Asynchronous DRL-based Bitrate Selection for 360-Degree Video Streaming over THz Wireless Systems

Mehdi Setayesh and Vincent W.S. Wong
Department of Electrical and Computer Engineering, The University of British Columbia, Vancouver, Canada
email: {setayeshm, vincentw}@ece.ubc.ca

***Abstract*—360° videos demand substantial bandwidth to deliver an immersive viewing experience to users. In wireless networks, this high data rate demand can be accommodated by utilizing the terahertz (THz) frequency band. However, THz band communications are susceptible to self-blockage. To ensure reliable transmission, this paper studies the streaming of 360° videos over THz wireless systems using multiple multi-antenna access points (APs). Users' requests for video tiles give rise to an optimization problem that involves asynchronous bitrate selection for those tiles and beamforming design for the APs. We formulate this problem as a macro-action decentralized partially observable Markov decision process (MacDec-POMDP). To efficiently tackle this problem for multiple users, we propose an asynchronous deep reinforcement learning (DRL) algorithm using a multi-agent actor-critic method to determine the bitrate selection policy. The APs' beamforming is determined by solving an optimization problem using the weighted minimum mean square error (WMMSE) algorithm. Results show that our proposed approach provides a higher average quality of experience (QoE) for the users when compared with two benchmark algorithms.**

I. Introduction

In 360° video streaming, users delve into an immersive visual experience using a head-mounted display (HMD). Compared to conventional video streaming, 360° video requires significantly higher bandwidth because it provides users with a high-resolution 360° visual field across three degrees of freedom [1]. The abundant bandwidth available in the terahertz (THz) frequency band can overcome this challenge [2].

Wireless systems operating in THz frequency band encounter a number of challenges, such as channel impairment due to molecular absorption and susceptibility to blockage by obstacles. Moreover, when a user with its HMD turns around to view another part of a 360° video, the THz link may be blocked by the user's own body, which is known as *self-blockage* [2]. 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 [3]. 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 time, each user watches a 360° video only from one direction. The region of the video that a user is watching at any given time is called a *viewport*. To efficiently utilize the network bandwidth, it is desirable to deliver to each user only its viewport with the maximum possible quality [4].

In practical 360° video streaming systems, the video is split 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 [5]. To prevent video stalling during playback, a prefetching scheme is employed to decide when and how the tiles for the upcoming video chunks should be sent to each user [1]. In such systems, 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 viewport of a user can reduce bandwidth consumption and enable a more flexible transmission mechanism through bitrate selection for the tiles [6].

Recently, streaming of 360° videos over wireless networks has received considerable attention. Yaqoob *et al.* in [4] proposed a combined field-of-view (FoV) prediction-assisted 360° video streaming algorithm and a priority-based bitrate adaptation algorithm. Huang *et al.* in [6] proposed a deep reinforcement learning (DRL) algorithm to maximize the achievable bitrate in an intelligent reflecting surface (IRS)-aided rate-splitting virtual reality (VR) streaming system. Zhao *et al.* in [7] proposed iterative algorithms to determine the beamforming vectors for maximizing the weighted sum average rate in a multicast VR streaming system. Yang *et al.* in [8] proposed a DRL-based algorithm to improve the users' visual experience in a millimeter wave (mmWave)-enabled VR streaming system. The aforementioned works consider either synchronized chunk requests or single-user video streaming. Thus, these works have not taken into account asynchronous requests from users for new video chunks based on their buffer status. Furthermore, self-blockage is a challenge for 360° video streaming in THz wireless networks. The algorithms proposed in [4], [6], [7] do not consider self-blockage.

In this paper, we study 360° video streaming in a THz wireless system with multiple multi-antenna APs. We propose a 360° video streaming approach to maximize the users' quality of experience (QoE). Our approach involves two algorithms: one for optimizing the bitrate selection for video tiles and another for optimizing the beamforming vectors at the APs. The main contributions of this paper are as follows:

- We formulate the bitrate selection for the video tiles as a macro-action decentralized partially observable Markov decision process (MacDec-POMDP) [9] due to the asynchronous decision-making inherent in this problem.
- We propose an asynchronous multi-agent deep deterministic policy gradient (DDPG) algorithm using a macro-action-based independent actor with individual central-

ized critic (Mac-IAICC) approach. This algorithm can effectively obtain the policy for bitrate selection.
- The APs require beamforming design in each time slot, resulting in a non-convex optimization problem. To tackle this issue, we develop a weighted minimum mean square error (WMMSE) algorithm for beamforming design.
- The results on a public 360° video dataset [10] show that our proposed approach outperforms two benchmark algorithms, including the combined FoV tile-based adaptive streaming (CFOV) algorithm [4] and the alternating optimization (AO) algorithm, in terms of the average QoE.

This paper is organized as follows. The system model is presented in Section II. Section III introduces the MacDec-POMDP problem formulation. In Section IV, we present our proposed DRL and WMMSE algorithms. Simulation results are presented in Section V. Conclusion is given in Section VI.

Notations: We represent vectors and matrices by boldface lowercase and uppercase letters, respectively. $|\cdot|$ denotes the cardinality of a set. $(\cdot)^H$ denotes conjugate transpose operator. $\boldsymbol{I}_N$ denotes an identity matrix of size N. $\|\cdot\|$ denotes the norm of a vector as well as the norm of a complex number. $\mathbb{1}(z \in \mathcal{Z})$ denotes the indicator function. We define $[z]^+ = \max\{0,\, z\}$.

