Multi-Task Decision-Making for Multi-User 360° Video Processing over Wireless Networks

Babak Badnava*, Jacob Chakareski†, Morteza Hashemi*
* Department of Electrical Engineering and Computer Science, University of Kansas
† College of Computing, New Jersey Institute of Technology
Emails: {babak.badnava,mhashemi}@ku.edu, jacobcha@njit.edu

***Abstract*—We study a multi-task decision-making problem f 360° video processing in a wireless multi-user virtual reality (V system that includes an edge computing unit (ECU) to deliv 360° videos to VR users and offer computing assistance for deco ing/rendering of video frames. However, this comes at the exper of increased data volume and required bandwidth. To balan this trade-off, we formulate a constrained quality of experien (QoE) maximization problem in which the rebuffering time a quality variation between video frames are bounded by us and video requirements. To solve the formulated multi-user Q maximization, we leverage deep reinforcement learning (DRL) f multi-task rate adaptation and computation distribution (MTR The proposed MTRC approach does not rely on any predefin assumption about the environment and relies on video playba statistics (*i.e.,* past throughput, decoding time, transmission time, etc.), video information, and the resulting performance to adjust the video bitrate and computation distribution. We train MTRC with real-world wireless network traces and 360° video datasets to obtain evaluation results in terms of the average QoE, peak signal-to-noise ratio (PSNR), rebuffering time, and quality variation. Our results indicate that the MTRC improves the users' QoE compared to state-of-the-art rate adaptation algorithm. Specifically, we show a 5.97 dB to 6.44 dB improvement in PSNR, a 1.66X to 4.23X improvement in rebuffering time, and a 4.21 dB to 4.35 dB improvement in quality variation.**

***Index Terms*—Quality of experience, wireless networks, 360° video processing, edge computing, mobile VR systems.**

I. Introduction

Next-generation wireless networks (6G and beyond) will enable new use cases and applications that demand significantly higher computational power and bandwidth. Augmented reality (AR), virtual reality (VR), and extended reality (XR) are examples of such applications [1–6]. For instance, VR applications capture entire spherical scenes and stream high-fidelity 360° video content to create an immersive experience for users. This process demands substantial computational and communication resources. Unlike traditional 2D video streaming, 360° video streaming requires additional computational power for encoding, decoding, spatial processing, stitching, and rendering [7, 8]. In fact, 360° video decoding entails spatio-temporal transformations for spherical projection. Viewport-adaptive streaming further increases the computational complexity by dynamically adjusting video segments based on the viewer's field of view (FoV). Moreover, 360° videos typically have higher resolution and larger file sizes compared to 2D videos, leading to higher bandwidth requirements. To satisfy these requirements, multi-access edge computing (MEC) and high-bandwidth mmWave wireless networks have been proposed in prior works [7, 9–11].

Numerous studies have focused on maximization of QoE for 2D video streaming (see, for example, [12–18]). On the other hand, due to the increased computational and communication requirements for 360° video streaming, it is necessary to develop joint resource allocation to ensure a satisfactory QoE for VR users. Such a joint optimization problem should incorporate several factors including (i) available communication data rates provided by the underlying wireless network, (ii) computational resources provided by the VR headset and/or MEC unit, and (iii) spatio-temporal characteristics of 360° videos.

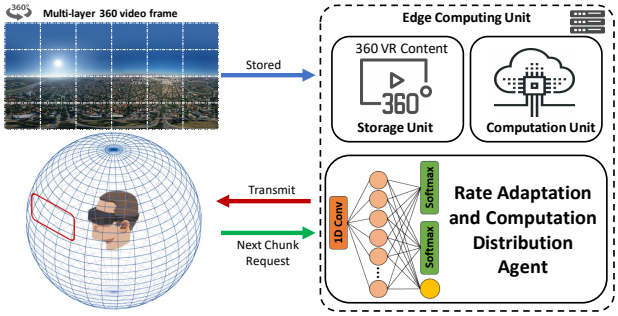


Fig. 1: Edge-assisted VR system model: multiple VR headset connected to an edge computing unit through a mmWave network

Extensive amounts of prior work have focused on different aspects of VR systems. For example, the authors in [7] propose a dual connectivity streaming system in which mmWave and Wi-Fi links are integrated to enable six degrees of freedom VR-based remote scene immersion. Furthermore, the authors in [19] propose a FoV prediction algorithm so that the predicted view is encoded in relatively high quality and transmitted in advance, which reduces the latency. Hsu in [9] investigates the optimization of caching and computing at the edge server to improve the QoE of users. Gupta et al. in [10] leverage an edge server for decoding to maximize the smallest immersion fidelity for the delivered 360° content across all VR users. The authors in [11] present an MEC computing framework to optimize energy efficiency and processing delay for AR applications. Moreover, [20] investigates the rate-distortion characteristics of ultra-high definition (UHD) 360° videos.

Despite extensive research on VR systems, to the best of our knowledge, no prior work has considered multi-task video processing for multi-user wireless VR systems. Therefore, in this paper we formulate a multi-task edge-assisted video quality maximization framework for 360° video streaming on wireless networks as depicted in Fig. 1. In this framework, the computational tasks (*i.e.,* decoding and rendering) necessary for processing 360° videos may be executed either by an ECU or by the users' headsets themselves. On one hand, the ECU has more computational resources to process the 360° videos

faster, which leads to lower computational latency and h QoE for users. However, decoding and rendering by the introduces a higher bandwidth requirement since proc videos have much larger sizes. Therefore, this leads to h communication latency, thus degrading the QoE. Further the ECU provides its best performance for a certain numb users due to limited computational resources.

