Efficient Content Delivery in Cache-Enabled VEN with Deadline-Constrained Heterogeneous Demands: A User-Centric Approach

Md Ferdous Pervej, Graduate Student Member, IEEE, Richeng Jin, Member, IEEE, Shih-Chun Lin, Member, IEEE, and Huaiyu Dai, Fellow, IEEE

Abstract—Modern connected vehicles (CVs) frequently require diverse types of content for mission-critical decision-making and onboard users’ entertainment. These contents are required to be fully delivered to the requester CVs within stringent deadlines that the existing radio access technology (RAT) solutions may fail to ensure. Motivated by the above consideration, this paper exploits content caching with a software-defined user-centric virtual cell (VC) based RAT solution for delivering the requested contents from a proximity edge server. Moreover, to capture the heterogeneous demands of the CVs, we introduce a preference-popularity tradeoff in their content request model. To that end, we formulate a joint optimization problem for content placement, CV scheduling, VC configuration, VC-CV association and radio resource allocation to minimize long-term content delivery delay. However, the joint problem is highly complex and cannot be solved efficiently in polynomial time. As such, we decompose the original problem into a cache placement problem and a content delivery delay minimization problem given the cache placement policy. We use deep reinforcement learning (DRL) as a learning solution for the first sub-problem. Furthermore, we transform the delay minimization problem into a priority-based weighted sum rate (WSR) maximization problem, which is solved leveraging maximum bipartite matching (MWBM) and a simple linear search algorithm. Our extensive simulation results demonstrate the effectiveness of the proposed method.

Index Terms—Connected vehicle (CV), content caching, delay minimization, software-defined networking (SDN), user-centric networking, vehicular edge network (VEN).

I. INTRODUCTION

ADVANCED driver-assistance systems (ADAS) and infotainment are two premier features for modern connected vehicles (CVs). With advanced radio access technologies (RATs), delivering the Society of Automotive Engineers (SAE) level 5 automation on the road seems more pragmatic day by day. Different government organizations such as the U.S. Department of Transportation’s National Highway Traffic Safety Administration in the United States [1], the Department for Transport in the U.K. [2], etc., set firm regulations for the CVs to ensure public safety on the road. For swift decision-making to satisfy the safety requirements, the CVs need fast, efficient, and reliable communication and data processing. As such, an efficient vehicular edge network (VEN) must ensure uninterrupted and ubiquitous wireless connectivity on the road. To deliver such services, the VEN demands advanced machine learning (ML) tools for resource management complementary to a RAT solution, such as the 5G new-radio (NR) vehicle-to-everything (V2X) communication [3].

With increased automation, in-car entertainment is also becoming a priority for modern CVs [4]. Modern CVs are expected to have many new features, such as vehicular sensing, onboard computation, virtual personal assistant, virtual reality, vehicular augmented reality, autopilot, high-definition (HD) map collection, HD content delivery, etc., [5], [6] that are interconnected for both ADAS and infotainment. For these demands, by exploiting the emerging content caching [7], the centralized core network can remarkably gain by not only ensuring local content distribution but also lessening the core network congestion [8], [9]. As such, VENs can reduce end-to-end delay significantly by storing the to-be-requested contents at the network edge [10], which is vital for the CVs’ mission-critical delay-sensitive applications. A practical RAT on top of content caching can, therefore, bring a promising solution for SAE level 5 automation on the road.

For diverse applications, such as mobile broadband and low latency (MBBLL), massive broadband machine-type (mBBMT), massive low-latency machine-type (mLLMT) communications, etc., the CVs urgently need an efficient RAT solution [11]. In the meantime, regardless of the applications, the VEN must ensure omnipresent connectivity to the CVs and deliver their requested contents timely. The so-called user-centric networking [12]-[15] is surging nowadays with its ability to shift network resources towards network edge. Note that, while the network-centric approach serves a user from only one base station, the user-centric approach enables serving a user from multiple transmission points [16]. The latter approach can, thus, not only provide ubiquitous connectivity but also provide higher throughput with minimized end-to-end latency for the end-users [17]. As such, a user-centric approach can combat the frequent changes in received signal strength - often experienced in VENs due to high mobility, by ensuring multipoint data transmission and receptions.

While the user-centric networking approach can bring universal connectivity and MBBLL/mBBMT/mLLMT solutions for the CVs, it induces a more complex network infrastructure. To ensure multipoint data transmission and reception, efficient baseband processing is required. Moreover, as the traditional hardware-based and closed network-centric approach is inflexible, the user-centric approach demands the use of software-defined networking [18], which can offer more efficient and agile node associations and resource allocations in the user-centric approach. With proper system design, it is possible to
create virtual cells (VCs) with multiple low-powered access points (APs) to ensure that the throughput and latency requirements of the CVs are satisfied. Moreover, amalgamating content caching with the user-centric RAT solution can indeed ensure timely payload delivery for stringent delay-sensitive application requirements of modern CVs. However, this requires a joint study for - content placement, CV scheduling, VC formulation, VC association with the scheduled CV, and radio resource allocation of the APs in the VCs.

A. Related Work

In literature, there exist several works [19]–[26] that considered cache-enabled VENs from the traditional network-centric approach. Huang et al. proposed a content caching scheme for the Internet of vehicles (IoVs) in [19]. They developed a delay-aware content delivery scheme exploiting both vehicle-to-infrastructure (V2I) and vehicle-to-vehicle (V2V) links. The authors minimized content delivery delays for the requester vehicles by jointly optimizing cache placement and vehicle associations. Nan et al. also proposed a delay-aware caching technique assuming that the vehicles could either - (a) decide to wait for better delivery opportunities, or (b) get associated with the roadside unit (RSU) that has the content, or (c) use one RSU as a relay to extract the content from the cloud in [20]. The authors exploited deep reinforcement learning (DRL) to minimize content delivery cost. However, these assumptions are not suitable for CVs because time-sensitivity plays a crucial role in the quick operation of CVs. [21] proposed quality-of-service ensured caching solution by bounding the content into smaller chunks.

Dai et al. leveraged blockchain and DRL to maximize caching gain [22]. Lu et al. proposed a federated learning approach for secure data sharing among the IoVs [23]. However, [22], [23] assumed that the data rate is perfectly known without any proper resource allocations for the RAT. Zhang et al. addressed proactive caching by predicting user mobility and demands in [24]. Similar prediction-based modeling has also been extensively studied in [7], [27], [28]. Moreover, [24] only analyzed cache hit ratio without incorporating any underlying RAT. Fang et al. considered a static popularity-based cooperative caching solution for roaming vehicles, which assumed constant velocity and downlink data rate and minimized content extraction delay [25]. Liu et al. considered coded caching for a typical heterogeneous network with one macro base station (MBS) overlaid on top of several RSUs [26]. Vehicles trajectory, average residence time within RSU’s coverage, and system information were assumed to be perfectly known to the MBS in [26]. Owing to the time-varying channel conditions in VENs, the authors further considered a two-time scaled model. Particularly, they assumed that content requests only arrive at the large time scale (LTS) slot, whereas MBS could decide to orchestrate resources in each small time scale (STS) slot - within the LTS slot. However, although [26] assumed LTS and STS considering time-varying wireless channels, it did not consider any communication model. Therefore, the study presented in [26] did not reflect delay analysis in VENs.

The study presented in [19]–[26] mostly considered that the content catalog consist of a fixed number of contents from a single category. In reality, each content belongs to a certain category, and the catalog consists of contents from different categories. Besides, these studies mainly assumed that the users request contents based on popularity. However, each CV may have a specific need for a particular type of content. For example, some CVs may need to have frequent operational information, whereas other CVs may purely consume entertainment-related content. Therefore, a VEN shall consider individual CV’s preference, as well as the global popularity.

Some literature also exploited user-centric RAT solutions for VENs [13], [16], [17], [29]–[31]. Considering the high mobility of the vehicles, [13] proposed an approach for user-centric VC creation and optimized resource allocation to ensure maximized network throughput. A power-efficient solution for the VC of the VENs was also proposed in [17]. Lin et al. proposed heterogeneous user-centric (HUC) clustering for VENs in [31]. Particularly, the authors considered creating HUC using both traditional APs and vehicular APs. The goal of [31] was to study how HUC migration helps in VEN. Considering both horizontal handover (HO) and vertical (VO), [31] studied the tradeoff between throughput and HO overhead. Xiao et al. showed that dynamic user-centric virtual cells could be used to multicast the same message to a group of vehicles in [29]. Particularly, [29] assumed that a group of vehicles could be considered as a hotspot (HS). If all vehicles inside the HS are interested in the same multicasted message, multiple APs could formulate a VC to serve the HS. [29] optimized power allocation to balance the signal-to-interference-plus-noise ratio for the vehicles in the HS. Similar studies were also presented in [16], [30].

B. Motivations and Our Contributions

As ubiquitous connectivity is essential for CVs, the existing RAT solutions may not be sufficient to meet the strict requirements of CVs for higher automation. Existing literature shows that user-centric networking can bring additional burdens that need rigorous studies, such as mobility and HO management [31]. Moreover, as multicasting delivers a common signal, the study presented in [16], [29], [30] is not suitable for CV-specific independent data requirements in delay-sensitive applications. However, an alternative software-defined networking approach with advanced ML algorithms can potentially bring the RAT solution [13], [17], [32]. Particularly, the VEN can deploy close proximity edge server that acts as the software-defined controller. The to-be-requested contents can be prefetched through the edge servers to ensure local delivery. Besides, multiple low-powered APs can be placed as RSUs. The controller can determine the user-centric VC configuration and the corresponding resource orchestration to meet the requirements of the CVs by controlling these APs.

Motivated by these, we consider a practical user-centric VEN where highly mobile CVs are served from multiple low-powered APs. We devise a joint cache placement and user-centric RAT solution. Particularly, our contributions are

- Considering the stringent requirements of the regulatory organizations, we propose a software-defined user-centric RAT solution that exploits multiple APs to provide ubiquitous and reliable connectivity to the CVs on the road.
TABLE I