II. System Model

Consider U users who are watching 360° videos in an indoor environment, using THz wireless links as shown in Fig. 1. 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. Let $\boldsymbol{l}_u^{\text{3D}} = (x_u,\, y_u,\, h_u)$ 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, respectively. 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 $\boldsymbol{l}_a^{\text{3D}} = \left(x_a,\, y_a,\, h^{\text{AP}}\right)$, 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. 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. 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}.

Let $\gamma_{u,a}$ denote the LoS path gain between AP $a \in \mathcal{A}$ and user $u \in \mathcal{U}$. Considering c_0 as the speed of light, we have $\gamma_{u,a} = \frac{c_0}{4\pi f_{\text{c}} \|\boldsymbol{l}_a^{\text{3D}} - \boldsymbol{l}_u^{\text{3D}}\|} e^{-\frac{1}{2}\kappa(f_{\text{c}})\|\boldsymbol{l}_a^{\text{3D}} - \boldsymbol{l}_u^{\text{3D}}\|}$, where $\kappa(f_{\text{c}})$ (in m^{-1}) denotes the molecular absorption coefficient at frequency f_{c} [3]. 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 from AP a to user u, respectively. The i-th element of vector $\mathbf{a}_a(\psi_{u,a}^{\text{AoD}})$ is equal to $e^{j\frac{2\pi d}{\lambda_{\text{c}}}(i-1)\sin(\psi_{u,a}^{\text{AoD}})}$ for $i \in \{1, \ldots, N_{\text{t}}\}$. The l-th element of vector $\mathbf{a}_u(\psi_{u,a}^{\text{AoA}})$ is equal to $e^{j\frac{2\pi d}{\lambda_{\text{c}}}(l-1)\sin(\psi_{u,a}^{\text{AoA}})}$ for $l \in \{1, \ldots, N_{\text{r}}\}$. 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 $\boldsymbol{G}_{u,a} = \sqrt{g_a g_u}\gamma_{u,a}\mathbf{a}_a(\psi_{u,a}^{\text{AoD}})\mathbf{a}_u(\psi_{u,a}^{\text{AoA}})^H$, where g_a and g_u are the antenna gains (in dBi) at AP a and user u, respectively.

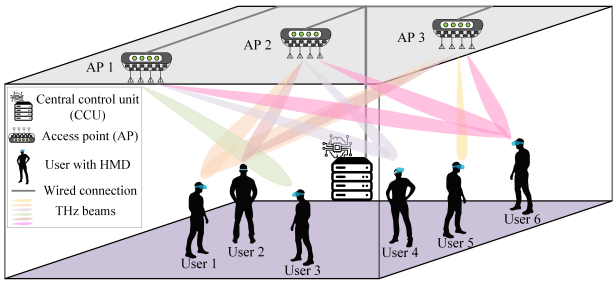

Fig. 1: A THz-enabled 360° video streaming system. 360° video streams are sent to each user by the APs which are not in the user's self-blockage region.

The APs are connected to a central control unit (CCU) via a wired connection. The beamforming vectors at the APs are determined by the CCU in each time slot. Let $\mathcal{T} = \{1, 2, \ldots\}$ denote the set of time slots, each with a duration of T^{slot}. 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

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

Let $\phi_u(t)$, where $0 \leq \phi_u(t) \leq 2\pi$, denote the longitude angle of user u's head orientation in time slot $t \in \mathcal{T}$. This angle is measured by the user's HMD. 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 [2]. Fig. 2(a) 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 $\mathcal{A}_u^{\text{nb}}(t) = \left\{a \,|\, a \in \mathcal{A},\, |\phi_u(t) - \phi_{u,a} - \pi| \geq \frac{\phi^{\text{blocked}}}{2}\right\}$, where $\phi_{u,a}$ denotes the longitude angle of the LoS link between user u and AP a. Let $r_u(t)$ denote the data rate of user u in time slot 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

$$r_u(t) = B \log_2 \left(1 + \boldsymbol{d}_{u,u}^H(t) \boldsymbol{\Gamma}_u^{-1}(t) \boldsymbol{d}_{u,u}(t)\right), \tag{2}$$

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.

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. 2(b), we consider that each video chunk has a fixed duration of T^{chunk} (in time slots) 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. A viewport prediction algorithm is used

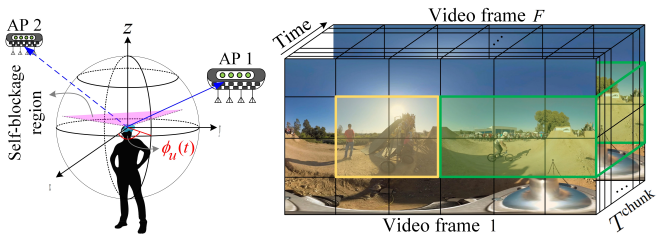

Fig. 2: (a) Illustration of a user's self-blockage region. AP 2 is located within the self-blockage region, whereas AP 1 is not. (b) Illustration of a video chunk with F video frames. The tiles in the viewport are shown in green, while those in the marginal region are shown in yellow.

to determine which video tiles should be transmitted to the users. Due to prediction error, in addition to the tiles covering the predicted viewport, some additional tiles, which cover a marginal region of a viewport, may also be sent. Let $\mathcal{N}^{\text{pred}}_{u,c,v}$ denote the set of tile indices that should be sent to user u upon its request for chunk c of video v. Specifically, $\mathcal{N}^{\text{pred}}_{u,c,v}$ is a prediction set containing tiles from both the viewport and the marginal region. We consider that $\mathcal{N}^{\text{pred}}_{u,c,v}$ is given using the viewport prediction algorithm proposed in [4].

