Asynchronous Hybrid Reinforcement Learning for Latency and Reliability Optimization in the Metaverse over Wireless Communications

Wenhan Yu, Terence Jie Chua, Jun Zhao

Abstract-Technology advancements in wireless communications and high-performance Extended Reality (XR) have empowered the developments of the Metaverse. The demand for the Metaverse applications and hence, real-time digital twinning of real-world scenes is increasing. Nevertheless, the replication of 2D physical world images into 3D virtual objects is computationally intensive and requires computation offloading. The disparity in transmitted object dimension (2D as opposed to 3D) leads to asymmetric data sizes in uplink (UL) and downlink (DL). To ensure the reliability and low latency of the system, we consider an asynchronous joint UL-DL scenario where in the UL stage, the smaller data size of the physical world images captured by multiple extended reality users (XUs) will be uploaded to the Metaverse Console (MC) to be construed and rendered. In the DL stage, the larger-size 3D virtual objects need to be transmitted back to the XUs. We design a novel multi-agent reinforcement learning algorithm structure, namely Asynchronous Actors Hybrid Critic (AAHC), to optimize the decisions pertaining to computation offloading and channel assignment in the UL stage and optimize the DL transmission power in the DL stage. Extensive experiments demonstrate that compared to proposed baselines. AAHC obtains better solutions with satisfactory training time.

Index Terms—Reinforcement learning, resource allocation, latency, reliability, wireless communications, Metaverse.

I. INTRODUCTION

A. Background

The Metaverse has brought about a revolution in the Internet space, and is an important element of the Web 3.0 [1], [2]. The introduction of the Metaverse opened doors to not only interactive socialization through Extended Reality (XR), but also through its highly connected virtual world and ecosystem. XR is one of the highlights of the Metaverse [1]. With the extensive research and development in XR technologies [2], the once-so-expensive equipment has become affordable for both professionals and ordinary people, entering our daily life. The increasing accessibility of XR equipment drives the penetration of XR technologies, and it is postulated to be a key feature in multi-player games, work-place meetings, and potentially, simulations for scientific research and engineering. The crux of XR is to integrate virtual world scenes with physical world environments. This integrated virtual-physical experience not only promotes greater interactivity for the users with virtual scenes, but also paints an illustrative experience for the users.

XR Computation Offloading. The real-time virtual entity transformation with XR technologies is undoubtedly a highlight application of the Metaverse. We can think of these virtual entities as a merger of our physical world with the virtual world [3]. Each physical object in the real world is 3-dimensional (3D), and it has to retain its 3D form when translated into the virtual world. Hence, scenes of real-world objects have to be converted into 3D virtual objects. This replication of physical world objects into the virtual space is also known as digital twinning, which is a key underlying component of XR [3]. In recent years, the real-time translation and rendering of videos and images to 3D objects have been thoroughly studied [4], which enables the bridge between our physical world and the Metaverse. An example is the physical world translation technique, NeuralRecon, proposed by Sun et al. [5], which obtains remarkable results in realtime 3D construction from videos. Meng et al. [6] addressed the synchronization between physical objects and the digital models by optimizing the sampling rate and the prediction horizon. Nevertheless, despite rapid development in XR technologies, even existing state-of-the-art user devices [7] do not have sufficient computing power to render an XR scenario. An alternative, yet feasible method to powering XR is to consider mobile edge computation offloading (MECO) [8], which needs us to thoroughly study this system over wireless communications.

xURLLC. In the case of XR computation offloading, the real-world scenes can first be captured by the user device and uploaded to an edge console. The console responsible for translating physical world scenes into virtual scenes (3D) will handle the translation and rendering task and subsequently send the rendered XR scenes back to the user device. The 2D to 3D translation and rendering can lead to much data increment in the downlink (DL) stage. For example, the DL data size can be tens of times the uplink (UL) data size [9]. Therefore, it could be difficult for the DL data to be transmitted back to users in the DL stage with an acceptable delay, which causes transmission failures, and the failures reflect the unreliability of the system. For example, an unexpectedly large UL transmission data would result in orders of magnitude larger DL data size, which may take an astronomical amount of time for DL transmission. To enable truly immersive XR applications, the users' seamless experience and low latency need to be guaranteed. Therefore, we need to consider the UL-DL transmission as a whole to fulfill the reliability of the system and different Key Performance Indicators (KPIs) in

The authors are all with the School of Computer Science and Engineering, Nanyang Technological University, Singapore. Emails:

wenhan002@e.ntu.edu.sg, terencej001@e.ntu.edu.sg, JunZhao@ntu.edu.sg Corresponding author: Jun Zhao

different transmission stages.

B. Scenario

To address the above issue, this paper considers a novel interactive and asymmetric joint UL-DL scenario. Multiple XR users (XUs) are in a localized indoor area, and each user has physical world scenes (data) to be uploaded to the only Metaverse Console (MC) in the building for the generation of a virtual replica (digital twin). The UL and DL stages are taken in turn. In each UL stage, XUs will upload their data (scenes) to the MC with fixed transmission power decided by their devices within a fixed uplink transmission time interval (UTTI) (in 6G, this is expected to be on the order of milliseconds [10]) to be rendered and transformed into a 3D virtual digital twin. The MC determines which XU will perform computation offloading, and establishes an appropriate MC-XU channel allocation in the UL stage. The XUs that are not allocated to any channel will be idle for this transmission and wait for later transmission intervals to upload their data. In the DL stage, the MC will transfer the generated data to XUs in a limited downlink transmission time interval (DTTI), and the MC arranges power selection for the DL transmission to each XU. The main reason for having the slotted structure with fixed UTTI and limited DTTI is to ensure the reliability of the whole system so that the transmission delay for each iteration (one iteration contains one UL and one DL transmission) does not take an astronomical amount of time, such that it results in a backlog of subsequent data to be transmitted. In the context of current 3D real-time transmissions, only keyframes are used for generation [5] to lessen the computation overhead. In other words, these keyframes are very important and we need to ensure that they will be transmitted to the MC when embedding MECO. Therefore, to guarantee reliability, if the DL delivery of a packet of data is not completed within the limit of the DTTI, the DL transmission is deemed as a failure and the packet will need to be re-transmitted. Therefore, the UL rate should not always be as fast as possible, and we are required to seek a self-adaptive approach for considering the UL-DL stages in tandem.

C. Methodology

Due to the asynchronous multi-stage, discrete-continuous mixed action space (decisions on computation offloading, channel assignment, and power allocation) nature of our problem setting, we propose a novel deep Reinforcement Learning (RL) structure, namely Asynchronous Actors Hybrid Critic (AAHC), in quest of a near-optimal solution. We design two interactive RL Agents to optimize the two transmission stages. As our problem is sequential and divided into stages, we decompose the reward into stage-specific rewards and an overarching global reward. We introduce the AAHC model to our system, which is inspired by the renowned *Hybrid Reward Architecture* [11], and use the state-of-art DRL algorithm, Proximal Policy Optimization (PPO) [12], as the backbone.

D. Challenges and motivation

We next explain our motivations regarding our scenarios and methodologies from the following aspects.

6G xURLLC in the Metaverse. Advancements in 6G wireless communications [13] and high-performance XR technology [14] have empowered the developments of the Metaverse, but new challenges arise within. Researchers have made great efforts in the 5G URLLC applications [15]. However, they do not study diverse mission-critical applications involving XR or within the Metaverse. To power mission-critical realtime applications through the seamless digital twinning and projection of complex and detailed real-world entities onto the Metaverse, a reliable and ultra-low latency communication system is required. Furthermore, to cater to an increasingly "virtual" population in the Metaverse, one has to consider a more reliable, efficient, and comprehensive 6G xURLLC communication system. Therefore, we design a problem setting with several XUs in which we optimize the multi-user XR wireless transmission problem in the Metaverse. Furthermore, running out of battery in the XR device (often battery-limited) can also be somehow understood as unreliable Metaverse experiences, from the perspective of the human user using the XR device. Hence, incorporating energy consumption into our optimization can help reliable Metaverse experiences by human users. Besides, the Metaverse is a gigantic digital world with immense computational devices, which can lead to serious energy waste and pollution. The "Green Metaverse Networking" is first proposed by Zhang et al. [16] to address the necessities of energy efficiencies and energy consumption optimizations. Lee et al. [3] also emphasize that green computing must be highly regarded, as the waste and pollution strongly influence the attitudes of people towards the Metaverse. Based on these, the optimization in our paper also includes energy consumption.

Joint Optimization in MECO. Three important variables in improving the MECO efficiency are decision to computation offloading, channel allocation, and power selection. In our proposed scenario, the first two are optimized in the UL stage. The decision on whether to offload computation refers to which XUs will do computation offloading, and the channel allocation refers to the assignment of each XU to a channel for communicating with the Metaverse Console (MC). Appropriate channel allocation is important in creating an efficient computation offloading system. If an XU is allocated to a channel with too many other XUs assigned, there will be great interference, resulting in an overall inefficient data transfer [17]. The power selection in DL refers to the selection of power for each user, to perform the data DL transmission. An inappropriate allocation of power can increase interference between users, which is detrimental to data transfer efficiency. Hence, it is paramount for us to jointly optimize these three variables in our work.

Asymmetric multi-process to Asynchronous MDP. In existing computation offloading works as discussed in [8], [18], the data sizes of downloaded scenes are often taken to be similar to the data sizes of scenes to be uploaded. These works mentioned above do not consider the asymmetric data size scenario. In such circumstances, the downloaded, rendered 3D virtual objects can possibly be orders of magnitude larger than the uploaded scenes. This difference in data sizes introduces an issue of sensitivity, where a slight change in the UL transmission size may possibly induce a dramatic change in the DL data size. Also, the UL and DL are connected and influenced by each other. In this scenario, the power allocation in the DL can only be obtained based on the actions in UL. Thus, the orchestration of the UL and DL transmissions has to be considered in tandem.

Furthermore, the consideration of an asymmetric multiprocess transmission and the existence of two agents with different states and actions lead to a new challenge: Asynchronous Markov Decision Process (AMDP). In traditional MDP, the transition in each slot can be formulated as $S_1^1 \times A_1^1 \rightarrow R_1^1, S_1^2$ (here the superscript and subscript denote the time step and agent index). However, in a twoagent AMDP, the transition has to be represented as such: $S_1^1 \times A_1^1 \times R_1^1 \times S_2^1 \times A_2^1 \times R_2^1 \to S_1^2, R_a^1$, which means the state of $Agent_2$ is dependent on and influenced by $Agent_1$, and it's not suitable for them to perform actions simultaneously. Correspondingly, in this scenario, $Agent_1$ is responsible for optimizing the DCOs and channel access, and $Agent_2$ should arrange the downlink power allocation based on the DCOs and channel arrangement. And they are influenced by each other through the relationship between UL and DL data sizes. Moreover, we set the global reward R_q as some important metrics like the transmission failure flag can only be obtained after all agents have performed actions in the iteration, and this global reward is related to both agents. Thus, we give it to both agents to offer them a more specific view of the global process for the networks. This complex asynchronous setting serves as the motivation for us to develop an entirely new approach reinforcement learning approach.

Multiple Objectives to Hybrid reward RL. In our work, we consider a complex problem scenario in which we take into account multiple processes and high-dimensional objectives. Traditional RL approaches struggle with handling such complex problems as they lack the ability to handle each underlying objective on a minuscule scale. Therefore, we proposed and designed a hybrid reward architecture that allows us to achieve optimality in an overarching objectives.