To capture these trade-offs and jointly optimize co nication and computation resource allocation, we pres novel learning-based decision-making algorithm, called M which considers the interaction between the communi and computation requirements of 360° videos. Learning- methods, particularly DRL, do not depend on pred models or system assumptions [21, 22]. Rather, they learn to make decisions exclusively by analyzing the outcomes of previous decisions and adapting accordingly. MTRC leverages a state-of-the-art DRL method to learn the optimal computation distribution (*i.e.,* ECU or headset) and the video bitrate in the VR arena by considering the playback statistics and video information. In summary, the main contributions of this paper are as follows:

- We consider an edge-assisted wireless VR streaming system in which an ECU provides 360° videos to VR users. We formulate a constrained video quality maximization problem, in which rebuffering time and video quality variation are bounded by user and video requirements, to find the best policy w.r.t network condition and spatio-temporal characteristics of multi-layer 360° videos.
- We develop a multi-task learning-based algorithm to find the optimal computation distribution and video bitrate for VR users to maximize their QoE. Our MTRC agent observes the playback statistics (*i.e.,* past throughput, decoding/transmission time, etc.) and video information, then decides the optimal bitrate and computation distribution to decode/render the 360° video.
- We develop a 360° VR streaming simulator using a real 360° video dataset, real-world VR user navigation information, and real-world mmWave network traces. Then, we perform an extensive simulation to analyze the behavior of our proposed method. We show that the proposed MTRC algorithm improves the PSNR by 5.97 dB to 6.44 dB, rebuffering time by 1.66X to 4.23X, and quality variation by 4.21 dB to 4.35 dB.

II. System Model

We consider a multi-user 360° VR video streaming application with N users, each connected to the ECU via a wireless access point. As depicted in Fig. 1, the users are equipped with VR headsets and request 360° videos that are stored on the ECU. The videos need to be processed (*i.e.,* decoded/rendered) and transmitted to VR headset before users can play them. Upon receiving a request for a video segment, a decision-making agent (*i.e.,* MTRC agent) makes a joint decision on the video bitrate allocated to each user and the computation distribution (*i.e.,* decoding and rendering on the ECU or on the headset). Then, computational resources will be allocated to process each video segment.

The ECU and all VR headsets are equipped with computational resources (*i.e.,* CPU and GPU) to process the video

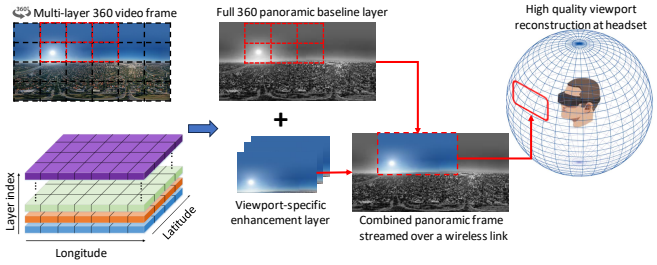


Fig. 2: Viewport-specific enhancement layers and a baseline layer are transmitted to a VR headset via a wireless link to be reconstructed.

segments. If the decision-making agent decides to process the video segment on the ECU, the ECU's computational resources are shared among users to process the video segments and then transmit the video segments to the users. However, if the decision-making agent decides to process the video segment on users' headset, the ECU sends the video segments to users and the preparation takes place at the headsets. Next, we present the models on multi-layer 360° videos, user headsets, and ECU.

Multi-layer 360° Video Model: We consider the scalable multi-layer 360° video viewpoint tiling design [7]. As depicted in Fig. 2, each panoramic 360° video frame is partitioned into $\mathscr{L}$ tiles arranged in a $L_H \times L_V$ grid. A block of consecutive video frames, compressed together with no reference to others, creates a group of pictures (GoP) or video segment. Each video is divided into M GoP with fixed time duration of Δt, and L layers of increased immersion fidelity for each tile in a GoP exist. The first layer is called the base layer, and the remaining layers are denoted as enhancement layers.

Each enhancement layer increases the video bitrate, hence the video quality. We denote $e_n^m \in \{0,1\}^L$ as a one-hot vector that determines the number of enhancement layers included in the m^{th} GoP requested by the n^{th} user. Then, the size of the m^{th} compressed GoP tile, denoted by $d(e_n^m)$, is determined by summing over all tiles' bitrates in the user's viewport. We assume a positive compression reduction factor of $\beta < 1$, which leads to $d(e_n^m)/\beta$ for the size of the m^{th} decoded GoP tile. After decoding, GoPs need to be rendered as well, which leads to an increase in size by a factor of $\alpha \geq 2$. Hence, the size of the m^{th} GoP after decoding and rendering is determined by $\alpha d(e_n^m)/\beta$.

Video Retrieval and Buffering: The VR headsets retrieve and store rendered videos in a buffer with a fixed duration time. Fig. 3 illustrates the buffer dynamics of the 360° video streaming application. The GoPs need to be processed in order to be buffered on headsets. At time t_n^m, the n^{th} user requests the m^{th} GoP. Then, the GoP will be processed (*i.e.,* either on the ECU or the headset) and buffered on headsets. The preparation of m^{th} GoP involves three key stages: the decoding time D_n^m, rendering time P_n^m, and transmission time T_n^m. Once the GoP is processed and buffered, the user waits for Δ_n^m seconds until requesting the next GoP. Thus, the next request time is: $t_n^{m+1} = t_n^m + D_n^m + P_n^m + T_n^m + \Delta_n^m$. The buffer occupancy evolves as GoPs are being prepared, and the video is being played by the user. The buffer occupancy of user n increases by Δt seconds after receiving GoP m. Let $B_n^m = B_n(t_n^m)$ denote the buffer occupancy of the n^{th} user at t_n^m. Then, we have:

$$B_n^{m+1} = B_n(t_n^{m+1}) = \left((B_n^m - P_n^m - D_n^m - T_n^m)_+ + \Delta t - \Delta_n^m\right)_+.$$

