Deep Reinforcement Learning and Permissioned Blockchain for Content Caching in Vehicular Edge Computing and Networks

Yueyue Dai, Du Xu, Ke Zhang, Sabita Maharjan, *Senior Member, IEEE* and Yan Zhang, *Fellow, IEEE*

Abstract—Vehicular Edge Computing (VEC) is a promising paradigm to enable huge amount of data and multimedia content to be cached in proximity to vehicles. However, high mobility of vehicles and dynamic wireless channel condition make it challenge to design an optimal content caching policy. Further, with much sensitive personal information, vehicles may be not willing to caching their contents to an untrusted caching provider. Deep Reinforcement Learning (DRL) is an emerging technique to solve the problem with high-dimensional and time-varying features. Permission blockchain is able to establish a secure and decentralized peer-to-peer transaction environment. In this paper, we integrate DRL and permissioned blockchain into vehicular networks for intelligent and secure content caching. We first propose a blockchain empowered distributed content caching framework where vehicles perform content caching and base stations maintain the permissioned blockchain. Then, we exploit the advanced DRL approach to design an optimal content caching scheme with taking mobility into account. Finally, we propose a new block verifier selection method, Proof-of-Utility (PoU), to accelerate block verification process. Security analysis shows that our proposed blockchain empowered content caching can achieve security and privacy protection. Numerical results based on a real dataset from Uber indicate that the DRL-inspired content caching scheme significantly outperforms two benchmark policies.

Index Terms—Deep Reinforcement Learning, Permissioned Blockchain, Content Caching, Vehicular Edge Computing

I. INTRODUCTION

With the rapid development of in-car touchscreen and autonomous driving systems (e.g., Tesla Autopilot), huge amount of data and content generated by in-vehicle sensors and vehicular infotainment applications [1], [2]. However, long distance between vehicles and cloud servers, and the limited backhaul link capacity pose significant challenges for supporting massive content delivery while also satisfying the low-latency requirement in vehicular networks [3]. Vehicular Edge Computing (VEC) is a promising paradigm where Base Stations (BS) and vehicles with a certain amount of computation resource and caching resource can be utilized as edge

servers to cooperatively cache content at the network edge [4], [5], [6], [7].

Caching content at edge servers can effectively alleviate mobile traffic on backhaul links and reduce content delivery latency [8]. The authors in [9] proposed to cache content on femto-cell base stations to minimize the total expected content delivery delay based on a given popularity distribution. Since state-of-the-art vehicles are equipped with a certain amount of caching resource, the vehicle with sufficient caching resource can be regarded as a caching provider to expand the caching capacity of the network edge. Vehicle-to-vehicle communication can further reduce average content transmission latency [10], [11], [12]. However, high mobility of vehicles leads to dynamic network topology and time-varying wireless channel condition which makes it difficult to design an optimal content caching policy [13], [14]. Moreover, a content usually involves much sensitive personal information of its generator such that vehicles may be not willing to store their contents to an untrusted caching provider.

Deep Reinforcement Learning (DRL) is an emerging technique which has the ability to learn and build knowledge about dynamic wireless communication environment [15]. By interacting with edge servers, the authors in [15] utilized DRL to observe the available computing and caching resource at the network edge and design the corresponding resource allocation scheme. Exploiting actor-critic reinforcement learning, the authors in [16] proposed a scheme to solve the joint content caching, computation offloading, and resource allocation problems in fog-enabled Internet of Things (IoT) networks. The authors in [17] proposed a deep Q-learning based task offloading scheme to select an optimal edge server for vehicles to maximize task offloading utility. However, security and privacy are not considered in the above works.

Blockchain is an open database which maintains an immutably distributed ledger to enable securely transactions among distributed entities without relying on a central intermediary [18], [19], [20]. Blockchain can be categorized into two main types: public blockchain and permissioned blockchain. In public blockchain, anyone can participate in the process of verifying transactions and creating blocks due to no access limitation, such as Bitcoin and Ethereum. In permissioned blockchain, only permissioned nodes can verify transactions and create blocks. The typical consensus in public blockchain is computation-intensive because nodes compete against each other for creating newly blocks by solving a difficult PoW puzzle. Because of no competitive PoW puzzle, permissioned

blockchain can build a distributed ledger with less energy and computation resource. On the other hand, in public blockchain, all distributed nodes have to participate in the process of consensus which results in a long time consumption. In contrast, in permissioned blockchain, the number of nodes participating in the consensus is quite few, such that this type of blockchain can achieve a very fast consensus. Thus, permissioned blockchain is suitable for energy-constrained and delay-sensitive networks. Because of the limited energy and computation resources of vehicles and the stringent delay requirement of applications, the existing work often select permissioned blockchain with low energy consumption and short consensus delay for vehicular networks [19], [21], [22], [23].