In this paper, we propose a novel multi-agent hybrid reinforcement learning approach to solve the joint optimization problem. Our contributions can be summarized as:

- **xURLLC in the Metaverse.** We formulated a novel joint optimization problem in a multi-process xURLLC system on the Metaverse, and proposed a feasible and practical RL-based solution to tackle the problem. The UL and DL are considered in tandem to ensure the reliability and low latency of the system.
- Joint optimization in multi-process transmissions. We conducted the joint optimization of decisions on computation offloading, channel assignment, and power allocation in the UL-DL transmissions.
- Asynchronous MDP. We first studied a novel AMDP wireless communication problem, and devised a new RL structure to solve it.
- Asymmetric Agents. We adopted two agents with separate local objectives (one discrete and the other continuous) and an overarching global objective.

• Hybrid Critic PPO. We proposed a novel approach that uses a hybrid Critic to guide the training and convergence of both agents. Our approach achieves the total delay, retransmission percentage, and energy cost improvements of 65.48%, 90.67%, 56.82% in the most complicated scenario compared to iterative RL [19].

The rest of this paper is organized as follows. (i) We cover related literature to this piece of work and emphasize the novelty of our work in Section II. (ii) The system model and problem formulation are then introduced in Section III. (iii) In Section IV, we explain how the RL settings are crafted for our proposed environment. (iv) The details of our algorithm and model implementation are then expounded in Section V. (v) Next, we compare our proposed approach with baseline models and provide in-depth analyses of the results in Section VI. (vi) Finally, we provide a summary and conclusion to our work in Section VII.

II. RELATED WORK

This paper studies joint optimization in an ultra-reliable and low-latency MECO communication system with a reinforcement learning approach. Therefore, the related work contains the aspects of (i) mobile edge URLLC that is relevant to our scenario, (ii) joint optimization with RL as we jointly optimize the decisions on computation offloading, channel assignment, and power allocation, (iii) multi-agent RL as we use multiple RL agents in different transmission stages, and (iv) hybrid reward RL as the objectives in our system are highdimensional and complicated.

A. Mobile edge ultra-reliable and low latency communication

Several works have utilized traditional convex optimization tools [20], game theory [21], reinforcement learning [22], distributed learning [23] approaches to tackle the channel allocation and power selection problem in MECO studies. However, most of these works consider only single-stage data transmission scenarios: optimizing either the uplink (UL) or DL process. While it may seem that the UL and DL transmission processes are two separate processes, it is important to consider the concurrent optimization of both stages. Besides, many works have studied the applications of 5G URLLC on MECO in enhancing the performance of wireless communication systems applications [24], [25]. However, none of them considers the scenarios and KPIs in asynchronous multiple transmission processes in XR or Metaverse. What sets this work apart from those discussed above is that we consider a multi-stage scenario with deep reinforcement learning method, handling both UL and DL transmission which would require two different agents in a realistic scenario.

B. Joint optimization with reinforcement learning

Recently, there are also some excellent works solving joint optimization problems with RL. Guo *et al.* [26] solved the handover control and power allocation joint problem using MAPPO and obtained satisfactory results. It aggregates all users' actions under two stages and follows the traditional Centralized Training Decentralized Execution (CTDE) paradigm.

However, the problem they aimed to address is not multi-stage and hence directly uses MAPPO. He *et al.* [27] studied joint power allocation and channel assignment problems with RL in NOMA system. This work models the channel assignment as a reinforcement learning task and conducts power allocation under the current channel assignment. This method optimizes channel assignment and power allocation in turn, but it doesn't solve a time-sequential problem like the ones specified in our work.

C. Multi-agent reinforcement learning

In our work, we employed multiple agents. However, the widely used MARL algorithms [28], [29] are not feasible methods to tackle our problem. These algorithms adopt the centralized training and decentralized execution (CTDE) approach [28], [30]. Specifically, each agent involved will have its own Critic that considers the actions of all the other agents in a single time step. The individual agents' Actors will have decentralized policies and execute actions in a decentralized fashion. However, the existing adaptations of the conventional MARL algorithm are not suitable methods to solve our proposed problem. Firstly, the agents under the MARL algorithm select actions simultaneously in each time step, and the agents' Critics consider all actions of other agents in each time step. Each agent's observation space does not efficiently take in sequential agent actions, as the observation space of each agent will be sparse in each time step. Secondly, the two agents in our problem are utterly distinct from each other. In the UL stage, the user-console allocation action by the UL agent is discrete, while the output power selection by the DL agent is continuous. The asynchronous nature of our system model and the asymmetric roles of our agents make it inappropriate and impractical to adopt the existing MARL approach, with each agent sharing information with others by allowing the Critic to take in all states and actions of both agents. Thus, we decompose the reward assignment into multiple transmission stages and propose our novel model to address such challenges. We detail and elaborate on our algorithm in Section V.

D. Hybrid reward reinforcement learning

High-dimensional objective functions are common in communication problems, as we usually need to consider multiple factors such as energy consumption and time delay. When using reinforcement learning to tackle such problems, it is important to design high-dimensional objectives as smaller components of additive rewards. This issue of using RL to solve a high dimensional objective function was first studied in [11]. In their work, they proposed a *Hybrid Reward Architecture* (HRA) for reinforcement learning which aims to decompose high-dimensional objective functions into several simpler objective functions. HRA permits the discovery of reasonably good solutions, easily.

Our proposed Hybrid Critic differs from the HRA [11] in several aspects. Primarily, the HRA breaks the main objective into simpler objectives, allocating each simpler objective to different agents. Then the state-action value pair for each

TABLE I: Notations

Symbol	Description				
i m t	Index of XR users channels and iteration/time				
ι, πι, ι	stop/TTI				
\mathcal{M}	Channels set: $\{1, 2, \dots, M\}$				
\mathcal{N}_{-}	XR users set: $\{1, 2, \ldots, N\}$				
\mathcal{T}	Iteration/time-step (TTI) set: $\{1, 2, \dots, T\}$				
$oldsymbol{z}^t = [z_n^t _{n \in \mathcal{N}}]$	Decisions on computation offloading				
$\boldsymbol{\kappa}^t = [\kappa_n^t _{n \in \mathcal{N}}]$	Channel access management				
d'_{n}^{t}	Downlink transmission delay of user n at iteration t				
r_n^t	Uplink transmission rate of user n at iteration t				
r'_n^t	Downlink transmission rate of user n at iteration t				
D_n^t	Data from uplink of user n at iteration t				
D'_{n}^{t}	Data need to be transmitted in downlink of user n				
	at iteration t				
B_n^t	Left data size in the buffer of user n at iteration t				
E^t	Energy consumption in downlink at t .				
W_m	Bandwidth of channel m				
$ au_u$	Fixed uplink transmission time interval (UTTI)				
$ au_d$	Downlink transmission time interval (DTTI) limit				
p_n	Uplink power of user n				
p'^t, p'^t_n	Allocated downlink power from MC to XUs				
$h_{i,m}^t$	Channel gain between user n in channel m and MC				
	at iteration t				
$(\sigma_{n\ m}^{t})^{2}, (\sigma_{MC}^{t})^{2}$	² Additive White Gaussian Noise (AWGN) parameter				
,	of XU n in channel m at t and the MC, respectively				
$\mathcal{N}_m^t(\boldsymbol{\kappa^t})$	the set of XUs that are allocated to the channel m				
	at t with total number of $N_m^t(\boldsymbol{\kappa^t})$				

agent is aggregated to produce a unified value. In our hybrid value network, our reward is decomposed across several Critic branches. Instead of aggregating sub-target values of the single agent as per the HRA, our loss is aggregated across each Critic branches for each part (UL, DL, global) to facilitate the cooperative training of the agents, striving to achieve their own objectives and the global objectives, together. The HRA aims to find an optimal solution to a complex scenario by decomposing the reward into target-specific parts. HRA is designed for the single-agent DRL, which excludes the possibility of us adopting it directly. Our adaption of HRA separates the Critic into three branches, for handling the UL, DL, and global parts, respectively.

Next, our system model will be expounded on in the following section.

III. PROBLEM FORMULATION

We first formally describe our proposed system model and scenario. Then, we present the communication system and give the achievable transmission rates. Following that, we introduce the UL and DL objectives and piece them together and present the overall objective function of the communication model.

A. System Model

Consider the **asynchronous** UL-DL transmission system of n XUs ($\mathcal{N} = \{1, 2, ..., N\}$) sharing a MC with m channels ($\mathcal{M} = \{1, 2, ..., M\}$) within a building. A slotted time structure is applied by using the clock signal from the MC for synchronization [31], and each slot (transmission time interval (TTI)) contains two sub-intervals: uplink transmission time interval (UTTI) and downlink transmission time interval (DTTI). Each XU $n \in \mathcal{N}$ has its own data buffer, in which the XU will note how much data remains to be transmitted

from the XU to the MC for digital replication. We assume that the data sizes of the images to be transmitted can be partitioned into subsets of any size [32]. For each UL, we set the UTTI as τ_{μ} (in ms denoting milliseconds) to transmit data to the MC. At the start of each TTI, the MC will collect XU's information about the remaining data and channel gain, and then decide which one will offload data, arranging the channel access for them. Since the control information can be reliably completed through the in-band channel, and the size of control information is relatively small, we neglect the time for this control-message transfer, and only consider each TTI containing one UTTI and one DTTI. As we design our scenario to have many more users as compared to MC channels, some users may be arranged with no channel to upload data at each slot, so as to avoid channel congestion. And then, they need to wait to be allocated with channels to transmit data in the following slots. We assume that the uplink transmission powers are decided by the XUs' devices, and they are not optimized in this paper. Consequently, the energy consumption in the UL stage is not considered explicitly. In addition, it can be understood that by reducing the number of UTTIs used to transmit the whole data, we are also reducing UL energy consumption.

In the DL stage, the MC will select transmission power for users assigned with a channel in the current transmission iteration, to facilitate the data DL process. We set the limit of the downlink transmission time interval (DTTI) as τ_d (in ms). Assume that d'_n^t is the transmission delay of XU n in DL at time step t, if the DL delay exceeds the τ_d ($d'_n^t > \tau_d$), we regard this circumstance as an XU's failed attempt to receive the DL data. If the data transmission for that round is considered a failed attempt, the data will have to be rescheduled to be transmitted in the subsequent rounds.

To simplify the problem, we assume that the UL and DL processes are carried out alternately without overlap, and we use $\mathcal{T} = \{1, 2, \dots, T\}$ as the notation for the iterations taken to complete the tasks. It should be noted that each iteration contains both a UL and a DL process.

We next introduce the communication system as well as the models of UL and DL stages, and tie together all descriptions above. The system model is shown in Fig. 1.

B. Communication system

We adopt the Non-Orthogonal Multiple Access (NOMA) technology for the communication model. In NOMA system, several users can be multiplexed on one channel by the successive interference cancellation (SIC) and superposition coding [33]. As there are more XUs sharing fewer channels, we apply NOMA signal interference structure for both UL and DL transmissions, and consider the interference between XUs on the same channel. Below we give detailed descriptions of the achievable rates from each XU to MC in UL, and the rates from MC to each XU in DL.

For each $n \in \mathcal{N}$, we use z_n^t to denote XU *n*'s Decision on Computation Offloading (DCO) at time step *t*. Specifically, $z_n^t = 1$ means that XU *n* will offload the computation to MC at time step *t*, whereas $z_n^t = 0$ means that XU *n* will idle at time step t. Then we define a row vector $\boldsymbol{z}^t = [z_1^t, z_2^t, \dots, z_N^t]$ to represent all N XUs' DCO at time step t. We further define a $N \times T$ matrix \boldsymbol{Z} such that the tth row is the vector \boldsymbol{z}^t for $t \in \{1, 2, \dots, T\}$; i.e., the tth row and nth column of \boldsymbol{Z} is z_n^t .

