_{u,c,v}$, the spatial quality smoothness of the tiles in the viewport $\ell^{\text{spatial}}_{u,c,v}$, the temporal quality smoothness of the tiles in the viewport $\ell^{\text{temp}}_{u,c,v}$, and the rebuffering delay $\tau^{\text{RD}}_{u,c,v}$ [8]. In the following, we describe how CCU obtains each of these QoE factors.

Let $\mathcal{N}^{\text{actual}}_{u,c,v}$ denote the set of tile indices that user $u\in\mathcal{U}$ has actually viewed for chunk $c\in\mathcal{C}_v$ of video $v\in\mathcal{V}$. $\bar{\ell}^{\text{view}}_{u,c,v}$ is obtained by averaging the quality of the tiles in set $\mathcal{N}^{\text{actual}}_{u,c,v}$. We have $\bar{\ell}^{\text{view}}_{u,c,v}=\frac{1}{\left|\mathcal{N}^{\text{actual}}_{u,c,v}\right|}\sum_{n\in\mathcal{N}^{\text{actual}}_{u,c,v}}\sum_{m\in\mathcal{M}}\beta_{u,n,m}m$.

The spatial quality smoothness factor measures the intra-chunk quality switch. In particular, the variance of the quality level of the tiles in the viewport, i.e., $\ell^{\text{spatial}}_{u,c,v}$, may lead to viewing irritation, cybersickness, and other physiological effects including fatigue, nausea, and aversion [8]. $\ell^{\text{spatial}}_{u,c,v}$ is obtained as $\ell^{\text{spatial}}_{u,c,v}=\frac{1}{\left|\mathcal{N}^{\text{actual}}_{u,c,v}\right|}\sum_{n\in\mathcal{N}^{\text{actual}}_{u,c,v}}\left(\sum_{m\in\mathcal{M}}\beta_{u,n,m}m-\bar{\ell}^{\text{view}}_{u,c,v}\right)^2$.

The temporal quality smoothness factor measures the inter-chunk quality switch. The user's QoE degrades when the average quality level of the tiles in the viewport differs between two consecutive chunks [7]. For $c=1$, we set $\ell^{\text{temp}}_{u,c,v}=0$. For $c>1$, we have $\ell^{\text{temp}}_{u,c,v}=\left|\bar{\ell}^{\text{view}}_{u,c,v}-\bar{\ell}^{\text{view}}_{u,c-1,v}\right|$.

The rebuffering delay $\tau^{\text{RD}}_{u,c,v}$ captures video stalling during playback [7]. A video is stalled when the downloading time of chunk c exceeds the user's buffer status at chunk c's request time. We have $\tau^{\text{RD}}_{u,c,v}=\left[\tau^{\text{TD}}_{u,c,v}-B_u\left(\tau^{\text{REQ}}_{u,c,v}\right)\right]^+$.

The QoE of user $u\in\mathcal{U}$ for chunk $c\in\mathcal{C}_v$ of video $v\in\mathcal{V}$ is the weighted sum of the mentioned factors. We have

$$\Upsilon_{u,c,v}=\bar{\ell}^{\text{view}}_{u,c,v}-\lambda^{\text{spatial}}\ell^{\text{spatial}}_{u,c,v}-\lambda^{\text{temp}}\ell^{\text{temp}}_{u,c,v}-\lambda^{\text{RD}}\tau^{\text{RD}}_{u,c,v},\quad u\in\mathcal{U},\ c\in\mathcal{C}_v,\ v\in\mathcal{V}, \quad (7)$$

where λ^{spatial}, λ^{temp}, and λ^{RD} are the non-negative weighting coefficients which penalize user u's QoE due to the nonzero intra-chunk quality switch, inter-chunk quality switch, and rebuffering delay, respectively.

IV. Optimizing Tile Bitrate Selection and Beamforming Design

Each user asynchronously requests a new video chunk based on its buffer status (see eqn. (6)). Upon receiving a chunk request, the CCU chooses an action (i.e., tile bitrate selection) given the observed system state information. The CCU aims to maximize the users' expected long-term QoE. To this end, the CCU should make *asynchronous* decisions on the quality level of the tiles requested by the users. Furthermore, the design of beamforming vectors in each time slot affects the rebuffering delay. At the beginning of each time slot, the CCU obtains an observation from the system and chooses an action (i.e., beamforming design) based on the selected bitrate for tiles. Given the asynchronous decision-making and hierarchical structure of this problem, we formulate it as a MacDec-POMDP [12], [13] with $T^{\max}$ decision epochs. In particular, we consider that for each user, an agent in the CCU is responsible to make decision on behalf of that user[2]. The agents cooperatively determine the bitrate of tiles by taking asynchronous macro-actions. The agents also cooperatively design the beamforming vectors through synchronized primitive-actions. Next, we describe the observation, action, and reward of each agent.

A. Observation

The global system state comprises the set of indices of the tiles that are actually viewed by the users (i.e., $\mathcal{N}^{\text{actual}}_{u,c,v}$). It also contains the set of APs which are not in the users' self-blockage region in each time slot (i.e., $\mathcal{A}^{\text{nb}}_u(t)$). However, the CCU does not have access to the global system state. Instead, it obtains a partial observation of the underlying system state. Among the N available tiles, let binary vector $\boldsymbol{v}_{u,c,v}\in\{0,1\}^N$ indicate the tile indices predicted by the viewport prediction framework to be transmitted to user $u\in\mathcal{U}$ upon its request for chunk $c\in\mathcal{C}_v$ of video $v\in\mathcal{V}$. The n-th element of vector $\boldsymbol{v}_{u,c,v}$ is equal to $\mathbb{1}(n\in\mathcal{N}^{\text{pred}}_{u,c,v})$. As we

[2]Considering one agent for each user makes our approach scalable. It also facilitates compatibility with existing adaptive bitrate streaming techniques.

will discuss in Section V-C, our proposed viewport prediction framework provides a fused feature map for each video frame by integrating the video saliency map and the user's head orientation map. We segment the fused feature map into N tiles. A higher feature value assigned to a tile on the map indicates that it is more likely for the user to view that tile in the corresponding video frame. We normalize the feature value of the tiles in the fused feature map to be within $[0,1]$. Let vector $\bar{\boldsymbol{\chi}}_{u,c,v} \in [0,1]^N$ denote the average feature value of tiles when chunk c of video v is requested by user u. The n-th element of vector $\bar{\boldsymbol{\chi}}_{u,c,v}$ is equal to the average feature value of the n-th tile across all video frames in chunk c. We also denote the average quality level of the tiles transmitted to user u upon its request for chunk c of video v by $\bar{\ell}^{\text{trans}}_{u,c,v}$. We have $\bar{\ell}^{\text{trans}}_{u,c,v} = \frac{1}{|\mathcal{N}^{\text{pred}}_{u,c,v}|}\sum_{n\in\mathcal{N}^{\text{pred}}_{u,c,v}}\sum_{m\in\mathcal{M}}\beta_{u,n,m}m$.

For the sake of brevity, we refer to the agent responsible to make decision on behalf of user $u \in \mathcal{U}$ as agent u. Let $\boldsymbol{o}^{\text{m}}_u(t)$ denote the macro-observation vector of agent u at the beginning of time slot $t \in \mathcal{T}$. When chunk $c \in \mathcal{C}_v$ of video $v \in \mathcal{V}$ is requested by user u, the macro-observation vector of agent u contains $\boldsymbol{v}_{u,c,v}$ and $\bar{\boldsymbol{\chi}}_{u,c,v}$. $\boldsymbol{o}^{\text{m}}_u(t)$ also contains the average quality level of the previously transmitted video chunk to user u (i.e., $\bar{\ell}^{\text{trans}}_{u,c-1,v}$) and the buffer status of user u in time slot $\tau_{u,c,v}$ (i.e., $B_u\left(\tau^{\text{REQ}}_{u,c,v}\right)$). For $t \in \{\tau^{\text{REQ}}_{u,c,v}, \tau^{\text{REQ}}_{u,c,v}+1\ldots, \tau^{\text{REQ}}_{u,c+1,v}-1\}$, we have $\boldsymbol{o}^{\text{m}}_u(t) = \left(\boldsymbol{v}_{u,c,v}, \bar{\boldsymbol{\chi}}_{u,c,v}, \bar{\ell}^{\text{trans}}_{u,c-1,v}, B_u\left(\tau^{\text{REQ}}_{u,c,v}\right)\right)$. Let $\mathcal{O}^{\text{m}}_u$ denote the macro-observation space over agent u. We have $\mathcal{O}^{\text{m}}_u = \{0,1\}^N \times [0,1]^N \times [1,M] \times \{0,\ldots,B^{\text{THR}}_u\}$.

Each agent $u \in \mathcal{U}$ selects the bitrate of the tiles in set $\mathcal{N}^{\text{pred}}_{u,c,v}$ based on its macro-observation vector $\boldsymbol{o}^{\text{m}}_u(t) \in \mathcal{O}^{\text{m}}_u$. At the beginning of each time slot $t \in \mathcal{T}$, the agents design the beamforming vectors based on their primitive-observation vector. We denote agent u's primitive-observation vector at the beginning of time slot t by $\boldsymbol{o}^{\text{p}}_u(t)$. At time slot t, the CCU does not have access to $\phi_u(t)$. Instead, using our proposed viewport prediction framework, the CCU obtains the predicted longitude angle of user u's head orientation (i.e. $\hat{\phi}_u(t)$). Thus, the CCU can *proactively* determine the self-blockage region of user u. We denote the set of APs which are predicted to be not in user u's self-blockage region in time slot t by $\hat{\mathcal{A}}^{\text{nb}}_u(t)$. Let binary vector $\boldsymbol{\varrho}_u(t) \in \{0,1\}^{N_{\text{AP}}}$ indicate the APs which are not in the self-blockage region of user u in time slot t. The a-th element of vector $\boldsymbol{\varrho}_u(t)$ is equal to $\mathbb{1}(a \in \hat{\mathcal{A}}^{\text{nb}}_u(t))$.

Let $\Delta^{\text{rem}}_u(t)$ denote the remaining bits of the requested chunk available for transmission to user $u \in \mathcal{U}$ in time slot $t \in \mathcal{T}$. At time slot $t = \tau^{\text{REQ}}_{u,c,v}$, $\Delta^{\text{rem}}_u(t)$ is initialized to be $T^{\text{chunk}}T^{\text{slot}}\sum_{n\in\mathcal{N}^{\text{pred}}_{u,c,v}}\sum_{m\in\mathcal{M}}\beta_{u,n,m}\nu_m$. For $t \in \{\tau^{\text{REQ}}_{u,c,v}+1, \tau^{\text{REQ}}_{u,c,v}+2, \ldots, \tau^{\text{REQ}}_{u,c+1,v}-1\}$, $\Delta^{\text{rem}}_u(t)$ is updated as $\Delta^{\text{rem}}_u(t) = \left[\Delta^{\text{rem}}_u(t-1) - T^{\text{slot}}r_u(t-1)\right]^+$. We also define parameter $\Delta^{\text{time}}_u(t)$ as the remaining time that user u can obtain its requested chunk without experiencing video stalling during playback. We initialize $\Delta^{\text{time}}_u(t) = B_u\left(\tau^{\text{REQ}}_{u,c,v}\right)$ at time slot $t = \tau^{\text{REQ}}_{u,c,v}$. For $t \in \{\tau^{\text{REQ}}_{u,c,v}+1, \tau^{\text{REQ}}_{u,c,v}+2, \ldots, \tau^{\text{REQ}}_{u,c+1,v}-1\}$, $\Delta^{\text{time}}_u(t)$ is updated as $\Delta^{\text{time}}_u(t) = \Delta^{\text{time}}_u(t-1) - 1$ when $\Delta^{\text{rem}}_u(t) > 0$, and is set to $\left[B_u\left(\tau^{\text{REQ}}_{u,c,v}\right) - \tau^{\text{TD}}_{u,c,v}\right]^+ + T^{\text{chunk}}$ when $\Delta^{\text{rem}}_u(t) = 0$.

At the beginning of time slot $t \in \mathcal{T}$, we have $\boldsymbol{o}^{\text{p}}_u(t) = \left(\boldsymbol{\varrho}_u(t), \Delta^{\text{rem}}_u(t), \Delta^{\text{time}}_u(t)\right)$ for each agent $u \in \mathcal{U}$. We denote the primitive-observation space over agent u by $\mathcal{O}^{\text{p}}_u = \{0,1\}^{N_{\text{AP}}} \times \mathbb{R}^+ \cup \{0\} \times \mathbb{Z}^- \cup \{0,\ldots,B^{\text{THR}}_u + T^{\text{chunk}} - 1\}$. Agent u takes a primitive-action in time slot t based on $\boldsymbol{o}^{\text{p}}_u(t) \in \mathcal{O}^{\text{p}}_u$.