In this paper, we integrate DRL and permissioned blockchain into vehicular networks to propose an intelligent and secure content caching scheme. We first propose a blockchain empowered distributed and secure content caching framework where vehicles act as caching requesters and caching providers to perform content caching and BSs act as verifiers to maintain permissioned blockchain. Due to high mobility of vehicles, we exploit the advanced DRL approach to learn dynamic network topology and time-varying wireless channel condition and then design an optimal content caching scheme between caching requesters and caching providers. We utilized permissioned blockchain to ensure a secure content caching among vehicles. To enable a fast and efficient blockchain consensus mechanism, we propose to select block verifiers based on Proof of Utility (PoU). The main contributions of this paper are summarized as follows:

- We propose a blockchain empowered distributed and secure content caching framework where vehicles perform content caching and BSs maintain permissioned blockchain to ensure an intelligent and secure content caching.
- We formulate the content caching problem as the form of DRL to maximize content caching with taking vehicular mobility into account and design a new DRL-inspired content caching scheme.
- We design a new block verifier selection method to enable a fast and efficient blockchain consensus mechanism. Security analysis shows that our proposed blockchain empowered content caching can achieve security and privacy protection. Numerical results based on a real dataset demonstrate the effectiveness of the proposed DRL-inspired content caching scheme.

The remainder of this paper is organized as follows. We introduce the architecture of blockchain empowered content caching in Section III. Then, we propose a DRL-inspired content caching scheme and introduce PoU consensus in Section IV and Section V, respectively. We present the security analysis and numerical results in Section VI. Finally, we conclude this paper in Section VII.

II. RELATED WORK

Recently, blockchain technology has attracted enormous attention of researchers and developers because of its feature such as decentralization, immutability, anonymity, and security. The authors in [21] proposed a neural-blockchain based drone-caching approach in unmanned aerial vehicles where blockchain ensures the high reliable communication among drones. The authors in [24] proposed a blockchain-based proactive caching in hierarchical wireless networks to enable autonomous caching-delivery among untrustworthy parties. The authors in [25] proposed a decentralized data management scheme for vehicular networks based on blockchain. However, these works establish blockchain by solving the meaningless PoW puzzle or Proof-of-Stake (PoS). The characteristics of nodes in vehicular edge computing networks, such as computing ability or QoS requirement, are not involved in the process of blockchain establishment.

To make a better integration of blockchain and vehicular networks, a few studies have utilized PoX consensus to replace the original PoW schemes. Because of no competitive PoW puzzle, the authors in [26] indicated that Delegated Proof-of-Stake (DPoS) is particularly suitable for lightweight vehicles to establish blockchain-based transportation systems. As a further exploration of [26], the authors in [27] proposed an enhanced DPoS consensus for a blockchain-based vehicular data sharing system, where reputation is used in the DPoS to measure the quality of RSUs. The authors in [23] and [28] also utilized reputation-based consensus to build blockchain for vehicular networks or cellular networks. The authors in [29] utilized proof-of-driving based blockchain to enable an intelligent vehicular data sharing among vehicles. The authors in [30] utilized proof-of-integrity in vehicular blockchain to manage the collected vehicle-related data from hundreds of sensors for a privacy-aware traffic accident diagnosis. However, the above researches are not suitable for vehicular edge computing networks, which are still not considering the limited computation resource of edge nodes and the stringent latency requirement s of users. In this paper, we proposed a proof-of-utility based consensus for vehicular edge caching, where the utility is comprehensive function to measure the computing and processing abilities of edge nodes and the latency requirements of vehicles.

III. BLOCKCHAIN AND ARTIFICIAL INTELLIGENCE CONTENT CACHING

In this section, we first present the proposed blockchain empowered vehicular content caching architecture, and then describe the detailed phases of the proposed blockchain empowered vehicular content caching.

A. Architecture of Vehicular Content Caching with Blockchain

We propose a new blockchain empowered content caching architecture which consists of a user plane and an edge plane, as illustrated in Fig. 1.

In the user plane, vehicles, equipped with multiple sensors and applications, can collect a variety of valuable content about vehicles, roads and their surrounds, such as entertainment videos, road maintenance information, parking lot occupancy and so on. Since state-of-the-art vehicles have a certain amount of caching resource, they can cache their content

Fig. 1: Blockchain empowered vehicular content caching

locally. However, due to the capacity limitation of caching resource, when a resource-constrained vehicle cannot store its collected content on its own cache, anyone of its neighbor can act as a caching provider to offer its unoccupied caching resource for content caching via Vehicle-to-Vehicle (V2V) communication. To encourage vehicles to contribute their unused caching resource, we utilize incentive mechanisms to motivate vehicles to participate in V2V content caching.

In the edge plane, several BSs are distributed in a specific area to work as edge servers with communication, computing capability, and AI functions. BSs can detect available caching resource of vehicles and deliver caching requests to the caching provider. In addition, BS can utilize computing capabilities and AI functions to predict the V2V transmission range and connection duration between caching requesters and caching providers, and perform caching pair matching to enhance system utility. There exists a central authority in the edge plane to manage the security parameters and keys of BSs and vehicles with a tamper-resistant hardware.

Caching content at vehicles can enhance spectrum utilization and reduce average content delivery latency between vehicles. However, since a content involves much sensitive and critical personal information of its generator, caching requesters are not willing to store their content to untrusted caching providers. To cope with this, each BS is equipped with a blockchain to enable untrustworthy vehicles to interact with each other for content caching in a secure manner.