We let $\kappa^t = [\kappa_1^t, \kappa_2^t, \dots, \kappa_N^t]$ be the channel access arrangement, and $p'^t = [p'_1^t, p'_2^t, \dots, p'_N^t]$ be the downlink power allocation, where $\kappa_n^t = m$ ($m \in \mathcal{M}$) means XU *n* is allocated to channel *m* at time step *t*, and p'_n^t is the power that the MC uses to communicate with XU *n* at time step *t*. The uplink transmission power is $\boldsymbol{p} = [p_1, p_2, \dots, p_N]$. Since we do not optimize \boldsymbol{p} for the uplink, we consider \boldsymbol{p} invariant with respect to time *t* for simplicity (the extension to time-variant \boldsymbol{p} just involves using more complex notation).

In the NOMA system, we follow [34] to consider that the decoders at the MC and XUs can recover the received signals from each channel through SIC, and multiple VUs can be multiplexed on each channel by superposition coding. The decoding sequence follows the rule explained in [34]. To better illustrate the decoding protocols, we define $\mathcal{N}_m^t(\kappa^t)$ as the set of $N_m^t(\kappa^t)$ XUs that are allocated to communicate with MC through channel m at time slot t, where " (κ^t) " represents that \mathcal{N}_m^t and N_m^t are functions of κ^t . Specifically,

$$\mathcal{N}_m^t(\boldsymbol{\kappa}^t) := \{ n \in \mathcal{N} | \kappa_n^t = m \}, \text{ and } N_m^t(\boldsymbol{\kappa}^t) = |\mathcal{N}_m^t(\boldsymbol{\kappa}^t)|.$$
(1)

For the UL NOMA, we order the XUs of $\mathcal{N}_m^t(\boldsymbol{\kappa}^t)$ in the descending order of the MC's received signals [34] sent from these XUs. After the above process, we denote the resulting indices of XUs in \mathcal{N}_m^t by $\bar{u}_1, \bar{u}_2, \ldots, \bar{u}_{N_m^t}$, where we simplify $\mathcal{N}_m^t(\boldsymbol{\kappa}^t)$ and $N_m^t(\boldsymbol{\kappa}^t)$ as \mathcal{N}_m^t and N_m^t . Then we have

$$p_{\bar{u}_1}|h_{\bar{u}_1,m}^t|^2 \ge p_{\bar{u}_2}|h_{\bar{u}_2,m}^t|^2 \ge \ldots \ge p_{\bar{u}_{N_m^t}}|h_{\bar{u}_{N_m^t},m}^t|^2, \quad (2)$$

where $h_{n,m}^t$ is the channel gain between XU n and the MC on channel m at time step t (this paper considers that uplink and downlink channel conditions are the same). Note that as each TTI is very short, we assume that the channel gain remains the same in each TTI (one UL plus one DL transmission), but varies across different TTIs. The detailed setting of $h_{n,m}^t$ will be explained in Section VI-C. According to [34], for $n \in \mathcal{N}_m^t$, the achievable **uplink rate** r_n^t for XU n is:

$$r_{n}^{t}(\boldsymbol{z}^{t},\boldsymbol{\kappa}^{t}) = W_{m} \log_{2} \left(1 + \frac{p_{n} |h_{n,m}^{t}|^{2}}{\sum_{j=\iota+1}^{N_{m}^{t}} p_{\bar{u}_{j}} |h_{\bar{u}_{j},m}^{t}|^{2} + W_{m}(\sigma_{MC}^{t})^{2}} \right)$$
(3)

where ι satisfies $\bar{u}_{\iota} = n$ (i.e., XU n is ranked at ι th position in \mathcal{N}_m^t according to descending order of received signals at the MC). W_m denotes the bandwidth of channel m, and $(\sigma_{MC}^t)^2$ is the power spectral density of Additive White Gaussian Noise (AWGN) at the MC.

For the DL, we also order XUs in a descending order based on channel-to-noise ratios, and denote the resulting indices of XUs using channel m as $\bar{d}_1, \bar{d}_2, \ldots, \bar{d}_{N_m^t}$, satisfying [34]:

$$\frac{|h_{\bar{d}_1,m}^t|^2}{(\sigma_{\bar{d}_1,m}^t)^2} \ge \frac{|h_{\bar{d}_2,m}^t|^2}{(\sigma_{\bar{d}_2,m}^t)^2} \ge \dots \ge \frac{|h_{\bar{d}_{N_{m,m}^t}}^t|^2}{(\sigma_{\bar{d}_{N_{m,m}^t}}^t)^2}.$$
 (4)



Fig. 1: Our proposed system model. The top left of the figure illustrates our problem scenario, while the bottom left showcases a snippet of our proposed algorithm. The right of the figure highlights our transmission mechanism.

Then the achievable **downlink rate** r'_n^t for XU n is:

$$r'_{n}^{t}(\boldsymbol{z}^{t}, \boldsymbol{\kappa}^{t}, \boldsymbol{p'}^{t}) = W_{m} \log_{2} \left(1 + \frac{p'_{n}^{t} |h_{n,m}^{t}|^{2}}{\sum_{j=1}^{\iota'-1} p'_{\bar{d}_{j}} |h_{n,m}^{t}|^{2} + W_{m}(\sigma_{n,m}^{t})^{2}} \right),$$
(5)

where ι' satisfies $\bar{d}_{\iota'} = n$ (i.e., XU *n* ranks ι' th in channel *m* in the descending order of channel-to-noise ratios), and $(\sigma_{n,m}^t)^2$ denotes the power spectral density of Additive White Gaussian Noise (AWGN) of XU *n* in channel *m* at time step *t*. Next, we will expound on the formulated problem.

C. Uplink

An important objective of the UL stage is to reduce the total delay in transmitting the data in all XUs' buffers to the MC. As described in Section III-A above, we aim to ensure the successful transmissions of these keyframes. Hence if the DL delay exceeds the DTTI limit τ_d (i.e., if $d'_n > \tau_d$), that set of data transmitted in that iteration is nullified and has to be retransmitted. Re-transmissions will increase overall delay (i.e., the number of TTIs used for finishing the whole task), and it is considered as a system **unreliability**. This is because in real applications, more transmissions may cause more packet losses, more energy usage and longer delay. Hence, the UL transmission delay is influenced by the DL power allocation, because poor actions selected by the DL agent (Agent₂) will

cause re-transmission. We let a UL agent $(Agent_1)$ arrange XUs to the available channels. Considering that this is a multistage sequential optimization problem, an XU n which is not arranged with any channel at time step t will not partake in both UL and DL stages, contributing no interference to other XUs.

With the UL transmission rate $r_n^t(\boldsymbol{z}^t, \boldsymbol{\kappa}^t)$ (abbreviated as r_n^t below) defined in Eq. (3), the data transmitted from XU n to MC at time step t is:

$$D_n^t = \min(B_n^t, r_n^t \times \tau_u), \ \forall n \in \mathcal{N},$$
(6)

where B_n^t means the remaining data of XU n at time step t to be transmitted. This set of data will be digitized on the MC and sent back to XUs in the DL stage. The size of the rendered data is formulated as $D'_n^t = f_n(D_n^t)$. The function $f_n(\cdot)$ is the data size translation and rendering formula from 2D to 3D. The expression of $f_n(\cdot)$ depends on the specific Metaverse application that XU n executes with the help from the MC.

The 3D real-time construction has already been applied with the local computation methods and obtained impressive performance. Even a single home edition GTX 2080 can handle this task in milliseconds [5], but we consider a server with much more computing power. Thus, this paper mainly studies the communication problem of this scenario, considering the asynchronous UL-DL communication, and seeks nearoptimal channel resources and power allocation. We take the computation resources at the MC as sufficient and assume the digital replication execution time on the MC as negligible. Thus, the sub-target of UL agent in each iteration t is to maximize the UL sum rate:

$$O_{u} : \max_{\boldsymbol{z}^{t}, \boldsymbol{\kappa}^{t}} \left(\sum_{n \in \mathcal{N}} r_{n}^{t} \right),$$
(7)
s.t. $\kappa_{n}^{t} \in \{1, 2, \dots, M\}, \ \forall n \in \mathcal{N}, \ \forall t \in \mathcal{T},$
 $z_{n}^{t} = 1, \ \forall n \in \mathcal{N}, \ \forall t \in \mathcal{T}.$

D. Downlink

In the DL stage, $Agent_2$'s main objective is to (i) minimize the total re-transmission counts and (ii) minimize energy spent for the DL transmission. As described in the previous section (III-A), to ensure the reliability of the whole system, if the DL delay exceeds the DTTI limit τ_d ($d'_n^t > \tau_d$) permitted for downlink, that set of data transmitted in that iteration is nullified and has to be re-transmitted.

In contrast to the fixed uplink transmission power in the UL stage, we adopt the variable $p'^t = [p'_1, p'_2, \dots, p'_N]$ as the power allocated to each XU by $Agent_2$ in the DL stage to be optimized. Therefore, the transmission delay d'_n^t of each XU in DL stage is represented as:

$${d'}_{n}^{t} = \frac{{D'}_{n}^{t}}{{r'}_{n}^{t}} = \frac{f_{n}(D_{n}^{t})}{{r'}_{n}^{t}},$$
(8)

where r'_n^t is short for the downlink transmission rate $r'_n^t(\boldsymbol{z^t}, \boldsymbol{\kappa^t}, \boldsymbol{p'^t})$. We note that each XU *n* will have a different DL delay d'_n^t . To simplify the problem of having a variable start time for each subsequent transmission, we assume that all XUs have the same UTTI and DTTI limit synchronized by the MC, and the next UL turn will start immediately after all XUs finish the current DL transmission.

In the desirable event that d'_n^t doesn't exceed the DTTI limit τ_d , this transmission is considered successful, and the remaining data in the XUs' buffers are as shown:

$$B_n^t = B_n^{t-1} - (1 - I_n^t) D_n^t.$$
(9)

where I_n^t is the failure flag:

$$I_{n}^{t} = \begin{cases} 0, & \text{if } d'_{n}^{t} \leq \tau_{d}. \\ 1, & \text{if } d'_{n}^{t} > \tau_{d}. \end{cases}$$
(10)

In addition, the energy E^t used at time step t is

$$E^{t} = \sum_{n \in \mathcal{N}: z_{n}^{t} = 1} {p'}_{n}^{t} \times \min({d'}_{n}^{t}, \tau_{d}).$$
(11)

The building floor-plan is taken to be a rectangle with length X and width Y, and the center of the rectangle has coordinates (0,0). The location of XU n is designed to be confined to the space. Therefore, the sub-targets of the DL agent in each iteration t is to minimize the number of XUs exceeding the pre-defined DL time limit τ_d and MC energy expenditure:

$$O_{d}: \min_{\boldsymbol{p}'^{t}} \left(w_{n} \sum_{n \in \mathcal{N}} I_{n}^{t} + w_{e} E^{t} \right).$$
(12)
s.t. $p'_{n}^{t} \in [p'_{\min}, p'_{\max}], \forall n \in \mathcal{N}, \forall t \in \mathcal{T}.$

where w_n and w_e are the weights of each sub-target, and they are represented by the reward instead of being directly set as constants in our proposed DRL algorithm.