After receiving the chunk request from a user, the CCU determines the quality level of the tiles. Let $\mathcal{M} = \{1, \ldots, M\}$ denote the set of quality levels, where 1 is the lowest quality. 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 $n \in \mathcal{N}^{\text{pred}}_{u,c,v}$ 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

$$\sum_{m \in \mathcal{M}} \beta_{u,n,m} \leq 1, \ u \in \mathcal{U}, \ n \in \mathcal{N}^{\text{pred}}_{u,c,v}, \ c \in \mathcal{C}_v, \ v \in \mathcal{V}. \quad (3)$$

Let $\tau^{\text{REQ}}_{u,c,v}$ (in time slots) denote the time when user $u \in \mathcal{U}$ requests the tiles of chunk $c \in \mathcal{C}_v$ of video $v \in \mathcal{V}$. Let $B_u\left(\tau^{\text{REQ}}_{u,c,v}\right)$ (in time slots) denote the buffer status of user u in time slot $\tau^{\text{REQ}}_{u,c,v}$. To prevent buffer overflow, each user only requests for a new video chunk when its buffer status is below a certain threshold [5]. Let B^{THR}_u (in time slots) denote the buffer size threshold for user u. When the buffer status of user u is not below B^{THR}_u, it should wait for a period of time before requesting the next video chunk. Let $\tau^{\text{WT}}_{u,c,v}$ denote the waiting time for user u after receiving chunk c of video v. We have $\tau^{\text{WT}}_{u,c,v} = \left[\left[B_u(\tau^{\text{REQ}}_{u,c,v}) - \tau^{\text{TD}}_{u,c,v}\right]^+ + T^{\text{chunk}} - B^{\text{THR}}_u\right]^+$, where $\tau^{\text{TD}}_{u,c,v}$ (in time slots) denotes the time it takes for chunk c of video v to be transmitted from the APs to user u. We have $\tau^{\text{TD}}_{u,c,v} = \min\left\{t' \in \mathcal{T} \,\middle|\, \sum_{t=\tau^{\text{REQ}}_{u,c,v}}^{t'} r_u(t) \geq T^{\text{chunk}} \sum_{n \in \mathcal{N}^{\text{pred}}_{u,c,v}} \sum_{m \in \mathcal{M}} \beta_{u,n,m}\nu_m\right\} - \tau^{\text{REQ}}_{u,c,v} + 1$. The next chunk of video v is requested by user u in time slot $\tau^{\text{REQ}}_{u,c+1,v} = \tau^{\text{REQ}}_{u,c,v} + \tau^{\text{TD}}_{u,c,v} + \tau^{\text{WT}}_{u,c,v}$. The buffer status of user u is updated as $B_u(\tau^{\text{REQ}}_{u,c+1,v}) = \left[\left[B_u(\tau^{\text{REQ}}_{u,c,v}) - \tau^{\text{TD}}_{u,c,v}\right]^+ + T^{\text{chunk}} - \tau^{\text{WT}}_{u,c,v}\right]^+$.

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 three 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 $\ell^{\text{spatial}}_{u,c,v}$, and the rebuffering delay $\tau^{\text{RD}}_{u,c,v}$ [4]. Next, 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. The variance of the quality level of the tiles in the viewport may lead to viewing irritation and other physiological effects such as fatigue. We have $\ell^{\text{spatial}}_{u,c,v} = \frac{1}{\left|\mathcal{N}^{\text{actual}}_{u,c,v}\right|} \sum_{n \in \mathcal{N}^{\text{actual}}_{u,c,v}} \left(\sum_{m \in \mathcal{M}} \beta_{u,n,m} m - \bar{\ell}^{\text{view}}_{u,c,v}\right)^2$.

The rebuffering delay $\tau^{\text{RD}}_{u,c,v}$ captures video stalling during playback. 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

$$\Upsilon_{u,c,v} = \bar{\ell}^{\text{view}}_{u,c,v} - \lambda^{\text{spatial}} \ell^{\text{spatial}}_{u,c,v} - \lambda^{\text{RD}} \tau^{\text{RD}}_{u,c,v}, \quad (4)$$

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

III. Problem Formulation

Each user asynchronously requests for a new video chunk based on its buffer status. The CCU aims to maximize the users' expected long-term QoE. Since the CCU should make *asynchronous* decisions on the quality level of the requested video tiles, we formulate the bitrate selection for those tiles as a MacDec-POMDP [9], [11] with $T^{\max}$ decision epochs. For each user, an agent in the CCU is responsible for providing high-quality video tiles to that user. Next, we describe the observation, action, and reward of each agent.

Observation: The CCU does not have access to the global system state (e.g., $\mathcal{N}^{\text{actual}}_{u,c,v}$). Instead, it obtains a partial observation of the underlying system state. Let binary vector $\boldsymbol{v}_{u,c,v} \in \{0,1\}^N$ indicate the tile indices predicted for transmission 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 $\mathbb{1}(n \in \mathcal{N}^{\text{pred}}_{u,c,v})$. We refer to the agent responsible for making a decision on behalf of user 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 user u requests chunk c of video v, the macro-observation vector of agent u contains $\boldsymbol{v}_{u,c,v}$ and the buffer status of user u in time slot $\tau^{\text{REQ}}_{u,c,v}$ (i.e., $B_u\left(\tau^{\text{REQ}}_{u,c,v}\right)$). For $\tau^{\text{REQ}}_{u,c,v} \leq t < \tau^{\text{REQ}}_{u,c+1,v}$, we have $\boldsymbol{o}^{\text{m}}_u(t) = \left(\boldsymbol{v}_{u,c,v}, B_u\left(\tau^{\text{REQ}}_{u,c,v}\right)\right)$.