B. Blockchain-based Vehicular Content Caching

In Fig. 1, there are two types of vehicles driving on the road. We define the vehicle requiring caching resource to store its content as caching requester and define the vehicle to provide caching resource as caching provider. Based on blockchain, a secure vehicular content caching can be achieved through the following phases.

1) **Identity Establishment and System Initialization:**

To implement V2V content caching, vehicles should register unique accounts and create their keys firstly. We utilize elliptic curve digital signature algorithm and asymmetric cryptography to establish identity. Specifically, vehicles and BSs register a legitimate identity after passing the authentication of the central authority. The legitimate identity consists of a public key, a private key, and a certificate, which can be described as $\{PK_{v_i}, SK_{v_i}, Cert_{v_i}\}$. The public key is regarded as the source address of the caching transaction which is used to verify the genuineness of transactions. The cryptographic private key is used to sign a transaction and the certificate is to uniquely identify the vehicle through binding registration information of the vehicle.

Each vehicle has a wallet. The wallet address is generated from its public key. At the system initialization stage, each vehicle requests the wallet addresses of other vehicles from the central authority. Specifically, each vehicle uploads its wallet address to a global account pool and then downloads other vehicles' wallet address for content caching from there. Note that vehicles can use changeable wallet address to preserve anonymity and privacy.

2) **Triggering Content Caching Smart Contract:**

For content caching, each BS gathers all vehicles' caching requests and monitors their available caching resource under its coverage. Vehicle v_i sends its caching request to the nearest BS b_j. The caching request message of vehicle v_i includes the required caching resource c_{v_i}, current location loc_{v_i}, public key PK_{v_i}, signature Sig_{v_i}, certificate $Cert_{v_i}$, and timestamp ts, which can be described as

$$Req^{v_i \to b_j} = E_{PK_{b_j}}(c_{v_i}||loc_{v_i}||PK_{v_i}||Sig_{v_i}||Cert_{v_i}||ts), \tag{1}$$

where $E_{PK_{b_j}}$ denotes that message $Req^{v_i \to b_j}$ is encrypted with public key PK_{b_j}, $Sig_{v_i} = Sign_{SK_{v_i}}(c_{v_i}||loc_{v_i})$ denotes that the digital signature of c_{v_i} and loc_{v_i} is with private key SK_{v_i}, and ts is the timestamp of the current message.

Vehicle v_p periodically sends its available caching resource to the nearest BS b_j for caching resource sharing. The message about available caching resource includes the available caching resource C_{v_p}, location loc_{v_p}, public key PK_{v_p}, signature Sig_{v_p}, certificate $Cert_{v_p}$, and timestamp ts, which can be described as

$$Mes^{v_p \to b_j} = E_{PK_{b_j}}(C_{v_p}||loc_{v_p}||PK_{v_p}||Sig_{v_p}||Cert_{v_p}||ts), \tag{2}$$

where $E_{PK_{b_j}}$ denotes that message $Mes^{v_p \to b_j}$ is encrypted with public key PK_{b_j}, $Sig_{v_p} = Sign_{SK_{v_p}}(C_{v_p}||loc_{v_p})$ denotes that the digital signature of C_{v_p} and loc_{v_p} is with private key SK_{v_p}, and ts is the timestamp of the current message.

After receiving caching requests and available caching resource of vehicles, BSs first verify their identity. To speed up the verification process, BSs adopt batch verification process which can verify the validity of a number of identities simultaneously [31]. Specifically, each BS abstracts the verification parameters from the received $Req^{v_i \to b_j}$ and $Mes^{v_p \to b_j}$ and constructs the verification parameters as $< PK, Mes, Sig >$, where Mes is the detailed message. Then, BS calls the batch verification algorithm for identity verification. If all $Ver(PK_{v_i}, Mes_{v_i}, Sig_{v_i}) = 1$ for all $i \in \mathcal{I} \cup \mathcal{P}$, we have $Batch((PK_{v_1}, Mes_{v_1}, Sig_{v_1}), ..., (PK_{v_i}, Mes_{v_i}, Sig_{v_i}), ...) = 1$ and batch verification is passed. If one or more than one of signatures are invalid, batch verification fails. After the batch verification, BSs perform V2V content caching mechanism to make caching pair

matching. More details on vehicular content caching will be given in Section IV.

When vehicular content caching mechanism is completed, each BS responses a message to caching requester and caching provider, respectively. $Resp_{req}^{b_j \to v_i}$ is the message that BS b_j responding to caching requester v_i and $Resp_{pro}^{b_j \to v_r}$ is the message that BS b_j responding to caching provider v_p, which can respectively be described as

$$
\begin{aligned}
Resp_{req}^{b_j \to v_i} &= E_{PK_{v_i}}(loc_{v_p}||chan_{ip}||PK_{v_p}||Sig_{b_j}||ts), \\
Resp_{pro}^{b_j \to v_p} &= E_{PK_{v_p}}(c_{v_i}||loc_{v_p}||chan_{ip}||Sig_{b_j}||ts),
\end{aligned}
\tag{3}
$$

where $chan_{ip}$ is the wireless channel between the caching requester and the caching provider. $E_{PK_{v_i}}$ and $E_{PK_{v_p}}$ denote that $Resp_{req}^{b_j \to v_i}$ and $Resp_{pro}^{b_j \to v_p}$ are encrypted with public key PK_{v_i} and PK_{v_p}, respectively. The digital signature in $Resp_{req}^{b_j \to v_i}$ is denoted as $Sig_{b_j} = Sign_{SK_{b_j}}(loc_{v_p}||chan_{ip})$, and the digital signature in $Resp_{pro}^{b_j \to v_p}$ is denoted as $Sig_{b_j} = Sign_{SK_{b_j}}(c_{v_i}||loc_{v_p}||chan_{ip})$.