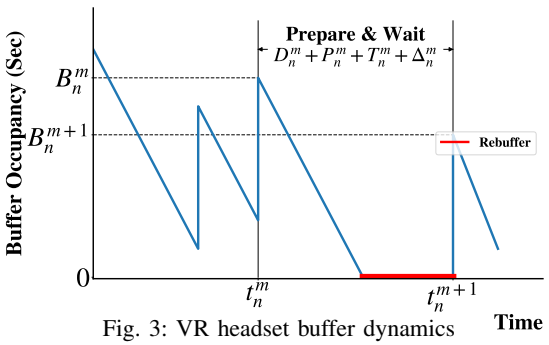


Fig. 3: VR headset buffer dynamics

Here, the notation $(x)_+ = \max\{0,x\}$ ensures that the buffer occupancy is non-negative. If the process and transmission times take longer than the amount of GoP stored in the buffer (*i.e.,* $B_n^m < P_n^m + D_n^m + T_n^m$), then *rebuffering* happens as shown in Fig. 3. We also assume that the waiting time Δ_n^m is zero, except when the buffer is full, which the headset waits until the buffer has enough space to accommodate the next GoP, which leads to: $\Delta_n^m = \left((B_n^m - P_n^m - D_n^m - T_n^m)_+ + \Delta t - B_n^{max}\right)_+$.

VR Headset Decoding and Rendering Model: The VR headsets are equipped with a CPU and GPU to process the videos, and each VR headset provides a maximum decoding speed of $\bar{Z}_n^{dec.}$, a maximum rendering speed of $\bar{Z}_n^{rend.}$, where n is the index of the VR headset. We incorporate a decoding and rendering model developed by [3] to compute the decoding and rendering time for each GoP. In this model, the decoding time of the m^{th} GoP for the n^{th} user is assessed as $\tilde{si}(e_n^m)/\bar{Z}_{dec.}^n$, where $\tilde{si}(.)$ returns the computational complexity of decoding a GoP (in bits), which is the induced data rate associated with the current viewport *i.e.,* $si(e_n^m) = d(e_n^m)$. Similarly, the rendering time is modeled as $si(e_n^m)/\bar{Z}_n^{rend.}$, where $si(.)$ returns the rendering computational complexity of a GoP (in bits), which is the induced data rate after decoding of the GoP $si(e_n^m) = d(e_n^m)/\beta$. Note that we assume that the viewport information is available on the headset.

Edge Computing Unit (ECU) Model: The ECU provides additional computational resources to assist VR users with decoding and rendering. This additional computational resources provide a maximum decoding speed of $Z_{ECU}^{dec.}$ and a maximum rendering speed of $Z_{ECU}^{rend.}$, which is shared among the users that decode/render their GoP on the ECU. We assume that the decision-making time is negligible since the decoding and rendering tasks are dominant overheads. Then, we incorporate a similar computation model on the ECU.

The decoding starts immediately after receiving the request for the next GoP if the decision-maker decides to decode the GoP on the ECU. This leads to $\tilde{si}(e_n^m)/\Psi_n^m$ seconds of decoding time. Here, Ψ_n^m denotes the amount of decoding resources, out of $Z_{ECU}^{dec.}$, allocated to the n^{th} user to decode the m^{th} GoP. Similarly, the ECU rendering time is modeled as $si(e_n^m)/\Theta_n^m$, where Θ_n^m denotes the amount of rendering resources, out of $Z_{ECU}^{rend.}$, allocated to the n^{th} user for the m^{th} GoP. Note that the total amount of computational resources allocated to users cannot exceed the maximum available resources, which means that $\sum_{n=1}^N \Psi_n^m \leq Z_{ECU}^{dec.}$ and $\sum_{n=1}^N \Theta_n^m \leq Z_{ECU}^{rend.}$ must be satisfied for each GoP.

Wireless Links Model: The ECU transmits GoPs through a mmWave wireless network. The expected transmission rate for a GoP is modeled as $C_n^m = \frac{1}{t_e - t_s}\int_{t_s}^{t_e} C_s^n ds$ where, t_s and t_e are transmission start and end times, respectively, and C_n^s is the throughput provided by the wireless channel for the n^{th} user. Hence, the transmission time for a compressed GoP follows $d(e_n^m)/C_n^m$. Similarly, we can model the transmission time for decoded and rendered GoPs.

III. Problem Formulation

Given the presented system model, we aim to improve QoE for multi-user 360° video streaming. To this end, we consider three major factors that impact QoE. The first factor is the Average Video Quality (AVQ) defined as the average per-GoP video quality for tiles in the user's FoV, expressed as: $Q(e_n) = \frac{1}{M}\sum_{m=1}^M q(e_n^m)$. While there are various choices for $q(.)$ [12], we use PSNR for the viewer's FoV [23], which can be calculated using the video distortion [20] as $q(e_n^m) = 10\log_{10}(255^2/MSE_n^m)$, where MSE_n^m is the distortion of the m^{th} GoP. The distortion has an inverse relation with the video bitrate, which is determined by the number of enhancement layers streamed to the user [20].

The second factor that impacts perceived QoE is the average quality variation (AQV) that captures quality variation in the user's FoV from one GoP to another. Therefore, we have $V(e_n) = \frac{1}{M-1}\sum_{m=1}^{M-1}\left|q(e_n^{m+1}) - q(e_n^m)\right|$. The third factor that affects QoE is rebuffering. Rebuffering occurs if the process time of a GoP is larger than the buffer occupancy level when the GoP was requested. Thus, the total rebuffering time (RT) is given by: $S(e_n,\phi_n) = \sum_{m=1}^M (D_n^m + P_n^m + T_n^m - B_n^m)_+$. Several previous studies [12, 13, 24] have defined the n^{th} user's QoE of GoP 1 through M by a weighted sum of the aforementioned components:

$$\hat{QoE}_M^n = Q(e_n) - \mu_0 S(e_n,\phi_n) - \mu_1 V(e_n), \tag{1}$$

where, μ_0 and μ_1 are non-negative weighting parameters corresponding to user sensitivity to rebuffering time and quality variation, respectively. Although this QoE metric allows us to model varying user preferences [12, 25], setting the values of μ_0 and μ_1 for various users' requirements is not straightforward. To resolve this issue, we will focus on optimizing the average video quality with constrained rebuffering time and average quality variations.