B. Action

When user $u \in \mathcal{U}$ requests chunk $c \in \mathcal{C}_v$ of video $v \in \mathcal{V}$ at time slot $t = \tau^{\text{REQ}}_{u,c,v}$, agent u will take a macro-action which remains unchanged for $t \in \{\tau^{\text{REQ}}_{u,c,v}, \tau^{\text{REQ}}_{u,c,v}+1, \ldots, \tau^{\text{REQ}}_{u,c+1,v}-1\}$. Let $\boldsymbol{a}^{\text{m}}_u(t)$ denote the macro-action which is taken by agent u at time slot t. From Section III-B, recall that the quality level of the tiles is selected from set $\mathcal{M}$. Let $\nu_{u,n}$ denote the bitrate of tile $n \in \mathcal{N}^{\text{pred}}_{u,c,v}$ when user u requests chunk c of video v. We set $\nu_{u,n} = 0$ for $n \notin \mathcal{N}^{\text{pred}}_{u,c,v}$. Agent u selects the bitrate of tile $n \in \mathcal{N}^{\text{pred}}_{u,c,v}$ using the following relaxed constraint:

$$\nu_1 \leq \nu_{u,n} \leq \nu_M,\ u \in \mathcal{U},\ n \in \mathcal{N}^{\text{pred}}_{u,c,v},\ c \in \mathcal{C}_v,\ v \in \mathcal{V}. \quad (8)$$

Thus, we have $\boldsymbol{a}^{\text{m}}_u(t) = (\nu_{u,n}, n \in \mathcal{N})$ for $u \in \mathcal{U}$ and $t \in \{\tau^{\text{REQ}}_{u,c,v}, \tau^{\text{REQ}}_{u,c,v}+1, \ldots, \tau^{\text{REQ}}_{u,c+1,v}-1\}$. After determining $\nu_{u,n}$, agent u rounds it down to the nearest possible bitrate based on the quality levels available in set $\mathcal{M}$. Note that the variables $\beta_{u,n,m}$, $u \in \mathcal{U}$, $n \in \mathcal{N}$, $m \in \mathcal{M}$ can be determined using the obtained bitrate for the tiles. We denote the macro-action space over agent u by $\mathcal{A}^{\text{m}}_u = \{\{0\} \cup [\nu_1, \nu_M]\}^N$. Each agent ends its previous macro-action upon receiving a new video chunk request.

At the beginning of each time slot $t \in \mathcal{T}$, agent $u \in \mathcal{U}$ will take a primitive-action to determine the beamforming vectors. Let $\boldsymbol{a}^{\text{p}}_u(t)$ denote the primitive-action which is taken by agent u at the beginning of time slot t. We have $\boldsymbol{a}^{\text{p}}_u(t) = (\boldsymbol{b}_{u,a}(t), a \in \mathcal{A})$, $u \in \mathcal{U}$, $t \in \mathcal{T}$. We denote the primitive-action space over agent u by $\mathcal{A}^{\text{p}}_u = \mathbb{C}^{N_{\text{t}} \times N_{\text{AP}}}$.

C. Reward

The agents aim to cooperatively maximize the QoE of the users. In each time slot $t \in \mathcal{T}$, we consider a *shared* extrinsic reward over agents denoted by $R^{\text{extr}}(t)$. Given $\Upsilon_{u,c,v}$ from eqn. (7) as the QoE of user $u \in \mathcal{U}$ for chunk $c \in \mathcal{C}_v$ of video $v \in \mathcal{V}$, we have

$$R^{\text{extr}}(t) = \sum_{u\in\mathcal{U}}\sum_{v\in\mathcal{V}}\sum_{c\in\mathcal{C}_v}\Upsilon_{u,c,v}\mathbb{1}\left(\tau^{\text{REQ}}_{u,c,v} + \tau^{\text{TD}}_{u,c,v} = t\right). \quad (9)$$

Based on the defined $R^{\text{extr}}(t)$ in (9), the agents receive a non-zero reward in time slot t when at least one of the users has completely downloaded its requested video chunk[3].

By performing macro-action $\boldsymbol{a}^{\text{m}}_u(t) \in \mathcal{A}^{\text{m}}_u$ in time slot $t = \tau_{u,c,v}$, the number of bits required for transmitting chunk $c \in \mathcal{C}_v$ of video $v \in \mathcal{V}$ to user u can be determined. The agents aim to maximize the system's sum-rate in each time slot while

[3]In this work, we use a 360° video dataset that includes the users' viewport during video playback. Thus, the CCU can obtain $\Upsilon_{u,c,v}$ once user u has downloaded chunk c of video v. However, without such a dataset, the CCU can obtain $\Upsilon_{u,c,v}$ only after user u has watched chunk c of video v from its playback buffer. Hence, obtaining a policy without using a recorded 360° video dataset may lead to a delayed reward environment [36]. While dealing with delayed rewards is not within the scope of this work, this issue can be addressed by redefining the reward function and the observation-action trajectories in our proposed algorithms.

satisfying the selected bitrate of the requested video tiles for users through a sequence of primitive actions. The agents receive an intrinsic reward, denoted by $R^{\text{intr}}(t)$, for performing primitive-actions $\boldsymbol{a}_u^{\text{p}}(t) \in \mathcal{A}_u^{\text{p}}$, $u \in \mathcal{U}$ in time slot $t \in \mathcal{T}$. $R^{\text{intr}}(t)$ is shared over the agents and is obtained as follows:

$$R^{\text{intr}}(t) = \sum_{u \in \mathcal{U}} r_u(t) - \lambda^{\text{intr}} \sum_{u \in \mathcal{U}} \left[\Delta_u^{\text{rem}}(t) - T^{\text{slot}} r_u(t)\right]^+, \quad (10)$$

where λ^{intr} is a positive scaling factor.

D. Problem Formulation

In a POMDP, an agent does not have direct access to the underlying system state. However, a history of observations and actions provides sufficient statistics for the agent to make decisions [37], [38]. Let $\boldsymbol{h}_u^{\text{m}}(t)$ and $\boldsymbol{h}_u^{\text{p}}(t)$, respectively, denote the macro- and primitive-observation-action history of agent u in time slot $t \in \mathcal{T}$. Both macro- and primitive-actions are high-dimensional continuous control variables. Each agent u selects a macro-action $\boldsymbol{a}_u^{\text{m}}(t) = \mu_u\left(\boldsymbol{h}_u^{\text{m}}(t)\right)$ using a deterministic policy $\mu_u : \mathcal{H}_u^{\text{m}} \rightarrow \mathcal{A}_u^{\text{m}}$ based on its macro-observation-action history $\boldsymbol{h}_u^{\text{m}}(t) \in \mathcal{H}_u^{\text{m}}$ in time slot t. Agent u also uses a deterministic policy $\pi_u : \mathcal{H}_u^{\text{p}} \rightarrow \mathcal{A}_u^{\text{p}}$ to select a primitive-action $\boldsymbol{a}_u^{\text{p}}(t) = \pi_u\left(\boldsymbol{h}_u^{\text{p}}(t)\right)$ given $\boldsymbol{h}_u^{\text{p}}(t) \in \mathcal{H}_u^{\text{p}}$ in time slot t. Let $\boldsymbol{\mu} = (\mu_u,\, u \in \mathcal{U})$ and $\boldsymbol{\pi} = (\pi_u,\, u \in \mathcal{U})$ denote the agents' joint policies for bitrate selection and beamforming design, respectively. We also denote the joint macro- and primitive-actions performed by all agents in time slot t by $\boldsymbol{a}^{\text{m}}(t) = (\boldsymbol{a}_u^{\text{m}}(t),\, u \in \mathcal{U})$ and $\boldsymbol{a}^{\text{p}}(t) = \left(\boldsymbol{a}_u^{\text{p}}(t),\, u \in \mathcal{U}\right)$, respectively.

By employing a hierarchical learning framework [39], both $\boldsymbol{\mu}$ and $\boldsymbol{\pi}$ are learned in order to maximize the expected discounted extrinsic and intrinsic rewards, respectively. Let $Q^*\left(\boldsymbol{h}^{\text{m}}, \boldsymbol{a}^{\text{m}}\right)$ denote the maximum action-value function when agents choose the joint macro-action $\boldsymbol{a}^{\text{m}}$ given the joint macro-observation-action history $\boldsymbol{h}^{\text{m}}$. For tile bitrate selection, we formulate the following optimization problem:

$$\mathcal{P}^{\text{m}}: \quad Q^*\left(\boldsymbol{h}^{\text{m}}, \boldsymbol{a}^{\text{m}}\right) = \underset{\boldsymbol{\mu}}{\text{maximize}}\ \mathbb{E}_{\boldsymbol{\mu}}\left\{ \sum_{t'=t}^{T^{\max}} \gamma^{t'-t} R^{\text{extr}}(t') \,\middle|\, \boldsymbol{h}^{\text{m}}(t) = \boldsymbol{h}^{\text{m}},\, \boldsymbol{a}^{\text{m}}(t) = \boldsymbol{a}^{\text{m}} \right\} \quad \text{subject to constraint (8)},$$

where γ is the discount factor. Problem $\mathcal{P}^{\text{m}}$ aims to learn a joint policy that maximizes the expected discounted extrinsic reward over a sequence of macro-actions when starting from $\boldsymbol{h}^{\text{m}}(t) = \boldsymbol{h}^{\text{m}} \in \prod_{u \in \mathcal{U}} \mathcal{H}_u^{\text{m}}$, having the agents take $\boldsymbol{a}^{\text{m}}(t) = \boldsymbol{a}^{\text{m}} \in \prod_{u \in \mathcal{U}} \mathcal{A}_u^{\text{m}}$, and thereafter following policy $\boldsymbol{\mu}$.

Let $Q^*\left(\boldsymbol{h}^{\text{p}}, \boldsymbol{a}^{\text{p}}; \boldsymbol{a}^{\text{m}}\right)$ denote the maximum action-value function when the agents choose the joint primitive-action $\boldsymbol{a}^{\text{p}}$ given the joint primitive-observation-action history $\boldsymbol{h}^{\text{p}}$ and the specified joint macro-action $\boldsymbol{a}^{\text{m}}$. The beamforming problem can be formulated as follows:

$$\mathcal{P}^{\text{p}}: \quad Q^*\left(\boldsymbol{h}^{\text{p}}, \boldsymbol{a}^{\text{p}}; \boldsymbol{a}^{\text{m}}\right) = \underset{\boldsymbol{\pi}}{\text{maximize}}\ \mathbb{E}_{\boldsymbol{\pi}}\left\{ \sum_{t'=t}^{T^{\max}} \gamma^{t'-t} R^{\text{intr}}(t') \,\middle|\, \boldsymbol{h}^{\text{p}}(t) = \boldsymbol{h}^{\text{p}},\, \boldsymbol{a}^{\text{p}}(t) = \boldsymbol{a}^{\text{p}},\, \boldsymbol{a}^{\text{m}}(t) = \boldsymbol{a}^{\text{m}} \right\}$$

subject to constraint (1).

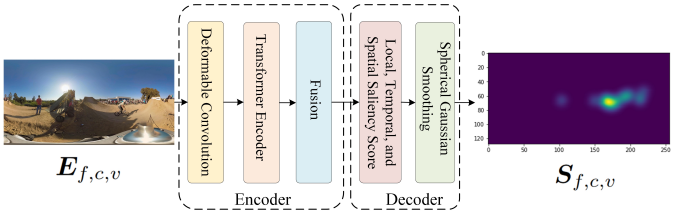


Fig. 5: PAVER model for saliency detection.

Problem $\mathcal{P}^{\text{p}}$ aims to learn a joint policy that leads to the maximum expected discounted intrinsic reward when starting from $\boldsymbol{h}^{\text{p}}(t) = \boldsymbol{h}^{\text{p}} \in \prod_{u \in \mathcal{U}} \mathcal{H}_u^{\text{p}}$, given the joint macro-action $\boldsymbol{a}^{\text{m}} \in \prod_{u \in \mathcal{U}} \mathcal{A}_u^{\text{m}}$ executed by the agents in time slot t. In problem $\mathcal{P}^{\text{p}}$, the expected discounted intrinsic reward is maximized over a sequence of primitive-actions, when the agents take $\boldsymbol{a}^{\text{p}}(t) = \boldsymbol{a}^{\text{p}} \in \prod_{u \in \mathcal{U}} \mathcal{A}_u^{\text{p}}$, and thereafter follow policy $\boldsymbol{\pi}$. Problems $\mathcal{P}^{\text{m}}$ and $\mathcal{P}^{\text{p}}$ are finite-horizon stochastic optimal control problems. These problems are difficult to solve due to their asynchronous decision-making and hierarchical structure. Moreover, the dynamics of users' viewports and their connections to APs are not known in advance. To address these challenges, we first propose a viewport prediction framework in Section V to predict the tiles in set $\mathcal{N}_{u,c,v}^{\text{pred}}$ as well as the head orientation of each user u. Then, in Section VI, we present two DRL algorithms based on multi-agent actor-critic methods to enable efficient macro- and primitive-action selection for the agents. These two algorithms are developed within a hierarchical learning framework to learn the joint policies $\boldsymbol{\mu}$ and $\boldsymbol{\pi}$ for problems $\mathcal{P}^{\text{m}}$ and $\mathcal{P}^{\text{p}}$, respectively.

V. Viewport Prediction Framework

In this section, we follow the idea proposed in our previous work [1] to develop a content-based viewport prediction framework. In particular, our proposed viewport prediction framework consists of three main components: a saliency detection model, a head movement prediction model, and an integration mechanism using fusion techniques. In the following subsections, we describe each of these three components.