Based on the above two messages, vehicles autonomous execute the pre-programmed smart contract. The smart contract consists of two modules. The first module is to carry out content delivery. The second module is to transfer a certain amount of coins from the wallet of content caching requester v_i to the wallet of content caching provider vehicle v_p.

3) **Recording Transactions:**

After finished content caching, caching requesters pays for its caching provider and generates a transaction to record the caching event. Specifically, caching requester v_i sends the generated transaction to the nearest BS b_j. The BS first verifies the received transaction, and then encrypts and broadcasts it to the entire blockchain network. The transaction includes the shared caching resource c_{v_i}, the coins that caching provider v_p obtains $coin^{v_i \to v_j}$, the wallet addresses of the content requester and the content provider, the signature of the content requester, and timestamp, namely,

$$
\begin{aligned}
Trans^{v_i \to b_j} =& E_{PK_{b_j}}(c_{v_i}||coin^{v_i \to v_p}||wallet_{addr}^{v_i}|| \\
& wallet_{addr}^{v_p}||Sig_{v_i}||ts),
\end{aligned}
\tag{4}
$$

where $E_{PK_{b_j}}$ denotes that $Trans^{v_i \to b_j}$ is encrypted with public key PK_{b_j}, $Sig_{v_i} = Sign_{SK_{v_i}}(c_{v_i}||coin^{v_i \to v_p})$ denotes that the digital signature of c_{v_i} and transferred coins $coin^{v_i \to v_j}$ with private key SK_{v_i}. The new generated transaction is broadcasted over the entire network for audit and verification.

The verified transactions are ordered and batched into a cryptographically tamper-evident data structure, named block. The blocks are linked in a linear chronological order by hash pointers to form a blockchain.

4) **Building Block and Performing Consensus Process:**

Each block is created by a specific BS in the consensus process. We define the BS to create the newly block as the leader. After the newly block created, the leader broadcasts the block with timestamp for block audit and verification. The other BSs verify the correctness of the newly created block. According to Bitcoin, the fastest node which solves Proof-of-Work (PoW) puzzle becomes the leader to create the newly block. However, PoW puzzle is a computation-intensive and energy-consuming task such that it is not suitable for vehicular networks [18]. Therefore, we need a fast and efficient blockchain consensus mechanism with low energy-consuming and time-consuming.

In this paper, we aim to design an intelligent and secure vehicular content caching scheme for the proposed architecture. However, there are two challenges to achieve this. One is in the user plane that high mobility of vehicles leads to dynamic network topology and time-varying wireless channel condition making it difficult to design an optimal content caching policy. The other is in the edge plane that how to achieve fast permission blockchain with low energy-consuming and time-consuming. To address such issues, we propose a DRL-inspired content caching scheme in Section IV and PoU consensus mechanism for permissioned blockchain in Section V.

IV. DEEP REINFORCEMENT LEARNING-BASED VEHICULAR CONTENT CACHING

In this section, we propose a DRL-inspired content caching algorithm with taking vehicular mobility into account to solve the challenge in the user plane.

A. Content Caching with Manhattan grid mobility model

We formulate V2V content caching problem to maximize system utility by focusing on a single cell with a BS and N vehicles. The BS can communicate with any vehicle under its coverage. We denote the set of caching requester as $\mathcal{I} = \{v_1, ..., v_I\}$ and the set of caching provider as $\mathcal{P} = \{v_1, ..., v_P\}$, where $\mathcal{I} \cap \mathcal{P} = \varnothing$ and $I + P = N$. The content generated by caching requester v_i can be described as $\{c_{v_i}, \tau_{v_i}\}$, where c_{v_i} and τ_{v_i} denote the required caching resource and the maximal content delivery latency, respectively.

We model the city as a Manhattan style grid, with a uniform block size across a fixed square area. The Manhattan grid model is introduced as a standard mobility by the European Telecommunications Standards Institute (ETSI) [32]. In Manhattan grid model, the map is composed of a number of horizontal and vertical streets. Each street has two lane for each direction (i.e., north and south direction for vertical streets, and east and west for horizontal streets). Vehicles move along streets and may turn at cross streets (i.e., intersection) with a given probability. Let η denote the driving direction of vehicles, where $\eta \in \{north, south, west, east\}$. The probability that each vehicle moves at an intersection can be denoted as

$$
\mathbb{P}_\eta = \frac{\frac{1}{\delta_{int}\nu}}{\frac{1}{\delta_{int}\nu} + \frac{T^{wait}P^{wait}}{2}} = \frac{2}{2 + T^{wait}P^{wait}\delta_{int}\nu},
\tag{5}
$$

where δ_{int} is the density of intersections, ν denotes constant velocity, T^{wait} denotes the maximum tolerant waiting time of vehicles at the intersection, P^{wait} is the probability that vehicles have to wait. Further, vehicles may stop at an intersection, which can be denoted as ζ. The probability that a vehicle stops at an intersection is represented as

$$
\mathbb{P}_\zeta = 1 - \mathbb{P}_\eta = \frac{T^{wait}P^{wait}\delta_{int}\nu}{2 + T^{wait}P^{wait}\delta_{int}\nu}.
\tag{6}
$$