| Parameter   | Definition                                                                                                                                 |
|-------------|-------------------------------------------------------------------------------------------------------------------------------------------|
| $\mathcal{B}$ | Set of APs, total number of APs, $b^{th}$ AP                                                                                             |
| $\mathcal{U}_b$ | Set of CVs, total number of CVs, $u^{th}$ CV                                                                                               |
| $\mathcal{W}_b(t)$ | Maximum possible VCs with $b$ APs, total created VCs in slot $t$                                                                          |
| $\lambda_b(t), \mu_b(t), \nu_b(t)$ | All VC configurations set, possible VC configurations in $\mathcal{W}_b(t)$; VC sets under $b^{th}$ configuration                        |
| $\mathcal{V}_b(t)$ | $b^{th}$ VC of $\lambda_b(t)$                                                                                                            |
| $\mathcal{V}_b(t)$ | Indicator function that defines whether AP $b$ is in $\mathcal{V}_b(t)$                                                                  |
| $\mathcal{I}_b(t)$ | Indicator function that defines whether VC $u$ is scheduled at $t$                                                                       |
| $\mathcal{I}_b(t)$ | Indicator function that defines whether VC $u$ is selected for user $u$ at time $t$                                                        |
| $Z, Z, z, w, x$ | Total network bandwidth, total orthogonal PRB, $zh$ PRB, size of the pRBs                                                                  |
| $F_b(t)$ | Large scale fading, log-Normal shadowing, fast fading, channel response, entire channel response, respectively                            |
| $\omega_b(t)$ | Remaining deadline and payload for the requested content in slot $t$                                                                      |
| $\mathcal{I}_{cb}(t)$ | Downlink received signal and downlink SNR at CV $a$ over $b^{th}$ PRB $c$, respectively, during slot $t$                                  |
| $\mathcal{I}_{cb}(t)$ | Downlink achievable rate at CV $a$ during slot $t$                                                                                       |
| $\mathcal{I}_{cb}(t)$ | Downlink achievable rate at CV $a$ during slot $t$                                                                                       |
| $\mathcal{I}_{cb}(t)$ | Content class set, total content class, $u^{th}$ content class                                                                           |
| $\mathcal{I}_{cb}(t)$ | Content class set in class $c$, total content in a, $f_c$ content of class $c$, entire content library                                   |
| $\mathcal{I}_{cb}(t)$ | Set of the content features, total number of features, $g_i$ feature, respectively, of content $c$                                      |
| $\mathcal{I}_{cb}(t)$ | Doyle, cache (re)-placement or Doyle change counter                                                                                       |
| $\mathcal{I}_{cb}(t)$ | Bernoulli random variable that defines whether CV $u$ places a content request at time $t$                                                 |
| $\mathcal{I}_{cb}(t)$ | Total content requests from all CVs during slot $t$                                                                                       |
| $\mathcal{I}_{cb}(t)$ | Intended signal, symbol, and beamforming vector of AP $b$ for CV $u$, respectively                                                      |
| $\mathcal{I}_{cb}(t)$ | Transmission power of AP $b$                                                                                                             |
| $\gamma_b(t)$ | Cosine similarity index of content $f_c$ and $f'_c$                                                                                       |
| $S, A, A'$ | Content size, storage cache size of edge server, cache storage to be filled with content from class $c$                                  |
| $\mathcal{I}_{cb}(t)$ | Indicator function that defines whether content is stored during cache placement content                                              |
| $d_{fc_c}, d_{fc_c}, d_{fc_c}$ | Maximum probability for content exploitation of CV $a$                                                                                   |
| $d_{fc_c}, d_{fc_c}, d_{fc_c}$ | CV $a$ a probability of selecting content class                                                                                         |
| $d_{fc_c}, d_{fc_c}, d_{fc_c}$ | Global popularity of content $f_c$, of class $c$                                                                                         |
| $d_{fc_c}, d_{fc_c}, d_{fc_c}$ | Content extraction delay from cloud, wait time of $U_c(t)$ before being scheduled, transmission delay                                     |
| $d_{fc_c}, d_{fc_c}, d_{fc_c}$ | Average delays for all $U_c(t)$                                                                                                          |
| $m_{cub}, m_{cub}$ | Cache placement action, position, cache size                                                                                             |
| $\mathcal{I}_{cb}(t)$ | Cache hit event for $U_c(t)$                                                                                                             |
| $h_b(t)$ | Total cache hit during slot $t$, cache hit rate during slot $t$                                                                       |
| $\tau_b(t)$ | Cache placement policy of the edge server                                                                                                |
| $\mathcal{I}_{cb}(t)$ | Slots of interests during slot $t$                                                                                                        |
| $\eta_{b}, \eta_{b}$ | Remaining deadline and payload for the requested content in slot $t$                                                                    |
| $d_{b}^{max}$ | Valid CV set during slot $t$, and their minimum remaining deadline and payload set, respectively                                      |
| $\mathcal{I}_{cb}(t)$ | Normalized weights of the CVs in valid CV set during slot $t$                                                                          |
| $\mathcal{I}_{cb}(t)$ | Scheduled CV set during slot $t$                                                                                                         |
| $\mathcal{I}_{cb}(t)$ | Weights sum rate of the VEN during slot $t$                                                                                              |
| $\mathcal{I}_{cb}(t)$ | Top-most popular and their $A'$-top similar contents matrix                                                                          |
| $\mathcal{I}_{cb}(t)$ | Content-specific requests history matrix of $a$ in past DoI                                                                            |
| $\mathcal{I}_{cb}(t)$ | Content-specific local cache hit history matrix of $a$ in past DoI                                                                       |
| $\mathcal{I}_{cb}(t)$ | Measured popularity of contents during $a$ based on past DoI                                                                         |
| $\mathcal{I}_{cb}(t)$ | State and instantaneous revenue of the edge server during DoI                                                                        |
| $\mathcal{I}_{cb}(t)$ | Online DNN and offline DNN of the edge server for learning the CPF                                                                   |
| $\mathcal{I}_{cb}(t)$ | Edge server’s memory buffer, maximum length of $mem_{max}$ for learning $X_c$                                                        |
| $G, K_c$ | Bipartite graph, weighted data rate matrix                                                                                              |
| $c_{v}, c, c_{v}, c_{v}$ | Edge connecting vertex $b$ and $c$, and corresponding weights                                                                         |
| $\mathcal{I}_{cb}(t)$ | Possible transmittable bits for CV $u$ during slot $t$                                                                                   |

Fig. 1. Proposed user-centric cache-enabled vehicular edge network

- To ensure fast decision-making for mission-critical operations and uninterrupted onboard entertainment, we exploit content prefetching at the edge server while introducing preference-popularity tradeoff into individual content requests owing to the CVs’ heterogeneous preferences. To deliver the requested contents within a hard deadline, we leverage the proposed RAT.

- We introduce a joint content placement, CV scheduling, VC configuration, CV-VC association and radio resource allocation problem to minimize content delivery delays.

- To tackle the grand challenges of the joint optimization problem, we decompose it into a cache placement subproblem and a delay minimization subproblem - given the cache placement policy. We use DRL to solve the first subproblem. Moreover, we transform the second subproblem into a weighted sum rate (WSR) maximization problem and solve it using graph theory and a simple linear search algorithm.

- Through analysis and simulation results, we verify that our proposed solution achieves better performance than the existing baselines.

The rest of the paper is organized as follows: Section II introduces our proposed software-defined user-centric system model. Section III presents the caching model, followed by the joint problem formulation in Section IV. Problem transformations are detailed in Section V, followed by our proposed solution in Section VI. Section VII presents extensive simulation results and discussions. Finally, Section VIII concludes the paper. The important notations are listed in Table I.

II. SOFTWARE-DEFINED USER-CENTRIC COMMUNICATION MODEL

A. Communication System Model

This paper considers a software-defined cache-enabled VEN. An edge server - controlled by a software-defined controller, is placed in proximity to the edge CVs. The edge server has dedicated radio resources with limited local cache storage and is connected to the cloud. Several low-powered APs are deployed as RSUs to provide omnipresent wireless connectivity to the CVs. These APs are connected to the edge server with high-speed wired links. The software-defined centralized controller can control the edge server and perform user scheduling, node associations, precoding, channel estimations, resource allocations, etc. In other words, the edge server acts as the baseband unit. Besides, unlike the legacy system models,
we consider a user-centric approach that uses multiple APs to serve the scheduled CVs. These APs are used as RSUs that only perform radio transmissions over the traditional Uu interface [33]. Denote the vehicle and AP set by \( \mathcal{W} = \{u\}_{u=1}^{U} \) and \( \mathcal{B} = \{b\}_{b=1}^{B} \), respectively. The VEN operates in slotted times. Given \( B \) APs at fixed locations, unlike the traditional network-centric approach, the proposed VEN partitions the AP set \( \mathcal{B} \) into \( W(t) \leq B \) subsets of APs at each slot \( t \). Without loss of generality, we define each subset as a VC. The proposed VEN can assign such a VC to a scheduled CV.

Let there be a set \( \mathcal{A}(W(t)) = \{a\}_{a=1}^{A(W(t))} \) that defines the possible ways to partition the \( B \) APs into \( W(t) \) subsets of APs, i.e., VCs. Denote the \( i \)-th VC of an \( a \in \mathcal{A}(W(t)) \) by \( V_{C_a}^i \subset \mathcal{B} \) and the set of VCs by \( \mathcal{R}_w^a(W(t)) = \{V_{C_a}^i\}_{i=1}^{A(W(t))} \). Furthermore, for each \( a \in \mathcal{A}(W(t)) \), the VC must obey the following rules:

\[
\begin{align*}
V_{C_a}^i & \neq \emptyset, \forall a \text{ and } i, \\
V_{C_a}^i \cap V_{C_a}^j & = \emptyset, \forall a \text{ and } i \neq j, \\
\bigcup_{i=1}^{W} V_{C_a}^i & = \mathcal{B}, \forall a.
\end{align*}
\]

The first rule in (1a) means that the VCs must not be empty, while the second rule in (1b) means the subsets are mutually exclusive. Besides, the last rule in (1c) represents that the union of the VCs must yield the original AP set \( \mathcal{B} \). When \( W(t) = 3 \) and \( B = 8 \), for one possible \( a \in \mathcal{A}(W(t)) \), the VC set \( \mathcal{R}_w^a(W(t)) \) is shown in Fig. 1 by the filled ovals.

Then, for a given \( W(t) \), the VEN can formulate the total VC configuration pool \( \mathcal{R}_w^a(W(t)) = \{\mathcal{R}_w^a(W(t))\}_{a=1}^{A(W(t))} \). Further, the VEN can partition the \( B \) APs into \( W(t) \) VCs following the above rules in \( A(W(t)) = W(t)!B_{w-c}(B,W(t)) \) ways, where \( B_{w-c}(B,W(t)) \) is calculated as:

\[
B_{w-c}(B,W(t)) = \frac{1}{W(t)!} \sum_{i=1}^{W(t)} (-1)^{i} \binom{W(t)}{i} (W(t) - i)^B.
\]

Note that \( B_{w-c}(B,W(t)) \) is commonly known as the Stirling number of the second kind [34].

To this end, as the VCs contain different AP configurations, denote the VC and AP mapping by

\[
\begin{align*}
I_b^{u_a}(t) = \begin{cases} 
1, & \text{if AP } b \text{ is in } V_{C_a}^i \text{ during slot } t, \\
0, & \text{otherwise},
\end{cases}
\end{align*}
\]

The edge server selects the total \( W(t) \) number of VCs to form and their configuration \( \mathcal{R}_w^a(W(t)) \in \mathcal{R}_w^a(W(t)) \). The VCs \( V_{C_a}^i \in \mathcal{R}_w^a(W(t)) \) are assigned to serve the scheduled CVs. Denote the CV scheduling and VC-CV association decisions by the following indicator function:

\[
\begin{align*}
I_u(t) = \begin{cases} 
1, & \text{if CV } u \text{ is scheduled in slot } t, \\
0, & \text{otherwise},
\end{cases}
\end{align*}
\]

We contemplate that the VEN operates in frequency division duplex (FDD) mode and has a fixed \( \tilde{Z} \) Hz bandwidth. The edge server uses this dedicated \( \tilde{Z} \) Hz bandwidth and divides it into \( Z \) orthogonal physical resource blocks (pRBs). Let the set of the orthogonal pRBs be \( \tilde{Z} = \{z\}_{z=1}^{Z} \) for the downlink infrastructure-to-vehicle (12V) communication. Denote the size of a pRB by \( \omega \), while we introduce the following indicator function for the pRB allocation

\[
\begin{align*}
I_{z}^{u_a}(t) = \begin{cases} 
1, & \text{if pRB } z \text{ is assigned to AP } b \text{ when } I_b^{u_a}(t) = 1 \\
0, & \text{otherwise},
\end{cases}
\end{align*}
\]

B. Communication Channel Modeling

We consider single antenna CVs whereas, the APs are equipped with \( L > 1 \) antennas. Let us denote the channel response at a CV \( u \) from the AP \( b \) over pRB \( z \), as follows:

\[
\tilde{h}_b^{u_z}(t) = \sqrt{\frac{P_b}{\tilde{I}_b^{u_z}(t)}} \bar{h}_b^{u_z}(t) \tilde{v}_b^{u_z}(t) \in \mathbb{C}^{L \times 1},
\]

where \( \bar{h}_b^{u_z}(t) \) and \( \tilde{v}_b^{u_z}(t) \) are \( [h_b^{u_z}(t), \ldots, h_b^{u_z}(t)]^T \in \mathbb{C}^{L \times 1} \) are large scale fading, log-Normal shadowing and fast fading channel responses from the \( L \) antennas, respectively. Besides, \( \tilde{h}_b^{u_z}(t) \) is the \( b \)-th row of \( \tilde{H}_b^{u_z}(t) \) that denotes the channel between \( u \) and the \( b \)-th antenna of AP \( b \) at time \( t \) over pRB \( z \). Note that we consider the UMa model [35] for modeling the path losses following 3GPP’s standard [33]. Then, for all pRBs in the system, we express the wireless channels from AP \( b \) to CV \( u \), at time \( t \), as \( \tilde{H}_b^{u_z}(t) = [\tilde{h}_b^{u_1}(t), \ldots, \tilde{h}_b^{u_Z}(t)] \in \mathbb{C}^{L \times Z} \). We consider the edge server has perfect channel state information (CSI)\(^1\) and all transceivers can mitigate the Doppler effect.