A. Saliency Detection Model

A saliency detection model determines the saliency map for each video frame. This map helps identify those parts of a video frame that are more interesting to users. In this work, we use the PAVER model, proposed in [40], for saliency detection. The PAVER model can be trained without requiring explicit supervision. In particular, when using this model, it is not required that the saliency map of the video frames be part of the training dataset. This is important because the availability of saliency datasets is limited, and collecting saliency maps for 360° videos through human supervision is expensive [20]. Another advantage of PAVER is its ability to leverage the pretrained weights of a vision transformer (ViT) model, which has been trained on 2D images or videos. By using these pretrained weights, the saliency detection performance of the PAVER model on 360° video frames can be improved.

Let $\boldsymbol{E}_{f,c,v} \in \mathbb{R}^{3 \times W \times H}$ and $\boldsymbol{S}_{f,c,v} \in \mathbb{R}^{W \times H}$ denote frame $f \in \mathcal{F}$ of chunk $c \in \mathcal{C}_v$ of 360° video $v \in \mathcal{V}$ in equirectangular projection (ERP) format and its corresponding

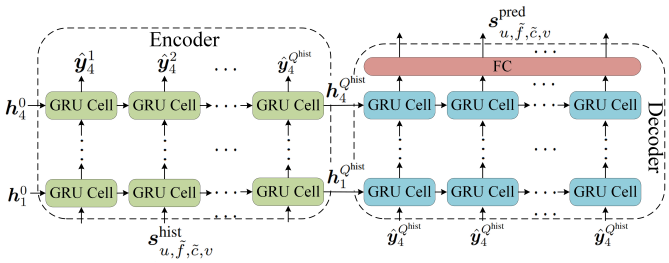


Fig. 6: GRU-based head movement prediction model.

saliency map, respectively. $W \times H$ denotes the video frame resolution (i.e., the number of pixels across the width and height of the video frame). As shown in Fig. 5, the PAVER model takes an ERP video frame as input. The output is the predicted saliency map. The input is divided into $I = \frac{W}{L} \times \frac{H}{L}$ patches, each having a resolution of $L \times L$. These patches pass through an encoder and a decoder module. The encoder module consists of a deformable convolution layer, a transformer encoder layer, and a fusion layer. The encoder module captures the local and global context of the input video frame.

The decoder module computes a saliency score for each patch. Local, temporal, and spatial saliencies are considered in determining the saliency score for each patch. All the saliency scores corresponding to the patches of the video frame $\boldsymbol{E}_{f,c,v}$ are collected in the saliency matrix $\hat{\boldsymbol{S}}_{f,c,v} \in \mathbb{R}^{\frac{W}{L} \times \frac{H}{L}}$. The final saliency map $\boldsymbol{S}_{f,c,v}$ is obtained using spherical Gaussian smoothing [40] to upsample from $\mathbb{R}^{\frac{W}{L} \times \frac{H}{L}}$ to $\mathbb{R}^{W \times H}$.

All 360° video frames are stored at the edge server. Thus, the saliency detection model can be trained entirely at the edge server. The weights of the deformable convolution and transformer encoder layers can be transferred from the pretrained ViT models without fine-tuning. Without any ground truth labels, the other layers in the PAVER model are trained by minimizing a weighted sum of the spatio-temporal consistency losses and the global context loss using Adam optimizer [41].

B. Head Movement Prediction Model

User $u \in \mathcal{U}$ is watching frame $\tilde{f} \in \mathcal{F}$ of chunk $\tilde{c} \in \mathcal{C}_v$ from its playback buffer, when it requests chunk c of video $v \in \mathcal{V}$ at time slot $\tau^{\text{REQ}}_{u,c,v}$. The head orientation angles of user u at time slot $\tau^{\text{REQ}}_{u,c,v}$, i.e., $\left(\theta_u(\tau^{\text{REQ}}_{u,c,v}), \phi_u(\tau^{\text{REQ}}_{u,c,v})\right)$, can be equivalently represented by $\left(\theta_{u,\tilde{f},\tilde{c},v}, \phi_{u,\tilde{f},\tilde{c},v}\right)$. To predict the head orientation angles of user u, a head movement prediction model takes a sequence with length Q^{hist} of the current and previous head movements (i.e., $\boldsymbol{s}^{\text{hist}}_{u,\tilde{f},\tilde{c},v}$) as input, and returns a sequence with length Q^{pred} which contains the predicted head movements for the frames of chunk c (i.e., $\boldsymbol{s}^{\text{pred}}_{u,\tilde{f},\tilde{c},v}$). To train the same model for all users, we set $Q^{\text{pred}} = F + \left\lfloor F \max_{u \in \mathcal{U}} B^{\text{THR}}_u / T^{\text{chunk}} \right\rfloor$.

Model: As shown in Fig. 6, the head movement prediction model comprises an encoder and a decoder, each containing four GRU layers. Each GRU layer has Q^{hist} and Q^{pred} GRU cells in the encoder and decoder modules, respectively. Let d^{GRU} denote the number of features in the hidden state of each GRU cell. $\boldsymbol{h}^0_i \in \mathbb{R}^{d^{\text{GRU}}}$ is the initial hidden state of layer $i \in \{1, \ldots, 4\}$. $\boldsymbol{h}^0_i$ is initialized to zero for the first input sequence of user u's head movements corresponding to each video. The input of the first GRU layer in the encoder module is the sequence $\boldsymbol{s}^{\text{hist}}_{u,\tilde{f},\tilde{c},v}$. The output of the last GRU layer in the decoder module passes through a fully connected (FC) layer. The output of the FC layer is the sequence $\boldsymbol{s}^{\text{pred}}_{u,\tilde{f},\tilde{c},v}$.

Algorithm 1 PFL-based Training Algorithm

1: Set the number of communication rounds R, the number of local update iterations ρ, and the learning rate η^{head}.
2: Randomly initialize $\boldsymbol{w}^{\text{GRU}}_0$ and $\boldsymbol{w}^{\text{FC}}_0$; set $\boldsymbol{w}^{\text{FC}}_u := \boldsymbol{w}^{\text{FC}}_0$, $u \in \mathcal{U}$.
3: **for** each communication round $r \in \mathcal{R} = \{1, \ldots, R\}$ **do**
4: **for** each user $u \in \mathcal{U}$ in parallel **do**
5: Given $\boldsymbol{w}^{\text{GRU}}_{r-1}$ and $\mathcal{L}^{\text{head}}_u$ in (11), perform ρ local update iterations to obtain the updated $\boldsymbol{w}^{\text{GRU}}_{u,r}$.
6: **end for**
7: $\boldsymbol{w}^{\text{GRU}}_r := \sum_{u \in \mathcal{U}} \alpha_u \boldsymbol{w}^{\text{GRU}}_{u,r}$.
8: **end for**
9: **for** each user $u \in \mathcal{U}$ in parallel **do**
10: Fine-tune the learning model $\boldsymbol{w}_u := \{\boldsymbol{w}^{\text{GRU}}_R, \boldsymbol{w}^{\text{FC}}_u\}$ using the local historical data.
11: **end for**

Loss Function: Let $\mathcal{V}^{\text{head-tr}}_u$ denote the set of videos for which user $u \in \mathcal{U}$ has its head movement measurements in the local training dataset. Let $\boldsymbol{w}_u$ denote the learning parameters of the head movement prediction model for user u. Each user u aims to determine $\boldsymbol{w}_u$ by minimizing the following loss function based on its local historical head movement:

$$\mathcal{L}^{\text{head}}_u = \frac{1}{|\mathcal{V}^{\text{head-tr}}_u|} \sum_{v \in \mathcal{V}^{\text{head-tr}}_u} \frac{1}{C_v} \sum_{\tilde{c} \in \mathcal{C}_v \setminus \{C_v\}} \frac{1}{F} \sum_{\tilde{f} \in \mathcal{F}} \epsilon^2_{u,\tilde{f},\tilde{c},v}, \quad (11)$$

where $\epsilon_{u,\tilde{f},\tilde{c},v} = \left\| \boldsymbol{s}^{\text{pred}}_{u,\tilde{f},\tilde{c},v} - \boldsymbol{s}^{\text{actual}}_{u,\tilde{f},\tilde{c},v} \right\|$, and $\boldsymbol{s}^{\text{actual}}_{u,\tilde{f},\tilde{c},v}$ is the sequence of user u's actual head movements in its dataset.

Training: We propose a PFL algorithm for training the head movement prediction model to address the data heterogeneity issue and preserve the users' privacy. Model decomposition has recently emerged as a promising method for PFL. For model decomposition, we use an approach similar to FedBABU [19] and PerFedMask [42]. For each user $u \in \mathcal{U}$, we decompose the learning model $\boldsymbol{w}_u$ into a global learning model $\boldsymbol{w}^{\text{GRU}}$ and a local head model $\boldsymbol{w}^{\text{FC}}_u$. We have $\boldsymbol{w}_u = \{\boldsymbol{w}^{\text{GRU}}, \boldsymbol{w}^{\text{FC}}_u\}$. $\boldsymbol{w}^{\text{GRU}}$ and $\boldsymbol{w}^{\text{FC}}_u$ contain the learning parameters of the GRU layers and the FC layer, respectively. All the local head models $\boldsymbol{w}^{\text{FC}}_u$, $u \in \mathcal{U}$, are initialized with the same random weights $\boldsymbol{w}^{\text{FC}}_0$ and are kept fixed during training of the global model $\boldsymbol{w}^{\text{GRU}}$. After convergence to a global model, each user u obtains its personalized model by fine-tuning the learning model $\boldsymbol{w}_u$ based on its local historical data.

The global model $\boldsymbol{w}^{\text{GRU}}$ is trained through communication rounds. Let $\mathcal{R} = \{1, \ldots, R\}$ denote the set of communication rounds. At the beginning of each communication round $r \in \mathcal{R}$, the users download the latest global model $\boldsymbol{w}^{\text{GRU}}_{r-1}$ from the server. $\boldsymbol{w}^{\text{GRU}}_0$ is initialized randomly. Each user $u \in \mathcal{U}$ initializes its global model $\boldsymbol{w}^{\text{GRU}}_{u,r}$ by the downloaded global model. We have $\boldsymbol{w}^{\text{GRU}}_{u,r} = \boldsymbol{w}^{\text{GRU}}_{r-1}$, $u \in \mathcal{U}$. Then, each user performs ρ local update iterations to update the global model using its local historical data. For each local update iteration, we have $\boldsymbol{w}^{\text{GRU}}_{u,r} \leftarrow \boldsymbol{w}^{\text{GRU}}_{u,r} - \eta^{\text{head}} \nabla_{\boldsymbol{w}^{\text{GRU}}} \mathcal{L}^{\text{head}}_u \big|_{\boldsymbol{w}^{\text{GRU}} = \boldsymbol{w}^{\text{GRU}}_{u,r}}$, where η^{head} is the learning rate. After completing the local update iterations, each user uploads its updated global model $\boldsymbol{w}^{\text{GRU}}_{u,r}$ to the server. At the end of each communication round r, the server computes a new global model $\boldsymbol{w}^{\text{GRU}}_r$ by aggregating the updated global models it has received from the users. We have

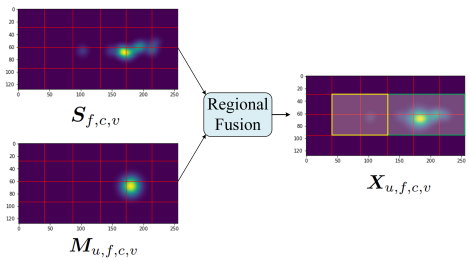


Fig. 7: Regional fusion for obtaining the fused feature map.

$\boldsymbol{w}_r^{\text{GRU}} = \sum_{u\in\mathcal{U}} \alpha_u \boldsymbol{w}_{u,r}^{\text{GRU}}$, where $\alpha_u = \frac{\sum_{v\in\mathcal{V}_u^{\text{head-tr}}} C_v}{\sum_{u'\in\mathcal{U}}\sum_{v'\in\mathcal{V}_{u'}^{\text{head-tr}}} C_{v'}}$ is the weight of user u in model aggregation. Algorithm 1 summarizes our PFL-based training.

Head Orientation Map: User $u \in \mathcal{U}$ uses its pre-trained head movement prediction model $\boldsymbol{w}_u$ to predict its head orientation for frame $f \in \mathcal{F}$ of chunk $c \in \mathcal{C}_v$ of video $v \in \mathcal{V}$. When user u requests video chunk c at time slot $\tau_{u,c,v}^{\text{REQ}}$, it also provides the server with its predicted head orientations $\left(\hat{\theta}_{u,f,c,v}, \hat{\phi}_{u,f,c,v}\right)$, $f \in \mathcal{F}$. Then, the server employs a Gaussian kernel [20] to obtain the head orientation maps of user u. We denote the head orientation map corresponding to user u's predicted head orientation, i.e., $\left(\hat{\theta}_{u,f,c,v}, \hat{\phi}_{u,f,c,v}\right)$, by $\boldsymbol{M}_{u,f,c,v} \in \mathbb{R}^{W\times H}$.

C. Integration of Saliency and Head Orientation Maps

When the server receives a request from user $u \in \mathcal{U}$ for chunk $c \in \mathcal{C}_v$ of video $v \in \mathcal{V}$, it aims to obtain the set of tile indices corresponding to viewport and marginal regions of each video frame $f \in \mathcal{F}$, i.e., $\mathcal{N}_{u,f,c,v}^{\text{view}}$ and $\mathcal{N}_{u,f,c,v}^{\text{marg}} \subset \mathcal{N}$, respectively. The saliency and the user's head orientation maps for video frame f are first divided into N tiles. Then, a fused feature map is obtained by integrating those maps using the *regional* fusion technique [1]. Finally, the tiles covering the viewport and marginal regions of a video frame are determined based on the feature value of the tiles in the fused feature map.