E. Overall

To sum up, our objectives are to minimize the total time used to complete the whole UL and DL processes, and to minimize the energy spent in the DL transmission. In our global objective, we do not include the energy spent on UL as we defined the UL output power as fixed. Therefore, the global objective can be written as:

$$\min_{\boldsymbol{z}^{t},\boldsymbol{\kappa}^{t},\boldsymbol{p}'^{t},T} \left\{ w_{1} \left[T \times \tau_{u} + \sum_{t=1}^{T} \min\left(\max_{n \in \mathcal{N}} ({d'}_{n}^{t}), \tau_{d} \right) \right] + (13a) \right. \\
\left. w_{2} \left[\sum_{t=1}^{T} \sum_{n \in \mathcal{N}: z_{n}^{t} = 1} p'_{n}^{t} \times \min({d'}_{n}^{t}, \tau_{d}) \right] + (13b) \right]$$

$$w_3\left[\sum_{t=1}^{T}\sum_{n\in\mathcal{N}}I_n^t\right]\right\}$$
(13c)

s.t. C1:
$$\sum_{t=1}^{1} (1 - I_n^t) D_n^t = B_n^0, \ \forall n \in \mathcal{N},$$
 (13d)

$$C2: p'_n^t \in [p'_{\min}, p'_{\max}], \ \forall n \in \mathcal{N}, \ \forall t \in \mathcal{T},$$
(13e)

$$C3: z_n^t \in \{0, 1\}, \ \forall n \in \mathcal{N}, \ \forall t \in \mathcal{T},$$
(13f)

$$C4: \kappa_n^t \in \{1, .., M\}, \ \forall n \in \mathcal{N}, \ \forall t \in \mathcal{T},$$
(13g)

where

$${l'}_{n}^{t} = \frac{f_{n}(\min(B_{n}^{t}, r_{n}^{t} \times \tau_{u}))}{{r'}_{n}^{t}}.$$
(14)

The term in (13a) refers to the total time taken in UL-DL iterations to complete the task. The term in (13b) denotes the energy spent in the DL transmission. The term in (13c) is the transmission failure count. Note that we do not include the UL energy consumption in the formulated problem, because we fix the UL transmission power and time in each iteration. Also, minimizing the number of iterations can be understood as implicitly minimizing the energy consumption in UL. All the weights (w_1, w_2, w_3) will be reflected in the reward settings in Section IV-C instead of being specific constants.

Constraint C1 means the whole system will finish with T TTIs, where T is also a variable in our optimization problem. In practice, we let the episode (one episode includes the entire execution of the above-mentioned process) continue if there is still data in any XU' s data buffer to fulfill this constraint. C2 is the range of downlink transmission power allocated by MC for communicating with each XU. This constraint will be satisfied by setting the DL agent action space in practice. C3 and C4 specify the domains for the decisions on computation offloading and channel resource arrangement, respectively, which will be guaranteed by designing the UL agent action space. According to the above-formulated problem, we propose an innovative model-free reinforcement learning method. The reasons for not using convex optimization techniques and model-based reinforcement learning strategies are explained as follows.

RL over Convex Optimization: Firstly, our defined problem scenario aims to be reflecting "true-to-real world" As such, our problem formulation (objective function and constraints) is not naturally convex. There are three options we considered: (1) Redefining our problem as a convex one. This would certainly make the defined problem much easier to solve, but it does not represent an accurate model of the real world. A clear example of problem redefinition for the proposed problem would be the constraint relaxation of the discrete MC-XU allocation into a continuous one. (2) Finding an approximate solution to the non-convex problem. Solution will then only be an approximate solution and it is difficult to gauge how far it is from the optimal solution. (3) Adopting deep reinforcement learning techniques. Deep reinforcement learning (RL) techniques may not provide the optimal solution. However, with sufficient model training, the RL agents are able to handle complex non-convex and sequential problems and provide near-optimal solutions. Secondly, the XUs' remaining data at each transmission iteration is sequential (collective) and changing, which makes the number of variables increase with T. This increases the solution search space indefinitely, rendering convex optimization techniques or heuristic search as infeasible approaches. Moreover, the discrete variables (DCOs and channel access) and the continuous variable (power) are highly coupled, which causes an Inseparable Mixedinteger Non-linear programming (MINLP) problem. This is NP-hard [35] and tough to tackle.

Model-free over Model-based RL: Model-based (MB) mainly differs from model-free (MF) RL in that model-based requires the specification of the entire model, such as the transition probabilities. In other words, MB RL algorithms use a predictive model to select the optimal actions, whereas MF RL algorithms involve the training of a control policy. MB RL algorithms have brilliant performance in many scenarios. However, due to the predictive model requirement, these algorithms are hard to implement in communication problems. There are too many random variables and unpredictable changes in communication models. The model-based methods cannot accommodate the randomly evolving environment and are impractical for implementation due to the daunting computational complexity. This is also another reason why convex optimization approaches are not suitable for our proposed scenario.

Next, we demonstrate how the problem formulated above can be transformed into an RL problem.

IV. REINFORCEMENT LEARNING SETTING

Deep reinforcement learning is a state-of-the-art technique to solve time-sequential optimization problems with randomly evolving environments. In this section, we will dissect our previously proposed problem and present the RL approach to solving it. The goal of a model-free RL algorithm is to find a near-optimal policy $\pi^*(a|s)$ of the sequential Markov Decision Process (MDP), which can guide the agent to select the best action *a* under state *s*. This can also be extended to the multi-agent scenario. When designing an RL environment, we will set rewards to guide the agents to find a near-optimal solution to our problem. Thus, in the subsequent subsections, we discuss our design of the most important parts of RL: state design, action formulation, reward decomposition, and their reflection on our environment setting.

A. State

Although sophisticated states provide the agent with more information and a more comprehensive view, complex states can result in erratic training. Therefore, the amount of information to be perceived by each XU needs to be limited. Hence, weeding out less relevant variables is essential. Key attributes to include in the states are those which are collective or sequential. Therefore, the state s_u^t in UL and state s_d^t in DL stages are as follows:

1) Uplink State s_u^t : (i) The remaining data in each XU's buffer B_n^t , (ii) the UL transmission power of each XU's device p_n , and (iii) channel gain of each XU n at TTI t: $h_{i,m}^t$.

2) Downlink State s_d^t : (i) The action by UL agent (combined by DCOs z^t and channel access management κ^t , which will be explained in the following part), (ii) The data buffer of each XU B_n^t , to equip the DL agent the overall view of the remaining data to be transmitted, (iii) The data size after generated and rendering: D'_n^t , and (iv) channel gain in current TTI $h_{i,m}^t$.

B. Action

The action space defines the boundaries of possible actions our agents can take. An appropriate and relevant setting, as per our earlier discussed optimization variables, is crucial. In our scenario, the action a_u^t and action a_d^t are respectively uplink action and downlink action, explained below.

1) Uplink Action a_u^t : The discrete action a_u^t in UL includes the DCOs and channel assignment:

$$a_u^t = \{ \boldsymbol{z}^t; \boldsymbol{\kappa}^t \}.$$
(15)

For discrete action space RL, we need to encode the actions into discrete indicators. Thus, we integrate the DCO and channel assignment as: $\Gamma_n^t = \{0, 1, 2, ..., M\}$. $\Gamma_n^t = 0$ means $z_n^t = 0$ (i.e., XU *n* does not offload the computation to MC at time step *t*). And $\Gamma_n^t = m$ for $m \in \mathcal{M}$ means *n* is allocated to channel *m*.

We use a tuple in which there are N elements corresponding to N users and each element can take M + 1 values, which corresponds to the number of MC channels, plus 1 for the possibility of a user not being assigned a channel. In practice, the discrete actions are in fact discrete indices to the DRL Agent. Therefore, we need to allocate consecutive but distinct indexes to each action. The most intuitive way of implementing this is



Fig. 2: The encoding method for the uplink (UL) action.

to use N-bit-(M+1)-number to denote the tuple, and convert it into a decimal action index (similar to binary to decimal):

$$a_u^t = \sum_{n=1}^N \Gamma_n^t (M+1)^{n-1}, \ \forall t \in \{1, \cdots, T\}.$$
 (16)

The encoding method is shown in Fig. 2.

2) *Downlink Action:* On the other hand, the continuous action power allocation to each XU is:

$$a_d^t = \{ {p'}_1^t, {p'}_2^t, \dots, {p'}_N^t \}, \ \forall {p'}_n^t \in [{p'}_{\min}, {p'}_{\max}].$$
(17)

C. Reward

In our scenario, the two agents not only have their own role-specific goals, but also have to fulfill a global target that works in the combined best interest of both agents. In our proposed scenario, we can observe some trade-offs: (i) While $Agent_1$ aims to fulfill its task of uploading the data to MC in the shortest time possible, it has to observe the data downlink speed to ensure that the uploaded data size can be managed by the DL agent in the downlink transmission. (2) While $Agent_2$'s objective is to minimize the DL transmission delay, it has to manage the energy spent for transmission at the same time. Due to these trade-offs between and within agents, we have to establish a unified priority by introducing a global target that serves as the global information to both agents. In our proposed work, minimizing the overall total time spent on UL and DL transmission (including re-transmissions) are related to both agents, and hence, are adopted as the global target. We also let the proposed algorithm learn the global reward separately with a global branch (which will be discussed in Section V-C), to give the networks a more specific view of the global process.

Therefore, we have chosen to adopt both role-specific rewards and global rewards for our scene. We introduced additional self-defined rewards to guide the agents' training, as sparse rewards detract the training progress of RL agents. In practice, the rewards are structured as follows:

1) Uplink Reward: R_u^t encompasses: (i) Upload efficiency penalty R_{ur}^t : a penalty will be given in each UL transmission in which a lower data transmission rate results in a bigger penalty.

2) Downlink Reward: R_d^t encompasses: (i) Download efficiency penalty R_{dr}^t : a larger downlink delay results in a larger penalty. (ii) Energy expenditure penalty R_{ene}^t : a penalty will be issued to the agent for the expense of energy. (iii) Power allocation guide $R_{i,qu}^t$: a small penalty will be

given when the XU that is not allocated a channel is assigned a power greater than 0.

3) Global Reward: R_g^t encompasses: (i) total delay penalty R_{ite}^t : a larger penalty will be added to every transmission iteration for both UL and DL agents, and (ii) Re-transmission penalty $R_{i,re}^t$: Higher transmission failures and hence, higher re-transmission counts result in a larger penalty added to both UL and DL agent.

In a hybrid reward setting where rewards of multiple agents are considered, we recommend standardizing the values across the different contributing rewards to smaller scales, to ease the training. Therefore, note that all the rewards are shaped (weighted) to easy-for-training scales in this paper. The numerical reward settings in this paper for reference are as follows:

• Uplink – upload efficiency penalty:

 $R_{ur}^t = -\sum_{n=1}^N \frac{(1-\frac{D_n^t}{B_n^0})}{N}$. This reward denotes the average XUs' transmitted portions of their total data in buffers, and the range of this reward is [-1, 0].

- **Downlink** download efficiency penalty: $R_{dr}^{t} = -\min(\sum_{n=1}^{N} \frac{d_{n}^{\prime n}}{\tau_{d}N}, 1)$. This reward is the average XUs' ratio of the actual required time to tolerant DL transmission delay. The range is [-1, 0].
- **Downlink** energy penalty: $R_{ene}^t = -\sum_{n=1}^{N} \frac{p'_n - p'_{\min}}{(p'_{\max} - p'_{\min})N} \times 0.5$. This is the average XUs' ratio of the allocated power to the maximum DL transmission power. Its range is [-0.5, 0].
- **Downlink** power allocation guide: $R_{i,gu}^t = -0.2$ if user *n* in channel 0 is allocated to power.
- Global total system delay penalty: $R_{ite}^t = -1$ for every iteration.
- Global re-transmission penalty:
- $R_{i,re}^t = -0.5$ for every re-transmission. The retransmissions also lead to an increased number of iterations. Therefore, we do not give the re-transmission a too large penalty.

V. METHODOLOGY

In this section, we introduce our proposed novel algorithm, Asynchronous Actors Hybrid Critic (AAHC). We will first introduce our inspiration behind this algorithm, and then introduce the preliminary algorithm: Proximal Policy Optimization (PPO). Finally, we will detail the derivation of AAHC from PPO, the structure of AAHC, and discuss the training mechanism.