Action: At time slot $t = \tau^{\text{REQ}}_{u,c,v}$, agent u will take a macro-action which remains unchanged for $\tau^{\text{REQ}}_{u,c,v} \leq t < \tau^{\text{REQ}}_{u,c+1,v}$. Let $\boldsymbol{a}^{\text{m}}_u(t)$ denote the macro-action which is taken by agent u at time slot t. Let $\nu_{u,n}$ denote the bitrate of tile $n \in \mathcal{N}^{\text{pred}}_{u,c,v}$ when user $u \in \mathcal{U}$ requests chunk $c \in \mathcal{C}_v$ of video $v \in \mathcal{V}$. 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:

$$\nu_1 \leq \nu_{u,n} \leq \nu_M, \ u \in \mathcal{U}, \ n \in \mathcal{N}^{\text{pred}}_{u,c,v}, \ c \in \mathcal{C}_v, \ v \in \mathcal{V}. \quad (5)$$

Thus, we have $\boldsymbol{a}_u^{\mathrm{m}}(t) = (\nu_{u,n},\, n \in \mathcal{N})$, $u \in \mathcal{U}$, $\tau_{u,c,v}^{\mathrm{REQ}} \le t < \tau_{u,c+1,v}^{\mathrm{REQ}}$. After determining $\nu_{u,n}$, agent u rounds it down to the nearest possible bitrate based on the quality levels available in set $\mathcal{M}$. The variables $\beta_{u,n,m}$, $n \in \mathcal{N}$, $m \in \mathcal{M}$ can be determined for user u using the obtained bitrate for the tiles.

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

$$R(t) = \sum_{u\in\mathcal{U}} \sum_{v\in\mathcal{V}} \sum_{c\in\mathcal{C}_v} \Upsilon_{u,c,v} \mathbb{1}\big(\tau_{u,c,v}^{\mathrm{REQ}} + \tau_{u,c,v}^{\mathrm{TD}} = t\big). \quad (6)$$

In a POMDP, a history of observations and actions provides sufficient statistics for the agent to make decisions. Let $\boldsymbol{h}_u^{\mathrm{m}}(t)$ denote the macro-observation-action history of agent u in time slot $t \in \mathcal{T}$. Each agent u selects a macro-action $\boldsymbol{a}_u^{\mathrm{m}}(t) = \mu_u\left(\boldsymbol{h}_u^{\mathrm{m}}(t)\right)$ using a deterministic policy μ_u based on $\boldsymbol{h}_u^{\mathrm{m}}(t)$ in time slot t. Let $\boldsymbol{\mu} = (\mu_u,\, u \in \mathcal{U})$ and $\boldsymbol{a}^{\mathrm{m}}(t) = (\boldsymbol{a}_u^{\mathrm{m}}(t),\, u \in \mathcal{U})$ denote the agents' joint policies and macro-actions, respectively. Let $Q^*\left(\boldsymbol{h}^{\mathrm{m}}, \boldsymbol{a}^{\mathrm{m}}\right)$ denote the maximum action-value function when agents choose the joint macro-action $\boldsymbol{a}^{\mathrm{m}}$ given the joint macro-observation-action history $\boldsymbol{h}^{\mathrm{m}}$. For tile bitrate selection, we formulate the following optimization problem:

$$\mathcal{P}^{\mathrm{m}}: \;\; Q^*\left(\boldsymbol{h}^{\mathrm{m}}, \boldsymbol{a}^{\mathrm{m}}\right) = \underset{\boldsymbol{\mu}}{\text{maximize}}\; \mathbb{E}_{\boldsymbol{\mu}}\Bigg\{ \sum_{t'=t}^{T^{\max}} \gamma^{t'-t} R(t') \;\Big|\; \boldsymbol{h}^{\mathrm{m}}(t) = \boldsymbol{h}^{\mathrm{m}},\, \boldsymbol{a}^{\mathrm{m}}(t) = \boldsymbol{a}^{\mathrm{m}} \Bigg\}, \quad \text{subject to constraint (5)},$$

where γ is the discount factor. Problem $\mathcal{P}^{\mathrm{m}}$ aims to learn a joint policy $\boldsymbol{\mu}$ that maximizes the expected discounted reward.