Multi-Task QoE Maximization Problem: To formulate the problem of multi-task QoE maximization, we define two sets of communication and computation decision variables. In particular, $\phi_n^m \in \{0,1\}^3$ is a binary vector of size three with one active element (*i.e.,* a one-hot vector), which determines where the decoding and rendering take place for each GoP and user. There are three states for ϕ_n^m as follows: (i) $\phi_{m,0}^n = 1$ corresponds to decoding and rendering on the ECU, (ii) $\phi_{m,1}^n = 1$ if we decode on ECU, but render on headset, and (iii) $\phi_{m,2}^n = 1$ implies that both decoding and rendering happen on headset.

In addition to the location of the computation, we consider the rate allocation decision variable $e^m \in \{0,1\}^{N\times L}$ that determines how many enhancement layers should be streamed to each user in the VR arena. In addition to these decision variables, ECU computation resources (*i.e.,* decoding resources ψ_n^m and rendering resources θ_n^m) should be allocated to the users, which we assume that they have been allocated proportional

to user's requirements. therefore, we formulate the following optimization problem:

$$\max_{\boldsymbol{\phi}_\mathbf{n},\mathbf{e}_\mathbf{n}} \quad Q(\mathbf{e_n}) \tag{2}$$

$$\text{s.t.} \quad S(\mathbf{e^n},\boldsymbol{\phi}^\mathbf{n}) \le \mathscr{H}_0 \tag{2a}$$

$$V(\mathbf{e^n}) \le \mathscr{H}_1 \tag{2b}$$

$$B_n^{m+1} = \left((B_n^m - P_n^m - D_n^m - T_n^m)_+ + \Delta t - \Delta_n^m\right)_+, \ \forall m \tag{2c}$$

$$\begin{aligned} t_n^{m+1} = {} & t_n^m \\ & + \phi_{m,1}^n \left[\frac{\tilde{si}(e_n^m)}{\Psi_n^m} + \frac{si(e_n^m)}{\Theta_n^m.} + \frac{\alpha d(e_n^m)/\beta}{C_n^m}\right] \\ & + \phi_{m,2}^n \left[\frac{\tilde{si}(e_n^m)}{\Psi_n^m} + \frac{si(e_n^m)}{Z_n^{rend.}} + \frac{d(e_n^m)/\beta}{C_n^m}\right] \\ & + \phi_{m,3}^n \left[\frac{\tilde{si}(e_n^m)}{Z_n^{dec.}} + \frac{si(e_n^m)}{Z_n^{rend.}} + \frac{d(e_n^m)}{C_n^m}\right] + \Delta_n^m, \ \forall m \end{aligned} \tag{2d}$$

$$\sum_{n=1}^{N} \psi_n^m \le Z_{ECU}^{dec.} \quad \forall m \quad , \quad \sum_{n=1}^{N} \theta_n^m \le Z_{ECU}^{rend.} \quad \forall m \tag{2e}$$

There are several key challenges that need to be addressed before solving the optimization problem outlined in Eq. 2. This is a constrained optimization problem, in which we maximize the average video quality such that the rebuffering time and average quality variation are constrained to a bound. Furthermore, the action space is an inter-dependent multi-task space. This implies that tasks (*i.e.,* rate allocation and computation distribution) put constraints on each other (*e.g.,* the maximum achievable rate is constrained by the rate adaptation decision). Another challenge arises from varying 360° video characteristics, which means the computational requirements may differ from one video to another. Finally, VR users experience time-varying and dynamic wireless network conditions (e.g., due to blockage [26], interference [27], etc.) that impact the streaming data rate. Thus, a decision-making algorithm that incorporates various dynamic and time-varying conditions in terms of video quality, wireless network, and available computational resources is desirable.

IV. Multi-Task Decision-Making Framework

To tackle the challenges mentioned above, we present the MTRC framework for multi-task rate adaptation and computation distribution algorithm. The MTRC solution leverages a state-of-the-art DRL method to learn the optimal rate adaptation and computation distribution policy, thereby maximizing QoE for all users. Next, we describe the proposed framework.

MTRC Agent: At each time step and after receiving the request for each GoP from all users, the DRL agent observes the state s^m of all users. The state input s^m includes *playback statistics*, and *video information*. The video playback statistics include six critical metrics for each user to fully describe the 360° video playback status. These metrics are past throughput, past decoding time, past transmission time, past rendering time, last allocated rate e_n^{m-1}, and current buffer level B_n^m. We take the future GoPs size and number of remaining GoPs for each user as video information. This will help our MTRC model to distinguish between videos with different spatio-temporal characteristics. All of these metrics are collected for all users and stacked together to represent the state information.

Once the state s^m is observed, the agent chooses an action a^m to decode, render, and transmit GoPs. In this case, the MTRC agent takes a joint action $a^m = (e^m, \phi^m)$. The rate allocation action $e^m \in \{0,1\}^{N\times L}$ determines how many enhancement layers will be streamed to each user. The computation distribution action $\phi^m \in \{0,1\}^{N\times 3}$ determines where each user decodes and/or renders the m^{th} GoP.

After taking the action, the state of the environment changes, and the agent receives a reward vector r^m that gives the reward for each user. The goal of our agent is to maximize the expected cumulative discounted reward for all users. However, the definition of the reward in such an unconstrained optimization problem is not straightforward. Hence, we modify the optimization problem outlined in Eq. 2 in order to capture the constraint violation and penalize the MTRC agent for not meeting the constraints. Leveraging the duality principle, we write the Lagrangian dual problem of Eq. 2:

$$\begin{aligned} \min_{\mu_0,\mu_1} \max_{\boldsymbol{\phi}_\mathbf{n},\mathbf{e}_\mathbf{n}} \ & \underbrace{Q(\mathbf{e_n}) + \mu_0\left(\mathscr{H}_0 - S(\mathbf{e_n},\boldsymbol{\phi}_\mathbf{n})\right) + \mu_1\left(\mathscr{H}_1 - V(\mathbf{e_n})\right)}_{:=QoE_n^M} \\ & \text{s.t. Eqs. } 2c, 2d, 2e, \end{aligned} \tag{3}$$