C. Transceiver Modeling

The transmitted signal at the AP \( b \) for CV \( u \) is \( s_b^{u_z}(t) = \sqrt{P_b} \tilde{v}_b^{u_z}(t) w_b^{u_z}(t) \in \mathbb{C}^{L \times 1} \), where \( P_b \) is the transmission power of AP \( b \). Besides, \( x_b^{u_z}(t) \) and \( w_b^{u_z}(t) \in \mathbb{C}^{L \times 1} \) are the unit powered intended signal and corresponding precoding vector over pRB \( z \), respectively, of the AP \( b \) for \( u \) during slot \( t \). Then, the transmitted signals for CV \( u \) from all APs is \( s_u(t) = [s_1^{u_z}(t), \ldots, s_B^{u_z}(t)]^T \in \mathbb{C}^{L \times Z} \). Moreover, as each AP transmits over orthogonal pRBs, the proposed VEN does not have any interference. To this end, we calculate the received signal at CV \( u \), over pRB \( z \), as

\[
\gamma_u^{u_z}(t) = \sum_{b=1}^{B} I_b^{u_a}(t) \sum_{i=1}^{W(t)} \sum_{b=1}^{B} I_b^{u_a}(t) \bar{h}_b^{u_z}(t) \tilde{v}_b^{u_z}(t) \left[ h_b^{u_z}(t) s_b^{u_z}(t) + \eta_b^{u_z}(t) \right],
\]  
where \( \eta_b^{u_z}(t) \sim CN(0, \sigma_b^2) \) is zero mean circularly symmetric Gaussian disturbed noise with variance \( \sigma_b^2 \). The corresponding signal-to-noise ratio (SNR), over pRB \( z \), is calculated as

\[
\Gamma_u^{u_z}(t) = \frac{\sum_{b=1}^{B} I_b^{u_a}(t) \sum_{i=1}^{W(t)} \sum_{b=1}^{B} I_b^{u_a}(t) \left[ P_b \bar{h}_b^{u_z}(t) \tilde{v}_b^{u_z}(t) \right]^2 \theta_b^{u_z}(t)}{\sum_{b=1}^{B} \sum_{i=1}^{W(t)} I_b^{u_a}(t) \cdot \left[ \sum_{b=1}^{B} I_b^{u_a}(t) \sigma_b^2 \right]}.
\]

Therefore, the total downlink achievable capacity for CV \( u \) is

\[
R_u(t) = \sum_{z=1}^{Z} \omega \cdot \log_2 (1 + \Gamma_u^{u_z}(t)).
\]

III. EDGE CACHING MODELING

A. Definitions and Assumptions

To avoid cross-domain nomenclature, we present necessary terms and our assumptions in the following.

1Although channel reciprocity does not hold in FDD, the edge server can use some feedback channels to estimate the CSI.
Definition 1 (Content). The source file that the CVs request is defined as content. These files can contain CVs’ operational information, geographic information, map/navigation information, weather conditions, compressed file with sensory data, local news, video/audio clips, etc.

Definition 2 (Content Class). Each content belongs to a class that defines the type/category of the content. Let there be F contents in each class and the set of the content class be \( C = \{ c \}_{c=1}^{F} \), where \( C \in \mathbb{Z}^+ \). Denote the content set of class \( c \) by \( \mathcal{F}_c = \{ f_c \}_{i=1}^{g_c} \), where \( f_c \) represents the \( i^{th} \) content of class \( c \). Moreover, let the content size be \( S \) bits.

Definition 3 (Content Features). Let the content in the \( c^{th} \) class have \( G_c \in \mathbb{Z}^+ \) features. Denote the feature set of content \( f_c \) by \( \mathcal{G}_c = \{ g_{cf} \}_{g=1}^{G_c} \).

Definition 4 (Content Library). The content library is comprised of all contents from all classes. Let \( \mathcal{F} = \bigcup_{c=1}^{C} \mathcal{F}_c \) be the content library.

Definition 5 (Duration of Interest (DoI)). The period for which the content library remains fixed is denoted as the duration of interest (DoI). Denote this period by \( \Upsilon \).

The content library \( \mathcal{F} \) is fixed. While the CVs may request contents in each slot \( t \), the edge servers can update its cached content only in each \( t = nT \) time slot, where \( n \in \mathbb{Z}^+ \) defines the cache (re)placement counter. Note that this assumption is made as it may not be practical to update the cache storage in each slot \( t \) due to hardware limitations. Moreover, we assume that the content update/refresh process is independent of the content request arrival process. To this end, we focus on request arrival modeling from the CVs, followed by modeling their preferences, i.e., their requested \( f_c \) in \( \mathcal{F} \) in slot \( t \).

B. User Request/Traffic Modeling

We assume that the CVs can make content requests in each slot \( t \) following Bernoulli distribution\(^2\). At slot \( t \), let \( \Theta_{u,t} \) denote a Bernoulli random variable - with success probability \( p_u \), which defines whether \( u \) makes a content request or not. The total number of requests during a DoI \( \Upsilon \) for an individual CV \( u \) follows Binomial distribution. Besides, at slot \( t \), the total number of requests from all CVs, i.e., \( \Psi_t = \sum_{u=1}^{U} \Theta_{u,t} \), follows a Poisson binomial distribution \([38]\) with probabilities \( \mathbf{p} = \{ p_u \}_{u \in \mathcal{U}} \). Moreover, the probability of the distribution of \( \Psi_t \), can be bounded using Proposition 1.

Proposition 1. Let \( \mu = E[\Psi_t] = \sum_{u=1}^{U} p_u \) and \( \bar{p} = \frac{1}{U} \sum_{u=1}^{U} p_u \) be the average success probability. Then, at slot \( t \), the probability that the distribution of the total number of requests of the CVs gets larger than some \( \xi = \mu + \delta \) and \( 0 < \delta < U - \mu \), is bounded above as follows:

\[
Pr \left\{ \Psi_t \geq \xi \right\} \leq \exp \left\{ -UD_{\bar{p}}(\chi) \right\},
\]

where \( D_{\bar{p}}(\chi) = \chi \ln \left( \frac{\bar{p}}{\chi} \right) + (1 - \chi) \ln \left( \frac{1 - \bar{p}}{1 - \chi} \right) \) is the relative entropy of \( \chi \) to that of \( \bar{p} \) and \( \chi = \frac{\mu}{\bar{p}} \).

Proof. Please see Appendix A.

\(^2\)Similar access modeling was also used in existing works \([36], [37]\).

C. Individual User Preference Modeling

Given that a CV makes a request, we now focus on the which question, i.e., given that \( \Theta_{u,t} = 1 \), which content shall that CV request at that slot? Let us express a particular content \( f_c \) requested by CV \( u \) during slot \( t \) by

\[
I_{u,c}(t) = \begin{cases} 1, & \text{if } \Theta_{u,t} = 1 \text{ and } u \text{ request } f_c, \\ 0, & \text{otherwise}, \end{cases}
\]

Unlike legacy modeling\(^3\), we consider that a CV’s choice depends both on its personal preference and global popularity. Particularly, a CV may prefer to consume a content based on its specific operational needs or likeness of a previously consumed content rather than requesting the globally popular content. Besides, it may also get influenced by the popularity of the contents. As such, we model the user’s content request as the exploitation-exploration tradeoff between personal preference and global popularity of the contents. We present this by the \( \varepsilon_u \)-policy, i.e., a CV exploits with probability \( \varepsilon_u \) and explores with probability \( (1 - \varepsilon_u) \).

1) Content Selection during Exploitation: In this case, the CV exploits its preferred contents from the same class it previously consumed a content. Given that CV has requested \( f_c \) in slot \( t \), it will request the most similar content to \( f_c \) in class \( c \) with probability \( \varepsilon_u \) if \( \Theta_{u,t} = 1 \). Note that similarity between \( f_c \) and \( f_c' \), where \( f_c' \neq f_c \) is calculated as

\[
\Omega_{f,c,c'} = \sqrt{\sum_{g \in \mathcal{G}_c} s_{f_c,g}^2 \sum_{g' \in \mathcal{G}_{c'}} s_{f_c',g'}^2},
\]

2) Content Selection during Exploration: Given that the CV requested content from class \( c \) previously, it will explore new content from a different class \( c' \neq c \). Denote CV \( u \)'s class selection probability by \( p_u^{c'} \), which follows a categorical distribution. Once the CV chooses the new content class \( c' \), it randomly selects a content from this class based on global popularity. Denote the global popularity of contents in class \( c \) by \( \mathbf{p} = \{ p_c \}_{c \in \mathcal{C}} \), where \( p_c \) is the popularity of content \( f_c \).

D. Content (Re)placement in the Cache

Recall that only the edge server has limited cache storage in the proposed VEN. Let the cache storage of the edge server be \( \Lambda \). Denote the cache placement indicator by the binary indicator function \( I_{f_c}(n) \) during cache placement counter \( n \).

The edge server obeys the following rules for cache placement: \( S' \cdot \sum_{c=1}^{C} \sum_{f=1}^{F} I_{f_c}(n) \leq \Lambda \), \((14) \) \( S' \cdot \sum_{f=1}^{F} I_{f_c}(n) = \Lambda' \), \((15) \) where \( \Lambda' \) is the total storage taken by the cached content from class \( c \in \mathcal{C} \). Moreover, \((14) \) ensures that the size of the total cached contents must not be larger than the storage capacity. At each \( t = nT \), the SD-controller pushes the updated contents into the cache storage. These contents remain at the edge servers’ cache storage till the next DoI update.

IV. DELAY MODELING AND PROBLEM FORMULATION

A. Content Delivery Delay Modeling

For higher automation (and uninterrupted entertainment of the onboard users), the CVs may need to continuously access
diverse contents within a tolerable delay to avoid fatalities (and quality of experiences (QoEs)). This motivates us to introduce a hard deadline requirement for the edge server to deliver the requested contents. We consider that requests arrive continuously following \( \Theta_{u}^{s} \). Each CV can make at most a single content request based on its preference if \( \Theta_{u}^{s} = 1 \). The requester has an associated hard deadline requirement \( d_{u}^{\text{max}} \), within which it needs the entire payload. The edge server, on the other hand, can have a shorter deadline, denoted by \( \delta_{u}^{(t)} \), associated with the requests as it replaces the content at the end of each DoI. Formally, the deadline for the edge server to fully onload a requested content is

\[
\delta_{u}^{(t)}(t) = \min\{d_{u}^{\text{max}}, (n+1)T - t\}, \quad \forall t \in [nT, (n+1)T].
\]  

(16)

Essentially, (16) ensures that the edge server cannot exceed the maximum of the maximum allowable delay threshold \( d_{u}^{\text{max}} \) and remaining time till the next cache replacement slot \((n+1)T\).