By performing macro-action $\boldsymbol{a}_u^{\mathrm{m}}(t)$ in time slot $t = \tau_{u,c,v}^{\mathrm{REQ}}$, the number of bits required for transmitting chunk $c \in \mathcal{C}_v$ of video $v \in \mathcal{V}$ to user $u \in \mathcal{U}$ is determined. The CCU determines the beamforming vectors by maximizing the system's sum-rate in each time slot while satisfying the selected bitrate of the requested video tiles for users. In each time slot $t \in \mathcal{T}$, the CCU solves the following optimization problem:

$$\mathcal{P}^{\mathrm{b}}: \underset{\substack{\boldsymbol{b}_{u,a}(t),\, \xi_u \ge 0,\\ u\in\mathcal{U},\, a\in\mathcal{A}}}{\text{maximize}} \quad \sum_{u\in\mathcal{U}} r_u(t) - \lambda^{\mathrm{b}} \xi_u$$

$$\text{subject to constraint (1)},$$

$$T^{\mathrm{slot}} r_u(t) \ge \Delta_u^{\mathrm{rem}}(t) - \xi_u,\; u \in \mathcal{U},$$

where λ^{b} is a positive scaling factor. $\Delta_u^{\mathrm{rem}}(t)$ denotes the remaining bits of the requested chunk available for transmission to user $u \in \mathcal{U}$ in time slot $t \in \mathcal{T}$. $\Delta_u^{\mathrm{rem}}(t)$ is initialized by $T^{\mathrm{chunk}} T^{\mathrm{slot}} \sum_{n\in\mathcal{N}_{u,c,v}^{\mathrm{pred}}} \sum_{m\in\mathcal{M}} \beta_{u,n,m}\nu_m$ at time slot $t = \tau_{u,c,v}^{\mathrm{REQ}}$. For $\tau_{u,c,v}^{\mathrm{REQ}} + 1 \le t < \tau_{u,c+1,v}^{\mathrm{REQ}}$, we have $\Delta_u^{\mathrm{rem}}(t) = \left[\Delta_u^{\mathrm{rem}}(t-1) - T^{\mathrm{slot}} r_u(t-1)\right]^+$. ξ_u is a slack variable for penalizing the objective function when the remaining bits cannot be fully transmitted to user u in the current time slot. Problems $\mathcal{P}^{\mathrm{m}}$ and $\mathcal{P}^{\mathrm{b}}$ are difficult to solve. The former is a finite-horizon stochastic optimal control problem. The latter is a non-convex optimization problem.

IV. Algorithms Design

To solve problem $\mathcal{P}^{\mathrm{m}}$, we 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. In particular, 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. The actor network of agent u specifies the policy of that agent for tile bitrate selection. We have $\boldsymbol{a}_u^{\mathrm{m}}(t) = \mu_{\boldsymbol{\omega}_u}\left(\boldsymbol{h}_u^{\mathrm{m}}(t)\right)$. The centralized critic network of agent u learns the action-value function $Q_{\boldsymbol{\vartheta}_u}\left(\boldsymbol{h}^{\mathrm{m}}(t), \boldsymbol{a}^{\mathrm{m}}(t)\right)$.

DDPG is able to learn the policy and action-value functions in a stable and robust manner using a replay buffer and separate target networks [12]. Since agents in a MacDec-POMDP asynchronously start and complete their macro-actions, a new replay buffer needs to be designed. To this end, we collect the macro-observation, macro-action, and reward of the agents into a buffer called macro-action concurrent experience replay trajectories (Mac-CERTs) at each time slot $t \in \mathcal{T}$. For $\tau_{u,c,v}^{\mathrm{REQ}} \le t < \tau_{u,c+1,v}^{\mathrm{REQ}}$, agent $u \in \mathcal{U}$ receives a cumulative reward for the executed macro-action at time slot $\tau_{u,c,v}^{\mathrm{REQ}}$ as $R_u^{\mathrm{c}}(t) = \sum_{t'=\tau_{u,c,v}^{\mathrm{REQ}}}^{t} \gamma^{t'-\tau_{u,c,v}^{\mathrm{REQ}}} R(t')$. Then, at the end of each time slot t, agent u stores its transition experience in the Mac-CERTs buffer as a tuple $\left(\boldsymbol{o}_u^{\mathrm{m}}(t), \boldsymbol{a}_u^{\mathrm{m}}(t), \boldsymbol{o}_u^{\mathrm{m}}(t+1), R_u^{\mathrm{c}}(t)\right)$. 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. For training agent u's actor network, 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. 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 $u \in \mathcal{U}$ updates $\boldsymbol{\vartheta}_u$ by minimizing the temporal difference (TD) error. Let $\boldsymbol{\omega}_u^-$ and $\boldsymbol{\vartheta}_u^-$, respectively, denote the weights of agent u's target actor and critic networks. The TD error is obtained as follows:

$$\mathcal{L}_u^{\mathrm{TD}}(\boldsymbol{\vartheta}_u) = \mathbb{E}_{\Lambda_u^{\mathrm{m}}\sim\mathcal{D}^{\mathrm{m}}}\Big\{ \big(Q_{\boldsymbol{\vartheta}_u}\left(\boldsymbol{h}^{\mathrm{m}}, \boldsymbol{a}^{\mathrm{m}}\right) - \hat{Q}_{\boldsymbol{\vartheta}_u^-}\left(\boldsymbol{h}^{\mathrm{m}}, \boldsymbol{a}^{\mathrm{m}}\right)\big)^2 \Big\}, \quad (7)$$

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

Each agent $u \in \mathcal{U}$ updates $\boldsymbol{\omega}_u$ using the action-value function gradient. The policy gradient is obtained as follows:

$$\nabla_{\boldsymbol{\omega}_u} \mathcal{L}_u^{\mathrm{PG}} = \mathbb{E}_{\boldsymbol{o}_u^{\mathrm{m}}\sim\mathcal{D}_u^{\mathrm{o}}}\Big\{ \nabla_{\boldsymbol{a}_u^{\mathrm{m}}} Q_{\boldsymbol{\vartheta}_u}\left(\boldsymbol{h}^{\mathrm{m}}, \boldsymbol{a}^{\mathrm{m}}\right)\big|_{\boldsymbol{a}_u^{\mathrm{m}} = \mu_{\boldsymbol{\omega}_u}(\boldsymbol{h}_u^{\mathrm{m}})} \times \nabla_{\boldsymbol{\omega}_u} \mu_{\boldsymbol{\omega}_u}\left(\boldsymbol{h}_u^{\mathrm{m}}\right) \Big\}, \quad (8)$$