which is equal to:

$$\begin{aligned} \min_{\mu_0,\mu_1} \quad & Q(\mathbf{e_n^*}) + \mu_0\left(\mathscr{H}_0 - S(\mathbf{e_n^*},\boldsymbol{\phi}_\mathbf{n}^*)\right) + \mu_1\left(\mathscr{H}_1 - V(\mathbf{e_n^*})\right) \\ \text{s.t.} \quad & \text{Eqs. } 2c, 2d, 2e, \end{aligned} \tag{4}$$

where $\mathbf{e_n^*}$ and $\boldsymbol{\phi}_\mathbf{n}^*$ are optimal rate allocation and computation distribution actions for user n. Then, the optimal coefficients for rebuffering time and quality variation, which correspond to Lagrangian multipliers, can be obtained by solving Eq. 3:

$$\begin{aligned} \mu_0^* &= \arg\min_{\mu_0} \ \mu_0\left(\mathscr{H}_0 - S(\mathbf{e_n^*},\boldsymbol{\phi}_\mathbf{n}^*)\right), \\ \mu_1^* &= \arg\min_{\mu_1} \ \mu_1\left(\mathscr{H}_1 - V(\mathbf{e_n^*})\right). \end{aligned} \tag{5}$$

Here, by minimizing the loss functions presented in Eq. 5, μ_0 and μ_1 are updated to increase (or decrease) if the rebuffering time or quality variation is higher (or lower) than the target rebuffering time and quality variation $\mathscr{H}_0$ and $\mathscr{H}_1$, respectively. This modification in the optimization problem effectively handles the reward magnitude change over time during training. Hence, one needs to set only the target rebuffering time $\mathscr{H}_0$ and the target quality variation $\mathscr{H}_1$ for each user and video, and then the coefficients μ_0 and μ_1 are automatically adjusted over time to meet the target constraints.

Now that the objective function in Eq. 3 accounts for both rebuffering time and quality variation, we employ the change in users' perceived QoE at each step as the reward term. This is defined as $r_n^{m+1} = QoE_n^{m+1} - QoE_n^m$. This reward captures the changes in the perceived QoE as a result of the last action taken by the MTRC agent, which enables the agent to learn the actions that lead to improvement in QoE.

DRL Implementation: We develop a sample-efficient multi-task DRL algorithm to tackle the complexity of the defined multi-task QoE maximization problem. The MTRC agent is composed of an actor network ω and a critic network ω_v. The actor network outputs the probabilities of both rate allocation action π_ω^e and computation distribution action π_ω^ϕ for

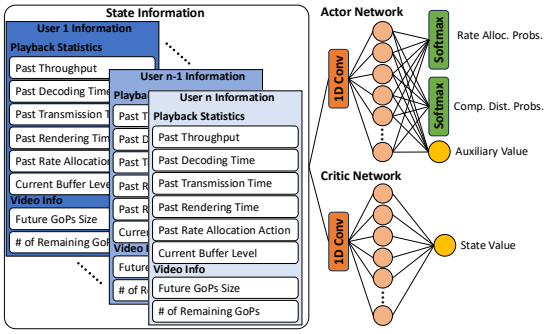


Fig. 4: MTRC Architecture

all users. The actor network also outputs an auxiliary vector that estimates the state value for each user separately. The critic network outputs the estimated state value for each user separately. Inspired by [28, 29], we employ a two-phase training procedure, composed of a policy training phase and an auxiliary training phase [30]. In the policy training phase, the actor and critic networks are trained by Dual-Clip PPO [31]:

$$\mathscr{L}^{DClip} = \hat{\mathbb{E}}\left[\mathbb{1}(\hat{A}^m < 0)\max(\mathscr{L}^{PPO}, c\hat{A}^m) + \mathbb{1}(\hat{A}^m \geq 0)\mathscr{L}^{PPO}\right]. \quad (6)$$

Here, $\mathbb{1}()$ denotes a binary indicator function, and $\mathscr{L}^{PPO}$ represents the surrogate vanilla PPO loss that is:

$$\mathscr{L}^{PPO} = \mathscr{L}^{Clip}(\pi_\omega, \hat{A}^m) + \beta H_\omega(s^m) + \mathscr{L}^{Value}, \quad (7)$$

where $H(s^m)$ is the entropy of all policies, and β is the entropy weight, which jointly balance the tradeoff between exploration and exploitation during the learning process. $\mathscr{L}^{Value}$ is the value network loss [30], and $\mathscr{L}^{Clip}$ is the Single-Clip policy loss:

$$\mathscr{L}^{Clip} = \min\left[\rho(\pi_\omega, \pi_{\omega_{old}}) \odot \hat{A}^m, clip(\rho(\pi_\omega, \pi_{\omega_{old}}), 1 \pm \varepsilon) \odot \hat{A}^m\right]. \quad (8)$$

Here, $\hat{A}^m = r_m + \gamma V_{\omega_v}(s^{m+1}) - V_{\omega_v}(s^m)$ is the advantage function that is calculated based on the current state-value estimate and the discount factor $\gamma = 0.99$, and $\odot$ is the element-wise multiplication. $\rho(\pi_\omega, \pi_{\omega_{old}}) = [\rho_1, \rho_2, \cdots, \rho_N]$ is a vector of size N that measures the changes in the new policy w.r.t. the old policy for all users (*i.e.,* the joint probability ratio of the new policy to the old policy for the joint action):

$$\rho_i = \frac{\pi^{\phi}_{\omega}(\phi^m_i|s^m)}{\pi^{\phi}_{\omega_{old}}(\phi^m_i|s^m)} \frac{\pi^{e}_{\omega}(e^m_i|s^m)}{\pi^{e}_{\omega_{old}}(e^m_i|s^m)}.$$

In Eq. 8, $\rho(\pi_\omega, \pi_{\omega_{old}}) \odot \hat{A}^m$ measures the gained/lost reward as the policy for each user changes, determining how a change in one user's policy affects the reward of other users. In the auxiliary phase, we further optimize the actor and critic networks according to a joint objective function $\mathscr{L}^{Joint}$, which is composed of a behavioral cloning loss and an auxiliary value loss. Due to space limitation, details of the auxiliary training phase are omitted. Interested readers can refer to [30].