Regional Fusion Technique: In this fusion technique, we first normalize each feature in the maps $\boldsymbol{S}_{f,c,v}$ and $\boldsymbol{M}_{u,f,c,v}$ to be within $[0,1]$. Let $\tilde{\boldsymbol{S}}_{f,c,v}$ and $\tilde{\boldsymbol{M}}_{u,f,c,v} \in \mathbb{R}^{W\times H}$ denote the normalized versions of $\boldsymbol{S}_{f,c,v}$ and $\boldsymbol{M}_{u,f,c,v}$, respectively. We segment $\tilde{\boldsymbol{S}}_{f,c,v}$ and $\tilde{\boldsymbol{M}}_{u,f,c,v}$ into N tiles. We obtain the maximum saliency score of the pixels in each tile of $\tilde{\boldsymbol{S}}_{f,c,v}$ and $\tilde{\boldsymbol{M}}_{u,f,c,v}$, and collect them in vectors $\boldsymbol{s}_{f,c,v}^{\max}$ and $\boldsymbol{m}_{u,f,c,v}^{\max} \in \mathbb{R}^N$, respectively. By integrating $\tilde{\boldsymbol{S}}_{f,c,v}$ and $\tilde{\boldsymbol{M}}_{u,f,c,v}$, the fused feature map $\boldsymbol{X}_{u,f,c,v}$ is obtained as shown in Fig. 7. We have $\boldsymbol{X}_{u,f,c,v} = \left(\max_{n\in\mathcal{N}} \boldsymbol{s}_{f,c,v}^{\max}[n] - \frac{1}{N}\sum_{n\in\mathcal{N}} \boldsymbol{s}_{f,c,v}^{\max}[n]\right)^2 \tilde{\boldsymbol{S}}_{f,c,v} + \left(\max_{n\in\mathcal{N}} \boldsymbol{m}_{u,f,c,v}^{\max}[n] - \frac{1}{N}\sum_{n\in\mathcal{N}} \boldsymbol{m}_{u,f,c,v}^{\max}\right)^2 \tilde{\boldsymbol{M}}_{u,f,c,v}$.

Tile Selection: The number of tiles that cover the viewport region depends on the FoV of the user's HMD. Thus, the size of FoV specifies $\left|\mathcal{N}_{u,f,c,v}^{\text{view}}\right|$. The server selects the tiles in the viewport region (i.e., $\mathcal{N}_{u,f,c,v}^{\text{view}}$) based on the adjacent tiles with the maximum feature values in the fused feature map $\boldsymbol{X}_{u,f,c,v}$. On the other hand, the number of tiles that cover the marginal region depends on the prediction accuracy of the user u's predicted head orientation [43]. The difference between the requested chunk index c and the last watched chunk index $\tilde{c}$ by user u at time slot $\tau_{u,c,v}^{\text{REQ}}$ has a direct impact on the prediction accuracy. Thus, we consider that the number of tiles in the marginal region is obtained as $N_{u,f,c,v}^{\text{marg}} = \left[\left\lfloor(\alpha^{\text{marg}}(c-\tilde{c})-1)\left|\mathcal{N}_{u,f,c,v}^{\text{view}}\right|\right\rfloor\right]^+$, where α^{marg} is a scaling factor that can be empirically determined. To obtain the tiles in set $\mathcal{N}_{u,f,c,v}^{\text{marg}}$, the server initializes $\mathcal{N}_{u,f,c,v}^{\text{pred}}$ to be $\mathcal{N}_{u,f,c,v}^{\text{view}}$ and invokes the following step $N_{u,f,c,v}^{\text{marg}}$ times: the server selects a tile $n \notin \mathcal{N}_{u,f,c,v}^{\text{pred}}$ which is adjacent to set $\mathcal{N}_{u,f,c,v}^{\text{pred}}$ and has the maximum feature value in the fused feature map to be added to set $\mathcal{N}_{u,f,c,v}^{\text{pred}}$. The tiles in the marginal region are determined as $\mathcal{N}_{u,f,c,v}^{\text{marg}} = \mathcal{N}_{u,f,c,v}^{\text{pred}} \backslash \mathcal{N}_{u,f,c,v}^{\text{view}}$.

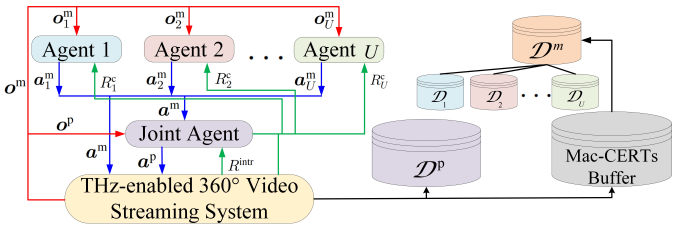


Fig. 8: Mac-IAICC and Prim-CAC are used within the agents and the joint agent, respectively. Each agent executes a macro-action based on its macro-observation. The joint agent executes the primitive-action based on the primitive-observation. At each time slot, the transition tuples of the agents and the joint agent are stored in the Mac-CERTs buffer and $\mathcal{D}^{\text{p}}$, respectively.

VI. DRL Algorithm Design

In this section, we develop a hierarchical learning framework using two DRL algorithms based on multi-agent cooperative actor-critic methods to solve problems $\mathcal{P}^{\text{m}}$ and $\mathcal{P}^{\text{p}}$. For each problem, we propose a DDPG algorithm to learn the agents' policy and action-value functions using actor and critic networks, respectively. Note that independent actor and critic networks for each agent may lead to a non-stationary learning environment in a multi-agent setting [44]. In particular, if each agent independently performs policy updating and exploring, it may not be possible for the agents to obtain high-quality cooperative policies. To address this issue, we leverage the centralized training with decentralized execution (CTDE) approach [12] to solve problem $\mathcal{P}^{\text{m}}$. By using this approach, we can learn a Mac-IAICC for each agent. Mac-IAICC facilitates offline training using centralized information and online execution in a decentralized manner for the agents. Thus, it is a suitable approach for selecting the bitrate of tiles requested asynchronously by users. To solve problem $\mathcal{P}^{\text{p}}$, which requires the agents to work in a synchronized manner, we use the Prim-CAC approach. This approach enables the agents to cooperatively design the beamforming vectors in each time slot. Fig. 8 shows an illustration of the interaction among the agents and the environment. Next, we present two algorithms for solving $\mathcal{P}^{\text{m}}$ and $\mathcal{P}^{\text{p}}$ using Mac-IAICC and Prim-CAC, respectively. Then, we propose a hierarchical learning framework to train the actor and critic networks.

A. Solving Problem $\mathcal{P}^m$ Using Mac-IAICC

To solve problem $\mathcal{P}^{\text{m}}$, we develop a DDPG algorithm to learn an independent actor and an individual centralized critic for each agent. Let $\boldsymbol{\omega}_u$ and $\boldsymbol{\vartheta}_u$ denote the learnable parameters of the neural networks corresponding to the actor and critic of agent $u \in \mathcal{U}$, respectively. Agent u's actor network specifies the policy of that agent for tile bitrate selection. We have

Algorithm 2 Parameters Update in Mac-IAICC with DDPG

1: **function** MacUpdateParams(B^{m}, $\mathcal{D}^{\mathrm{m}}$, $\mathcal{D}_u$, $\boldsymbol{\vartheta}_u$, $\boldsymbol{\omega}_u$)
2: Sample a random minibatch of B^{m} consecutive transition tuples from the sets $\mathcal{D}^{\mathrm{m}}$ and $\mathcal{D}_u$.
3: Obtain the gradient of $\mathcal{L}_u^{\mathrm{TD}}(\boldsymbol{\vartheta}_u)$ in (12) and update $\boldsymbol{\vartheta}_u$ using Adam optimizer.
4: Determine the gradient in (13) and update $\boldsymbol{\omega}_u$ using Adam optimizer.
5: **Return** $\boldsymbol{\vartheta}_u$ and $\boldsymbol{\omega}_u$
6: **end function**

Algorithm 3 Parameters Update in Prim-CAC with DDPG

1: **function** PrimUpdateParams(B^{p}, $\mathcal{D}^{\mathrm{p}}$, $\boldsymbol{\vartheta}$, $\boldsymbol{\omega}$)
2: Sample a random minibatch of B^{p} consecutive transition tuples from the set $\mathcal{D}^{\mathrm{p}}$.
3: Obtain the gradient of $\mathcal{L}^{\mathrm{TD}}(\boldsymbol{\vartheta})$ in (14) and update $\boldsymbol{\vartheta}$ using Adam optimizer.
4: Determine the gradient in (15) and update $\boldsymbol{\omega}$ using Adam optimizer.
5: **Return** $\boldsymbol{\vartheta}$ and $\boldsymbol{\omega}$
6: **end function**

$\boldsymbol{a}_u^{\mathrm{m}}(t) = \mu_{\boldsymbol{\omega}_u}(\boldsymbol{h}_u^{\mathrm{m}}(t))$. The centralized critic network of agent u learns the action-value function $Q_{\boldsymbol{\vartheta}_u}(\boldsymbol{h}^{\mathrm{m}}(t), \boldsymbol{a}^{\mathrm{m}}(t))$. Note that $\boldsymbol{h}_u^{\mathrm{m}}(t)$ can be generated implicitly for the actor and critic networks by leveraging an LSTM layer within their neural network architectures [12].

DDPG is an off-policy algorithm that is able to learn the policy and action-value functions in a stable and robust manner using a replay buffer and separate target networks [38]. Since agents in a MacDec-POMDP asynchronously start and complete their macro-actions, a new replay buffer needs to be designed. Thus, we collect the macro-observation, macro-action, and extrinsic reward of the agents into a buffer called macro-action concurrent experience replay trajectories (Mac-CERTs) in each time slot. The Mac-CERTs buffer has been used in [12] to solve a MacDec-POMDP problem using policy gradient algorithm. In this work, we utilize Mac-CERTs buffer to train the actor and critic networks of each agent using a DDPG algorithm. For $t \in \{\tau_{u,c,v}^{\mathrm{REQ}}, \tau_{u,c,v}^{\mathrm{REQ}}+1, \ldots, \tau_{u,c+1,v}^{\mathrm{REQ}}-1\}$, agent $u \in \mathcal{U}$ receives a cumulative reward for the macro-action at time slot $\tau_{u,c,v}^{\mathrm{REQ}}$ as $R_u^{\mathrm{c}}(t) = \sum_{t'=\tau_{u,c,v}^{\mathrm{REQ}}}^{t} \gamma^{t'-\tau_{u,c,v}^{\mathrm{REQ}}} R^{\mathrm{extr}}(t')$. Then, at the end of each time slot t, agent u stores its transition experience in the Mac-CERTs buffer as a tuple $(\boldsymbol{o}_u^{\mathrm{m}}(t), \boldsymbol{a}_u^{\mathrm{m}}(t), \boldsymbol{o}_u^{\mathrm{m}}(t+1), R_u^{\mathrm{c}}(t))$. Note that $\boldsymbol{o}_u^{\mathrm{m}}(t)$, $\boldsymbol{a}_u^{\mathrm{m}}(t)$, and $\boldsymbol{o}_u^{\mathrm{m}}(t+1)$ remain unchanged until agent u completes its current macro-action at the end of time slot $t = \tau_{u,c+1,v}^{\mathrm{REQ}} - 1$. Thus, a macro action $\boldsymbol{a}_u^{\mathrm{m}}(t)$ takes $\Delta_\tau(\boldsymbol{a}_u^{\mathrm{m}}(t)) = \tau_{u,c+1,v}^{\mathrm{REQ}} - \tau_{u,c,v}^{\mathrm{REQ}}$ time slots to complete. When training the actor network of agent u, we only consider the tuples in the Mac-CERTs buffer that correspond to agent u. We filter those tuples by selecting the ones that agent u completes its macro-action. However, training each agent's critic network requires all the joint macro-observation-action information. To train the critic networks, we filter the tuples in the Mac-CERTs buffer by selecting the ones that an agent has completed its macro-action. Let $\mathcal{D}_u$ and $\mathcal{D}^{\mathrm{m}}$ denote the set of tuples which are used for training agent u's actor and critic networks, respectively.