Vehicle v_p as caching provider has a local cache with capacity of C_p. We define $x_{ip} \in \{0, 1\}$ as the content caching variable. If the content of caching requester v_i is cached at caching provider v_p, $x_{ip} = 1$. Otherwise, $x_{ip} = 0$. When v_i caches its content on v_p, it has to make a certain payment for caching resource usage. The payment that v_i pays to v_p is defined as $coin^{v_i \rightarrow v_p} = x_{ip}\varsigma c_{v_i}$, where $\varsigma > 0$ is the price for unit caching resource. Since the amount of caching resource on each vehicle is limited, the total occupied cache resource of all contents on v_p cannot exceed its caching capacity, i.e., $\sum_{i \in \mathcal{I}} x_{ip}c_{v_i} \leq C_p$.

Vehicles can communicate with each other if the distance between them does not exceed the communication distance, i.e., $d_{ip} < \gamma$ [33]. The communication data rate between vehicle v_i and vehicle v_p can be expressed as

$$R_{ip} = b \log_2(1 + \frac{p_i h_{ip} d_{ip}^{-\alpha}}{\sigma^2}), \qquad (7)$$

where b is the channel bandwidth, p_i is the transmission power of v_i, h_{ip} is the channel gain, α denotes the path loss exponent, and σ^2 is the noise power.

According to (7), the V2V content transmission latency is

$$T_{ip} = \frac{c_{v_i}}{R_{ip}}. \qquad (8)$$

Since the content of caching requester v_i should be transmitted within τ_{v_i}, we have $\sum_{p \in \mathcal{P}} x_{ip}T_{ip} \leq \tau_{v_i}$.

The total consumed energy consists of two parts: transmission energy consumption and content caching energy consumption. Let β denote the price per energy consumption. The energy cost for V2V content caching is

$$E_{ip} = \beta\{p_i \frac{c_{v_i}}{R_{ip}} + e_0 * c_{v_i}\}, \qquad (9)$$

where e_0 is the unit energy consumption per caching resource.

Under the constraints of caching capacity and maximum content delivery latency, the problem to maximize system utility is formulated as follows:

$$\max \sum_{i \in \mathcal{I}} \sum_{p \in \mathcal{P}} (x_{ip}\varsigma c_{v_i} - x_{ip}E_{ip})$$

$$\sum_{i \in \mathcal{I}} x_{ip}c_{v_i} \leq C_p, \quad \forall p \in \mathcal{P} \qquad (10a)$$

$$\sum_{p \in \mathcal{P}} x_{ip}T_{ip} \leq \tau_{v_i}, \quad \forall i \in \mathcal{I} \qquad (10b)$$

$$x_{ip} \in \{0, 1\}, \quad \forall i \in \mathcal{I}, \ p \in \mathcal{P} \qquad (10c)$$

Since x_{ip} is a binary variable, the feasible set and objective function of problem (10) are not convex. Though we can use an approximate algorithm to solve it, the scalability of the solution is very weak as the solution may fail with the increasing number of vehicles. Moreover, the movement of vehicles leads to time-varying wireless channel such that the conventional optimization method is impractical. Since deep reinforcement learning is suitable for decision-making problems with high-dimensional and time-varying features, here we attempt to utilize it to solve problem (10).

B. DRL-based V2V Content Caching Solution

We first reformulate problem (10) as deep reinforcement learning form with system state, action, and reward, as shown in Fig. 2. Then, we propose the DRL-based V2V content caching algorithm.

System state is a space to reflect the observed vehicular environment. Let $\mathcal{S}$ denote the system state space. The state $s_t \in \mathcal{S}$ at time slot t can be defined as

$$s_t = \{R(t), T(t), E(t), \eta(t), F_i, C\}, \qquad (11)$$

where

- $R(t) = [R_{11}(t), ..., R_{IP}(t)]$: is a vector which represents V2V communication data rate between vehicles at time slot t;
- $T(t) = [T_{11}(t), ..., T_{IP}(t)]$: is a vector which represents content latency via V2V transmission at time slot t;
- $E(t) = [E_{11}(t), ..., E_{IP}(t)]$: is a vector which represents energy consumption of V2V content delivery at time slot t;
- $\eta(t) = [\eta_1(t), ..., \eta_I(t), \eta_{I+1}(t)..., \eta_{I+P}(t)]$: is a vector which represents each vehicle's driving direction at time slot t;
- $F_i = \{[c_{v_1}, t_{v_1}], .., [c_{v_I}, t_{v_I}]\}$: is a matrix which represents the required caching resource and the maximal content delivery latency of caching requesters;
- $C = [C_1, .., C_p]$: is a vector which represents the caching capacities of caching providers.