A. Inspiration

In our problem definition, we aim to minimize the overall task delay (including re-transmission delay) and transmission energy consumption to fulfill the "Green Metaverse" demand while achieving each agent's objectives. Given the multitude of objectives, reinforcement learning agents struggle in achieving them and may fail to achieve convergence. We propose a novel multi-input, multi-objective (output) Critic which aims to be able to handle complex scenarios such as the one presented in our work, and simplify the objectives into bite-size challenges. Our Critic's objectives can be broken down as such: The UL Critic branch guides the UL agent in (i) increasing the UL data transmission rate in every UL. The DL Critic branch guides the DL agent in (ii) decreasing the DL transmission delay and (iii) minimizing MC DL transmission energy consumption. We employ an additional, overarching branch within our Critic which handles the global objective, (iv) to reduce the overall time taken to complete the task (UL and DL), and (v) total energy spent by MC for the DL transmission, as these are important objectives of our system. It is crucial to note that agent policies that minimize transmission failure and retransmission could potentially assist in decreasing the overall system transmission delay.

Our multi-input, multi-objective Critic is inspired by Hybrid Reward Architecture (HRA) [11], which has not been used for solving problems related to wireless communications to the best of our knowledge. HRA can utilize the domain knowledge by decomposing the reward into simpler parts, which has been demonstrated by extensive experiments in multiple fields, e.g., video games [36]. However, unlike our proposed AAHC, HRA does not consider the multi-agent setting and the asynchronous interaction between agents. Our reward is decomposed across two agents, and uses a similar way of updating the value network (Critic) by the weighted sum of the loss functions. In the next sub-section, we will introduce the preliminary RL algorithm, Proximal Policy Optimization (PPO), which is chosen as the backbone of our AAHC algorithm.

B. Preliminary: Proximal Policy Optimization (PPO)

PPO is a state-of-the-art, effective RL algorithm, which has been actively used in wireless communication research [37]. PPO is a suitable RL algorithm to tackle our proposed scenario as PPO can handle both discrete and continuous action spaces through fitting different output heads on the Actor network. Next we will expound PPO, focusing on its three main underlying features: (i) Policy gradient, (ii) Importance sampling and (iii) Policy constrain.

Based on policy gradient methods. The widely used policy gradient method computes an estimator and embeds it into a stochastic gradient ascent algorithm to maximize the expected reward:

$$J(\theta) = \sum_{\tau} \pi_{\theta}(\tau) R(\tau), \qquad (18)$$
$$\nabla_{\theta} J(\theta) = \sum_{\tau} \nabla_{\theta} \pi_{\theta}(\tau) R(\tau)$$
$$= \sum_{\tau} \pi_{\theta}(\tau) \nabla_{\theta} log \pi_{\theta}(\tau) R(\tau)$$
$$= \mathbb{E}_{\tau \sim \pi_{\theta}} [\nabla_{\theta} log \pi_{\theta}(\tau) R(\tau)], \qquad (19)$$

where π_{θ} is a stochastic policy, R(...) denotes the reward function, and τ denotes the trajectories including $(s^0, a^0, ..., s^t, a^t)$. Recent works use an advantage function instead of a reward function to make training more stable. Thus, we rewrite Eq. (18) into:

$$J(\theta) = \mathbb{E}_{\tau \sim \pi_{\theta}}[A(\tau)]. \tag{20}$$

Note that this expected value $\mathbb{E} = [...]$ represents the average value of the sampled data.

Use of importance sampling. Importance sampling (IS) [38] is a method where an expectation with respect to a target distribution is approximated from another distribution. Hence, IS is an important trick adopted in PPO as it allows PPO to use different policies for sampling and evaluating trajectories, increasing the overall sample efficiency [12].

Here we use π_{θ} as the policy for evaluation and $\pi_{\theta'}$ as the policy for sampling data for training, and Eq. (20) can be rewritten as:

$$\mathbb{E}_{\tau \sim \pi_{\theta}}[A(\tau)] = \mathbb{E}_{\tau \sim \pi_{\theta'}}\left[\frac{\pi_{\theta}(\tau)}{\pi_{\theta'}(\tau)}A(\tau)\right].$$
 (21)

However, in practice, we use state-action pairs instead of trajectories to update the gradient. Thus, the objective function of the Actor can be written as:

$$J(\theta) = \mathbb{E}_{(s^t, a^t) \sim \pi_{\theta'}} \left[\frac{\pi_{\theta}(s^t, a^t)}{\pi_{\theta'}(s^t, a^t)} A^t \right],$$

$$\approx \mathbb{E}_{(s^t, a^t) \sim \pi_{\theta'}} \left[\frac{\pi_{\theta}(a^t | s^t)}{\pi_{\theta'}(a^t | s^t)} A^t \right],$$
(22)

where A^t is short for $A(s^t, a^t)$. Note that we use the \approx instead of the = symbol as calculating the exact probabilities of π_{θ} and $\pi_{\theta'}$ is impractical. This is because some states within our proposed scenario occur infrequently. Therefore, we assume that $\pi_{\theta}(s^t) = \pi_{\theta'}(s^t)$ [12].

Add KL-divergence penalty. After switching to $\pi_{\theta'}$ for data sampling, there remains an issue of unequal variances. Although Eq.s (20) and (21) have the same expectations, their variances are very different, as shown below:

$$Var_{\tau \sim \pi_{\theta}}[A(\tau)] = \mathbb{E}_{\tau \sim \pi_{\theta}}[A^{2}(\tau)] - (\mathbb{E}_{\tau \sim \pi_{\theta}}[A(\tau)])^{2}, \qquad (23)$$

$$Var_{\tau \sim \pi_{\theta}}\left[\pi_{\theta}(\tau) A(\tau)\right]$$

$$= \sum_{\tau \sim \pi_{\theta'}} \frac{\pi_{\theta'}^2(\tau)}{\pi_{\theta'}^2(\tau)} A^2(\tau) \pi_{\theta'}(\tau) - \left(\sum_{\tau \sim \pi_{\theta'}} \frac{\pi_{\theta}(\tau)}{\pi_{\theta'}(\tau)} A(\tau) \pi_{\theta'}(\tau)\right)^2$$
$$= \sum_{\tau \sim \pi_{\theta'}} \frac{\pi_{\theta}^2(\tau)}{\pi_{\theta'}(\tau)} A^2(\tau) - \left(\sum_{\tau \sim \pi_{\theta'}} \pi_{\theta}(\tau) A(\tau)\right)^2$$
$$= \mathbb{E}_{\tau \sim \pi_{\theta}} \left[\frac{\pi_{\theta}(\tau)}{\pi_{\theta'}(\tau)} A^2(\tau)\right] - (\mathbb{E}_{\tau \sim \pi_{\theta}} [A(\tau)])^2. \tag{24}$$

From the two variances (23) and (24), we can see that the distance between the distributions θ and θ' can not be large. Therefore, PPO adds a KL divergence penalty to the Actor objective function to constrain the distance:

$$J(\theta) = \mathbb{E}_{(s^t, a^t) \sim \pi_{\theta'}}[r(\theta)A^t]$$
(25)

s.t.
$$D_{KL}(\pi_{\theta}||\pi_{\theta'}) \le \sigma,$$
 (26)

where $r^t(\theta) = \frac{\pi_{\theta}(a^t|s^t)}{\pi_{\theta'}(a^t|s^t)}$ is the probability ratio between old and new policies. $D_{KL}(\cdot||\cdot)$ denotes the Kullback–Leibler (KL) divergence for measuring the distance between π_{θ} and $\pi_{\theta'}$.

These strategies make PPO an effective and reliable RL algorithm to tackle wireless communication optimization problems. Nevertheless, this KL divergence is impractical to calculate in practice as this constraint is imposed on every observation. Thus, in [12], the objective function is finally represented as:

$$\mathbb{E}_{(s^t, a^t) \sim \pi_{\theta'}}[f^t(\theta, A^t)], \tag{27}$$

where

$$f^{t}(\theta, A^{t}) = \min\{r^{t}(\theta)A^{t}, clip(r^{t}(\theta), 1-\epsilon, 1+\epsilon)A^{t}\}.$$
 (28)

The problem in (27) is solved by gradient ascent, therefore, the gradient is finally written as:

$$\Delta \theta = \mathbb{E}_{(s^t, a^t) \sim \pi_{\theta'}} [\nabla_{\theta} f^t(\theta, A^t)].$$
⁽²⁹⁾

Critic loss. In terms of the Critic, PPO uses a Critic with an identical network to the Actor, just like in other Actor-Critic algorithms. We use Generalized advantage estimation (GAE) [39] to calculate the advantage in practice. Thus, the loss function is formulated in [26] as:

$$L(\phi) = (V_{\phi}(s^{t}) - V_{target}^{t})^{2}, \qquad (30)$$

$$V_{target}^t = A^{GAE} + \gamma V_{\phi'}(s^t). \tag{31}$$

 A^{GAE} is the advantage calculated using GAE, which will be explained in the next part, and V(s) is the widely used statevalue function [40], which is estimated by a learned Critic network with parameter ϕ . We update ϕ by minimizing the $L(\phi)$, and the parameter ϕ' of the target state-value function periodically with ϕ . Using target value is a prevailing trick in RL, which has been used in many algorithms [41], [42].

C. Asynchronous Actors Hybrid Critic (AAHC)

This paper proposes a novel structure AAHC that selects the state-of-art RL algorithm PPO as the backbone. In our work, we equipped the AAHC with a discrete-action space Actor, a continuous-action space Actor, and a multi-head hybrid Critic. Different from existing algorithms like independent multi-agent RL in [19] that uses two separate independent agents, and the widely used CTDE-based algorithm like MAPPO [26] that uses the concatenation of states and actions in different stages, AAHC uses two asynchronous Actors and a hybrid Critic with three branches to learn the information in UL, DL, and global stages separately, and better utilizes the domain knowledge from the separate states, actions and rewards in different stages.

Function process: In each episode, the initial state s_u^0 will be observed by the UL Actor, which will output the selected action a_u . Then, the current UL transmission can be accomplished with the selected a_u , and the environment will provide a feedback reward R_u for the action a_u and the DL state s_d at the current stage. Following the UL transmission, the DL Actor will generate the power allocation a_d in the environment to accomplish the DL transmission task, and obtain the DL reward R_d for the choice of a_d , and global

reward R_g for the whole iteration. These rewards (R_u, R_d, R_g) will be used to generate the advantages (A_u, A_d, A_g) by GAE [39] for updating the Actors, and the state-values for UL, DL, global and loss functions of Critic will be calculated. This process repeats until the end of an episode. The abovementioned process is illustrated in Fig. 4. Next, we will explain the mechanisms of how Actors and Critic update, respectively.

Asynchronous Actors: In AAHC, there are two Actors, one is responsible for making decisions on computation of-floading and arranging channel resources in the UL stage, and the other is responsible for allocating transmission power in the DL stage. Apart from their asymmetric task, the action space types for both UL and DL agents, and hence policy parameterizations, are dissimilar. Thus, we adopt the action space methods proposed by Sutton [40]. We compute learned probabilities for each of the many actions for the Actor in the UL transmission stage, and learn the probability distribution of the policy for the Actor in DL transmission stage. In practice, continuous action values are chosen from a normal (Gaussian) distribution.

Each agent attempts to achieve its own role-specific and global objectives. However, both agents are designed to not have the ability to observe the other's role-specific objective as the success of the other agent's objectives is not within an agent's control. An example would be that $Agent_1$ is rewarded for having an overall short UL transmission time, while the UL transmission time is not within $Agent_2$'s control (which is DL power selection). In this case, allowing $Agent_2$ to receive rewards based on $Agent_1$'s control can be "confusing" and even "conflicting" to $Agent_2$'s policy training.

In Eq. (29), we established the policy gradient for PPO Actor, and in AAHC we have the gradients $\Delta \theta_1$, $\Delta \theta_2$ of $Agent_1$ and $Agent_2$ as:

$$\Delta \theta_1 = \mathbb{E}_{(s_u^t, a_u^t) \sim \pi_{\theta_1}'} [\nabla_{\theta_1} f^t(\theta_1, (A_u^t + A_g^t)], \quad (32)$$

$$\Delta \theta_2 = \mathbb{E}_{(s_d^t, a_d^t) \sim \pi_{\theta_2'}} [\nabla_{\theta_2} f^t(\theta_2, (A_d^t + A_g^t)], \qquad (33)$$

where s_u^t , s_d^t denote the states at time step t in UL and DL stages, respectively. A_u, A_d, A_g are the UL, DL and global advantage functions.