Each agent u updates the learnable parameters of its critic network (i.e., $\boldsymbol{\vartheta}_u$) by minimizing the temporal difference (TD) error over the tuples sampled from set $\mathcal{D}^{\mathrm{m}}$. We create a copy of the actor and critic networks to be used as target networks. Let $\boldsymbol{\omega}_u^-$ and $\boldsymbol{\vartheta}_u^-$, respectively, denote the weights of agent u's target actor and critic networks. The weights of agent u's target networks are updated by slowly tracking the learned parameters of its actor and critic networks. The TD error is obtained as follows:

$$\mathcal{L}_u^{\mathrm{TD}}(\boldsymbol{\vartheta}_u) = \mathbb{E}_{\Lambda_u^{\mathrm{m}} \sim \mathcal{D}^{\mathrm{m}}}\Big\{\big(Q_{\boldsymbol{\vartheta}_u}(\boldsymbol{h}^{\mathrm{m}}, \boldsymbol{a}^{\mathrm{m}}) - \hat{Q}_{\boldsymbol{\vartheta}_u^-}(\boldsymbol{h}^{\mathrm{m}}, \boldsymbol{a}^{\mathrm{m}})\big)^2\Big\}, \tag{12}$$

where $\Lambda_u^{\mathrm{m}} = \big(\boldsymbol{o}^{\mathrm{m}}, \boldsymbol{a}^{\mathrm{m}}, \boldsymbol{o}'^{\mathrm{m}}, R_u^{\mathrm{c}}\big)$ and $\hat{Q}_{\boldsymbol{\vartheta}_u^-}\big(\boldsymbol{h}^{\mathrm{m}}, \boldsymbol{a}^{\mathrm{m}}\big) = R_u^{\mathrm{c}} + \gamma^{\Delta_\tau(\boldsymbol{a}_u^{\mathrm{m}})} Q_{\boldsymbol{\vartheta}_u^-}\big(\boldsymbol{h}'^{\mathrm{m}}, \boldsymbol{a}'^{\mathrm{m}}\big)$. We obtain $\boldsymbol{a}'^{\mathrm{m}}$ using the target actor networks of the agents. We have $\boldsymbol{a}'^{\mathrm{m}} = \big(\mu_{\boldsymbol{\omega}_u^-}\big(\boldsymbol{h}'^{\mathrm{m}}_u\big), u \in \mathcal{U}\big)$.

Each agent $u \in \mathcal{U}$ updates the learnable parameters of its actor network (i.e., $\boldsymbol{\omega}_u$) using the gradient of the action-value function. The policy gradient is obtained as follows [38]:

$$\nabla_{\boldsymbol{\omega}_u} \mathcal{L}_u^{\mathrm{PG}} = \mathbb{E}_{\boldsymbol{o}_u^{\mathrm{m}} \sim \mathcal{D}_u^{\boldsymbol{o}}}\Big\{\nabla_{\boldsymbol{a}_u^{\mathrm{m}}} Q_{\boldsymbol{\vartheta}_u}(\boldsymbol{h}^{\mathrm{m}}, \boldsymbol{a}^{\mathrm{m}})\big|_{\boldsymbol{a}_u^{\mathrm{m}} = \mu_{\boldsymbol{\omega}_u}(\boldsymbol{h}_u^{\mathrm{m}})} \times \nabla_{\boldsymbol{\omega}_u} \mu_{\boldsymbol{\omega}_u}(\boldsymbol{h}_u^{\mathrm{m}})\Big\}, \tag{13}$$

where $\mathcal{D}_u^{\boldsymbol{o}} = \big\{\boldsymbol{o}_u^{\mathrm{m}} \,\big|\, \big(\boldsymbol{o}_u^{\mathrm{m}}, \boldsymbol{a}_u^{\mathrm{m}}, \boldsymbol{o}'^{\mathrm{m}}_u, R_u^{\mathrm{c}}\big) \in \mathcal{D}_u\big\}$. Algorithm 2 describes the MacUpdateParams function, which updates the actor and critic network parameters for each agent u.

B. Solving Problem $\mathcal{P}^{\mathrm{p}}$ Using Prim-CAC

The agents aim to cooperatively solve problem $\mathcal{P}^{\mathrm{p}}$ in a synchronized manner. This problem can be efficiently solved by treating all the agents as a joint agent. As a result, the learning environment turns into a fully centralized learning setting, where the joint agent trains both the centralized actor and critic networks to obtain $\boldsymbol{\pi}_{\boldsymbol{\omega}}\big(\boldsymbol{h}^{\mathrm{p}}\big)$ and $Q_{\boldsymbol{\vartheta}}\big(\boldsymbol{h}^{\mathrm{p}}, \boldsymbol{a}^{\mathrm{p}}; \boldsymbol{a}^{\mathrm{m}}\big)$, respectively. To this end, the joint agent collects the joint primitive-observations and actions, as well as the intrinsic rewards into a replay buffer. Recall from Section IV-C that $R^{\mathrm{intr}}(t)$ represents the agents' intrinsic reward in time slot $t \in \mathcal{T}$. At the end of each time slot t, the tuple $\big(\boldsymbol{o}^{\mathrm{p}}(t), \boldsymbol{a}^{\mathrm{p}}(t), \boldsymbol{o}^{\mathrm{p}}(t+1), R^{\mathrm{intr}}(t)\big)$ is stored in the replay buffer. Let $\mathcal{D}^{\mathrm{p}}$ denote the set of tuples in the replay buffer. The learnable parameters of the centralized critic network (i.e., $\boldsymbol{\vartheta}$) are updated by minimizing the following TD error:

$$\mathcal{L}^{\mathrm{TD}}(\boldsymbol{\vartheta}) = \mathbb{E}_{\Lambda^{\mathrm{p}} \sim \mathcal{D}^{\mathrm{p}}}\Big\{\big(Q_{\boldsymbol{\vartheta}}(\boldsymbol{h}^{\mathrm{p}}, \boldsymbol{a}^{\mathrm{p}}; \boldsymbol{a}^{\mathrm{m}}) - \hat{Q}_{\boldsymbol{\vartheta}^-}(\boldsymbol{h}^{\mathrm{p}}, \boldsymbol{a}^{\mathrm{p}}; \boldsymbol{a}^{\mathrm{m}})\big)^2\Big\}, \tag{14}$$

where $\Lambda^{\mathrm{p}} = \big(\boldsymbol{o}^{\mathrm{p}}, \boldsymbol{a}^{\mathrm{p}}, \boldsymbol{o}'^{\mathrm{p}}, R^{\mathrm{intr}}\big)$ and $\hat{Q}_{\boldsymbol{\vartheta}^-}\big(\boldsymbol{h}^{\mathrm{p}}, \boldsymbol{a}^{\mathrm{p}}; \boldsymbol{a}^{\mathrm{m}}\big) = R^{\mathrm{intr}} + \gamma Q_{\boldsymbol{\vartheta}^-}\big(\boldsymbol{h}'^{\mathrm{p}}, \boldsymbol{\pi}_{\boldsymbol{\omega}^-}\big(\boldsymbol{h}'^{\mathrm{p}}\big); \boldsymbol{a}'^{\mathrm{m}}\big)$. We update the centralized actor network's parameters using the following policy gradient:

$$\nabla_{\boldsymbol{\omega}} \mathcal{L}^{\mathrm{PG}} = \mathbb{E}_{\boldsymbol{o}^{\mathrm{p}} \sim \mathcal{D}^{\boldsymbol{o}}}\Big\{\nabla_{\boldsymbol{a}^{\mathrm{p}}} Q_{\boldsymbol{\vartheta}}(\boldsymbol{h}^{\mathrm{p}}, \boldsymbol{a}^{\mathrm{p}}; \boldsymbol{a}^{\mathrm{m}})\big|_{\boldsymbol{a}^{\mathrm{p}} = \boldsymbol{\pi}_{\boldsymbol{\omega}}(\boldsymbol{h}^{\mathrm{p}})} \times \nabla_{\boldsymbol{\omega}} \boldsymbol{\pi}_{\boldsymbol{\omega}}(\boldsymbol{h}^{\mathrm{p}})\Big\}, \tag{15}$$

where $\mathcal{D}^{\boldsymbol{o}} = \big\{\boldsymbol{o}^{\mathrm{p}} \,\big|\, \Lambda^{\mathrm{p}} \in \mathcal{D}^{\mathrm{p}}\big\}$. Algorithm 3 describes the PrimUpdateParams function, which updates the actor and critic network parameters for the joint agent.

C. Hierarchical Learning Framework

The parameters of the Mac-IAICC for each agent and the parameters of the Prim-CAC for the considered joint agent are learned at different time scales. In particular, the joint

Algorithm 4 Our Proposed Hierarchical Learning Framework

Set the maximum number of episodes $E^{\max}$, the minibatch sizes B^{m} and B^{p}, the learning rate for the actor network η^{a}, the learning rate for the critic network η^{c}, and the soft target network update constant $\varepsilon \in (0,1)$.
Randomly initialize the learnable parameters $\boldsymbol{\vartheta}_u$ and $\boldsymbol{\omega}_u$ for each agent $u \in \mathcal{U}$, and the learnable parameters $\boldsymbol{\vartheta}$ and $\boldsymbol{\omega}$ for the joint agent.
Set the target network weights $\boldsymbol{\vartheta}_u^- := \boldsymbol{\vartheta}_u$ and $\boldsymbol{\omega}_u^- := \boldsymbol{\omega}_u$ for each agent $u \in \mathcal{U}$, as well as $\boldsymbol{\vartheta}^- := \boldsymbol{\vartheta}$ and $\boldsymbol{\omega}^- := \boldsymbol{\omega}$ for the joint agent.
Initialize the set $\mathcal{D}_u$ for each agent $u \in \mathcal{U}$, as well as the sets $\mathcal{D}^{\mathrm{m}}$ and $\mathcal{D}^{\mathrm{p}}$ by performing random exploration for $T^{\text{warm-up}}$ time slots.
for each episode **do**
  Initialize the macro-observation vector $\boldsymbol{o}_u^{\mathrm{m}}$ for each agent $u \in \mathcal{U}$.
  Determine the macro-action $\boldsymbol{a}_u^{\mathrm{m}} \leftarrow \mu_{\boldsymbol{\omega}_u}(\boldsymbol{h}_u^{\mathrm{m}}) + \boldsymbol{\varrho}^{\mathrm{m}}$ for each agent u.
  Obtain the joint primitive-observation $\boldsymbol{o}^{\mathrm{p}}$.
  for each time slot $t \in \{1, \ldots, T^{\max}\}$ **do**
    Determine the joint primitive-action $\boldsymbol{a}^{\mathrm{p}} = \boldsymbol{\pi}_{\boldsymbol{\omega}}(\boldsymbol{h}^{\mathrm{p}}) + \boldsymbol{\varrho}^{\mathrm{p}}$.
    Obtain the next macro-observation vector $\boldsymbol{o}'^{\mathrm{m}}_u$ and the cumulative reward R_u^{c} for each agent u.
    for each agent $u \in \mathcal{U}$ **do**
      Store the tuple $(\boldsymbol{o}_u^{\mathrm{m}}, \boldsymbol{a}_u^{\mathrm{m}}, \boldsymbol{o}'^{\mathrm{m}}_u, R_u^{\mathrm{c}})$ in the Mac-CERTs buffer.
      if macro-action $\boldsymbol{a}_u^{\mathrm{m}}$ is completed **then**
        $\boldsymbol{o}_u^{\mathrm{m}} \leftarrow \boldsymbol{o}'^{\mathrm{m}}_u$ and $\boldsymbol{a}_u^{\mathrm{m}} \leftarrow \mu_{\boldsymbol{\omega}_u}(\boldsymbol{h}_u^{\mathrm{m}}) + \boldsymbol{\varrho}^{\mathrm{m}}$.
      end if
    end for
    Obtain the next observation $\boldsymbol{o}'^{\mathrm{p}}$ and the intrinsic reward R^{intr}.
    Store the tuple $(\boldsymbol{o}^{\mathrm{p}}, \boldsymbol{a}^{\mathrm{p}}, \boldsymbol{o}'^{\mathrm{p}}, R^{\text{intr}})$ in set $\mathcal{D}^{\mathrm{p}}$.
    Update $\boldsymbol{\vartheta}_u$ and $\boldsymbol{\omega}_u$ using Algorithm 2 for each agent $u \in \mathcal{U}$.
    Update $\boldsymbol{\vartheta}$ and $\boldsymbol{\omega}$ using Algorithm 3 for the joint agent.
    Update the target network weights for each agent $u \in \mathcal{U}$ as follows: $\boldsymbol{\vartheta}_u^- \leftarrow \varepsilon\boldsymbol{\vartheta}_u + (1-\varepsilon)\boldsymbol{\vartheta}_u^-$ and $\boldsymbol{\omega}_u^- \leftarrow \varepsilon\boldsymbol{\omega}_u + (1-\varepsilon)\boldsymbol{\omega}_u^-$.
    Update the target network weights for the joint agent as follows: $\boldsymbol{\vartheta}^- \leftarrow \varepsilon\boldsymbol{\vartheta} + (1-\varepsilon)\boldsymbol{\vartheta}^-$ and $\boldsymbol{\omega}^- \leftarrow \varepsilon\boldsymbol{\omega} + (1-\varepsilon)\boldsymbol{\omega}^-$.
    $\boldsymbol{o}^{\mathrm{p}} \leftarrow \boldsymbol{o}'^{\mathrm{p}}$.
  end for
end for
Outputs are $\boldsymbol{\omega}_u$ for each agent $u \in \mathcal{U}$, and $\boldsymbol{\omega}$ for the joint agent.

agent's transition experiences are collected at each time slot. However, to train the Mac-IAICC for each agent, we use the agents' transition experiences when their macro-actions are completed. Furthermore, the agents' macro-actions will change the primitive-observation vector for the joint agent, while the joint agent's action affects the extrinsic rewards obtained by the agents. To effectively learn both policies $\boldsymbol{\mu}$ and $\boldsymbol{\pi}$, we propose a hierarchical learning framework which considers the interaction among the agents' policies at different levels of temporal abstraction and trains their actor-critic networks accordingly. Algorithm 4 describes our proposed hierarchical learning framework.