Because of mobility, the location of each vehicle is time-varying such that V2V communication data rates, content transmission latency, and the energy consumption of V2V content delivery are time-varying.

The action of V2V content caching is to match caching pairs. Let $\mathcal{A}$ denote the action space. The action $a_t \in \mathcal{A}$ at time slot t is defined as

$$a_t = [x_{11}(t), ..., x_{IP}(t)], \qquad (12)$$

After taking action a_t, the system will receive an immediate reward $\Upsilon(s_t, a_t)$. Since the objective of problem (10) is to maximize system utility, we define the immediate reward as

$$\Upsilon(s_t, a_t) = \begin{cases} \mathbb{E}\left[\sum_{i \in \mathcal{I}} \sum_{p \in \mathcal{P}} (x_{ip}(t)\varsigma c_{v_i} - x_{ip}(t)E_{ip}(t))\right], \\ \qquad \text{if (10a) and (10b)}; \\ plt, \qquad \text{otherwise}; \end{cases}$$
$$(13)$$

If the action of caching pairs satisfies constraints (10a) and (10b), the immediate reward is the current system utility. Otherwise, the system will receive a penalty and $\Upsilon(s_t, a_t) = plt$, where plt is a negative constant. The optimal V2V content caching strategy is to maximize the long-term reward which can be defined as

$$Reward = \max E\left[\sum_{t=0}^{T-1} \epsilon^t \Upsilon(s_t, a_t)\right], \qquad (14)$$

where $\epsilon \in [0, 1]$ is the discounted factor.

Based on system state, action, and reward, we attempt to utilize DRL to solve the proposed V2V content caching problem.

Fig. 2: DRL-empowered V2V content caching

There are three common DRL algorithms: Q-learning, Deep Q Network (DQN), and Deep Deterministic Policy Gradient (DDPG). Q-learning is a classical deep reinforcement learning algorithm which computes the Q-function of each state-action pair for action exploration. However, Q-learning is not an ideal algorithm for the problem with a high-dimensional observation space. DQN is a kind of deep reinforcement learning algorithm which uses deep neural networks instead of Q-function to explore actions. DQN is a powerful tool that can learn optimal policies with high-dimensional observation spaces but it can only hand low-dimensional action spaces [34]. DDPG is an actor-critic and model-free algorithm that can learn policies in high-dimensional observation spaces and high-dimensional action spaces. In this paper, we exploit the deep deterministic policy gradient [34], to solve V2V content caching problem.

According to DDPG, caching agent is composed of three modules: primary network, target network, and replay memory. Primary network aims to match content caching pairs by policy gradient method. Primary network consists of two deep neural networks, namely primary actor neural network and primary critic neural network. Target network is used to generate target value for training primary network. The structure of target network is similar to the structure of primary network but with different parameters. Replay memory is used to store experience tuples. Experience tuples include current state, the selected action, reward, and next state, which can be randomly sampled for training primary network and target network. The detailed interaction processes among these models are shown in Fig. 2.

The explored policy can be defined as a function parametrized by θ_π, mapping current state to an action $\hat{a} = \pi(s_t|\theta_\pi)$ where $\hat{a}$ is a proto-actor action generated by the mapping and $\pi(s_t|\theta_\pi)$ is the explored edge caching and content delivery policy produced by primary actor neural network. By adding an Ornstein-Uhlenbeck noise $\mathfrak{N}_t$, the constructed action can be described as [34]

$$a_t = \pi(s_t|\theta_\pi) + \mathfrak{N}_t. \tag{15}$$

The primary actor neural network updates network parameter θ_π using the sampled policy gradient, computed as

$$\nabla_{\theta_\pi} J \approx \mathbb{E}\left[\nabla_a Q(s,a|\theta_Q)|_{s=s_t,a=\pi(s_t)} \nabla_{\theta_\pi} \pi(s|\theta_\pi)|_{s=s_t}\right], \tag{16}$$

where $Q(s,a|\theta_Q)$ is an action-value function and will be introduced in the following. Specifically, at each training step, θ_π is updated by a mini-batch experience $< s_t, a_t, \mathcal{R}^{imm}, s_{t+1} >$, $t \in \{1,...,V\}$, randomly sampled from replay memory,

$$\theta_\pi = \theta_\pi - \frac{\alpha_\pi}{V} \sum_{t=1}^{V} \left[\nabla_a Q(s,a|\theta_Q)|_{s=s_t,a=\pi(s_t)} \nabla_{\theta_\pi} \pi(s|\theta_\pi)|_{s=s_t}\right], \tag{17}$$

where α_π is the learning rate of the primary actor neural network.

The primary critic neural network evaluates the performance of the selected action based on the action-value function. The action-value function is calculated by the Bellman optimality equation and can be expressed as

$$Q(s_t, a_t|\theta_Q) = \mathbb{E}\left[\mathcal{R}^{imm}(s_t, a_t) + \varepsilon Q(s_{t+1}, \pi(s_{t+1})|\theta_Q)\right], \tag{18}$$

Here, the primary critic neural network takes both current state s_t and next state s_{t+1} as input to calculate $Q(s_t, a_t|\theta_Q)$ for each action.