To that end, we calculate the associated delay of delivering the requested contents from all CVs \( \{u \in \mathcal{U}\} \). This delay depends on whether the requested content has been prefetched and the underlying wireless communication infrastructure. Particularly, for the cache miss event\(^{4}\), the requested content is extracted from the cloud. This extraction causes an additional delay and involves the upper layers of the network\(^{5}\). Denote the delay for extracting content \( f_{u} \), requested by CV \( u \) during slot \( t \), from the cloud by \( d_{u}^{(f,m)} \). Moreover, we consider two more additional delays. The first one is the wait time of a request before being scheduled for transmission by the edge server, given that the edge server has either prefetched the content during the cache placement slot or the upper layers have already processed the requested content from the cloud. The second one is the transmission delay. Denote these two delays, i.e., wait time and transmission delays, for \( I_{u}(t) \) by \( d_{u}^{(f,t)} \) and \( d_{u,f,t}^{(t)} \), respectively. Therefore, the total delay of delivering the entire content is calculated as

\[
d_{u}^{(t)}(t) = \left[ 1 - f_{u}(n) \right] d_{u}^{(f,m)} + d_{u,f}^{(t)} + d_{u,f}^{(t)}. \]  

(17)

Thus, the average content delivery delay for all CVs is

\[
\bar{d}(t) = \frac{1}{U} \sum_{u=1}^{U} \sum_{t=1}^{T} I_{u}(t) \cdot d_{u}^{(t)}(t). \]  

(18)

B. Joint Problem Formulation

We aim to find joint cache placement \( I_{f}(n) \), user scheduling \( I_{u}(t) \), total W (t) VC to form, the VC configuration \( \mathcal{B}_{w}(W(t)) \), VC association \( I_{u}(t) \) and radio resource allocation for the serving APs in the selected VCs, i.e., \( I_{u}^{(f,t)}(t) \)'s to minimize long-term expected average content delivery delay for the CVs. As such, we pose our joint optimization problem as

\[
\begin{align*}
\text{minimize} \quad d = \limsup_{T \to \infty} E \left[ \frac{1}{T} \sum_{t=1}^{T} \bar{d}(t) \right] \\
\text{s. t.} \quad C_{1} : \sum_{u=1}^{U} \sum_{t=1}^{T} I_{u}(t) \leq 1, \quad \forall u, t \quad (19a) \\
C_{2} : (14), (15), \quad \forall n, \quad (19b) \\
C_{3} : \sum_{u=1}^{U} I_{u}(t) \leq W(t), \quad \forall t, \quad (19c) \\
C_{4} : \sum_{u=1}^{U} I_{u}(t) = 1, \quad \forall u, \quad (19d)
\end{align*}
\]

\( C_{5} : \sum_{u=1}^{U} I_{u}^{(f)}(t) = 1, \quad \forall i, \) \quad (19e)

\( C_{6} : \sum_{u=1}^{U} \sum_{t=1}^{T} I_{u}^{(t)}(t) = W(t), \) \quad (19f)

\( C_{7} : W(t) = \min \left\{ \sum_{u=1}^{U} I_{u}(t), W_{\max} \right\}, \) \quad (19g)

\( C_{8} : \sum_{b=1}^{B} I_{u}^{(f)}(b) = 1, \quad \forall z, u, \) \quad (19h)

\( C_{9} : \sum_{z=1}^{Z} I_{u}^{(f)}(z) = 1, \quad \forall b, u, \) \quad (19i)

\( C_{10} : \sum_{u=1}^{U} I_{u}^{(t)}(t) = 1, \quad \forall z, b, \) \quad (19j)

\( C_{11} : \sum_{u=1}^{U} \sum_{t=1}^{T} I_{u}^{(t)}(t) = Z, \) \quad (19k)

\( C_{12} : d_{u}^{(t)}(t) \leq \delta_{u}^{(t)}(t), \) \quad (19l)

\( C_{13} : I_{f}(n), I_{u}(t), I_{u}^{(f,t)}(t), I_{u}^{(f,m)}(t) \in \{0, 1\}, \) \quad (19m)

where \( W_{\max} \) is the maximum allowable number of VCs in the system. Constraint \( C_{1} \) ensures that each CV can request at most one content in each slot \( t \). The constraints (14) and (15) in \( C_{2} \) are due to physical storage limitations. Constraint \( C_{3} \) in (19c) restricts the total number of scheduled CVs to at max the total number of created VCs \( W(t) \). Constraints (19d), (19e) and (19f) make sure that each CV can get at max one VC, each VC is assigned to at max one CV and summation of all assigned VCs is at max the total number of available VCs, respectively. Besides, constraints \( C_{7} \) in (19g) restricts the total VCs \( W(t) \) to be at max the minimum of the total scheduled CVs and \( W_{\max} \). Furthermore, constraints (19h), (19i) and (19j) ensure that each AP can get at max one pRB, each pRB is assigned to at max one AP and each CV gets non-overlapping resources, respectively. Constraint \( C_{12} \) in (19k) ensures that all available radio resources are utilized. \( C_{12} \) is introduced to satisfy the entire payload delivery delay of a requested content to be within the edge server’s hard deadline \( d_{u}^{(t)}(t) \). Finally, constraints in (19m) are the feasibility space.

Remark 1. (Intuitions behind the constraints)

Constraint \( C_{1} \) incorporates CV’s content request, while \( C_{2} \) is for the cache placement at the edge server. Constraint \( C_{3} \) is for CV scheduling. Constraints \( C_{4} - C_{7} \) are for the user-centric RAT’s VC formation and associations. Besides, constraints \( C_{8} - C_{11} \) are for radio resource allocation. Moreover, \( C_{12} \) is introduced to satisfy the hard deadline for delivering the CV’s requested contents, which holds if \( \sum_{t=1}^{T} I_{u}(t) \cdot \kappa \cdot R_{u}(t) \geq S \), where \( \kappa \) is the transmission time interval (TTI).

Remark 2. The total delay associated with each content request, calculated in (17), depends on both cache placement and the RAT. More specifically, an efficient cache placement solution can minimize cache miss events, i.e., minimize \( d_{u}^{(f,m)} \). On the other hand, \( d_{u}^{(f,t)} \) and \( d_{u,f}^{(t)} \) depend on the CV scheduling, and total VC \( W(t) \). VC configuration \( \mathcal{B}_{w}(W(t)) \), CV-VC association \( I_{u}^{(f,t)}(t) \) and radio resource allocation \( I_{u}^{(f,m)}(t) \).

Note that the optimization problem in (19) is an average Markov decision process (MDP) over an infinite time horizon with different combinatorial optimization variables. The exact solution for this grand problem is hard to find. In the subsequent section, we will prove that even the reduced problems of this complex joint optimization are NP-hard. Moreover, the decision variables are different in different time slots. As such, we decompose the original problem into

\( d_{u}^{(t)}(t) \) but it does not have content \( f_{u} \) in its local cache storage is known as the cache miss.

\( 5 \)We assume that each miss event needs to be handled by the upper layers.

\( 6 \)When the edge server needs to serve content \( f_{u}^{(t)}(t) \) it
two sub-problems. The first sub-problem transforms the cache placement problem, which will be solved using a learning solution. The second sub-problem introduces a joint CV scheduling, total \( W(t) \) VC formation, association and resource allocation optimization problem for minimizing the average content delivery delay, given that the edge server knows the cache placement decisions. The learning solution for the cache placement depends on the following preliminaries of DRL.

C. Preliminary of Deep Reinforcement Learning

An MDP contains a set of states \( \mathcal{X} = \{ x_i \}_{i=1}^{\mathcal{F}} \), a set of possible actions \( \mathcal{A} = \{ m \}_{m=1}^{\mathcal{M}} \), a transition probability \( P_{mf}(m) \) from the current state \( x_i \in \mathcal{X} \) to the next state \( x_{i'} \in \mathcal{X} \) when action \( m \) is taken, and an immediate reward \( R_r(m) \) for this state transition [39]. RL perceives the best way of choosing actions in an unknown environment through repeated observations and is widely used for solving MDP. The RL agent learns policy \( \pi : \mathcal{X} \times \mathcal{A} \rightarrow [0, 1] \), where \( \pi(x_i, m) = \Pr\{m|x_i\} \) denotes the probability of taking action \( m \) given the agent is at state \( x_i \). Following \( \pi \), the agent takes the action \( m \) with the probability \( \pi(x_i, m) \), and then the agent observes \( s' \) and receives a reward \( r \) from that state onward. The expected return denotes how good it is to be at that state and is measured by the following state-value function:

\[
V_\pi(x_i) = \mathbb{E}[R|x_i, \pi] = \mathbb{E} \left[ \sum_{t'=t}^{T_{end}} \gamma^{t'-t} R_t(m)|x_t, \pi \right],
\]

where \( \gamma \in [0, 1] \) is the discount factor, \( T_{end} \) is the time step at which the episode ends, and \( R_t(m) \) is the reward at step \( t \). Moreover, (20) follows an MDP [39] and can be rewritten as \( V_\pi(x_i) = R_t(m) + \gamma \sum_{s' \in \mathcal{X}} P_{mf}(m)v_\pi(s') \). Moreover, the quality of an action taken at a state is ascertained by the following action-value function - known as the Q\((x, m)\) value [39]:

\[
Q(x_i, m) = R_t(m) + \gamma \sum_{s' \in \mathcal{X}} P_{mf}(m)v_\pi(s'),
\]

The agent’s goal is to find optimal policy \( \pi^* \) to maximize \( Q^*(x_i, m) = R_t(m) + \gamma \sum_{s' \in \mathcal{X}} P_{mf}(m)v^*(s') \), where \( V^*(s') = \max_m Q^*(x_i, m) \). Moreover, as the environment is unknown, the agent updates this Q value as [40]

\[
Q(x_i, m) \leftarrow (1 - \alpha)Q(x_i, m) + \alpha \tilde{y}_t Q(x_i, m) = \max_{m \in \mathcal{A}} Q(x_i, m)
\]

where \( \alpha \) is the learning rate and \( \tilde{y}_t = R_t(m) + \gamma \max_m Q^*(x_{i'}, m) \) is commonly known as the temporal target. With sufficient observations, (23) is guaranteed to converge to \( Q^*(x, m) \) [40].

When the number of states or actions is quite large, this classical RL becomes intractable and suffers from the curse of dimensionality [39]. A remedy to this is to approximate the action-value function using deep neural networks (DNN) [41], which is known as the so-called deep-RL (DRL). Let the DNN be parameterized by its weight \( \theta \). Denote the approximation by \( Q^*(x_i, m) = Q(x, m; \theta) \). The essential idea is to store the agent’s experiences \( \{ x_i, m, R_r, x_{i+1} \} \) into a memory buffer \( \mathcal{D} \). The DRL agent is then trained by randomly sampling \( S_\theta \) batches from the buffer \( \mathcal{D} \) and performing stochastic gradient descent (SGD) to minimize the following loss function [41]:

\[
L(\theta) = |\tilde{y}_t(\theta) - Q(x_i, m; \theta)|^2,
\]

where \( \tilde{y}_t(\theta) = R_t(m) + \gamma \max_m Q(x_{i'}, m; \theta) \). Note that while the same DNN \( \theta \) can be used to predict both \( Q(x_i, m; \theta) \) and the target \( \tilde{y}_t(\theta) \), this often creates increased oscillations or divergence of the policy \( \pi \) [41]. To increase learning stability, a separate target DNN - parameterized by \( \theta^- \) is used to predict \( \tilde{y}_t(\theta^-) \). This target DNN \( \theta^- \) is used in an offline manner, while the DNN \( \theta \) is used in each learning step \( t \) to predict \( Q(x_i, m) \) and defined as the online DNN. Moreover, in every \( \eta \) learning step, the weights of the offline DNN \( \theta^- \) get replaced by the weights of the online DNN weights \( \theta \) [41].