Algorithm 1 Training Algorithm for DRL-based Bitrate Selection

1: Set the maximum number of episodes $E^{\max}$, the minibatch size B^{m}, and the soft target network update constant $\varepsilon \in (0,1)$.
2: Randomly initialize the learnable parameters $\boldsymbol{\vartheta}_u$ and $\boldsymbol{\omega}_u$, $u \in \mathcal{U}$.
3: Set the target network weights $\boldsymbol{\vartheta}_u^- := \boldsymbol{\vartheta}_u$ and $\boldsymbol{\omega}_u^- := \boldsymbol{\omega}_u$, $u \in \mathcal{U}$.
4: **for** each episode **do**
5: Initialize $\boldsymbol{o}_u^{\mathrm{m}}$ and determine $\boldsymbol{a}_u^{\mathrm{m}} \leftarrow \mu_{\boldsymbol{\omega}_u}(\boldsymbol{h}_u^{\mathrm{m}}) + \boldsymbol{\varrho}^{\mathrm{m}}$ for each agent u.
6: **for** each time slot $t \in \{1, \ldots, T^{\max}\}$ **do**
7: Determine beamforming vectors using Algorithm 2.
8: Obtain the next macro-observation vector $\boldsymbol{o'}_u^{\mathrm{m}}$ and the cumulative reward R_u^{c} for each agent u.
9: **for** each agent $u \in \mathcal{U}$ **do**
10: Store the tuple $\left(\boldsymbol{o}_u^{\mathrm{m}}, \boldsymbol{a}_u^{\mathrm{m}}, \boldsymbol{o'}_u^{\mathrm{m}}, R_u^{\mathrm{c}}\right)$ in the Mac-CERTs buffer.
11: **if** macro-action $\boldsymbol{a}_u^{\mathrm{m}}$ is completed **then**
12: $\boldsymbol{o}_u^{\mathrm{m}} \leftarrow \boldsymbol{o'}_u^{\mathrm{m}}$ and $\boldsymbol{a}_u^{\mathrm{m}} \leftarrow \mu_{\boldsymbol{\omega}_u}(\boldsymbol{h}_u^{\mathrm{m}}) + \boldsymbol{\varrho}^{\mathrm{m}}$.
13: **end if**
14: **end for**
15: Sample a random minibatch of B^{m} from the sets $\mathcal{D}^{\mathrm{m}}$ and $\mathcal{D}_u$.
16: Determine the required gradients based on (7) and (8).
17: Update $\boldsymbol{\vartheta}_u$ and $\boldsymbol{\omega}_u$ for each agent $u \in \mathcal{U}$ using Adam optimizer.
18: $\boldsymbol{\vartheta}_u^- \leftarrow \varepsilon\boldsymbol{\vartheta}_u + (1-\varepsilon)\,\boldsymbol{\vartheta}_u^-$ and $\boldsymbol{\omega}_u^- \leftarrow \varepsilon\boldsymbol{\omega}_u + (1-\varepsilon)\,\boldsymbol{\omega}_u^-$, $u \in \mathcal{U}$.
19: **end for**
20: **end for**
21: Outputs are $\boldsymbol{\omega}_u$ for each agent $u \in \mathcal{U}$.

where $\mathcal{D}_u^{\boldsymbol{o}} = \left\{\boldsymbol{o}_u^{\mathrm{m}} \mid \left(\boldsymbol{o}_u^{\mathrm{m}}, \boldsymbol{a}_u^{\mathrm{m}}, \boldsymbol{o'}_u^{\mathrm{m}}, R_u^{\mathrm{c}}\right) \in \mathcal{D}_u\right\}$. Algorithm 1 describes our proposed training algorithm for DRL-based bitrate selection. We train the actor and critic networks of the agents for $E^{\max}$ episodes, each with $T^{\max}$ time slots. To encourage effective exploration, we add an exploration noise $\boldsymbol{\varrho}^{\mathrm{m}}$ to the actions determined by the actor networks. We use the soft update technique with constant ε to update the target networks. In Line 7 of Algorithm 1, we employ the WMMSE algorithm described below to solve $\mathcal{P}^{\mathrm{b}}$ in each time slot.