The primary critic neural network updates the network parameter θ_Q by minimizing the loss function $Ls(\theta_Q)$. The loss function is defined as

$$Ls(\theta_Q) = \mathbb{E}\left[(y_t - Q(s_t, a_t|\theta_Q))^2\right], \tag{19}$$

where y_t is the target value and can be obtained by

$$y_t = \mathcal{R}^{imm}(s_t, a_t) + \varepsilon Q'(s_{t+1}, (\pi)'(s_{t+1}|\theta_\pi^T)|\theta_Q^T). \tag{20}$$

where $Q'(s_{t+1}, \pi'(s_{t+1}|\theta_\pi^T)|\theta_Q^T)$ is obtained through the target network, i.e., the network with parameters θ_π^T and θ_Q^T.

The gradient of loss function $Ls(\theta_Q)$ is calculated by its first derivative, which can be denoted as [34]

$$\nabla_{\theta_Q} Ls = \mathbb{E}\left[2(y_t - Q(s_t, a_t|\theta_Q)) \nabla_{\theta_Q} Q(s_t, a_t)\right]. \tag{21}$$

According to (21), the parameter θ_Q of primary critic neural network can be updated. Specifically, at each training step, θ_Q is updated with a mini-batch experiences $< s_t, a_t, \mathcal{R}^{imm}, s_{t+1} >$, $t \in \{1,...,V\}$, that randomly sampled from replay memory,

$$\theta_Q = \theta_Q - \frac{\alpha_Q}{V} \sum_{t=1}^{V} \left[2(y_t - Q(s_t, a_t|\theta_Q)) \nabla_{\theta_Q} Q(s_t, a_t)\right], \tag{22}$$

Algorithm 1 DRL-based V2V content caching algorithm

Input: The parameters about mobility model δ_{int}, ν, T^{wait}, and P^{wait} ;

        The parameters about V2V communication, γ, transmission power, bandwidth, channel gain, and path loss exponent;

        The state of the observed vehicular environment s_t;

Output: The explored caching pairs;

1: Initialize $\mu(s|\theta^\mu)$ and $Q(s,a|\theta^Q)$ of the primary network with parameters θ^μ and θ^Q;

2: Initialize the target network with parameters $\theta^{\mu'} \leftarrow \theta^\mu$ and $\theta^{Q'} \leftarrow \theta^Q$; Initialize replay memory;

3: **for** each episode **do**

4:     Setup vehicular environment;

5:     **for** each time step t **do**

6:         Execute action a_t based on $\mu(s|\theta^\mu)$ and state s_t .

7:         Observe reward $\Upsilon(s_t, a_t)$ and state s_{t+1} based on (13);

8:         Store the tuple $< s_t, a_t, \Upsilon(s_t, a_t), s_{t+1} >$ into replay memory;

9:         Sample a mini-batch of tuples from replay memory;

10:        Compute the target value y_t and update θ^Q by minimizing the loss function (19);

11:        Update $\mu(s|\theta^\mu)$ using the sampled policy gradient (16);

12:        Update target networks with:
$$\theta^{\mu'} \leftarrow \omega\theta^\mu + (1-\omega)\theta^{\mu'}$$
$$\theta^{Q'} \leftarrow \omega\theta^Q + (1-\omega)\theta^{Q'}$$

13:     **end for**

14: **end for**

Algorithm 2 Construct the edges of bipartite graph $\mathcal{G}$

Input: The outputs of the explored caching pairs;

Output: The constructed bipartite graph $\mathcal{G}(\mathcal{I}, \mathcal{P}, \mathcal{E})$;

1: Set $\mathcal{E} \leftarrow \varnothing$.

2: **if** $P_p \leqslant 1$ **then**

3:     There is only one node v_{p1} corresponding to caching provider p.

4:     **for** each $x'_{ip} > 0$ **do**

5:         Add edge (i, v_{p1}) into edge set $\mathcal{E}$ and let the edge weight as $e_{ip1} = x'_{ip}$.

6:     **end for**

7: **else**

8:     Find the minimum index i_s where $\sum_{i'=1}^{i_s} x'_{ip} \geqslant s$.

9:     **if** $i = i_{s-1} + 1, .., i_s - 1$, and $x'_{ip} > 0$ **then**

10:        Add edge (i, v_{ps}) into edge set $\mathcal{E}$ with weight $e_{ips} = x'_{ip}$.

11:     **else if** $i = i_s$ **then**

12:        Add edge (i, v_{ps}) into edge set $\mathcal{E}$ with $e_{ips} = 1 - \sum_{i=1}^{i_s-1} x'_{ip}$.

13:     **else**

14:        Add edge $(i, v_{p(s+1)})$ into edge set $\mathcal{E}$ with weight $e_{ip(s+1)} = \sum_{i=1}^{i_s} x'_{ip} - s$.

15:     **end if**

16: **end if**

---

where α_Q is the learning rate of the primary critic neural network.

The target network can be regarded as an old version of the primary network with different parameters θ_π^T and θ_Q^T. At each iteration, the parameters θ_π^T and θ_Q^T are updated based on the following definition:

$$\theta_\pi^T = \omega\theta_\pi + (1-\omega)\theta_\pi^T,$$
$$\theta_Q^T = \omega\theta_Q + (1-\omega)\theta_Q^T, \tag{23}$$

where $\omega \in [0, 1]$.