Algorithm 1 presents the training process of MTRC agent, which continues for multiple iterations until convergence. Each iteration is composed of four phases. In the first phase, we perform the current policy π_ω on a randomized environment to collect new experiences (*i.e.,* rollout process). We encapsulated our edge-assisted VR system into a gym-like environment, which allows the MTRC agent to interact with the system effectively. The MTRC agent learns the policy through its

Algorithm 1 MTRC Training Process

```
for epoch = 1,2,... do
    Perform rollout under current policy π
    for i = 1,2,...,N_Policy do
        Optimize Eq. 6 (i.e., L^DClip) w.r.t. ω, ω_v
    end for
    for i = 1,2,...,N_aux do
        Optimize L^Joint w.r.t. ω
        Optimize L^Value w.r.t. ω_v
    end for
    Update μ_0 and μ_1 according to Eq. 5
end for
```

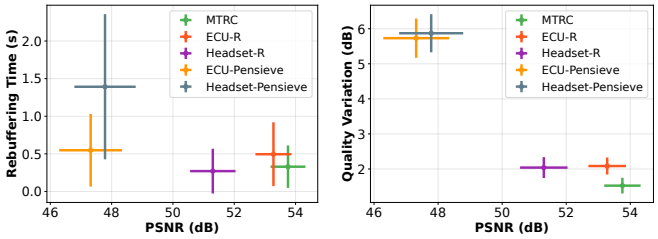
interaction with the environment. Both users' video and network conditions are randomized in this environment so that the agent learns the optimal computation distribution and rate adaptation policy for various videos and network conditions. In the second phase, we update both the actor and critic networks. We compute the Dual-Clip PPO loss $\mathscr{L}^{DClip}$, and use newly collected experiences (*i.e.,* resulted from the rollout process) to update the networks. In the third phase, we use all collected experiences to update both the actor and critic networks by optimizing the behavioral cloning and value losses. Finally, in the fourth phase, we update μ_0 and μ_1 according to Eq. 5.

V. Evaluation

Simulation and Training: In our simulations, we employ a full UHD 360° video dataset [20]. This dataset includes 9 videos with various spatio-temporal characteristics. Each video is represented using the multi-layer 360° model presented in Section II, and the video frames are partitioned into a 8×8 grid. Bitrate information for seven layers, each offering progressively higher levels of immersion fidelity for each tile, is provided. Additionally, head movement data for multiple users is included, allowing us to determine the viewport location for each user. Moreover, we use a dataset of wireless network throughput traces [32], which were collected from commercial operators (T-Mobile and Verizon) in two cities in the U.S.

Baselines: We evaluate our proposed framework through an extensive simulation against Pensieve [12], a state-of-the-art rate adaptation algorithm. Pensieve is designed for 2D video streaming applications, and only adjust the video rate based on user state, while our method is a multi-task rate adaptation and computation distribution algorithm. Thus, we employ two variants of Pensieve, namely ECU-Pensieve and Headset-Pensieve. ECU-Pensieve performs all the computations (*i.e.,* decoding and rendering) on the ECU, while Headset-Pensieve performs all the computations on the users' headset. Both use the original Pensieve with no modification to adaptively adjust the users' bitrates. Moreover, we modify our MTRC framework and present two rate adaptation algorithms, ECU-R and Headset-R. ECU-R and Headset-R use a neural network with the same architecture as shown in Fig. 4 for rate adaptation, except that the computation distribution is not decided by the neural network and all computations are performed on the ECU or headsets, respectively.

We train our MTRC agent in an environment with $N = 6$ VR users, ECU decoding speed of $Z^{dec.}_{ECU} = 7.5$ Gbps, ECU rendering speed of $Z^{rend.}_{ECU} = 20$ Gbps, headsets' decoding and



(a) RT vs PSNR (b) QV vs PSNR

Fig. 5: Performance trade-offs between rebuffering time, quality variation, and PSNR during testing stage.

rendering speeds of $Z_n^{dec.} = 0.2$ Gbps and $Z_n^{rend.} = 9.4$ Gbps, respectively. The 360° videos and network traces are randomly chosen in each episode of learning, and we train all the baselines for 5000 episodes. Then, we run 300 testing episodes and report the performance of our MTRC method compared to other baselines in the testing stage.

Testing Performance: Fig. 5 demonstrates the performance trade-offs between rebuffering time, quality variation, and PSNR. Each point demonstrates the average rebuffering time (or quality variation) and PSNR experienced by the users. The vertical and horizontal bars represent the standard deviation of the rebuffering time (or quality variation) and PSNR, respectively. A small rebuffering time (or quality variation) and high PSNR with small variation is desirable, which is represented by a point in the lower right corner of these plots. Overall, the MTRC agent achieves an improvement of 5.97 dB to 6.44 dB in PSNR, 1.66X to 4.23X improvement in rebuffering time, and 4.21 dB to 4.35 dB improvement in quality variation.

Furthermore, Table I reports the average and standard deviation of PSNR, rebuffering time, and quality variations for groups of users that play a specific video. Table I shows that our proposed method is able to capture spatio-temporal characteristics of various 360° videos and provide a fair QoE for groups of users with various requirements.