V. Problem Transformations

Our goal is to devise a learning solution for the cache placement policy (CPP) during slot \( t = nT \), which will be used to re-design the delay minimization problem.

A. Cache Placement Policy (CPP) Optimization Sub-Problem

We want to learn the CPP \( \pi_{ca} \) that provides the optimal cache placement decision \( I_{fc}(n) \) in the cache placement time slots \( t = nT \), \( \forall n \). We have total \( m_{ca} = \prod_{t'=1}^{T} m_{ca} = \prod_{t'=1}^{T} (\mathcal{F}) \) ways for content placement as \( \mathcal{A}' \) and \( \Lambda \) are of the unit of content size \( S \) based on constraints (14) and (15). Moreover, the CPP \( \pi_{ca} \) is a mapping between the state space \( x^n_{ca} \) and an action \( m_{ca} \) in the joint action space with \( m_{ca}(n) \) possible actions. To this end, let us define a cache hit event by

\[
I_{f}(t) = \begin{cases} 
1, & \text{if } \bar{I}_{f}(t) = 1 \text{ and } I_{fc}(n) = 1, \\
0, & \text{otherwise}.
\end{cases}
\]

Besides, the total cache hit at the edge server is calculated as the summation of the locally served requests and is calculated as \( h(t) = \sum_{n=1}^{U} I_{f}(t) \). Thus, we calculate the cache hit ratio (CHR) as

\[
\text{CHR}(t) = h(t)/\left( \sum_{n=1}^{U} I_{f}(t) \right).
\]

Next, we devise the CPP of the edge server that ensures a long-term CHR while satisfying the cache storage constraints. Formally, we pose the optimization problem as follows:

\[
\max_{\pi_{ca}} \text{CHR}(\pi_{ca}) = \lim_{T \to \infty} \mathbb{E} \left[ \frac{1}{T} \sum_{t=1}^{T} \text{CHR}(t) \right],
\]

\[\text{s. t. } C_1, C_2, I_{fc}(n) \in \{0, 1\}, \]

where \( C_1 \) and \( C_2 \) are introduced in (19).

Theorem 1. The CHR maximization problem (27) is NP-hard. Proof. Please see Appendix B.

B. Joint Optimization Problem for the User-Centric RAT

Note that as the delay of extracting a content from the cloud during a cache miss event is fixed, the first term in (17) will be minimized if the CPP \( \pi_{ca} \) ensures maximum CHR. In this sub-problem, we focus on the other two delays \( d_{n, fc}^{u} \) and \( d_{n, fc}^{d} \) in (17) by jointly optimizing scheduling, VC formation, VC association and radio resource allocation of the proposed user-centric RAT solution assuming the cache placement is known at the edge server. Therefore, we pose the following modified content delivery delay minimization problem.

\[
\min_{I_{u}(t) , W(t), \mathcal{X}(t), I_{f}(t), I_{f}(t), I_{f}(t)} \mathbb{E} \left[ \frac{1}{T} \sum_{t=1}^{T} \bar{d}(t) \right],
\]

\[\text{s. t. } C_3, C_4, C_5, C_6, C_7, C_8, C_9, C_{10}, C_{11}, C_{12}, \]

(28a)

(28b)

where the constraints in (28a) and (28b) are taken for the same reasons as in the original problem in (19).
Sub-problem (28) contains combinatorial optimization variables and, thus, is NP-hard. An exhaustive search for optimal parameters is also infeasible due to the large search space as well as sequential dependencies for the deadline constraints in \(C_{12}\). Besides, as each content request arrives with a deadline constraint, we consider that the edge server adopts priority-based scheduling. Intuitively, given the fact that the edge server does not know the transmission delay \(d_{f,c}\) due to channel uncertainty and it needs to satisfy constraint \(C_{12}\) for all \(I_c^f(t)\), it should schedule the CVs with earliest-deadline-first (EDF)\(^6\) followed by optimal VC formation, association and radio resource allocation. Note that EDF is widely used for scheduling in real-time operating systems [44]. If EDF cannot guarantee zero deadline violation for the tasks, no other algorithm can [43]. In our case, scheduling also depends on the availability of the requested content at the edge server. In cache miss event, the edge server must wait for \(d_{f,c}\), so that the upper layers can extract the content from the cloud.

Upon receiving a content request \(I_c^f(t)\), the edge server checks \(I_c(n)\). If \(I_c(n) = 0\), the request is forwarded to the upper layers. The upper layer initiates the extraction process from the cloud. At each slot \(t\), before making the scheduling and VC formation decisions, the edge server considers previous \(\mathcal{Z}_{\text{Sol}}\) slots information. These \(\mathcal{Z}_{\text{Sol}}\) slots are termed as our slots of interest (SoI) and are calculated in (29).

\[
P^t_{\text{Sol}} = \{ \min\{0, t - d_{f,max} + \zeta\} \}^{t_{f,max}}_{\zeta=1}.
\]

This SoI captures the previous slots that may still have undelivered payloads with remaining deadlines at the current slot \(t\). Denote the remaining time and payload for \(I_c^f(t)\) in current slot \(t\) by \(T_{u,\text{rem}}\) and \(P_{u,\text{rem}}\), respectively. Particularly, for all \(I_c^f(t)\), the edge server first checks whether the content is available at the edge server’s local cache storage or by the upper layers. If it is available, the edge server calculates the remaining deadline and payloads for the requests in all slots of \(\mathcal{Z}_{\text{Sol}}\). The edge server finds a set of candidate requestor CVs \(\mathcal{W}_{\text{val}}\subseteq \mathcal{W}\), their minimum remaining deadline set \(\mathcal{Z}_{\text{rem}}\) and corresponding left-over payload set \(\mathcal{P}_{\text{rem}}\). This procedure is summarized in Algorithm 1.

After extracting the valid CV set \(\mathcal{W}_{\text{val}}\), the edge server can formulate total \(W(t)\) VCs based on the following equation:

\[
W(t) = \min\{ |\mathcal{W}_{\text{val}}|, W_{\text{max}} \},
\]

where \(|\mathcal{W}_{\text{val}}|\) is the cardinality of the set \(\mathcal{W}_{\text{val}}\). Besides, the edge server calculates the priorities of the valid CVs set based on their remaining time using the following equation:

\[
\phi_u(t) = \frac{\phi_u(t)}{\sum_{u \in \mathcal{W}_{\text{val}}} \phi_u(t)},
\]

where \(\phi_u(t) = \frac{\mathcal{Z}_{\text{val}}^i[t] / \mathcal{Z}_{\text{rem}}^i[t]}{\mathcal{W}_{\text{val}}[u]}\). Note that (31) sets the highest priority to the CV with the least available deadline. The edge server then picks the top-\(W(t)\) CVs for scheduling based on the priorities \(\phi_u(t)\). Denote the scheduled CV set during slot \(t\) by \(\mathcal{W}_{\text{sch}}^t \subseteq \mathcal{W}\). Given that the edge server makes scheduling decisions based on top-\(W(t)\) priorities of (31), to satisfy the hard deadline constraint in \(C_{12}\), we aim to maximize a WSR, which is calculated as

\[
\bar{R}(t) = \sum_{u \in \mathcal{W}_{\text{sch}}^t} T_{u}(t) \cdot R_{u}(t) \cdot \phi_u(t),
\]

where the weights are set based on the CV’s priority \(\phi_u(t)\). Again, the intuition for this is that with the underlying RAT solution, due to channel uncertainty, the edge server expects to satisfy constraint \(C_{12}\) by prioritizing the CVs based on (31) and follow optimal VC configuration, their association and radio resource allocation. As such, we pose the following WSR maximization problem for the edge server:

\[
\begin{align*}
\text{maximize} & \quad \mathcal{P}_w^u(W(t)) \cdot \mathcal{L}^u(t) \cdot \bar{R}(t) \\
\text{subject to} & \quad C_4, C_5, C_6, C_8, C_9, C_{10}, C_{11}, \quad (33a)
\end{align*}
\]

where the constraints in (33a) and (33b) are taken for the same reasons as in the original problem in (19).

**Remark 3.** The edge server finds \(W(t)\) VCs and \(I_c^f(t)\)s using (30) and (31), respectively. Given that the contents are placed following \(\pi_{ac}\) during slot \(t = t\), the edge server knows \(W(t)\) and \(I_c^f(t)\)s, the joint optimization problem in (28) is simplified to a joint VC configuration, CV-VC association and radio resource allocation problem in (33).

VI. PROBLEM SOLUTION

A. Learning Solution for the CPP

To find the CPP \(\pi_{ac}\), the edge server uses some key information from the environment and learns the underlying environment dynamics. Recall that the CVs requests are modeled by the exploration and exploitation manner. At the beginning of each DOI, the edge server determines top-\(\Lambda^c\) popular contents in each class and also calculates top-\(\Lambda^c\) similar contents for each of these popular contents as

\[
\mathcal{F}_{\text{top}}^c(n)[c, f_c] = \begin{cases} 
1, & \text{if } f_c \text{ is top-}\Lambda^c \text{ similar content of } f_c^{\text{top}}, \\
0, & \text{otherwise},
\end{cases}
\]

where \(f_c^{\text{top}}\) is in top-\(\Lambda^c\) popular content list of class \(c\). Besides, the edge server also keeps track of the content requests coming from each CV and corresponding cache hit based on the stored content during the previous DOI. Let the edge server store the
content-specific request from CV $u$ into a $\mathbb{R}^{C \times F}$ matrix $\mathbf{P}_{\text{req}}(n)$ during all slots of the DoI. Similarly, let there be a matrix $\mathbf{P}_{\text{hit}}(n) \in \mathbb{R}^{C \times F}$ that captures content-specific cache hit $I_{\text{hit}}^{n}(t)$’s during all $t$ within the DoI. Furthermore, we also provide the measured popularity matrix $\mathbf{P}_{f}(n)$ during the current DoI based on the CVs requests in the previous DoI change interval $(n-1)$. As such, the edge server designs state $x_{ca}^{n}$ as the following tuple:

$$x_{ca}^{n} = \left\{ \left( \mathbf{P}_{\text{req}}(n) \right)_{U=1}^{U=U}, \left( \mathbf{P}_{\text{hit}}(n) \right)_{U=1}^{U=U}, \mathbf{P}_{f}(n), \mathbf{P}_{f}(n) \right\}.$$ (34)

The intuition behind this state design is to provide the edge server some context on how individual CV’s preferences and global content popularity may affect the overall system reward.