In Algorithm 4, each agent $u \in \mathcal{U}$ and the joint agent randomly explore the environment for $T^{\text{warm-up}}$ time slots and initialize the sets $\mathcal{D}_u$, $\mathcal{D}^{\mathrm{m}}$, and $\mathcal{D}^{\mathrm{p}}$. Then, we train the actor and critic networks of the agents and the joint agent for $E^{\max}$ episodes, each with $T^{\max}$ time slots. To encourage effective exploration and learning, we add an exploration noise $\boldsymbol{\varrho}^{\mathrm{m}}$ to the actions determined by the actor network of each agent. We also add an exploration noise $\boldsymbol{\varrho}^{\mathrm{p}}$ to the actions determined by the actor network of the joint agent. Both $\boldsymbol{\varrho}^{\mathrm{m}}$ and $\boldsymbol{\varrho}^{\mathrm{p}}$ follow the Ornstein-Uhlenbeck process with parameters $(\theta_{\boldsymbol{\varrho}^{\mathrm{m}}}, \sigma_{\boldsymbol{\varrho}^{\mathrm{m}}})$ and $(\theta_{\boldsymbol{\varrho}^{\mathrm{p}}}, \sigma_{\boldsymbol{\varrho}^{\mathrm{p}}})$, respectively. To stabilize the training process, we update the target networks at the end of each time slot using the soft update technique with constant ε, where $0 < \varepsilon < 1$.

D. Computational Complexity

For the actor and critic networks, we consider neural networks comprising one LSTM layer and two FC layers. There is a leaky rectified linear unit (leaky ReLU) activation layer between the LSTM and the first FC layer, as well as between the two FC layers. A hyperbolic tangent (tanh) activation layer is utilized after the second FC layer in the actor networks. Let $d_{\mathrm{h}}^{\mathrm{A}}$ and $d_{\mathrm{h}}^{\mathrm{J}}$ denote the hidden sizes of each agent's LSTM layer and the joint agent's LSTM layer, respectively. The output sizes of the first and second FC layers corresponding to the neural networks of each agent are denoted by $d_{\mathrm{fc}}^{\mathrm{A}}$ and $d_{\mathrm{out}}^{\mathrm{A}}$, respectively. We denote the output sizes of the first and second FC layers corresponding to the neural networks of the joint agent by $d_{\mathrm{fc}}^{\mathrm{J}}$ and $d_{\mathrm{out}}^{\mathrm{J}}$, respectively.

After training, the computational complexity of the online tile bitrate selection using the pre-trained actor network obtained by Algorithms 2 and 4 for each agent is $O(d_{\mathrm{in}}^{\mathrm{A}} d_{\mathrm{h}}^{\mathrm{A}} + (d_{\mathrm{h}}^{\mathrm{A}})^2 + d_{\mathrm{h}}^{\mathrm{A}} d_{\mathrm{fc}}^{\mathrm{A}} + d_{\mathrm{fc}}^{\mathrm{A}} d_{\mathrm{out}}^{\mathrm{A}})$, where $d_{\mathrm{in}}^{\mathrm{A}}$ is the size of the input layer for the agent's actor network. Similarly, after training, the computational complexity of the online design of beamforming vectors using the pre-trained actor network obtained by Algorithms 3 and 4 for the joint agent is $O(d_{\mathrm{in}}^{\mathrm{J}} d_{\mathrm{h}}^{\mathrm{J}} + (d_{\mathrm{h}}^{\mathrm{J}})^2 + d_{\mathrm{h}}^{\mathrm{J}} d_{\mathrm{fc}}^{\mathrm{J}} + d_{\mathrm{fc}}^{\mathrm{J}} d_{\mathrm{out}}^{\mathrm{J}})$, where $d_{\mathrm{in}}^{\mathrm{J}}$ is the size of the input layer for the joint agent's actor network.

VII. Performance Evaluation

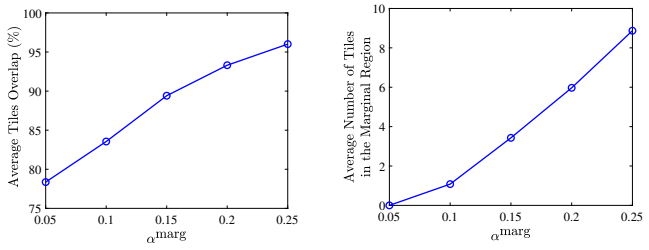
A. Experimental Setup

Wireless Environment: We consider a 10 m $\times$ 10 m $\times$ 4 m indoor environment and three APs as shown in Fig. 2. The APs are located at $\boldsymbol{l}_1 = (9, 1, 4)$, $\boldsymbol{l}_2 = (5, 5, 4)$, and $\boldsymbol{l}_3 = (1, 9, 4)$. We consider the APs to be operating at a carrier frequency of $f_{\mathrm{c}} = 1.05$ THz. The molecular absorption coefficient is determined as $\kappa(f_{\mathrm{c}}) = 0.07512$ m^{-1}. Unless stated otherwise, we consider $U = 6$ users, each with a height of $h_u = 1.6$ m. The floor of the indoor environment is divided into U equal areas. Each user is positioned at the center of an area. We set the self-blockage angle of the users to $\phi^{\text{blocked}} = \pi$. We set the number of antenna elements for each AP N_{t} and each user's HMD N_{r} to be 6 and 2, respectively.

Dataset: We conduct our experiments using a public 360° video dataset [14]. The dataset consists of 104 videos including five sports events: basketball, parkour, BMX, skateboarding, and dance. There are 27 viewers in this dataset. Each video has been watched by at least 18 viewers. For each viewer, the eye gaze points are recorded when watching the videos.

Simulation Setting: A frame rate of 30 frames per second and a chunk duration of one second are considered for the videos. Each video frame is divided into 24 tiles using 6×4 tiling pattern as shown in Fig. 4. The time slot duration T^{slot} is set to 100 ms. The FoV of each user's HMD is set to $90° \times 135°$. We consider five quality levels. We select the bitrate value for encoding the tiles from set $\{28, 33, 38, 43, 48\}$ Mbps.

Algorithms Parameters Setting: For the head movement prediction model, we set $Q^{\text{hist}} = 90$ and $d^{\text{GRU}} = 64$. In Algorithm 1, we set $R = 50$, $\rho = 3$, and $\eta^{\text{head}} = 0.01$. For each user, we consider 80 and 24 videos in the training dataset $\mathcal{V}_u^{\text{head-tr}}$ and test dataset $\mathcal{V}_u^{\text{head-tst}}$, respectively. Since all the video frames are available at the server and our proposed viewport prediction framework decouples the saliency detection and head movement prediction models, the saliency



(a) (b)

Fig. 9: (a) Average tiles overlap and (b) average number of tiles in the marginal region versus the scale factor α^{marg}.

TABLE I: Simulation Parameters

Parameter	Value	Parameter	Value	Parameter	Value	Parameter	Value
η^{a}, η^{c}	10^{-4}	$\sigma_{\boldsymbol{\theta}^{\text{m}}}$, $\sigma_{\boldsymbol{\theta}^{\text{p}}}$	0.15	$E^{\max}$	5000	$P^{\max}$	5 dBm
ε	10^{-2}	λ^{spatial}, λ^{temp}	0.5	$T^{\max}$	153	g_a	25 dBi
B^{m}, B^{p}	512	λ^{RD}	0.5	$T^{\text{warm-up}}$	5200	g_u	15 dBi
$\theta_{\boldsymbol{\theta}^{\text{m}}}$, $\theta_{\boldsymbol{\theta}^{\text{p}}}$	0.1	λ^{intr}	2	γ	0.99	σ^2	-77 dBm

map corresponding to each video frame can be obtained and stored at the server in advance. In particular, each user sends its predicted head orientations to the server when requesting a new video chunk. Upon receiving a new video chunk request, the server can retrieve the saliency maps of the video frames corresponding to the requested video chunk, as well as the user's head orientation maps corresponding to those video frames. Given the saliency maps and the user's head orientation maps, the tiles covering the viewport and marginal regions of the requested video frames can be determined using the integration mechanism described in Section V-C, which has a computational complexity of $O(N)$. We empirically determine the value of α^{marg} by comparing the average tiles overlap and the average number of tiles in the marginal region for various values of α^{marg}. The tiles overlap is obtained as $\left|\mathcal{N}^{\text{pred}}_{u,c,v} \cap \mathcal{N}^{\text{actual}}_{u,c,v}\right| / \left|\mathcal{N}^{\text{actual}}_{u,c,v}\right|$ for tiles that are predicted to be transmitted for user $u \in \mathcal{U}$ upon its request for chunk $c \in \mathcal{C}_v$ of video $v \in \mathcal{V}^{\text{head-tst}}_u$. Fig. 9 shows that increasing α^{marg} leads to a higher average tiles overlap and a higher average number of tiles in the marginal region. To achieve an average tiles overlap above 90% without consuming excessive network bandwidth on the transmission of additional tiles, we set $\alpha^{\text{marg}} = 0.15$. For the agents, we set $d^{\text{A}}_{\text{h}} = 512$ and $d^{\text{A}}_{\text{fc}} = d^{\text{A}}_{\text{out}} = 256$. For the joint agent, we set $d^{\text{J}}_{\text{h}} = 1024$ and $d^{\text{J}}_{\text{fc}} = d^{\text{J}}_{\text{out}} = 512$. Other simulation parameters are summarized in Table I.

Benchmarks: We compare the performance of our proposed algorithms with the following algorithms as benchmarks:

- **Video streaming with viewport prediction using FedAvg**: In this algorithm, instead of using our proposed viewport prediction framework, we use FedAvg to train the head movement prediction model as proposed in [15].
- **Combined FoV tile-based adaptive streaming algorithm [8]**: In this algorithm, the viewport is predicted by combining the current chunk's viewport and another viewport obtained by spherical walk. A priority-based bitrate adaptation algorithm is used to select the bitrate of tiles. Each user's data rate is predicted by dividing the number of bits in the previous video chunk by its transmission delay. Since users' head movements are not predicted, we consider that the APs in the self-blockage region of users are determined only after detecting beam failure, a process that takes 300 ms.
- **Video streaming with WMMSE beamforming algorithm**: In this algorithm, the joint agent employs a WMMSE algorithm [25], [33] to obtain the beamforming vectors instead of utilizing our actor-critic algorithm.

B. Experimental Results

In Fig. 10, we study the impact of increasing the maximum buffer size threshold on the 360° video streaming system performance. The users can prefetch more video chunks when the buffer size threshold increases. The results in Fig. 10(a) illustrate that as the maximum buffer size threshold increases, the average viewport tile quality decreases for the combined FoV tile-based adaptive streaming algorithm and video streaming with viewport prediction using FedAvg. This is because the increase in the maximum buffer size threshold leads to a degradation in viewport prediction performance for these algorithms. Moreover, such degradation in viewport prediction performance results in lower average intra- and inter-chunk quality switch as shown in Figs. 10(b) and 10(c), respectively. However, by transmitting more video tiles based on the value of α^{marg} and by using the Mac-IAICC approach for tile bitrate selection, our proposed video streaming approach can achieve a higher average viewport tile quality and a lower average intra- and inter-chunk quality switch. By increasing the maximum buffer size threshold, users have more time to prefetch their requested chunks without experiencing video stalling. Thus, as shown in Fig. 10(d), the average rebuffering delay decreases for all the algorithms. In video streaming with viewport prediction using FedAvg, only the predicted tiles for viewport are sent to the users. Thus, this algorithm can achieve a lower average rebuffering delay and a higher average sum-rate as shown in Figs. 10(d) and 10(e), respectively. The combined FoV tile-based adaptive streaming algorithm uses a reactive THz beam failure detection. Thus, it has the highest average rebuffering delay and the lowest average sum-rate. Moreover, the results in Figs. 10(d) and 10(e) illustrate that the Prim-CAC approach employed by the joint agent in our proposed video streaming approach effectively reduces the average rebuffering delay and increases the average sum-rate compared with using the WMMSE beamforming algorithm. Overall, as shown in Fig. 10(f), our proposed approach provides a higher average QoE compared with the considered benchmarks when the maximum buffer size threshold increases.

In Fig. 11, we investigate the impact of increasing the number of users on average QoE. With more users, designing the beamforming vectors becomes more challenging due to the limited bandwidth of the APs and the self-blockage region of the users. The results in Fig. 11 illustrate that when the number of users is equal to 12, our proposed video streaming approach can achieve an average QoE which is 23.77%, 62.7%, and 116.57% higher than that of the video streaming with WMMSE beamforming algorithm, combined FoV tile-based adaptive streaming algorithm, and video streaming with viewport prediction using FedAvg, respectively.