#	Video Name	PSNR [dB]	RT [Sec.]	AQV [dB]
0	Academic	52.05 ± 1.10	0.13 ± 0.32	1.58 ± 0.41
1	Basketball	53.72 ± 1.34	0.34 ± 0.60	2.29 ± 0.44
2	Bridge	55.00 ± 0.85	0.57 ± 1.17	1.27 ± 0.36
3	Gate Night	55.39 ± 1.09	0.25 ± 0.62	1.33 ± 0.35
4	Runner	52.56 ± 0.78	0.25 ± 0.57	1.81 ± 0.40
5	Siyuan	52.96 ± 0.98	0.10 ± 0.26	1.88 ± 0.40
6	South Gate	53.22 ± 1.47	0.26 ± 0.65	1.99 ± 0.48
7	Studyroom	54.23 ± 1.06	0.13 ± 0.33	1.36 ± 0.38
8	Sward	50.28 ± 1.97	0.68 ± 1.02	1.83 ± 0.62

TABLE I: QoE percived by groups of users who play an specific video

Effect of Network Condition: Table II demonstrates the effect of the network condition on QoE perceived by users. To generate this table, we analyzed the QoE perceived by two groups of users in our simulation. The first group experiences a network condition that leads to low throughput with an average and standard deviation of 662.22 ± 359.25 Mbps, while the second group experiences high throughput with an average and standard deviation of 1454.55 ± 381.04 Mbps. We report the average and standard deviation of the PSNR (*i.e.,* video quality), rebuffering time, and quality variation for these two groups of users. Overall, the first group experiences an improvement of 5.81 dB to 6.25 dB in PSNR, 1.71X to 3.97X improvement in rebuffering time, and 4.14 dB to 4.29 dB improvement in quality variation. The second group experiences even more improvements. The MTRC agent demonstrates an improvement of 6.22 dB to 6.74 dB in PSNR, a up to 13.83X improvement in rebuffering time, and a 4.33 dB to 4.45 dB improvement in quality variation for this group.

Network Condition	**Low Throughput:** 662.22 ± 359.25 **Mbps**		
Baseline	**PSNR [dB]**	**RT [s]**	**AQV [dB]**
MTRC	$\mathbf{53.63 \pm 1.51}$	0.38 ± 0.74	$\mathbf{1.57 \pm 0.62}$
ECU-R	53.21 ± 1.42	0.64 ± 1.25	2.12 ± 0.59
Headset-R	51.29 ± 2.19	$\mathbf{0.31 \pm 0.79}$	2.04 ± 1.10
ECU-Pensieve	47.38 ± 2.58	0.65 ± 1.17	5.71 ± 1.40
Headset-Pensieve	47.82 ± 2.49	1.51 ± 2.51	5.86 ± 1.43
Network Condition	**High Throughput:** 1454.55 ± 381.04 **Mbps**		
MTRC	$\mathbf{53.96 \pm 1.51}$	0.06 ± 0.05	$\mathbf{1.45 \pm 0.55}$
ECU-R	53.71 ± 1.55	0.06 ± 0.05	1.90 ± 0.56
Headset-R	51.52 ± 1.95	0.09 ± 0.11	2.00 ± 1.15
ECU-Pensieve	47.22 ± 2.69	$\mathbf{0.06 \pm 0.04}$	5.78 ± 1.49
Headset-Pensieve	47.74 ± 2.56	0.83 ± 1.67	5.90 ± 1.45

TABLE II: PSNR, rebuffering time, and quality variation for two groups of users in different network conditions

VI. Conclusion

In this paper, we considered the problem of multi-task rate adaptation and computation distribution in a VR arena for a 360° video streaming platform, where a learning-based multi-task agent decides on the video bitrate allocated to each user and computation distribution (*i.e.,* whether each video segment should be decoded/rendered on the ECU or on the headset). The overall objective is to maximize the QoE of users under dynamic and time-varying conditions in terms of video requests, available computational resources, and communication bandwidth. Using the state-of-the-art DRL algorithm, we developed MTRC that utilizes playback statistics and video information to make a joint rate adaptation and computation distribution decision. Through numerical simulation using real-world network traces and 360° video information, we showed that the MTRC agent learns to balance the existing trade-offs in the system and outperforms the state-of-the-art rate adaptation algorithm. Specifically, MTRC demonstrates an improvement of 5.97 dB to 6.44 dB in PSNR, 1.66X to 4.23X improvement in rebuffering time, and 4.21 dB to 4.35 dB in quality variation.

VII. Acknowledgement

The material is based upon work supported in part by the National Science Foundation (NSF) grants 1955561, 2212565, and 2323189. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of NSF. The work of Jacob Chakareski has been supported in part by NSF under awards CCF-2031881, ECCS-2032387, CNS-2040088, CNS-2032033, and CNS-2106150; by the National Institutes of Health (NIH) under award R01EY030470; and by the Panasonic Chair of Sustainability at the New Jersey Institute for Technology.

REFERENCES

[1] SeaGate, "State of the Edge," https://www.seagate.com/www-content/enterprise-storage/it-4-0/images/Data-At-The-Edge-UP1.pdf, 2019, [Online].

[2] J. Chakareski, M. Khan, and M. Yuksel, "Towards Enabling Next Generation Societal Virtual Reality Applications for Virtual Human Teleportation," *"IEEE Signal Processing Magazine"*, 2022.

[3] J. Chakareski, M. Khan, T. Ropitault, and S. Blandino, "Millimeter Wave and Free-Space-Optics for Future Dual-Connectivity 6DOF Mobile Multi-User VR Streaming," *ACM Trans. Multimedia Comp. Comm. App.*, 2023.

[4] Statista, "AR and VR: Market data & analysis," https://www.statista.com/outlook/amo/ar-vr/worldwide, [Online].

[5] J. Chakareski, "UAV-IoT for next generation virtual reality," *IEEE Transactions on Image Processing*, 2019.

[6] J. Chakareski and M. Khan, "Live 360° Video Streaming to Heterogeneous Clients in 5G Networks," *IEEE Transactions on Multimedia*, 2024.

[7] J. Chakareski, M. Khan, T. Ropitault, and S. Blandino, "6DOF Virtual Reality Dataset and Performance Evaluation of Millimeter Wave vs. Free-Space-Optical Indoor Communications Systems for Lifelike Mobile VR Streaming," in *Proceedings of 54th ACSSC*, 2020.