At each $t = nT$, the edge server takes a cache placement action $m_{ca}$ to prefetch the contents in its local storage. At the end of the DoI, it gets the following reward $r_{ca}^{n}$

$$r_{ca}^{n} = \frac{1}{T} \sum_{t=1}^{T} r_{ca}(t),$$ (35)

where $r_{ca}(t) = \sum_{c=1}^{C} \sum_{i=1}^{I} \sum_{n_{u}} r_{ca}(c, f, c_{u})$. Moreover, $r_{ca}(c, f, c_{u})$ is calculated in (36), where $\delta_{\text{pop}}$ and $\delta_{\text{hit}}$ are two hyper-parameters. Note that these hyper-parameters balance the cache hit for the top-$N$ contents and the other stored contents in the edge server’s cache storage. Empirically, we have observed $\delta_{\text{pop}} > \delta_{\text{hit}}$ works well.

$$r_{ca}(c, f, c_{u}) = \begin{cases} \delta_{\text{pop}} \cdot U \cdot \frac{1}{T_{u}}(t), & \text{if } \mathbf{P}(n)[c, f] = 1 \\ \delta_{\text{hit}} \cdot \frac{U}{T_{u}}(t), & \text{if } \mathbf{P}(n)[c, f] \neq 1 \text{ and } U_{l}(t) > 0, \\ -\sum_{f=1}^{F} U_{l}(t), & \text{otherwise}. \end{cases}$$ (36)

We consider that the edge server learns the CPP $\pi_{ca}$ offline. It uses two DNNs - $\theta_{ca}$ and $\theta_{ca}^{-}$, and learns $\pi_{ca}$ following the basic principles described in Section IV-C. During the cache placement slot $t = nT$, the edge server observes its state $x_{ca}^{n}$ and takes action following $\varepsilon$-greedy policy [39] using $\theta_{ca}$. Upon taking the action, it gets rewards $r_{ca}^{n}$ and the environment transits to the next state $x_{ca}^{n'}$. Moreover, the edge server stores its experiences tuple $\{x_{ca}^{n}, m_{ca}, r_{ca}^{n}, \theta_{ca}^{-}\}$ into its memory buffer $\text{mem}_{ca}$, which can hold at max $\text{mem}_{ca}$ number of samples. The edge server randomly samples $S_{ca}$ samples from $\text{mem}_{ca}$ in every $\eta_{ca}$ steps and uses the $\theta_{ca}$ and $\theta_{ca}^{-}$ to get the $Q(x_{ca}^{n}, m_{ca}, \theta_{ca})$ and the target value $y_{ca}^{n}(\theta^{-})$, respectively. It then trains the DNN $\theta_{ca}$ by minimizing the loss function shown in (24) using stochastic gradient descent. After $\eta_{ca}$ steps, the offline DNN $\theta_{ca}^{-}$ gets updated by $\theta_{ca}$. Algorithm 2 summarizes this CPP learning process.

### B. WSR Maximization

Recall that once the edge server determines $W(t)$, all possible VC configurations $\mathcal{B}_{ve}(W(t)) = \{ \mathcal{B}_{ve}(W(t)) \}_{u=1}^{u=U}$ can be generated following the VC formation rules defined in (1a)-(1c). Besides, each VC configuration $\mathcal{B}_{ve}(W(t))$ has exactly $W(t)$ number of VCs. Moreover, the edge server schedules $|\mathcal{S}_{ve}^{\text{sch}}| = W(t)$ CVs in each slot $t$ based on the priority $\phi_{ve}(t)$. Let the $i^{th}$ CV in $\mathcal{S}_{ve}^{\text{sch}}$ be assigned to the $i^{th}$ VC in $\mathcal{B}_{ve}(W(t))$. This assigns each CV to exactly one VC and all VCs are assigned to all scheduled CVs. Therefore, essentially, for a selected VC configuration $\mathcal{B}_{ve}(W(t))$, by assigning the VCs in the above mentioned way, the edge server can satisfy constraints C$_3$, C$_4$, C$_5$, and C$_6$. To this end, given that the selected VC configuration $\mathcal{B}_{ve}(W(t))$ and $I_{va}^{n}(t)$ are known at the edge server, we can rewrite (33) as follows:

$$\text{maximize } \bar{R}(t),$$ (37)

subject to $C_{8}, C_{9}, C_{10}, C_{11}, I_{va}^{n}(t) \in \{0, 1\}$. (37a) As the CSI is perfectly known at the edge server, it can choose maximal ratio transmission to design the precoding vector $w_{b}^{\text{ve}}$. In other words, given $I_{va}^{n}(t) = 1$, the edge server chooses $w_{b}^{\text{ve}}(t) = \frac{y_{va}^{n}(t)}{\|y_{va}^{n}(t)\|^2}$ received at CV $u$ from AP $b$ over pRB $z$ during slot $t$:

$$\left[ R_{[b, z]} = r_{2}(u, b, z) \cdot \phi_{ve}(t) \right] ;$$

Algorithm 3: Get Weighted Data Rate Matrix

```
Input: $\mathbf{P}_{u}(n), \mathbf{P}_{u}(n), \phi_{ve}(n)$, $H_{s}(n)$
1: Initialize $R_{u} = \text{zeros}(B \times Z)$;
2: for $a \in \mathcal{A}_{u}(n)$ do
3:  Get assigned VC $u$ using $\mathcal{B}_{ve}(t)$;
4:  for $b \in \mathcal{V}_{u}$ do
5:  for $z \in \mathcal{Z}_{b}$ do
6:  Calculate $r_{1}(u, b, z) = \log_{2}\left(1 + \frac{H_{s}(b, z)w_{b}^{ve}(t)}{\sigma_{b}^{2}}\right)$ received at CV $u$ from AP $b$ over pRB $z$ during slot $t$;
7:  $R_{[b, z]} = r_{1}(u, b, z) \cdot \phi_{ve}(t)$;
8: end
9: end
10: Output: $R_{u}$
```
Algorithm 4: Optimal VC Configuration and pRB Allocation

Input: $W(t)$, $\mathcal{W}_{sch}$: $\{\phi_{t}(i)\}_{i\in\mathcal{W}_{sch}}$, $\mathbf{I}_{t}^{(i)}$;
1. Get all VC configuration set $\mathcal{B}_{sch}(W(t)) = \{ \mathcal{B}_{sch}(W(t)) \}_{i=1}^{\mathcal{M}}$;
2. Initiate WSR vector $r_t = \text{zeros}(\mathcal{A}_{W}(t))$ and empty pRB allocation set $\mathbf{I}(t) = \emptyset$;
3. for $\mathcal{B}_{sch}(W(t)) \in \mathcal{B}_{sch}(W(t))$ do
4. Get $\mathbf{R}_{t}$ matrix from Algorithm 3 for this VC configuration $\mathcal{B}_{sch}(W(t))$;
5. Solve the MWBM problem using Hungarian algorithm [46] to get optimal pRB allocation set $\mathbf{I}_{t}^{(i)} = \{ \mathbf{I}_{t}^{(i)}(b) \}_{b=1}^{\mathcal{M}}$ and get the optimal sum-weights $r_{t}(b, z)$ from the optimal edges $e^{*}(b, z)$;
6. $\mathbf{R}_{t}[a] = r_{t}(a)$;
7. $\mathbf{I}(t).append(\mathbf{I}_{t}^{(i)})$;
end
9. Find the max($\mathbf{r}_{t}$) and corresponding index $a^{*}$;
10. Take best VC configuration $\mathcal{B}_{sch}(W(t))$ and corresponding optimal pRB allocation set $\mathbf{I}_{t}^{(a^{*})}$ and $\mathbf{I}_{t}^{(a^{*})}$;
Output: $\mathcal{B}_{sch}(W(t))_{t}$ and $\mathbf{I}_{t}^{(a^{*})}$

Algorithm 5: Content Delivery Model
1. Set $n = 0$;
2. for all $t$ do
3. Check if the requested contents are in the cache storage, if any upper layer for extraction from cloud;
4. Calculate Sol $\mathcal{S}_{sol}$ using (29);
5. Find eligible CV set $\mathcal{W}_{sch}$ using Algorithm 1;
6. Find total number of VC to formulate, i.e., $W(t)$ using (30);
7. Calculate eligible CVs’ priorities using (31);
8. Get the CV set $\mathcal{W}_{sch}$ to schedule by picking the top-$W(t)$ $\phi_{t}(s)$;
9. Find optimal VC configuration $\mathcal{B}_{sch}(W(t))$ and optimal pRB allocations $\mathbf{I}_{t}^{(a^{*})}$ by running Algorithm 4;
11. Based on VC configuration $\mathcal{B}_{sch}(W(t))$ and $\mathbf{I}_{t}^{(a^{*})}$ calculate CVs SNRs $\Gamma_{t}(s) = (\mathcal{B}_{sch}(W(t)))_{s \in \mathcal{W}_{sch}}$ using (9);
12. Calculate $\mathbf{R}_{t}^{(s)}(t)$ using (38) for all $s \in \mathcal{W}_{sch}$;
13. Offload $\mathbf{R}_{t}^{(s)}(t)$ bits from the remaining payloads of all CVs $u \in \mathcal{W}_{sch}$;
14. Update all $u \in \mathcal{W}_{sch}$ remaining payload and deadline;
end

monly known as the maximum weighted bipartite matching (MWBM) problem [45]. The edge server needs to find the set of edges $e^{*}(b, z) \in \mathcal{E}$ that maximizes the summation of the weights of the edges. Moreover, the edge server uses well-known Hungarian algorithm [46] to get the optimal edges $e^{*}(b, z)$, i.e., pRB allocations $\mathbf{I}_{t}^{(a^{*})}$ in polynomial time. This pRB allocation is, however, optimal only for the selected VC configuration $\mathcal{B}_{sch}(W(t))$. In order to find the best VC configuration $\mathcal{B}_{vc}(W(t))$, the edge server performs a simple linear search over all $\mathcal{A}_{W}(t)$ VC configurations. This procedure is summarized in Algorithm 4.

C. Content Delivery Process

Contents are placed using the trained CPP $\pi_{ca}$ during each cache placement slot $t = nT$, while the CVs make content requests in each $t$ following Section III-C. Upon receiving the $\mathbf{I}_{t}^{(a^{*})}(t)$’s, the edge server checks whether $I_{f}(n) = 1$ or $I_{f}(n) = 0$. If $I_{f}(n) = 1$, $f_{c}$ can be delivered locally. All cache miss events are forwarded to the VEN’s upper layers. The upper layer extracts each cache missed content from the cloud with an additional delay of $d_{msf}^{m}$. In all $t$, the edge server calculates the Sol $\mathcal{S}_{sol}$ using (29). It then finds the eligible CV set $\mathcal{W}_{sch}$ and forms total $W(t)$ VCUs using Algorithm 3 and (30), respectively. To that end, the edge server calculates

| Item/Description | Value |
|------------------|-------|
| Total number of APs $B$ | 6 |
| Maximum pRB per slot $w_{max}$ | 9 |
| THz | 1 ms |
| DoT update interval | 30 s |
| Carver frequency | 2 GHz |
| pRB size $a_{0}$ | 180 KHz |
| Noise power $d_{n}$ | -174 dBm/Hz |
| AP coverage radius | 250 m |
| Antenna $\frac{\lambda}{\theta}$ | 4 |
| AP antenna height | 25 m |
| CV antenna height | 1.5 m |
| Transmission power $P_{TX}$ | 30 dBm |
| AP transmitter antenna gain $G_{TX}$ | 8 dB |
| CV receiver antenna gain $G_{RX}$ | 3 dB |
| CV receiver noise figure $F_{RX}$ | 9 dB |
| Total content class $c$ | 10 |
| Contents per class $\sigma_{c}$ | 4 |
| Feature per content $\epsilon_{c}$ | 5 × $\times$ |
| AN cache size $A$ | $[3, 8, 9, 12] \times 8$ |
| Max allowable delay $d_{max}$ | 10.0 s |
| Content extraction delay $d_{extract}$ | 5.0 s |
| CV active probability $p_{a}$ | Uniform(0.1, 1.1) |
| CV’s inclination to similarity/popularity $p_{s}$ | Uniform(0.1, 1.1) |

Fig. 2. Simulated RoI
Fig. 3. CPP learning: average return during training

The priorities $\phi_{t}(s)$ using (31) and selects top-$W(t)$ CVs to schedule. Once the edge server knows $W(t)$, $\phi_{t}(s)$ and $\mathcal{W}_{sch}$, it runs Algorithm 4 to get the VC configuration and pRB allocations that maximizes the WSR of (33). Algorithm 4 returns the $\mathcal{B}_{sch}$ and $\mathcal{I}_{t}^{(a^{*})}(t)$ which can then be used to get the SNRs $\Gamma_{t}(s)$ from (9). Upon receiving the SNRs $\Gamma_{t}(s)$, the edge server can calculate the possible transmitted bits for the CVs as follows:

$$R_{t}^{(s)}(t) = \kappa \cdot R_{t}(t).$$

(38)

The edge server then delivers the remaining $t - d_{max}^{m} + \zeta$ sequentially. This entire process is summarized in Algorithm 5.