In Fig. 12, we show the impact of the number and location of APs on the performance of the considered THz-enabled 360° video streaming system. Due to self-blockage, the system

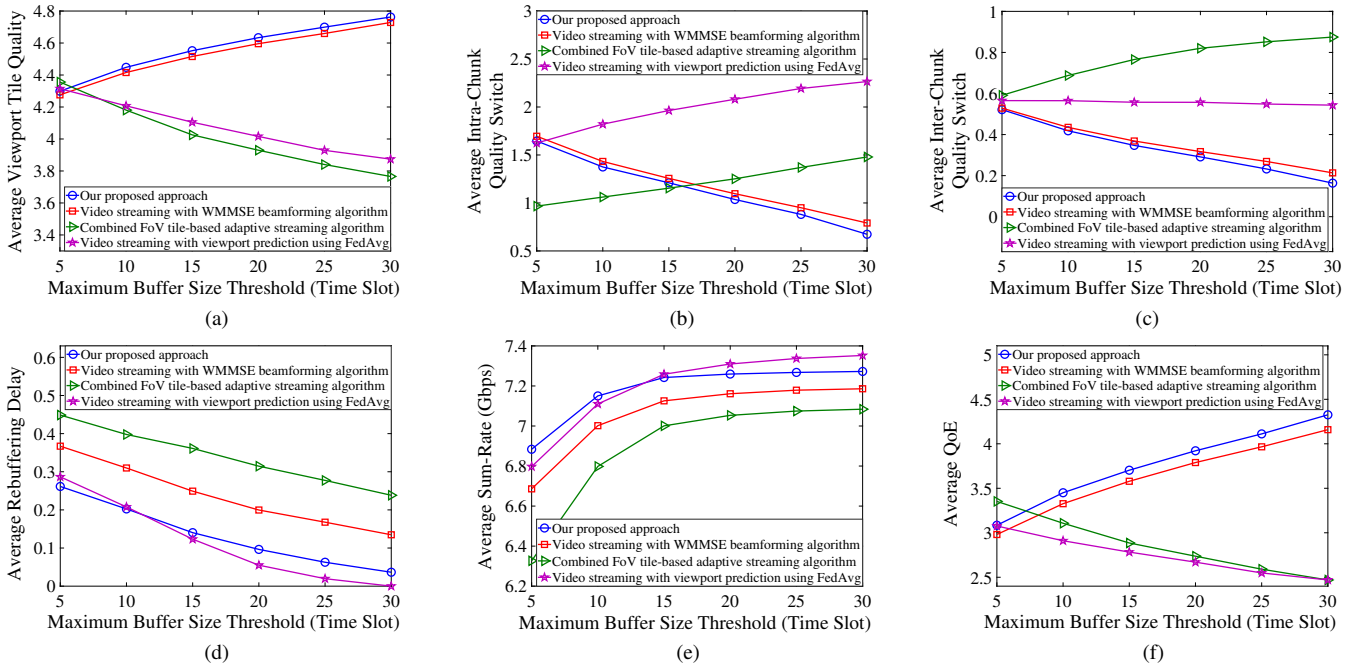


Fig. 10: (a) Average viewport tile quality, (b) average intra-chunk quality switch, (c) average inter-chunk quality switch, (d) average rebuffering delay, (e) average sum-rate, and (f) average QoE over users versus the maximum buffer size threshold. We set $B = 0.5$ GHz.

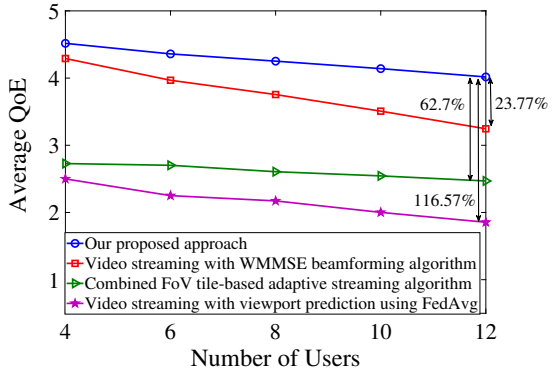


Fig. 11: Average QoE over users when the number of users is increased. To support more users, we set $B = 5$ GHz and $\max_{u\in\mathcal{U}} B_u^{\text{THR}} = 30$ time slots.

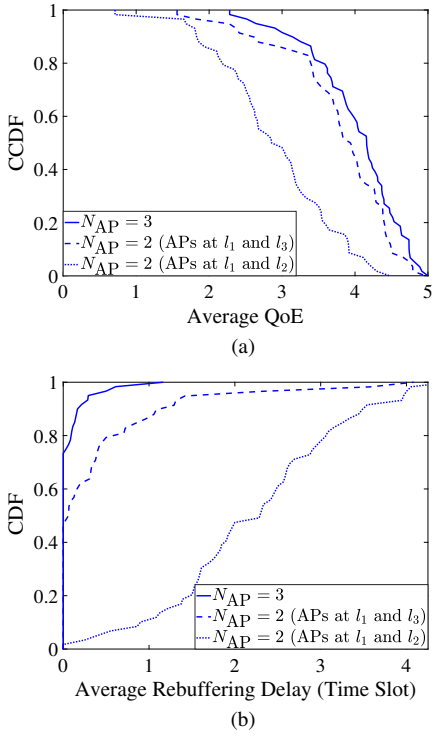


Fig. 12: (a) CCDF of the average QoE and (b) CDF of the average rebuffering delay. We set $B = 0.5$ GHz and $\max_{u\in\mathcal{U}} B_u^{\text{THR}} = 30$ time slots.

requires more than one AP. Also, as shown in Fig. 2, there is a symmetry in the APs' locations. Thus, we consider the following three scenarios: all three APs are available; only two APs located at $\boldsymbol{l}_1 = (9, 1, 4)$ and $\boldsymbol{l}_3 = (1, 9, 4)$ are available; and only two APs located at $\boldsymbol{l}_1 = (9, 1, 4)$ and $\boldsymbol{l}_2 = (5, 5, 4)$ are available to transmit the video tiles to the users. Fig. 12(a) illustrates the complementary cumulative distribution function (CCDF) of the average QoE. A higher CCDF indicates a higher probability of obtaining an average QoE above a given threshold value. Fig. 12(b) presents the cumulative distribution function (CDF) of the average rebuffering delay for each of the considered scenarios. A higher CDF indicates a higher probability of obtaining an average rebuffering delay below a given threshold value. The results in Fig. 12 show that with three APs, we can consistently achieve a higher average QoE (> 2.28) and a lower average rebuffering delay (< 1.67 time slots) with a higher probability compared to scenarios with only two APs. Furthermore, Fig. 12 illustrates that the APs' locations have a significant impact on the system performance when only two APs are available for video tile transmission to the users. In particular, locating two APs at $\boldsymbol{l}_1$ and $\boldsymbol{l}_3$ can better cope with self-blockage. For example, as shown in Fig. 12(b), the probability of having an average rebuffering delay below 2 time slots is approximately 0.96 when APs are located at $\boldsymbol{l}_1$ and $\boldsymbol{l}_3$, while this probability is close to 0.47 when APs

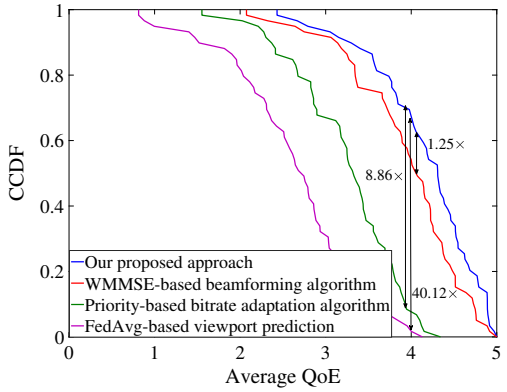


Fig. 13: CCDF of the average QoE across users. We set $B = 0.5$ GHz and $\max_{u\in\mathcal{U}} B_u^{\text{THR}} = 30$ time slots.

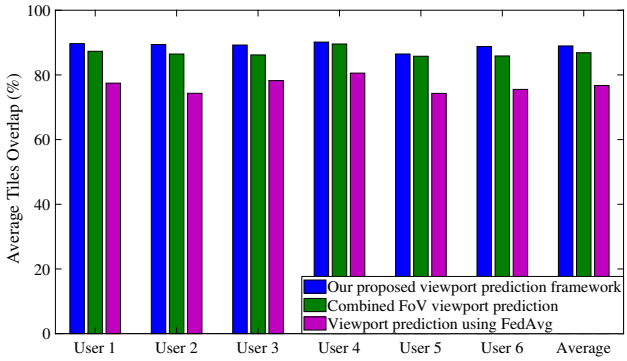


Fig. 14: Average tiles overlap of our proposed viewport prediction framework for different users and the average tiles overlap obtained by averaging across users compared to the considered benchmarks.

are located at $\boldsymbol{l}_1$ and $\boldsymbol{l}_2$. Thus, a more reliable transmission of 360° videos over THz links is provided if the two APs are located at $\boldsymbol{l}_1$ and $\boldsymbol{l}_3$.

As shown in Fig. 1, our proposed 360° video streaming approach consists of three algorithms: a viewport prediction framework, a multi-agent DDPG algorithm for bitrate selection, and a multi-agent DDPG algorithm for beamforming design. To demonstrate the performance gains contributed by each algorithm in our approach, we assess their impact by fixing two of our proposed algorithms and replacing the third one with an existing algorithm from the literature. Fig. 13 illustrates the CCDF of the average QoE for our proposed approach compared with three benchmarks. In the first benchmark, we replace the DDPG-based beamforming design algorithm with a WMMSE-based beamforming algorithm. In the second benchmark, we replace the DDPG-based bitrate selection algorithm with a priority-based bitrate adaptation algorithm. In the third benchmark, our proposed viewport prediction framework is replaced by a FedAvg-based viewport prediction algorithm. As shown in Fig. 13, the probability of achieving an average QoE above 4 with our proposed approach is 1.25 times higher than with the WMMSE-based beamforming algorithm, 8.86 times higher than with the priority-based bitrate adaptation algorithm, and 40.12 times higher than with the FedAvg-based viewport prediction. Thus, each of the algorithms comprising our proposed approach definitely has its own merits on providing the users with an immersive viewing experience.

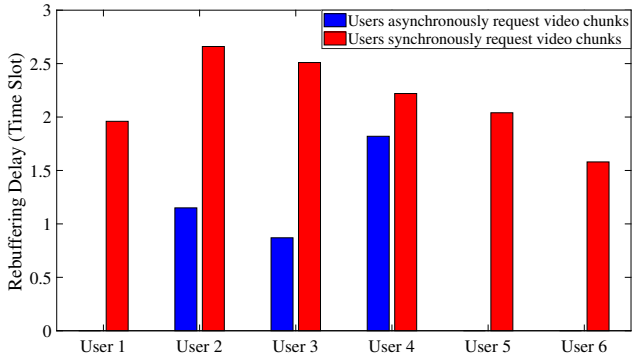


Fig. 15: Rebuffering delay of each user. We set $B = 0.5$ GHz.

In Fig. 14, we compare the performance of our proposed viewport prediction framework with the algorithms proposed in [8] and [15]. The results in Fig. 14 show that our proposed content-based viewport prediction framework can achieve an average tiles overlap which is on average 2.41% and 15.93% higher than that of the viewport prediction algorithms proposed in [8] and [15], respectively.

In Fig. 15, we show the rebuffering delay of each user for two scenarios. The first scenario is when users request their next video chunks asynchronously. The second scenario is when they request them synchronously. The results in Fig. 15 illustrate that asynchronous video chunk requests can provide a lower rebuffering delay for each user (even zero rebuffering delay for users 1, 5, and 6) compared with synchronous video chunk requests. This is due to the fact that in synchronous video chunk request: (a) each user may need to wait an extra amount of time during which the user watches the prefetched video tiles in its buffer, leading to a lower value of $B_u\left(\tau_{u,c,v}^{\text{REQ}}\right)$; and (b) all users request their next video chunk simultaneously and compete for the limited wireless resources, leading to a higher transmission delay $\tau_{u,c,v}^{\text{TD}}$.

VIII. Conclusion

In this paper, we proposed a content-based viewport prediction framework for 360° videos. To address users' privacy concerns and the data heterogeneity issue, our proposed framework employs a PFL algorithm for training the users' head movement prediction models. We applied the proposed viewport prediction framework in a THz-enabled 360° video streaming system to determine which video tiles should be transmitted to the users. We modeled the bitrate selection for video tiles and the design of beamforming vectors for the APs, as a MacDec-POMDP. To solve this problem, we proposed a hierarchical DRL framework consisting of two multi-agent DDPG algorithms. We determined a policy for tile bitrate selection using the Mac-IAICC approach and a policy for beamforming design using the Prim-CAC approach. The results on a public 360° video dataset showed that our proposed video streaming approach provides a higher QoE for the users compared with three benchmark algorithms. One direction for future work is to employ our proposed video streaming approach in Metaverse systems, where users are mobile and uplink transmission is used to send their motion tracking data.

REFERENCES

[1] M. Setayesh and V. W.S. Wong, "A content-based viewport prediction framework for 360° video using personalized federated learning and fusion techniques," in *Proc. of Int'l Conf. on Multimedia and Expo (ICME)*, Brisbane, Australia, Jul. 2023.

[2] ——, "Asynchronous DRL-based bitrate selection for 360-degree video streaming over THz wireless systems," in *Proc. of IEEE Int'l Conf. Commun. (ICC)*, Denver, CO, Jun. 2024.

[3] R. Zhang, J. Liu, F. Liu, T. Huang, Q. Tang, S. Wang, and F. R. Yu, "Buffer-aware virtual reality video streaming with personalized and private viewport prediction," *IEEE J. Sel. Areas in Commun.*, vol. 40, no. 2, pp. 694–709, Feb. 2022.

[4] C. Chaccour, M. N. Soorki, W. Saad, M. Bennis, and P. Popovski, "Can terahertz provide high-rate reliable low-latency communications for wireless VR?" *IEEE Internet Things J.*, vol. 9, no. 12, pp. 9712–9729, Jun. 2022.

[5] A. Shafie, N. Yang, S. Durrani, X. Zhou, C. Han, and M. Juntti, "Coverage analysis for 3D terahertz communication systems," *IEEE J. Sel. Areas Commun.*, vol. 39, no. 6, pp. 1817–1832, Jun. 2021.