In terms of the **advantage function**, most techniques compute it with a learned state-value function V(s), and generalized advantage estimation (GAE) [39] is undoubtedly one of the most renowned methods. Running the policy for \overline{T} time steps and using them to update is a widely accepted method popularized in [43]. Following their paradigm, we use the truncated version of GAE as:

$$A_{u}^{t} = \delta_{u}^{t} + (\gamma\lambda)\delta_{u}^{t+1} + \dots + (\gamma\lambda)^{\bar{T}-1}\delta_{u}^{t+\bar{T}-1}, \qquad (34)$$

where
$$\delta_{u}^{t} = R_{u}^{t} + \gamma V_{\phi'}(s_{u}^{t+1}) - V_{\phi'}(s_{u}^{t}).$$
 (35)

and A_d, A_g are as the same. \overline{T} specifies the length of given trajectory segment, and γ specifies the discount factor, and λ denotes the GAE parameter.

Hybrid Critic: In our problem, the objectives of each agent are very different, while there remains an overall arching unified goal. Thus, we propose the Hybrid Critic to aid the estimation of the value function in the hybrid reward problem.



* d(x) means the dimension of x; h is the hidden layer size; in x out is the input-output dimensions of the Linear layer. In practice, the input a is batch size x dimension of a, and all layers are linear.

Fig. 3: Asynchronous Actor Hybrid Critic (AAHC) structure. Top of the figure illustrates the architecture of AAHC. Below are the networks of the Uplink Actor, Downlink Actor, and Hybrid Critic, respectively.

This Hybrid Critic has three heads and the value is divided into three parts $V_{\phi}^{u}, V_{\phi}^{d}, V_{\phi}^{g}$. The value function can be estimated by the three-head value network V_{ϕ} as shown below:

$$V^u_\phi = V_\phi(s_u),\tag{36}$$

$$V_{\phi}^{d} = V_{\phi}(s_{d}), \tag{37}$$

$$V_{\phi}^{g} = V_{\phi}(\{s_{u}; s_{d}\}). \tag{38}$$

Then, we calculate the three losses $L^{u}(\phi), L^{d}(\phi), L^{g}(\phi)$ with the advantages and values, and sum the weighted losses of each head according to Eq. (30) and Eq. (31) as the loss function of the Hybrid Critic:

$$L^{u}(\phi) = (V_{\phi}(s_{u}^{t}) - A_{u}^{t} - \gamma V_{\phi'}(s_{u}^{t}))^{2},$$
(39)

$$L^{d}(\phi) = (V_{\phi}(s_{d}^{t}) - A_{d}^{t} - \gamma V_{\phi'}(s_{d}^{t}))^{2},$$
(40)

$$L^{g}(\phi) = (V_{\phi}(\{s_{u}^{t}; s_{d}^{t}\}) - A_{a}^{t} - \gamma V_{\phi'}(\{s_{u}^{t}; s_{d}^{t}\}))^{2}, \quad (41)$$

$$L(\phi) = w_u \times L^u(\phi) + w_d \times L^d(\phi) + w_g \times L^g(\phi), \quad (42)$$

where ϕ , ϕ' are the weights of the network and the target network, and w_u, w_d, w_g are the weights of each head's loss that are all set as 1 in the experiments. We update the Hybrid Critic through Eq. (42) to improve the estimate of the hybrid value function. The intricacies of our algorithm are illustrated in Fig. 4. And the algorithm flow is in Algorithm. 1.

VI. EXPERIMENTS

In this section, we conduct several experiments to highlight the outstanding performance achieved by our proposed AAHC algorithm. We compare our algorithm performance against baseline models: (i) iterative independent RL [19], (ii) and iterative CTRL framework, based on commonly adopted metrics. The numerical settings and extensive experimental results are illustrated as well.

A. Baseline

To demonstrate the superiority of our proposed algorithm, we adopt commonly used baseline models. All baseline models are similar to our proposed model in that they contain two separate agents which work asynchronously, and that they are fitted with the PPO algorithm as their backbone.

iteRL. The most intuitive way of using RL in such a cooperative interactive environment is to implement two single standard iterative RL (iteRL) agents, similar to [19]. However,

Algorithm 1 Our proposed Asynchronous Actors Hybrid Critic (AAHC) algorithm.

- **Initiate:** uplink Actor parameter θ_1 , downlink Actor parameter θ_2 , Critic parameter ϕ and target network ϕ' , initial state $s_u^0, s_u^t \leftarrow s_u^0$;
- 1: **for** iteration = 1, 2... **do**
- 2: Agent₁ execute action according to $\pi_{\theta'_1}(a_u^t | s_u^t)$;
- Get reward R_u^t and next downlink state s_d^{t+1} ; 3:
- Agent₂ execute action according to $\pi_{\theta'_2}(a_d^t|s_d^t)$; 4:
- Get reward R_d^t, R_g^t and next uplink state s_u^{t+1} ; 5:
- if iteration ≥ 2 then 6:
- Sample $(s_u^t, a_u^t, R_u^t, s_u^{t+1}, s_d^t, a_d^t, R_d^t, R_d^t, s_d^{t+1})$ itera-7: tively;
- end if 8:
- 9:
- $\begin{array}{l} s_d^t \leftarrow s_d^{t+1}, \, s_u^t \leftarrow s_u^{t+1}; \\ \text{Compute advantages } \{A_u^t, A_d^t, A_g^t\} \text{ and target values} \\ \end{array}$ 10: $\{V_{u,targ}^t, V_{d,targ}^t, V_{g,targ}^t\}$ using current Hybrid Critic; for k = 1, 2, ..., K do
- 11:
- Shuffle the data's order, set batch size bs; 12:
- for $j = 0, 1, \dots, \frac{\text{trajectory length}}{bs} 1$ do 13:
- Compute gradient for uplink, downlink Actors by 14: Eq. (32) and (33);
- Update Actors by gradient ascent; 15:
- Update Critic with MSE loss using Eq. (42); 16:
- end for 17:
- Assign target network parameters $\phi' \leftarrow \phi$ for every 18. C steps;
- end for 19:
- 20: end for

in [19] they use two iterative deep Q networks, while we use two more advanced PPO networks for this baseline. In terms of the reward, we give the UL agent $R_U^t = R_u^t + R_q^t$, and the DL agent $R_D^t = R_d^t + R_g^t$. In other words, these two agents are not entirely separated but are still affiliated through the global reward. This baseline is to testify the performance if we apply no modification to the structure.

CTRL. The widely used, standard Centralized Training with Decentralized Execution (CTDE) based algorithms like MAPPO [28] cannot be used right out-of-the-box for our scenario in the execution stage, as the agents in our scenario are required to select actions asynchronously rather than at the same time. In order to compare our proposed methods with the CTDE-based MAPPO, we adapt the Centralized training reinforcement learning (CTRL) algorithm for comparison. We use a centralized Critic with two Actors, and in this baseline, we use the sum of rewards $R = R_u^t + R_d^t + R_q^t$ as the reward received by the Critic, instead of computing three different losses and using the sum-loss for updating the Critic. This baseline is to examine the performance if no hybrid reward structure is embedded in the Critic.

Random Allocation. In addition to the above-mentioned algorithms, we implement an algorithm that assigns both UL and DL random channel allocation and power selection, respectively. The random channel assignment and power se-



Fig. 4: Illustrations of iteRL and CTRL. iteRL uses two separate DRL agents with the UL agent taking in $R_u^t + R_a^t$, and the DL agent using $R_d^t + R_q^t$ to update the networks. CTRL only has one critic, which takes in the $R_u^t + R_d^t + R_a^t$ for updating the networks.

lection algorithm will serve as an intuitive baseline.

B. Metrics (KPIs)

To fairly compare the performance of the algorithms tackling our proposed scenario, we introduce several commonlyadopted metrics as key performance indicators (KPIs).

Given that we are using an RL approach to tackle our proposed problem, the most obvious performance metric would be the rewards. The objective of DRL algorithms is to obtain as high accumulated rewards in one episode as possible, and the rewards are directly related to multiple objectives, which reflect the algorithms' abilities to handle the proposed problem. Therefore, the training rewards can well serve as an intuitive and overall performance of the algorithms. Specifically, there are three main rewards in our proposed scenario: (i) Uplink reward, which stands for the uplink efficiency penalty, (ii) Downlink reward, which encapsulates the downlink efficiency and energy expenditure penalty, (iii) Global reward, which encompasses the total delay and retransmission penalty. Aside from the rewards, the training efficiency of an algorithm is another important metric. It reflects on an algorithm's ability to learn optimal policies quickly. Therefore, we illustrate the (iv) Training time of different methods.

In our proposed scenario, we consider the (v) Total latency taken to complete the digital twinning task as the most important objective. Hence, it makes complete sense to include the total time taken including the latency caused by re-transmissions. (vi) Re-transmission percentage (retransmission times divided by the number of total transmission iterations) reflects the reliability of the system and directly impacts the total time taken for UL and DL transmissions. To satisfy the "Green Metaverse" requirement, we are concerned with (vii) the Energy consumption. Additionally, we use (viii) the Maximum uplink rate of all XUs in one iteration to testify if the UL agent can handle the channel access task and be influenced by the DL transmission.

Note that we log the number of transmission iterations (UL and DL) during training instead of the total time (in milliseconds) as it is more clear and more intuitive.

C. Experimental settings

For our experiments, the bandwidth W_m for every channel is simulated to be 10 Ghz [10], and all the Gaussian noise power spectral densities are simulated to be -174 dBm/Hz. To simplify the simulation, we set the data augmentation function $f_n(\cdot)$ as a proportional function with slope c_n^t (i.e., $D'_{n}^{t} = c_{n}^{t} \times D_{n}^{t}$, where c_{n}^{t} is sampled from a uniform distribution [5,15] in this paper. The domain of c_n^t is referenced from the experimental results from demos of NeuralRecon [5] and Monster Mash [9], which are two impressive 3D reconstruction techniques that are abreast of the times. The initial buffer sizes B_n^0 and uplink transmission power of each XU are uniformly selected from [10, 20] Megabits (Mb) and [3, 10]Watts, respectively. The XUs and MC are uniformly located in a $100 \times 100 \ m^2$ indoor space. To ensure the adaptability of our method, these variables vary in every episode, and the solutions in each episode are not the same. The DTTI limit and UTTI in each iteration are simulated to 1.5ms and 0.5ms [44], respectively. And the minimum and maximum downlink transmission powers p'_{max} are 0 and 20 Watt, respectively. The max training steps (iterations) in one episode are 100, and the total training steps are 2 million steps¹. As the random values influence the experiment outcome to a certain degree, we conducted each experiment using global random seeds from 0 to 10 to ascertain a consistent and reliable study. We then draw the error bands to better quantify the performance of each algorithm. The detailed implementation and hyper-parameters are explained in Appendix A.

In terms of channel gain, Ibrahim *et al.* [45] studied the model of 6G indoor reconfigurable intelligent surface (RIS). Similar to them, we use the Rician fading as the small-scale fading. It is given by:

$$h_{i,m}^t = \sqrt{\beta_n^t} g_{i,m}^t, \tag{43}$$

where

$$\beta_n^t = \beta_0(L_n^t)^{-\alpha},\tag{44}$$

$$g_{i,m}^{t} = \sqrt{\frac{K}{K+1}\bar{g}} + \sqrt{\frac{1}{K+1}\tilde{g}},$$
(45)

$$L_n = \sqrt{(X_n - X_{MC})^2 + (Y_n - Y_{MC})^2 + H^2}.$$
 (46)

 (X_n, Y_n) is the location of XU *n*, and H is the height. L_n^t represents the distance between XU *n* and the MC, and β_n^t represents the large-scale channel gain of XU *n* at iteration *t*. \bar{g} and \tilde{g} stand for the Line-Of-Sight (LOS) component and the Non-LOS (NLOS) component, respectively. Here, \tilde{g} follows the standard complex normal distribution $\mathcal{CN}(0,1)$ distribution. β_0 denotes the channel gain at the reference distance $L_0 = 1$ m, and α denotes the path-loss exponent which is simulated as 2. The Rician factor K is simulated as 3.