VII. PERFORMANCE EVALUATION

A. Simulation Setting

We consider $U$ CVs roam over a region of interest (RoI) and deploy $B = 6$ APs alongside the road to cover the entire RoI. Table II shows other key simulation parameters used in this paper. We consider a 300 meters by 200 meters Manhattan grid model [33] with two-way roads as shown in Fig. 2. For realistic microscopic CV mobility modeling, we use the widely known simulation of urban mobility (SUMO) [47]. The CVs are deployed with some initial routes with a maximum speed of 45 miles/hour and later randomly rerouted from the intersections on this RoI. In SUMO, we have used car-following mobility model [48] and extracted the CVs’ locations using the Traffic Control Interface [49] application programming interface.

To design our simulation episode, we consider 1000$\kappa$ milliseconds of CVs activities. For the CPP learning, the edge
server uses DNN $\theta_{ca}$ that has the following architecture: $\text{Conv2d} \rightarrow \text{Conv2d} \rightarrow \text{Linear} \rightarrow \text{Linear}$. We train $\theta_{ca}$ in each cache placement slots with a batch size $S_{ca} = 512$. Besides, we choose $\gamma = 0.995$, $\epsilon_{\text{max}} = 1$, $\epsilon_{\text{min}} = 0.005$, $\nu = 0.6$, $\text{Mem}_{\text{ca}}^{\text{init}} = 15000$, $T_{\text{epoch}} = 15000$, $\hat{\eta}_{ca} = 4T$. For training, we use Adam as the optimizer with a learning rate of 0.001. Using our simulation setup, the edge server first learns $\pi_{ca}$ using Algorithm 2 for $T_{\text{epoch}}$ episodes. The average per state returns during this learning is shown in Fig. 3. As the training progresses, we observe that the edge server learns to tune its policy to maximize the expected return. After sufficient exploration, the edge server is expected to learn the CPP that gives the maximized expected return. As a result, it is expected that the reward will increase as the learning proceeds. Fig. 3 also validates this and shows the convergence of Algorithm 2. As such, we use this trained CPP $\pi_{ca}$ for performance evaluations in what follows.

### B. Performance Study

We first show the performance comparisons of the learned CPP with the following baselines without any RAT solution.

**Genie-Aided cache replacement (Genie):** The to-be-requested contents are known beforehand during the start of the DoI provided by a Genie. In this best case, we then store the top-$\Lambda^c$ requested contents from all $c \in \calC$ in all $n$.

**Random cache replacement (RCR):** In this case, contents from each class are selected randomly for cache placement.

**K-Popular (K-PoP) replacement** [50]: In this popularity-based caching mechanism, we store the most popular $K = \Lambda^c$ contents during the past DoI for each content class $c \in \calC$.

**Modified K-PoP+LRU (K-LRU) replacement:** We modify the popularity-driven $K$-PoP with classical least recently used (LRU) [51] cache replacement. The least popular contents in the $K$-PoP contents are replaced by the most recently used but not in $K$-PoP contents to prioritize recently used contents.

To this end, we vary the cache size of the VEN and show the average CHR during an episode in Fig. 4. The general intuition is that when we increase the cache size $\Lambda$, more contents can be placed locally. Therefore, by increasing $\Lambda$, the average CHR is expected to increase. $K$-PoP and K-LRU do not capture the heterogeneous preferences of the CVs. Similarly, as contents are replaced randomly with the naive RCR baseline, it should perform poorly. However, when the cache size is relatively small, solely popularity-based K-PoP performs even worse than RCR. This means that popularity does not dominate the content demands of the CVs. Moreover, when the cache size becomes moderate, K-PoP and K-LRU outperform the naive RCR baseline. On the other hand, the proposed CPP aims to optimize $\pi_{ca}$ by capturing the underlying preference-popularity tradeoff of the CVs. Therefore, the average CHR is expected to be better than the baselines. Fig. 4 also reflects these analysis. Moreover, notice that the performance gap with the Genie-aided average CHR and our proposed CPP is lower. In the VEN, we do not know the future and CVs’ content demands. Therefore, we can only predict the future and tune the CPP $\pi_{ca}$ accordingly. Particularly, when the cache storage is reasonable, the performance gap of the proposed CPP is much lower. For example, at $\Lambda = 9$ and $\Lambda = 12$ the proposed CPP delivers around 93% and 98% of the Genie-aided solution. Moreover, the baselines perform poorly regardless of $\Lambda$. For example, at $\Lambda = 9$, the proposed CPP is around 49%, 23% and 24% better than RCR, K-PoP and K-LRU, respectively.

Fig. 5 shows the average CHR variation over 10 test episodes in 100 simulation runs and corresponding standard deviations. As expected, the performance of the proposed CPP is very close to the Genie-aided performance in these test runs. Particularly, the proposed CPP delivers around 98% of the Genie-aided performance. Moreover, the other baselines’ average CHRs largely deviate from the Genie-aided solution. We observe that the proposed CPP is around 52%, 16% and 14% better than RCR, K-PoP and K-LRU, respectively, even when $\Lambda$ is 80% of the content catalog $\calF$, which validate the effectiveness of the proposed method.

As content requests arrive following preference-popularity tradeoff, the CHR also gets affected by the total number of CVs in the VEN. Intuitively, as the CVs’ preferences are heterogeneous, when the total number of CVs in the VEN increases, the content requests largely diversify. Therefore, even with the Genie-aided solution, the CHR may degrade when the number of CVs in the VEN increases. This is also reflected in our simulated results in Fig. 6. The performance of the proposed CPP algorithm is stable regardless of the number of CVs in the VEN. We observe a slight performance gap between the CPP and the Genie-aided solution. This gap gets smaller and smaller as the total number of CVs in the VEN increases. Particularly, we observe that the proposed CPP delivers an average 97% CHR for the considered CV numbers. Besides, it delivers around 47%, 21% and 22% better performance than RCR, K-PoP and K-LRU, respectively. Therefore, we will use this CPP $\pi_{ca}$ to find $I_f(n)$ for all $n$ and show performance analysis of our proposed user-centric RAT solution.

To that end, we compare the performance of the proposed RAT solution with legacy network-centric RAT (NC-RAT). In the NC-RAT, a base station (BS) is located at a fixed suitable location which has $Z = 6$ pRBs and total transmission power of 46 dBm. We use the same scheduling and deadline-based priority modeling for the NC-RAT as the proposed user-centric case. Besides, we distributed the total transmission power proportionally to the scheduled CVs’ priorities. Moreover, we performed the same WSR maximization problem for getting the pRB allocation using Hungarian algorithm [46]. In the following, this legacy RAT solution is termed NC-RAT and used with the cache placement baselines. On the other hand, the ‘Proposed’ method uses the proposed CPP and user-centric RAT solution.

Intuitively, with an increased $\Lambda$, the edge server can store more contents locally which increases the total number of local delivery by assuring lower cache miss events. Therefore, with a proper RAT solution, the content delivery delay is expected to decrease if we increase the cache size of the edge server. We also observe this trend with both NC-RAT and our proposed user-centric RAT solution in Fig. 7. However, note that NC-RAT is inflexible, and depending on the location of the CVs, NC-RAT may not even have expected radio-
link qualities. This can, therefore, cause link failure and may increase the content delivery delay for the CVs’ requested content. On the other hand, the proposed user-centric RAT solution can design the appropriate VC configuration, VC associations and proper radio resource allocation to deliver the content timely. Therefore, we expect the user-centric RAT solution to outperform the traditional NC-RAT. Fig. 7 shows the average content delivery delay $d = \frac{1}{M} \sum_{i=1}^{M} d_i(t)$, where $d_i(t)$ is calculated in (18) with $W_{\text{max}} = 5$. As we can see, the proposed solution outperforms the baselines. Particularly, the average gain of the proposed solution on content delivery delay is around 15% over the baselines.

The effectiveness of the proposed solution is more evident in Fig. 8, which shows the percentage of deadline violations in a test episode when the content size is $S = 4$ KB. As a general trend, the deadline violations decrease as $\Lambda$ increases. Besides, among the cache placement baselines, as we have seen in the performance comparisons of the CPP, even RCR delivers lower deadline violations than solely popularity-based $K$-PoP when the cache size is small. Moreover, we observe around 28% higher deadline violations with the baseline NC-RAT over our proposed user-centric RAT solution. Recall that this deadline violation is essentially the violation of constraint $C_{12}$, which means the requester CVs have not received the requested content by their required deadlines. As such, these requester CVs may experience fatalities and degraded QoEs with the existing RAT and cache placement baselines.

Content size $S$ also affects the delivery delays and corresponding deadline violations. Intuitively, content delivery delay shall increase if the payload increases when the network resources are unchanged. This also increases the likelihood of deadline violations. Fig. 9 shows how the delivery delay gets affected by content size $S$. Note that transmission delay is directly related to channel quality between the transmitter and receiver. This channel uncertainty can cause fluctuations in the content delivery delay. However, the general expectation is that...
the content delivery delay will increase when the payload size increases. We also observe these in Fig. 9. Particularly, when $S = 2.5$ KB, the performance gain of the proposed solution is around 30% over the RCR+NCRAT and around 27% over the K-PoP+NCRAT and K-LRU+NCRAT baselines.

Recall that delay cannot exceed the hard deadline. Therefore, higher content delivery delay leads to deadline violations. Fig. 10 shows how the payload size affects the deadline violations in the proposed VEN. As expected, even when the payload size is small, we observe that the legacy NC-RAT solution cannot ensure guaranteed delivery within the deadline. Moreover, when $S$ increases, the deadline violation percentage of our proposed solution performs significantly better than the NCRAT-based baselines. For example, when $S = 4$ KB, the deadline violation percentage with our proposed solution is around 12%, whereas the NCRAT-based baselines have around 47% deadline violations. From Fig. 7 - Fig. 10, we can clearly see that the traditional NC-RAT is not sufficient to deliver the demands of the CVs.

To that end, we show the efficacy of the proposed RAT solution by considering all cache placement baselines accompanied by the proposed RAT solution for delivering the requested contents of the CVs. Fig. 11 shows how the content delivery delay gets affected by different cache sizes. Particularly, the proposed solution delivers requested contents around 14%, 7% and 8% faster than the RCR+Proposed-RAT, K-PoP+Proposed-RAT and K-LRU+Proposed-RAT, respectively, when $\Lambda = 9$. Recall that the proposed CPP (without RAT) had a performance gain of around 49%, 23% and 24% over the traditional NC-RAT. Therefore, $d$ is expected to increase if $\Lambda$ is increased, which significantly compensates for the cache miss events.

Moreover, Fig. 12 shows delay vs total number of CVs $U$ in the VEN. Intuitively, if $U$ increases, the edge server receives a larger number of content requests. Then, with the limited VCs, the edge server can at max schedule only $W_{\max}$ number of CVs. Therefore, $d$ is expected to increase if $U$ increases, which is also reflected in Fig. 12. Notice that in both Fig. 11 and Fig. 12, while the proposed solution outperforms the other cache placement baselines, the performance gaps are small because all cache placement baselines now use our proposed user-centric RAT solution for delivering the requested contents.