[6] A. Shafie, N. Yang, S. A. Alvi, C. Han, S. Durrani, and J. M. Jornet, "Spectrum allocation with adaptive sub-band bandwidth for terahertz communication systems," *IEEE Trans. Commun.*, vol. 70, no. 2, pp. 1407–1422, Feb. 2022.

[7] N. Kan, J. Zou, C. Li, W. Dai, and H. Xiong, "RAPT360: Reinforcement learning-based rate adaptation for 360-degree video streaming with adaptive prediction and tiling," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 32, no. 3, pp. 1607–1623, Mar. 2022.

[8] A. Yaqoob and G.-M. Muntean, "A combined field-of-view prediction-assisted viewport adaptive delivery scheme for 360° videos," *IEEE Trans. Broadcast.*, vol. 67, no. 3, pp. 746–760, Sept. 2021.

[9] J. Li, L. Han, C. Zhang, Q. Li, and Z. Liu, "Spherical convolution empowered viewport prediction in 360 video multicast with limited FoV feedback," *ACM Trans. on Multimedia Comput. Commun. Appl.*, vol. 19, no. 1, pp. 1–23, Jan. 2023.

[10] F. Qian, B. Han, Q. Xiao, and V. Gopalakrishnan, "Flare: Practical viewport-adaptive 360-degree video streaming for mobile devices," in *Proc. of ACM Int'l Conf. Mobile Comput. Netw. (MobiCom)*, New Delhi, India, Oct. 2018.

[11] P. Maniotis, E. Bourtsoulatze, and N. Thomos, "Tile-based joint caching and delivery of 360 videos in heterogeneous networks," *IEEE Trans. Multimedia*, vol. 22, no. 9, pp. 2382–2395, Sept. 2020.

[12] Y. Xiao, W. Tan, and C. Amato, "Asynchronous actor-critic for multi-agent reinforcement learning," in *Proc. of Conf. Neural Inf. Process. Syst. (NIPS)*, New Orleans, LA, Nov. 2022.

[13] X. Lyu, A. Banitalebi-Dehkordi, M. Chen, and Y. Zhang, "Asynchronous, option-based multi-agent policy gradient: A conditional reasoning approach," in *Proc. of IEEE/RSJ Int'l Conf. on Intell. Robots Syst. (IROS)*, Detroit, MI, Oct. 2023.

[14] Z. Zhang, Y. Xu, J. Yu, and S. Gao, "Saliency detection in 360° videos," in *Proc. of Eur. Conf. on Comput. Vis.*, Munich, Germany, Sept. 2018.

[15] X. Liu, Y. Deng, C. Han, and M. Di Renzo, "Learning-based prediction, rendering and transmission for interactive virtual reality in RIS-assisted terahertz networks," *IEEE J. Sel. Areas in Commun.*, vol. 40, no. 2, pp. 710–724, Feb. 2022.

[16] F.-Y. Chao, C. Ozcinar, and A. Smolic, "Transformer-based long-term viewport prediction in 360° video: Scanpath is all you need." in *Proc. of Int'l Workshop Multimedia Signal Process. (MMSP)*, Tampere, Finland, Oct. 2021.

[17] X. Liu, X. Li, and Y. Deng, "Learning-based prediction and proactive uplink retransmission for wireless virtual reality network," *IEEE Trans. Veh. Technol.*, vol. 70, no. 10, pp. 10 723–10 734, Oct. 2021.

[18] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proc. of Int'l Conf. Artificial Intelligence and Statistics (AISTATS)*, Ft. Lauderdale, FL, Apr. 2017.

[19] J. Oh, S. Kim, and S.-Y. Yun, "FedBABU: Towards enhanced representation for federated image classification," in *Proc. of Int'l Conf. on Learning Representations (ICLR)*, Apr. 2022.

[20] A. Nguyen, Z. Yan, and K. Nahrstedt, "Your attention is unique: Detecting 360-degree video saliency in head-mounted display for head movement prediction," in *Proc. of ACM Int'l Conf. on Multimedia*, Seoul, Republic of Korea, Oct. 2018.

[21] C. Wu, R. Zhang, Z. Wang, and L. Sun, "A spherical convolution approach for learning long term viewport prediction in 360 immersive video," in *Proc. of AAAI Conf. on Artif. Intell.*, New York, NY, Feb. 2020.

[22] F. Hu, Y. Deng, W. Saad, M. Bennis, and A. H. Aghvami, "Cellular-connected wireless virtual reality: Requirements, challenges, and solutions," *IEEE Commun. Mag.*, vol. 58, no. 5, pp. 105–111, May 2020.

[23] L. Zhao, Y. Cui, S. Yang, and S. S. Shitz, "An optimization framework for general rate splitting for general multicast," *IEEE Trans. Wireless Commun.*, vol. 22, no. 3, pp. 1573–1587, Mar. 2022.

[24] P. Yang, T. Q. Quek, J. Chen, C. You, and X. Cao, "Feeling of presence maximization: mmWave-enabled virtual reality meets deep reinforcement learning," *IEEE Trans. Wireless Commun.*, vol. 21, no. 11, pp. 10 005–10 019, Nov. 2022.

[25] R. Huang, V. W.S. Wong, and R. Schober, "Rate-splitting for intelligent reflecting surface-aided multiuser VR streaming," *IEEE J. Sel. Areas Commun.*, vol. 41, no. 5, pp. 1516–1535, May 2023.

[26] C. Perfecto, M. S. Elbamby, J. Del Ser, and M. Bennis, "Taming the latency in multi-user VR 360°: A QoE-aware deep learning-aided multicast framework," *IEEE Trans. Commun.*, vol. 68, no. 4, pp. 2491–2508, Apr. 2020.

[27] J. Zhang, H. Wu, X. Tao, and X. Zhang, "Adaptive bitrate video streaming in non-orthogonal multiple access networks," *IEEE Trans. Veh. Technol.*, vol. 69, no. 4, pp. 3980–3993, Apr. 2020.

[28] Y. Zhang, P. Zhao, K. Bian, Y. Liu, L. Song, and X. Li, "DRL360: 360-degree video streaming with deep reinforcement learning," in *Proc. of IEEE Conf. Comput. Commun. (INFOCOM)*, Paris, France, Apr. 2019.

[29] G. Xiao, M. Wu, Q. Shi, Z. Zhou, and X. Chen, "DeepVR: Deep reinforcement learning for predictive panoramic video streaming," *IEEE Trans. Cogn. Commun. Netw.*, vol. 5, no. 4, pp. 1167–1177, Dec. 2019.

[30] J. M. Jornet and I. F. Akyildiz, "Channel modeling and capacity analysis for electromagnetic wireless nanonetworks in the terahertz band," *IEEE Trans. Wireless Commun.*, vol. 10, no. 10, pp. 3211–3221, Oct. 2011.

[31] A. Mehrabian and V. W.S. Wong, "Joint spectrum, precoding, and phase shifts design for RIS-aided multiuser MIMO THz systems," *IEEE Trans. Commun.*, vol. 72, no. 8, pp. 5087–5101, Aug. 2024.

[32] M. Hu, X. Luo, J. Chen, Y. C. Lee, Y. Zhou, and D. Wu, "Virtual reality: A survey of enabling technologies and its applications in IoT," *J. Netw. Comput. Appl.*, vol. 178, p. 102970, Mar. 2021.

[33] Q. Shi, M. Razaviyayn, Z.-Q. Luo, and C. He, "An iteratively weighted MMSE approach to distributed sum-utility maximization for a MIMO interfering broadcast channel," *IEEE Trans. Signal Process.*, vol. 59, no. 9, pp. 4331–4340, Sept. 2011.

[34] Z. Zhang, M. Zeng, M. Chen, D. Liu, W. Saad, S. Cui, and H. V. Poor, "Joint user grouping, version selection, and bandwidth allocation for live video multicasting," *IEEE Trans. Commun.*, vol. 70, no. 1, pp. 350–365, Jan. 2022.

[35] K. Long, Y. Cui, C. Ye, and Z. Liu, "Optimal wireless streaming of multi-quality 360 VR video by exploiting natural, relative smoothness-enabled, and transcoding-enabled multicast opportunities," *IEEE Trans. Multimedia*, vol. 23, pp. 3670–3683, Oct. 2021.

[36] B. Han, Z. Ren, Z. Wu, Y. Zhou, and J. Peng, "Off-policy reinforcement learning with delayed rewards," in *Proc. of Int'l Conf. on Machine Learning (ICML)*, Baltimore, MD, Jul. 2022.

[37] C. Yu, X. Yang, J. Gao, J. Chen, Y. Li, J. Liu, Y. Xiang, R. Huang, H. Yang, Y. Wu, and Y. Wang, "Asynchronous multi-agent reinforcement learning for efficient real-time multi-robot cooperative exploration," in *Proc. of Int'l Conf. on Auton. Agents and Multiagent Syst. (AAMAS)*, London, United Kingdom, May 2023.

[38] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," in *Proc. of Int'l Conf. on Learning Representations (ICLR)*, San Juan, Puerto Rico, May 2016.

[39] T. D. Kulkarni, K. Narasimhan, A. Saeedi, and J. Tenenbaum, "Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation," in *Proc. of Conf. Neural Inf. Process. Syst. (NIPS)*, Barcelona, Spain, Dec. 2016.

[40] H. Yun, S. Lee, and G. Kim, "Panoramic vision transformer for saliency detection in 360° videos," in *Proc. of Eur. Conf. on Comput. Vis. (ECCV)*, Tel Aviv, Israel, Oct. 2022.

[41] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. of Int'l Conf. Learning Representations (ICLR)*, San Diego, CA, May 2015.

[42] M. Setayesh, X. Li, and V. W.S. Wong, "PerFedMask: Personalized federated learning with optimized masking vectors," in *Proc. of Int'l Conf. on Learning Representations (ICLR)*, Kigali, Rwanda, May 2023.

[43] L. Teng, G. Zhai, Y. Wu, X. Min, W. Zhang, Z. Ding, and C. Xiao, "QoE driven VR 360° video massive MIMO transmission," *IEEE Trans. Wireless Commun.*, vol. 21, no. 1, pp. 18–33, Jan. 2022.

[44] R. Lowe, Y. Wu, A. Tamar, J. Harb, P. Abbeel, and I. Mordatch, "Multi-agent actor-critic for mixed cooperative-competitive environments," in

Proc. of Conf. Neural Inf. Process. Syst. (NIPS), Long Beach, CA, Dec. 2017.

Mehdi Setayesh (S'20) received the B.Sc. and M.Sc. degrees from the Sharif University of Technology, Tehran, Iran, in 2014 and 2016, respectively, and the Ph.D. degree from the University of British Columbia (UBC), Vancouver, Canada, in 2024. From 2020 to 2023, he was a Mitacs Accelerate intern with Rogers Communications Canada Inc. He is currently a Senior Researcher with Huawei Noah's Ark Lab Canada. His research interests include Network for AI (Net4AI), AI for Network (AI4Net), machine learning, algorithm design, and optimization in wireless communication systems. From 2020 to 2023, he received the Graduate Support Initiative (GSI) Award from the Faculty of Applied Science at UBC. He received the Best Paper Award at the *IEEE GLOBECOM* 2020.

Vincent W.S. Wong (S'94, M'00, SM'07, F'16) received the B.Sc. degree from the University of Manitoba, Canada, in 1994, the M.A.Sc. degree from the University of Waterloo, Canada, in 1996, and the Ph.D. degree from the University of British Columbia (UBC), Vancouver, Canada, in 2000. From 2000 to 2001, he worked as a systems engineer at PMC-Sierra Inc. (now Microchip Technology Inc.). He joined the Department of Electrical and Computer Engineering at UBC in 2002 and is currently a Professor. His research areas include protocol design, optimization, and resource management of communication networks, with applications to 5G/6G wireless networks, Internet of things, mobile edge computing, smart grid, and energy systems. Dr. Wong is the Editor-in-Chief of *IEEE Transactions on Wireless Communications*. He has served as an Area Editor of *IEEE Transactions on Communications* and *IEEE Open Journal of the Communications Society*, an Associate Editor of *IEEE Transactions on Mobile Computing* and *IEEE Transactions on Vehicular Technology*, and a Guest Editor of *IEEE Journal on Selected Areas in Communications*, *IEEE Internet of Things Journal*, and *IEEE Wireless Communications*. Dr. Wong was the General Co-Chair of *IEEE INFOCOM* 2024; Tutorial Co-Chair of *IEEE GLOBECOM* 2018; Technical Program Co-Chair of *IEEE VTC*2020-*Fall* and *IEEE SmartGridComm* 2014; and Symposium Co-Chair of *IEEE ICC*'18, *IEEE SmartGridComm* ('13, '17) and *IEEE GLOBECOM*'13. He received the 2022 Best Paper Award from *IEEE Transactions on Mobile Computing* and Best Paper Awards at the *IEEE ICC* 2022 and *IEEE GLOBECOM* 2020. He has served as the Chair of the IEEE Vancouver Joint Communications Chapter and IEEE Communications Society Emerging Technical Sub-Committee on Smart Grid Communications. He was an IEEE Communications Society Distinguished Lecturer from 2019 to 2020 and is an IEEE Vehicular Technology Society Distinguished Lecturer for the term of 2023−2025. Dr. Wong is a Fellow of the IEEE and the Engineering Institute of Canada (EIC).