Here, we use the notation m - n to denote the scenario where there are m channels and n extended reality users (XUs). In conducting our experiments, we fix the number of channels at 3 and vary the number of XUs across 4 to 8, for comparison.

D. Results

In this part, we first choose to only display the experiment results of the most complex scenario: the 3-8 configuration, to show the model convergence and characteristics. The complete set of experiments which include all other configurations are shown in Appendix B. Then, the metrics (KPIs) across different configurations are evaluated and discussed. The overall numerical results are shown in Table II.

1) Train-time model performance in 3-8 configuration: For ease of discussion, we compared the performance of our proposed method against other algorithms in the most complex setting: the 3-8 configuration. Although the action space in this configuration is extremely huge, AAHC has demonstrated superior performance in handling the task when compared to the iteRL and CTRL baseline models.

In terms of rewards, AAHC has achieved the highest uplink, downlink, and global reward across the training episodes, when compared to the baselines. The global reward encompasses the total delay and re-transmission penalty. Despite having gradually increasing global rewards, the maximum uplink transmission rate of XUs is not increasing steadily across the training episodes. We observe from Fig. 5(b) and Fig. 5(f) that the maximum uplink rate reaches a peak at about 1.5 million training steps and then decreases sharply. Then, it increases slowly from thereon, while the global reward increases almost steadily (Fig. 5(a)) in this stage. We can speculate that although the UL agent is able to achieve a higher max uplink rate, it may bring about higher transmission failure (re-transmission) counts, which can result in lower rewards. Therefore, AAHC learns to slowly decrease the max uplink rate after 1.5 million steps to avoid unacceptable retransmissions.

The MC DL transmission energy consumption initially increases in the early training episodes, and subsequently decreases in the later training episodes. We believe that the agent attempts to maximize reward by minimizing transmission failure (and hence the overall total time taken), through increasing DL transmission power output and lowering retransmission counts. However, this is at the expense of having a higher energy consumption which negatively impacts the global reward. Subsequently, it is likely that the agent learns to control the MC DL power output, using significantly lower power while maintaining re-transmission counts at a decently low rate. Overall, this improves the downlink reward and demonstrates the prowess of our proposed model in handling the complex scenario and reward structure.

Across the different performance metrics, iteRL performs the worst. The improvement in reward attainment of the UL agent across the training episodes is at the expense of the DL agent's performance. This poorer DL agent performance is reflected in its increasing DL transmission energy consumption, while the improving UL agent's performance is reflected in its overall decreasing re-transmission percentage, across

¹As PPO is an on-policy algorithm that only uses the latest trajectories sampled from the current stochastic policy, it will fill the trajectory buffer first and use it for training, then empty the buffer and refill. Therefore, the training steps here are not the actual training times, but the sample times. And the training times should be equal to training steps divided by buffer capacity (trajectory length).



Fig. 5: Training in 3-8 scenarios. Considering the randomly evolving environment, all experiments are conducted with global random seeds from 0-10, and the error bands are drawn.

the training. Although the global reward increases slightly across the training episodes, it achieves a much lower reward when compared to our proposed AAHC and CTRL algorithms. From the above observations, this signifies that the agents in the iteRL are non-cooperative and do not achieve an overall good channel arrangement and downlink power selection when compared to the AAHC and CTRL algorithms in this scenario. Nevertheless, the iteRL algorithm still achieves relatively good performance in less complex configurations such as 3 - 4 in Fig. 9.

The CTRL algorithm fails to find an optimal solution in the complex 3-8 configuration as the UL agent chose to keep the UL transmission rates at a low value so that the DL retransmission percentage, and hence re-transmission percentage and energy consumption stays low. This sub-par performance is reflected in its considerably low eventual uplink reward and consistently high downlink reward. This problem likely results from the ability of the UL agent to perceive and consider the DL agent's reward and objective. In such a case, the UL agent is partially rewarded based on the DL agent's actions and it is unable to decipher how its action influences its own objectives. Hence, it could have fallen into a local optimal when it finds that decreasing the uplink rate can lead to a relatively higher overall reward because of the lower re-transmission counts and energy consumption.

Complexity analysis. We also analyze the computational complexity with the aid of real training and execution time illustrations. Let (L_u, L_d, L_c) , (X_u^l, X_d^l, X_c^l) be the total number of layers, and the number of neurons in layer *l*'s of the three networks: UL Actor, DL Actor, and hybrid Critic. Let d(s) for state *s* be the input dimension, which is proportional to the state dimension described in Section IV-A. Then we have s_u^t and s_d^t for UL Actor and DL Actor respectively in

the *t*th training step. Here we first analyze the complexity in a single training step. In AAHC, one training step involves the updates of two Actors and the update of one Critic, and the hybrid Critic has three input-output branches. Therefore, the complexity of the *t*th training step can be derived as:

$$\begin{split} &O(B\{d(s_u^t)X_u^1 + d(s_d^t)X_d^1 + [d(s_u^t) + d(s_d^t) + d(s_u^t; s_d^t)]X_c^1 \\ &+ \sum_{l=1}^{L_u-1} X_u^l X_u^{l+1} + \sum_{l=1}^{L_d-1} X_d^l X_d^{l+1} + 3\sum_{l=1}^{L_c-1} X_c^l X_c^{l+1}\}), \end{split}$$

where B is the batch size decided in experiments. According to [46], the total computation complexity should be the complexity in one training step multiplied by the total steps used for convergence, which is hard to quantify. To better show the complexity, we illustrate the training times in one step and execution time (both agents in one iteration) of different algorithms under different scenarios in Fig. 5(h). We can observe from it that AAHC, iteRL, and CTRL all have similar training time and execution time in a single step. Although the hybrid Critic in AAHC needs to take in three different states and calculate three values, it needs only one back-propagation with the summed losses, while iteRL needs two back-propagation for two separate Critics. CTRL is slightly faster during training, but the difference is not obvious. As the execution stage is not related to the Critics, they are expected to have similar time in a single execution (evaluate) step. In practice, the training stage with huge computational complexity can be performed offline on the MC first. The training and execution are all conducted on a GTX 2080 Ti.

2) Metric performance across different configurations: As our model performance results are taken from a constantly improving RL model, we take the performance results of each RL model by evaluating the trained model on a new but



Fig. 6: Metrics with different numbers of XUs.

identical environment and averaging the results. The results are shown in Fig 6.

In general, the total delay, re-transmission times, and energy consumption all increase with the rise in the number of users. Nevertheless, as compared to the other baseline RL models, AAHC displays greater capability in finding a near-optimal solution in more complicated scenarios (i.e., configuration 3 - 8). This capability can be seen in the much slower increment in total delay and re-transmission counts for the AAHC, as the number of users increases, when compared to the iteRL and CTRL methods.

Nevertheless, as the number of XU rises, the training complexity of the problem rises precipitously. When compared to AAHC, CTRL and iteRL methods fail to find satisfactory solutions in the 3-7, 3-8 scenarios as the total delays are 2 or 3 times that of AAHC. Furthermore, according to the error bars drawn in Fig. 6, AAHC also has the lowest variance under different random seeds. Therefore, we infer from the results that AAHC is more efficient and stable than CTRL and



(b) 3-8 heat map after 2 million training steps with AAHC.

Fig. 7: 3-8 heat maps with AAHC.

iteRL, especially in more complicated scenarios. AAHC learns a conservative policy of restraining the UL rate to avoid retransmission, reducing the total time delay. This demonstrates that our proposed methods enable the agents to have a global view through the shared losses, instead of directly sharing the extra state or action information from each other. This is not unexpected because in such a hybrid-reward scenario, giving the agent extra information from the other agent is not always advisable. In this scenario, the energy consumption in each DL can just be influenced by the actions of the downlink agent, and it is not related to UL agent. Thus, the information of DL energy consumption is redundant to UL agent, and it will even impinge on the action evaluation of UL agent.

To better visualize the improvement during training, we sample trajectories from 1 evaluation episode from each of two pre-trained AAHC models with identical initial settings. The two AAHC models differ in that one of the pre-trained models has been pre-trained for 2 million iterations, while the other has undergone disrupted training in the early iterations. As

Scenario	Number of	Total	Re-trans	Max uplink	Energy		
	iterations	delay (ms)	rate (%)	rate (Gbps)	cost (J)		
AAHC							
3-4	3.5	3.7	0.52	17.08	0.07		
3-5	2.9	3.2	0.13	18.33	0.03		
3-6	12.1	11.8	1.43	7.63	0.08		
3-7	14.3	16.1	1.01	6.95	0.17		
3-8	31.1	34.9	1.86	3.23	0.19		
iteRL							
3-4	11.9	11.5	1.91	8.36	0.09		
3-5	21.9	23.9	4.30	12.46	0.14		
3-6	23.2	30.1	4.44	2.75	0.12		
3-7	42.8	54.3	8.49	1.94	0.23		
3-8	83.9	101.1	20.26	0.52	0.44		
CTRL							
3-4	9.9	12.0	2.11	8.77	0.04		
3-5	25.2	29.0	3.92	5.73	0.14		
3-6	29.2	33.6	3.55	2.38	0.15		
3-7	38.7	43.9	2.21	1.88	0.11		
3-8	69.2	86.7	4.54	0.44	0.20		
random							
3-4	73.5	114.9	29.32	3.71	0.22		
3-5	81.4	129.2	32.94	4.32	0.28		
3-6	92.6	176.0	40.13	1.96	0.42		
3-7	94.7	183.7	44.23	1.01	0.57		
3-8	97.2	180.1	49.33	0.32	1.09		

TABLE II: Overall results

shown in Fig. 7, the y-axis *channel-user* denotes the allocation of users to the channel, with a sizeable node referring that a particular user having been assigned to the specified channel. Nodes with larger sizes and deeper colors denote that the DL agent allocates more power for the DL transmission to the specified user in a particular channel. The x-axis indicates the number of training iterations.

In the initial stages of training (i.e., 50000 iterations), as illustrated in Fig. 7(a), the policy is poorly trained and is unable to select good actions under states. It is unsurprising, as agents lack exploration in the early stages of training. Coupled with being faced with a complex scenario, the agent struggles to make optimal decisions under various states. As a consequence, the actions chosen by the agent lacked pliability and variability when faced with different states. Evidently, the lack of variability in chosen actions by the agent is inconsistent with the fact that the environment is constantly evolving, in which channel gains and XUs' locations are changing in each time step. Furthermore, the 50,000 iterations pre-trained model exhibit in-efficiency by continuously allocating power to XUs that are arranged with no channel (channel 0). The 2 million iterations pre-trained model performed much better, as shown in Fig. 7(b). The user-channel arrangement is much more dispersive, and the decision-makings by the two agents exhibit much more variability in response to granular changes in the state. Compared to the 50,000 iterations pre-trained model, which uses 63 iterations to finish the whole task, the 2 million iterations pre-trained model needs only 25 iterations.

VII. CONCLUSION

In this work, we have investigated an asynchronous hybridreward joint optimization problem, the real-time 3D reconstruction in xURLLC over wireless communication with multiple XR users, where the uplink and downlink are considered in tandem. We formulate the problem as an asynchronous multi-agent reinforcement learning task and propose the novel AAHC algorithm. Multiple KPIs such as the total delay, energy consumption, and re-transmission percentage are studied to fulfill the reliability and low latency of our communication system. And extensive experiments demonstrate that AAHC has more granular views on each agent and performs better in asynchronous and hybrid tasks with a preferable training time. We hope this work can provide more insights into the asynchronous cooperative tasks, as they are common in communication problems, and important to guarantee the reliability of the whole system.