VIII. CONCLUSION

Considering the higher automation demand on the road, in this paper, we propose a user-centric RAT solution for delivering the CVs’ requested content with a learning solution for the cache placement. From the results and analysis, we can conclude that existing cache placement baselines may not be sufficient to capture the heterogeneous demands and preferences of the CVs. Moreover, the existing NC-RAT may cause severe fatalities on the road as it yields frequent deadline violations. Even for continuous deadline-constrained demand arrivals in each time slot, the proposed software-defined user-centric RAT solution has shown significant potential for offloading the payloads timely. The results suggest that our proposed cache placement policy delivers practical near-optimal cache hit ratio while the proposed user-centric RAT efficiently delivers the requested contents within the allowable deadline.

APPENDIX

A. Proof of Proposition 1

Assuming $t > 0$, we write the following:

\[
\Pr \{ \Psi_t \geq \xi \} = \Pr \left\{ e^{\Psi_t} \geq e^{\xi} \right\} = \frac{E \left\{ e^{\Psi_t} \right\}}{e^{\xi}}, \quad (39a)
\]

\[
e^{-t \xi} \sum_{u=1}^{U} \left( 1 - p_u + p_u e^{\xi} \right)^{U} \leq e^{-t \xi} \left( 1 - \bar{p} + \bar{pe}^{\xi} \right)^{U}, \quad (39c)
\]

\[
= \exp \left[ -t \xi + U \ln \left( 1 - \bar{p} + \bar{pe}^{\xi} \right) \right], \quad (39d)
\]

where (a) follows Markov inequality, (b) is true as $\Theta_{u}$ are independent and identically distributed, (c) follows as $E \left\{ e^{\Psi_t} \right\}$ is the moment generating function of $\Theta_t$, and (d) is obtained following the inequality of arithmetic and geometric means.

To this end, we find $e^{\xi} = \frac{\xi \left( 1 - \bar{p} \right)}{\bar{p} U^{\xi}}$ that minimizes (39). Plugging this value in (39), we obtain the bound as

\[
\Pr \{ \Psi_t \geq \xi \} \leq \exp \left[ U \left\{ \ln \left( \frac{1 - \bar{p}}{1 - \chi} \right) - \chi \ln \left( \frac{\chi}{\bar{p}} \right) \right\} \right], \quad (40a)
\]

\[
= \exp \left[ -UD_{\bar{p}} (\chi) \right], \quad (40c)
\]

where $\chi = \frac{\xi}{\bar{p}}$ in (a) and $D_{\bar{p}} = \chi \ln \left( \frac{\chi}{\bar{p}} \right) + (1 - \chi) \ln \left( \frac{\chi}{1 - \bar{p}} \right).

B. Proof of Theorem 1

We show that an instance of our problem in (27) reduces to an instance of a well-known NP-hard problem. Particularly, we only consider a single cache placement step $t = n\Gamma$ and assume that $I_{i,k}(n), \forall i \in \{ \mathcal{N}, (n+1)\mathcal{N} \}$ are known at the edge server beforehand\footnote{This assumption is only for the sake of this proof. The edge server does not know the future.}. Then, we re-write our (27) instance as

\[
\max_{I_{i,k}(n), \forall i \in \mathcal{N}, f_{i,k} \in \mathcal{F}} \sum_{i \in \mathcal{N}, (n+1)\mathcal{N}} \operatorname{CHR}(t), \quad (41)
\]

\[
\sum_{k=1}^{c} \sum_{f_{i,k} \in \mathcal{F}_i} S \cdot I_{i,k}(n) \leq \Lambda, \forall i \in \mathcal{C}, \quad (41a)
\]

\[
i_{f_{i,k}}(n) \in \{ 0, 1 \}, \forall c = 1, \ldots, C; f_{i,k} \in \mathcal{F}_i, \quad (41b)
\]

where the constraints are taken for the same reasons as in (27).

To that end, if $\mathcal{A}_{\mathcal{N}} = \mathcal{S} = 1$, we could rewrite the second constraint as $\sum_{f_{i,k} \in \mathcal{F}_i} I_{i,k}(n) = 1$. Then, it is easy to recognize that an instance of the well-known multiple-choice knapsack problem (MCKP) [52] has reduced to this instance of our CHR maximization problem. As MCKP is a well-known NP-hard problem [52], we conclude that the cache placement problem for each $t = n\Gamma$ is NP-hard even when the to-be requested contents are known beforehand. As such, the long-term policy optimization problem in (27) is NP-hard.

REFERENCES

[1] “Automated Vehicles for Safety,” United States Department of Transportation, National Highway Traffic Safety Administration, Accessed: February 17, 2022. [Online]. Available: https://www.nhtsa.gov/technology-innovation/automated-vehicles-safety
[2] “Connected and Automated Vehicles in the UK: 2020 information booklet,” centre for Connected and Autonomous Vehicles, Department for Transportation, UK, Accessed: February 17, 2022. [Online]. Available: https://www.gov.uk/government/publications/connected-and-automated-vehicles-in-the-uk-2020-information-booklet

[3] M. Noor-A-Rahim, Z. Liu, H. Lee, M. O. Khyam, J. He, D. Pesch, K. Monninger, L. Sun, and H. V. Poor, “QG for vehicle-to-everything (V2X) communications: Enabling technologies, challenges, and opportunities,” arXiv preprint arXiv:2012.07753, 2020.

[4] Y. F. Payalan and M. A. Guevans, “Towards next-generation vehicles featuring the vehicle intelligence,” IEEE Trans. Intelligent. Transport. Syst., vol. 21, no. 1, pp. 30–47, June 2020.

[5] “I-n-base® connected and connected vehicles: Current situation, future directions, and challenges,” IEEE Commun. Stand. Mag., vol. 2, no. 3, pp. 59–65, Sept. 2018.

[6] S. Chandupatla, N. Yerram, and B. Achuthan, “Augmented reality projection for driver assistance in autonomous vehicles,” SAE Technical Paper, Tech. Rep., 2020.

[7] M. F. Pervej, L. T. Tan, and R. Q. Hu, “User preference learning aided collaborative edge caching for small cell networks,” in Proc. IEEE Globecom, Dec. 2020.

[8] 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 Commun. Surveys Tutor., vol. 20, no. 4, pp. 3098–3130, May 2018.

[9] M. F. Pervej, L. T. Tan, and R. Q. Hu, “Artificial intelligence assisted collaborative edge caching in small cell networks,” in Proc. IEEE Globecom, Dec. 2020.

[10] L. Zhao, H. Li, N. Lin, M. Lin, C. Fan, and J. Shi, “Intelligent content caching strategy in autonomous driving toward 6g,” IEEE Trans. Intell. Transport. Syst., pp. 1–11, Sept. 2021.

[11] W. Qi, Q. Li, Q. Song, L. Guo, and A. Jamalipour, “Extensive edge intelligence for future vehicular networks in 6g,” IEEE Wireless Commun., vol. 28, no. 4, pp. 128–135, Mar. 2021.

[12] P. Wu, L. Ding, Y. Wang, L. Li, H. Zheng, and J. Zhang, “Performance analysis for uplink transmission in user-centric ultra-dense v2i networks,” IEEE Trans. Vehicular Technol., vol. 69, no. 9, pp. 9342–9355, Mar. 2020.

[13] M. F. Pervej and S.-C. Lin, “Dynamic power allocation and virtual cell formation for Throughput-Optimal vehicular edge networks in highway transportation,” in Proc. IEEE ICC Workshops, June 2020.

[14] Y. Zhou and Y. Zhang, “User-centric data communication service strategy for 5g vehicular networks,” IET Com., Jul. 2021.

[15] Z. Cheng, D. Zhu, Y. Zhao, and C. Sun, “Flexible virtual cell design for ultradense networks: A machine learning approach,” IEEE Access, vol. 9, pp. 91 575–91 583, June 2021.

[16] T. Şahin, M. Klügel, C. Zhou, and W. Kellerer, “Virtual cells for 5G v2x communications,” IEEE Commun. Standards Mag., vol. 2, no. 1, pp. 22–28, Mar. 2018.

[17] M. F. Pervej and S.-C. Lin, “Eco-Vehicular edge networks for connected transportation: A distributed multi-agent reinforcement learning approach,” in Proc. IEEE VTC2020-Fall, Oct. 2020.

[18] D. Kreutz, F. M. V. Ramos, P. E. Verissimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig, “Software-defined networking: A comprehensive survey,” Proc. IEEE, vol. 103, no. 1, pp. 14–76, Dec. 2015.

[19] X. Huang, K. Xu, Q. Chen, and J. Zhang, “Delay-aware caching in internet of vehicles networks,” IEEE Internet Things J., 2021.

[20] Z. Nan, Y. Jia, Z. Ren, Z. Chen, and L. Liang, “Delay-aware content delivery with deep reinforcement learning in internet of vehicles,” IEEE Trans. Intell. Transport. Syst., 2021.

[21] S. O. Somuyiwa, A. György, and D. Gündüz, “A reinforcement-learning approach to proactive caching in wireless networks,” IEEE J. Sel. Areas Commun., vol. 36, no. 6, pp. 1313–1344, 2018.

[22] T. P. Anderson, K. G. Nagananda, D. Gündüz, and H. V. Poor, “Caching with time-varying popularity profiles: A learning-theoretic perspective,” IEEE Trans. Commun., vol. 66, no. 9, pp. 3837–3847, 2018.

[23] Y. H. Wang, “On the number of successes in independent trials,” Statistica Sinica, pp. 295–312, 1993.

[24] R. S. Sutton and A. G. Barto, Reinforcement learning: An introduction. MIT press, 2018.

[25] C. J. Watkins and P. Dayan, “Q-learning,” Machine learning, vol. 8, no. 3-4, pp. 279–292, 1992.

[26] V. Mnih et al., “Human-level control through deep reinforcement learning,” nature, vol. 518, no. 7540, pp. 529–533, 2015.

[27] T. Guo and A. Suárez, “Enabling 5G ran slicing with edf slice scheduling,” IEEE Trans. Vehicular Technol., vol. 68, no. 3, pp. 2865–2877, May 2019.

[28] P. Guan and X. Deng, “Maximize potential reserved task scheduling for urllc transmission and edge computing,” in Proc. VTC2020-Fall, 2020.

[29] G. C. Buttazzo, Hard real-time computing systems: predictable scheduling algorithms and applications. Springer Science & Business Media, 2011, vol. 24.

[30] D. B. West et al., Introduction to graph theory. Prentice hall Upper Saddle River, 2001, vol. 2.

[31] H. W. Kuhn, “The hungarian method for the assignment problem,” Naval research logistics quarterly, vol. 2, no. 1-2, pp. 83–97, 1955.

[32] P. A. Lopez, M. Behrisch, L. Bieker-Walz, J. Erdmann, Y. Flößertörd, R. Hilbrich, L. Lücke, J. Rummel, P. Wagner, and E. Wiesner, “Traffic micro-simulation using sumo,” in Proc. ITSC, Nov. 2018.

[33] R. R. Roy, Handbook of mobile ad hoc networks for mobility models. Springer, 2011, vol. 170.

[34] A. Wegener, M. Piorkowski, M. Rayya, H. Hellbrück, S. Fischer, and J.-P. Hubaux, “Traci: an interface for coupling road traffic and network simulators,” in Proc. CNSN, 2008.

[35] T. Jiang, R. Wang, M. Ghaffari, M. F. Alhamid, and M. Guizani, “Heterogeneous information network-based content caching in the internet of vehicles,” IEEE Trans. Vehicular Technol., vol. 68, no. 10, pp. 10216–10226, Aug. 2019.

[36] S. Podlipnig and L. Bösžeményi, “A survey of web cache replacement strategies,” ACM Comput. Surv., vol. 35, no. 4, pp. 374–398, dec 2003.

[37] H. Kellerer, U. Pferschy, and D. Pisinger, “Multidimensional knapsack problems,” in Knapsack problems, Springer, 2004, pp. 235–283.