[8] J. Chakareski, "Viewport-Adaptive Scalable Multi-User Virtual Reality Mobile-Edge Streaming," *IEEE Trans. Image Processing*, 2020.

[9] C.-H. Hsu, "MEC-Assisted FoV-Aware and QoE-Driven Adaptive 360° Video Streaming for Virtual Reality," in *Proceedings of 16th Intern. Conf. on Mobility, Sensing and Networking (MSN)*, 2020.

[10] S. Gupta, J. Chakareski, and P. Popovski, "mmWave Networking and Edge Computing for Scalable 360° Video Multi-User Virtual Reality," *IEEE Transactions on Image Processing*, 2023.

[11] J. Ren, Y. He, G. Huang, G. Yu, Y. Cai, and Z. Zhang, "An Edge-Computing Based Arch. for Mobile Aug. Reality," *IEEE Network*, 2019.

[12] H. Mao, R. Netravali, and M. Alizadeh, "Neural Adaptive Video Streaming with Pensieve," in *Proc. ACM SIGCOMM*, 2017.

[13] K. Spiteri, R. Urgaonkar, and R. K. Sitaraman, "BOLA: Near-Optimal Bitrate Adaptation for Online Videos," *IEEE Trans. on Networking*, 2020.

[14] B. Badnava, S. Reddy Chintareddy, and M. Hashemi, "QoE-Centric Multi-User mmWave Scheduling: A Beam Alignment and Buffer Predictive Approach," in *Proc. IEEE ISIT*, 2022.

[15] J. Chakareski and P. Frossard, "Distributed collaboration for enhanced sender-driven video streaming," *IEEE Transactions on Multimedia*, 2008.

[16] A. B. Reis, J. Chakareski, A. Kassler, and S. Sargento, "Distortion optimized multi-service scheduling for next-generation wireless mesh networks," in *Proc. IEEE Conference on Computer Communications Workshops*, 2010.

[17] J. Chakareski, J. Apostolopoulos, W.-T. Tan, S. Wee, and B. Girod, "Distortion chains for predicting the video distortion for general packet loss patterns," in *Proc. Int'l Conf. Acoustics, Speech, and Signal Processing*. IEEE, 2004.

[18] J. Chakareski, J. Apostolopoulos, S. Wee, W.-T. Tan, and B. Girod, "Rate-Distortion Hint Tracks for Adaptive Video Streaming," *IEEE Trans. Circuits and Systems for Video Technology*, 2005.

[19] X. Hou, S. Dey, J. Zhang, and M. Budagavi, "Predictive Adaptive Streaming to Enable Mobile 360-Degree and VR Experiences," *IEEE Trans. on Multimedia*, 2021.

[20] J. Chakareski, R. Aksu, V. Swaminathan, and M. Zink, "Full UHD 360-Degree Video Dataset and Modeling of Rate-Distortion Characteristics and Head Movement Navigation," in *Proc. ACM Multimedia Sys.*, 2021.

[21] B. Badnava, K. Roach, K. Cheung, M. Hashemi, and N. B. Shroff, "Energy-Efficient Deadline-Aware Edge Computing: Bandit Learning with Partial Observations in Multi-Channel Systems," in *Proceedings of IEEE Global Communications Conference*, 2023.

[22] B. Badnava, T. Kim, K. Cheung, Z. Ali, and M. Hashemi, "Spectrum-Aware Mobile Edge Computing for UAVs Using Reinforcement Learning," in *Proceedings of IEEE/ACM Symposium on Edge Computing (SEC)*, 2021.

[23] M. Yu, H. Lakshman, and B. Girod, "A Framework to Evaluate Omnidirectional Video Coding Schemes," in *Proceedings of 2015 IEEE International Symposium on Mixed and Augmented Reality*, 2015.

[24] L. Wang, S. Singh, J. Chakareski, M. Hajiesmaili, and R. K. Sitaraman, "BONES: Near-Optimal Neural-Enhanced Video Streaming," *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, 2024.

[25] X. Yin, A. Jindal, V. Sekar, and B. Sinopoli, "A Control-Theoretic Approach for Dynamic Adaptive Video Streaming over HTTP," in *Proceedings of the 2015 ACM Conf. on Special Interest Group on Data Comm.*, 2015.

[26] M. Ghazikor, K. Roach, K. Cheung, and M. Hashemi, "Interference-Aware Queuing Analysis for Distributed Transmission Control in UAV Networks," in *arXiv:2401.11084*, 2024.

[27] ——, "Exploring the Interplay of Interference and Queues in Unlicensed Spectrum Bands for UAV Networks," in *Proceedings of 57th Asilomar Conference on Signals, Systems, and Computers*, 2023.

[28] D. Ye, Z. Liu, M. Sun, B. Shi, P. Zhao, H. Wu, H. Yu, S. Yang, X. Wu, Q. Guo, Q. Chen, Y. Yin, H. Zhang, T. Shi, L. Wang, Q. Fu, W. Yang, and L. Huang, "Mastering Complex Control in MOBA Games with Deep Reinforcement Learning," in *Proceedings of The 34 AAAI Conf.*, 2020.

[29] C. W. L. S. Tianchi Huang ; Chao Zhou ; Rui-Xiao Zhang, "Buffer Awareness Neural Adaptive Video Streaming for Avoiding Extra Buffer Consumption," in *Proceedings of IEEE INFOCOM Conf.*, 2023.

[30] K. Cobbe, J. Hilton, O. Klimov, and J. Schulman, "Phasic Policy Gradient," *arXiv:2009.04416*, 2020.

[31] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal Policy Optimization Algorithms," *arXiv:1707.06347*, 2017.

[32] A. Narayanan, X. Zhang, R. Zhu, A. Hassan, S. Jin, X. Zhu, X. Zhang, D. Rybkin, Z. Yang, Z. M. Mao, F. Qian, and Z.-L. Zhang, "A Variegated Look at 5G in the Wild: Performance, Power, and QoE Implications," in *Proceedings of ACM SIGCOMM*, 2021.