Let $\boldsymbol{v}_u(t) \in \mathbb{C}^{N_{\mathrm{r}}}$ denote the receive beamformer used by user $u \in \mathcal{U}$ in time slot $t \in \mathcal{T}$. The MSE of user u in time slot t is obtained as $e_u(t) = \|1 - \boldsymbol{v}_u^H(t)\boldsymbol{d}_{u,u}(t)\|^2 + \sum_{u' \in \mathcal{U}\setminus\{u\}} \|\boldsymbol{v}_u^H(t)\boldsymbol{d}_{u,u'}(t)\|^2 + \sigma^2\|\boldsymbol{v}_u(t)\|^2$. Problem $\mathcal{P}^{\mathrm{b}}$ is equivalent to the following optimization problem [13]:

$$\mathcal{P}^{\mathrm{MSE}}: \underset{\substack{\boldsymbol{b}_{u,a}(t),\, \boldsymbol{v}_u(t),\\ w_u(t),\, \xi_u \geq 0,\\ u\in\mathcal{U},\, a\in\mathcal{A}}}{\text{minimize}} \quad \sum_{u\in\mathcal{U}} w_u(t)e_u(t) - \log_2 w_u(t) + \lambda^{\mathrm{b}}\xi_u$$

subject to constraint (1),

$$w_u(t)e_u(t) - \log_2 w_u(t) \leq 1 - (\Delta_u^{\mathrm{rem}}(t) - \xi_u)/(T^{\mathrm{slot}}B),\ u \in \mathcal{U},$$

where $w_u(t)$ is a weight factor for user u in time slot t. To solve problem $\mathcal{P}^{\mathrm{MSE}}$, we develop a WMMSE algorithm. We initialize the beamforming vectors with a feasible solution. Given $\boldsymbol{J}_u(t) = \sum_{u' \in \mathcal{U}} \boldsymbol{d}_{u,u'}(t)\boldsymbol{d}_{u,u'}^H(t) + \sigma^2\boldsymbol{I}_{N_{\mathrm{r}}}$, the users' receive beamformers are obtained as follows:

$$\boldsymbol{v}_u(t) = \boldsymbol{J}_u^{-1}(t) \sum_{a \in \mathcal{A}_u^{\mathrm{nb}}(t)} \boldsymbol{G}_{u,a}^H \boldsymbol{b}_{u,a}(t),\ u \in \mathcal{U}. \tag{9}$$

For each user u, the weight factor $w_u(t)$ is updated as $w_u(t) = e_u^{-1}(t)$. For fixed $\boldsymbol{v}_u(t)$ and $w_u(t)$, $u \in \mathcal{U}$, $\mathcal{P}^{\mathrm{MSE}}$ is solved using convex optimization to obtain new beamforming vectors. A suboptimal solution of $\mathcal{P}^{\mathrm{MSE}}$ is obtained by iteratively optimizing $\boldsymbol{v}_u(t)$, $w_u(t)$, and $\boldsymbol{b}_{u,a}(t)$, $a \in \mathcal{A}$, $u \in \mathcal{U}$. Algorithm 2 describes our WMMSE algorithm.

Algorithm 2 WMMSE Algorithm for Beamforming Design

1: Initialize $\boldsymbol{b}_{u,a}(t)$, $u \in \mathcal{U}$, $a \in \mathcal{A}$ with a feasible solution.
2: **Repeat**
3: Update $\boldsymbol{v}_u(t)$ based on (9) and set $w_u(t) := e_u^{-1}(t)$ for $u \in \mathcal{U}$.
4: Obtain the updated $\boldsymbol{b}_{u,a}(t)$, $u \in \mathcal{U}$, $a \in \mathcal{A}$ by solving $\mathcal{P}^{\mathrm{MSE}}$ when $\boldsymbol{v}_u(t)$ and $w_u(t)$, $u \in \mathcal{U}$ are fixed.
5: **Until** $\sum_{u\in\mathcal{U}} \log_2 w_u(t)$ converges.
6: Outputs are $\boldsymbol{b}_{u,a}(t)$, $u \in \mathcal{U}$, $a \in \mathcal{A}$.

TABLE I: Simulation Parameters

Parameter	Value	Parameter	Value	Parameter	Value	Parameter	Value
f_{c}	1.05 THz	g_a, g_u	25, 15	$E^{\max}$	5000	ε	10^{-2}
$P^{\max}$	5 dBm	N_{t}, N_{r}	6, 2	$T^{\max}$	153	λ^{b}	2
σ^2	-77 dBm	ϕ^{blocked}	π	T^{chunk}	1 s	γ	0.99
B	0.5 GHz	$\lambda^{\mathrm{spatial}}$, λ^{RD}	0.5	T^{slot}	100 ms	B^{m}	256

For the actor and critic networks of the agents, we consider the neural networks comprising two long short-term memory (LSTM) layers and two fully connected (FC) layers. Each LSTM layer has a hidden size of d_{h}. The output size of the first and second FC layers is d_{fc} and d_{out}, respectively. After training, the computational complexity of the online tile bitrate selection using the pre-trained actor network obtained by Algorithm 1 for each agent is $O(d_{\mathrm{in}}d_{\mathrm{h}} + d_{\mathrm{h}}^2 + d_{\mathrm{h}}d_{\mathrm{fc}} + d_{\mathrm{fc}}d_{\mathrm{out}})$, where d_{in} is the size of the actor network's input layer. Furthermore, solving the optimization problem $\mathcal{P}^{\mathrm{MSE}}$ in Line 4 of Algorithm 2 has a polynomial computational complexity.