REFERENCES

- G. S. Research, "Framing the future of Web 3.0: Metaverse edition," Dec 2021. [Online]. Available: https://www.goldmansachs.com/insights/ pages/framing-the-future-of-web-3.0-metaverse-edition.html
- M. M. David Grider, "The Metaverse: Web 3.0 virtual cloud economies." [Online]. Available: https://grayscale.com/wp-content/uploads/2021/11/ Grayscale_Metaverse_Report_Nov2021.pdf
- [3] L.-H. Lee, T. Braud, P. Zhou, L. Wang, D. Xu, Z. Lin, A. Kumar, C. Bermejo, and P. Hui, "All one needs to know about Metaverse: A complete survey on technological singularity, virtual ecosystem, and research agenda," arXiv preprint arXiv:2110.05352, 2021.
- [4] X.-F. Han, H. Laga, and M. Bennamoun, "Image-based 3D object reconstruction: State-of-the-art and trends in the deep learning era," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 43, no. 5, pp. 1578–1604, 2019.
- [5] J. Sun, Y. Xie, L. Chen, X. Zhou, and H. Bao, "Neuralrecon: Real-time coherent 3D reconstruction from monocular video," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 15 598–15 607.
- [6] Z. Meng, C. She, G. Zhao, and D. De Martini, "Sampling, communication, and prediction co-design for synchronizing the real-world device and digital model in Metaverse," *IEEE Journal on Selected Areas in Communications*, vol. 41, no. 1, pp. 288–300, 2022.
- [7] "Meta quest 2: Our most advanced new all-in-one VR headset." [Online]. Available: https://store.facebook.com/quest/products/quest-2/
- [8] L. Lin, X. Liao, H. Jin, and P. Li, "Computation offloading toward edge computing," *Proceedings of the IEEE*, 2019.
- [9] M. Dvorožňák, D. Sýkora, C. Curtis, B. Curless, O. Sorkine-Hornung, and D. Salesin, "Monster Mash: a single-view approach to casual 3D modeling and animation," ACM Transactions on Graphics (TOG), 2020.
- [10] P. Yang, Y. Xiao, M. Xiao, and S. Li, "6G wireless communications: Vision and potential techniques," *IEEE Network*, 2019.
- [11] H. Van Seijen, M. Fatemi, J. Romoff, R. Laroche, T. Barnes, and J. Tsang, "Hybrid reward architecture for reinforcement learning," *Advances in Neural Information Processing Systems*, vol. 30, 2017.
- [12] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.
- [13] K. B. Letaief, W. Chen, Y. Shi, J. Zhang, and Y.-J. A. Zhang, "The roadmap to 6G: AI empowered wireless networks," *IEEE Communications Magazine*, vol. 57, no. 8, pp. 84–90, 2019.
- [14] I. F. Akyildiz and H. Guo, "Wireless extended reality (XR): Challenges and new research directions," *ITU J. Future Evol. Technol*, vol. 3, pp. 1–15, 2022.
- [15] I. Parvez, A. Rahmati, I. Guvenc, A. I. Sarwat, and H. Dai, "A survey on low latency towards 5G: RAN, core network and caching solutions," *IEEE Communications Surveys & Tutorials*, 2018.
- [16] S. Zhang, W. Y. B. Lim, W. C. Ng, Z. Xiong, D. Niyato, X. S. Shen, and C. Miao, "Towards Green Metaverse Networking: Technologies, Advancements and Future Directions," *IEEE Network*, 2023.
- [17] J. G. Andrews, "Interference cancellation for cellular systems: a contemporary overview," *IEEE Wireless Communications*, 2005.
- [18] Y. Sun, X. Guo, J. Song, S. Zhou, Z. Jiang, X. Liu, and Z. Niu, "Adaptive learning-based task offloading for vehicular edge computing systems," *IEEE Transactions on vehicular technology*, 2019.
- [19] Z. Wang, L. Li, Y. Xu, H. Tian, and S. Cui, "Handover control in wireless systems via asynchronous multiuser deep reinforcement learning," *IEEE Internet of Things Journal*, 2018.
- [20] C. Kai, H. Zhou, Y. Yi, and W. Huang, "Collaborative cloud-edge-end task offloading in mobile-edge computing networks with limited communication capability," *IEEE Transactions on Cognitive Communications and Networking*, vol. 7, no. 2, pp. 624–634, 2020.
 [21] Z. Xiao, X. Dai, H. Jiang, D. Wang, H. Chen, L. Yang, and F. Zeng,
- [21] Z. Xiao, X. Dai, H. Jiang, D. Wang, H. Chen, L. Yang, and F. Zeng, "Vehicular task offloading via heat-aware mec cooperation using gametheoretic method," *IEEE Internet of Things Journal*, 2019.

- [22] T. Alfakih, M. M. Hassan, A. Gumaei, C. Savaglio, and G. Fortino, "Task offloading and resource allocation for mobile edge computing by deep reinforcement learning based on sarsa," *IEEE Access*, 2020.
- [23] L. Qian, P. Yang, M. Xiao, O. A. Dobre, M. Di Renzo, J. Li, Z. Han, Q. Yi, and J. Zhao, "Distributed learning for wireless communications: Methods, applications and challenges," *IEEE Journal of Selected Topics* in Signal Processing, vol. 16, no. 3, pp. 326–342, 2022.
- [24] M. Merluzzi, P. Di Lorenzo, S. Barbarossa, and V. Frascolla, "Dynamic computation offloading in multi-access edge computing via ultra-reliable and low-latency communications," *IEEE Transactions on Signal and Information Processing over Networks*, 2020.
- [25] C.-F. Liu, M. Bennis, M. Debbah, and H. V. Poor, "Dynamic task offloading and resource allocation for ultra-reliable low-latency edge computing," *IEEE Transactions on Communications*, 2019.
- [26] D. Guo, L. Tang, X. Zhang, and Y.-C. Liang, "Joint optimization of handover control and power allocation based on multi-agent deep reinforcement learning," *IEEE Transactions on Vehicular Technology*, 2020.
- [27] C. He, Y. Hu, Y. Chen, and B. Zeng, "Joint power allocation and channel assignment for noma with deep reinforcement learning," *IEEE Journal* on Selected Areas in Communications, 2019.
- [28] R. Lowe, Y. I. Wu, A. Tamar, J. Harb, O. Pieter Abbeel, and I. Mordatch, "Multi-agent actor-critic for mixed cooperative-competitive environments," Advances in Neural Information Processing Systems, 2017.
- [29] T. Rashid, M. Samvelyan, C. Schroeder, G. Farquhar, J. Foerster, and S. Whiteson, "Qmix: Monotonic value function factorisation for deep multi-agent reinforcement learning," in *International Conference on Machine Learning*. PMLR, 2018, pp. 4295–4304.
- [30] J. Foerster, G. Farquhar, T. Afouras, N. Nardelli, and S. Whiteson, "Counterfactual multi-agent policy gradients," in *Proceedings of the* AAAI Conference on Artificial Intelligence, vol. 32, no. 1, 2018.
- [31] X. Chen, "Decentralized computation offloading game for mobile cloud computing," *IEEE Transactions on Parallel and Distributed Systems*, vol. 26, no. 4, pp. 974–983, 2014.
- [32] O. Munoz, A. Pascual-Iserte, and J. Vidal, "Optimization of radio and computational resources for energy efficiency in latency-constrained application offloading," *IEEE Transactions on Vehicular Technology*, vol. 64, no. 10, pp. 4738–4755, 2014.
- [33] Z. Ding, X. Lei, G. K. Karagiannidis, R. Schober, J. Yuan, and V. K. Bhargava, "A survey on non-orthogonal multiple access for 5G networks: Research challenges and future trends," *IEEE Journal on Selected Areas in Communications*, vol. 35, no. 10, pp. 2181–2195, 2017.
- [34] L. Dai, B. Wang, Z. Ding, Z. Wang, S. Chen, and L. Hanzo, "A survey of non-orthogonal multiple access for 5G," *IEEE Communications Surveys* & *Tutorials*, vol. 20, no. 3, pp. 2294–2323, 2018.
- [35] L. Liberti, "Undecidability and hardness in mixed-integer nonlinear programming," *RAIRO-Operations Research*, 2019.
- [36] N. Justesen, P. Bontrager, J. Togelius, and S. Risi, "Deep learning for video game playing," *IEEE Transactions on Games*, 2020.
- [37] X. Li, Q. Wang, J. Liu, and W. Zhang, "Trajectory design and generalization for uav enabled networks: A deep reinforcement learning approach," in *IEEE Wireless Communications and Networking Conference*, 2020.
- [38] A. Owen and Y. Zhou, "Safe and effective importance sampling," *Journal of the American Statistical Association*, 2000.
- [39] J. Schulman, P. Moritz, S. Levine, M. Jordan, and P. Abbeel, "Highdimensional continuous control using generalized advantage estimation," arXiv preprint arXiv:1506.02438, 2015.
- [40] R. S. Sutton, D. McAllester, S. Singh, and Y. Mansour, "Policy gradient methods for reinforcement learning with function approximation," *Advances in Neural Information Processing Systems*, vol. 12, 1999.
- [41] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing atari with deep reinforcement learning," arXiv preprint arXiv:1312.5602, 2013.
- [42] 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," arXiv preprint arXiv:1509.02971, 2015.
- [43] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," in *International Conference on Machine Learning*. PMLR, 2016, pp. 1928–1937.
- [44] C. She, C. Yang, and T. Q. Quek, "Joint uplink and downlink resource configuration for ultra-reliable and low-latency communications," *IEEE Transactions on Communications*, vol. 66, no. 5, pp. 2266–2280, 2018.
- [45] I. Yildirim, A. Uyrus, and E. Basar, "Modeling and analysis of reconfigurable intelligent surfaces for indoor and outdoor applications in future wireless networks," *IEEE Transactions on Communications*, vol. 69, no. 2, pp. 1290–1301, 2021.
- [46] C. Jin, Z. Allen-Zhu, S. Bubeck, and M. I. Jordan, "Is Q-learning provably efficient?" Advances in Neural Information Processing Systems, vol. 31, 2018.

Appendix A

IMPLEMENTATION DETAILS

For all these experiments, Adam is used as the optimization algorithm. The discount factor γ and GAE factor λ are fixed at 0.99 and 0.95. The batch size is set to 64. The learning rates for uplink Actor, downlink Actor, and hybrid Critic are 10^{-4} , 10^{-4} , 5×10^{-5} , respectively. The entropy coefficient is set as 10^{-3} . As the essential objective of RL is to maximize the expected return (reward), the reward setting is always crucial. We highly recommend standardizing the values across kinds of rewards, or the parameter tuning will become a daunting task, and the large losses will make it slow to converge.

In terms of the activation functions, we recommend trying to use tanh instead of ReLU first, especially in the simpler network (in fact, networks in this kind of scenario where there is only deep neural network (DNN) but not convolutional neural network (CNN) are always much simpler than those in computer vision domains). Because tanh is zero-centered. Hence we can easily map the output values as strongly negative, neutral, or strongly positive. In other words, this reduces the difficulty of reward settings. However, tanh in hidden layers faces the problem of vanishing gradient. Thus, we should be careful when using this.

APPENDIX B

ADDITIONAL EXPERIMENTS

We have provided selected experimental results (metrics under the 3-8 scenario during training) in Section VI-D. Additional experimental results are presented in Figure 8 below and in Figure 9 on the next page.



Fig. 8: Re-transmission percentage in 3-4 to 3-7 scenarios.



Fig. 9: Total iterations, rewards, UL rate, energy consumption in 3-4 to 3-8 scenarios during training.