V. Performance Evaluation

We consider a 10 m $\times$ 10 m $\times$ 4 m indoor environment and three APs. The APs are located at $(9, 1, 4)$, $(5, 5, 4)$, and $(1, 9, 4)$. We consider six users, each with a height of 1.6 m. We conduct our experiments using a public 360° video dataset [10]. We use the viewport prediction algorithm proposed in [4], where the video tiles for transmission are predicted by combining the current chunk's viewport and another viewport obtained by spherical walk. A frame rate of 30 frames per second is considered for the videos. Each video frame is divided into 24 tiles. We consider five quality levels, resulting in the maximum achievable QoE for users being 5. We select the bitrate value for encoding the tiles from set $\{28, 33, 38, 43, 48\}$ Mbps. We set $d_{\mathrm{h}} = 512$ and $d_{\mathrm{fc}} = 256$. The learning rate in Algorithm 1 is set to 10^{-4}. Other simulation parameters are summarized in Table I. We implement our algorithms in Python 3.7 using PyTorch framework. For comparison, we consider the following benchmark algorithms:

- **Combined FoV tile-based adaptive streaming (CFOV) algorithm [4]**: In this algorithm, a priority-based bitrate adaptation algorithm is used to select the bitrate of tiles. The APs in the users' self-blockage region are determined after detecting beam failure, a process that takes 300 ms.
- **Alternating optimization (AO) algorithm**: In this algorithm, the average QoE and sum-rate maximization subproblems are solved iteratively. In each iteration, given the beamforming vectors, the average QoE is maximized for the users who request a new video chunk. Then, given the video tile bitrates, $\mathcal{P}^{\mathrm{MSE}}$ is solved using Algorithm 2.

In Fig. 3, we study the impact of increasing the maximum buffer size threshold on the 360° video streaming system

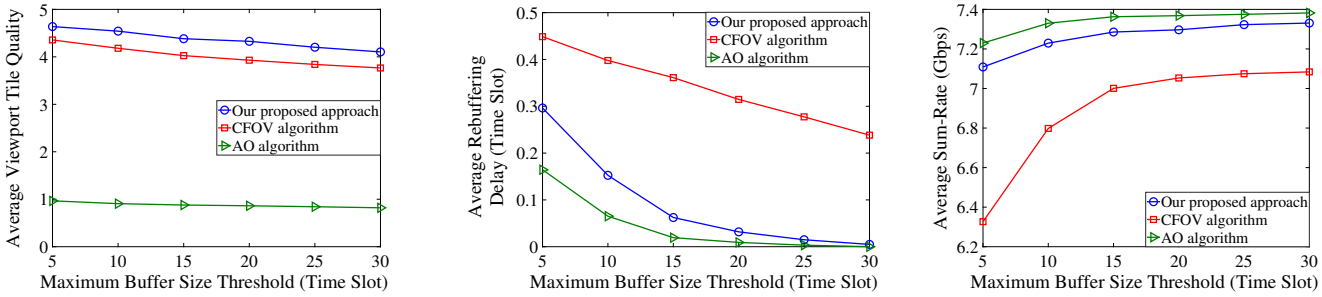

Fig. 3: (a) Average viewport tile quality, (b) average rebuffering delay, and (c) average sum-rate across users versus the maximum buffer size threshold.

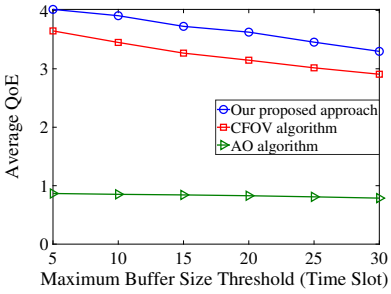

Fig. 4: Average QoE across users versus the maximum buffer size threshold.

performance. The users can prefetch more video chunks when the buffer size threshold increases. The results in Fig. 3(a) illustrate that our proposed approach provides a higher average viewport tile quality compared to the considered benchmark algorithms. 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. 3(b), the average rebuffering delay decreases for all the algorithms. The AO algorithm provides a conservative solution for bitrate selection by reducing the penalizing terms in eqn. (4). Thus, it achieves a lower average viewport tile quality, a lower average rebuffering delay, and a higher average sum-rate as shown in Fig. 3. The CFOV algorithm does not consider self-blockage and uses a reactive THz beam failure detection. Therefore, it has the highest average rebuffering delay and the lowest average sum-rate. Our proposed approach effectively reduces average rebuffering delay and increases the average sum-rate, while maintaining a high average viewport tile quality.

The results in Fig. 4 illustrate that, considering all the QoE factors in eqn. (4) , our proposed approach provides a higher average QoE compared to the considered benchmark algorithms. The average QoE decreases as the maximum buffer size threshold increases. This is due to the performance degradation of the employed viewport prediction algorithm.

VI. Conclusion

In this paper, we considered a THz-enabled multi-user 360° video streaming system with multiple multi-antenna APs. We modeled the bitrate selection for video tiles as a MacDec-POMDP. We proposed an asynchronous DRL-based algorithm for bitrate selection. We employed a WMMSE algorithm for the design of beamforming vectors at the APs. The results showed that our proposed video streaming approach can provide a higher QoE for the users compared with two benchmark algorithms. For future work, we will consider improving the performance of viewport prediction as well as replacing the WMMSE algorithm with a DRL algorithm [14].

References

[1] 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.

[2] 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.

[3] 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.

[4] 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.

[5] 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.

[6] 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.

[7] 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.

[8] 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.

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

[10] 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.

[11] 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.

[12] 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.

[13] S. S. Christensen, R. Agarwal, E. De Carvalho, and J. M. Cioffi, "Weighted sum-rate maximization using weighted MMSE for MIMO-BC beamforming design," *IEEE Trans. Wireless Commun.*, vol. 7, no. 12, pp. 4792–4799, Dec. 2008.

[14] M. Setayesh and V. W.S. Wong, "Viewport prediction, bitrate selection, and beamforming design for THz-enabled 360-degree video streaming," *arXiv preprint arXiv:2401.13114*, Jan. 2024.