DRL-based V2V content caching algorithm is shown in Algorithm 1. First, caching agent initializes policy $\mu(s|\theta^\mu)$ with parameter θ^μ and initializes action-value faction $Q(s,a|\theta^Q)$ with parameter θ^Q. The parameters of the target network are also initialized. Then, for each time step, primary network generates action a_t based on current policy $\mu(s|\theta^\mu)$ and current state s_t. Observing reward $\Upsilon(s_t, a_t)$ and next state s_{t+1}, caching agent constructs a tuple $< s_t, a_t, \Upsilon(s_t, a_t), s_{t+1} >$ and stores it into replay memory. Based on mini-batch technique, caching agent updates parameter θ^Q by minimizing loss function $Ls(\theta^Q)$ and updates θ^μ using the sampled policy gradient. The parameters of the target networks are updated based on θ^μ, θ^Q, and ω, where $\omega \in [0, 1]$.

C. Action Refinement

The outputs from DRL-based content caching are continuous values. However, content caching variables are integer values, i.e., $x_{ip} \in \{0, 1\}$. Therefore, we need to refine the outputs of DRL. Here, we adopt rounding technique to make action refinement. The rounding technique has three steps: 1) find the continuous solution from a_t, 2) construct a weighted bipartite graph to establish the relationship between vehicles and BSs, 3) find an integer matching to obtain the integer solution.

1) *Find the continuous solution from a_t:* We define the input sets as $\mathbf{z} = [x'_{11}, .., x'_{IP}]$, where $x'_{ip} \in [0, 1]$.

2) *Construct bipartite graph:* We construct the weighted bipartite graph $\mathcal{G}(\mathcal{I}, \mathcal{P}, \mathcal{E})$ to establish the relationship between caching requesters and providers. $\mathcal{I}$ represents the caching requesters in the network. $\mathcal{V} = \{v_{ps} : j =, 1, .., P; s = 1, ..., P_p\}$, where $P_p = \lceil \sum_{i=1}^I x_{ip} \rceil$ implies caching provider p can serve the number of P_p caching requesters. The nodes $\{v_{ps} : s = 1, .., P_p\}$ correspond to caching provider p. The most important procedure for constructing graph $\mathcal{G}$ is to set the edges and the edge weight between $\mathcal{I}$ and $\mathcal{P}$. The edges in $\mathcal{G}$ are constructed using Algorithm 2.

3) *Action refinement:* We utilize the Hungarian algorithm [35] to find a complete max-weighted bipartite matching M_{match}. According to the M_{match}, we obtain the detailed caching pairs matching. Specifically, if (i, v_{ps}, e_{ips}) is in the M_{match}, we set $x_{ip} = 1$; otherwise, $x_{ip} = 0$.

The outputs of DRL-based content caching are fractional solutions, the rounding results are integer solutions.

V. Proof-of-Utility Consensus in Vehicular Networks

In this section, we present the details of PoU consensus mechanism for permissioned blockchain in the edge plane and propose how to evaluate BS utility for block verifier selection.

A. PoU Consensus

DPoS is a fast and efficient blockchain consensus mechanism which leverages voting and selection to protect blockchain from centralization and malicious usage [36]. Compared to PoW and PoS, the number of entities participating in DPoS consensus is very small, making it possible to effectively reduce the time consumption and energy consumption to reach consensus. Inspired by DPoS, the proposed PoU consensus consists of two parts: 1) delegate selection, 2) block production and verification, as shown in Fig.3. Different from DPoS, we are not use the stake of users but use the utility of base station to select delegates. The details about PoU consensus are introduced in the following.

1) Delegate Selection: Since coin transfer in content caching transactions occurs among vehicles and these vehicles are non-trusted, we define vehicles as token holders to dominate delegate selection process. Because delegates involve the key process of consensus algorithms (i.e., block production and validation), they are preferably neutral nodes [36]. In content caching process, BSs are not directly participating in content delivery and coin payment, which means they do not get any profit from the caching process, such that they are ideal neutral nodes and can be regarded as delegate candidates.

At each selection, vehicles vote for their preferred BSs with the highest utility. The utility is utilized to measure the quality of BSs. Higher utility means BS is equppied with more powerful computing and processing abilities to generate and verify block. The details of utility evaluation is given in the following subsection V-B. Each vehicle has one vote per round and the voting weight is proportional to the number of coins it holds. The top $\widehat{n}$ candidates with the most votes are selected to form a delegate commission, where $\widehat{n}$ is an odd integer and no greater than the number of BSs.

2) Block Production and Verification: In the block production and verification process, delegates are divided into two roles: leader and verifier, where leader is responsible for transaction collection and block production, and verifier is responsible for block verification. At each block production process, one of $\widehat{n}$ delegates acts as the leader and the other delegates act as verifiers. Leader is generated in a round-robin manner among delegates which indicates each delegate can become a leader to produce block. For example, in Fig. 3, delegate B is the leader responsible for creating Block B.

In a specific block production and verification process, the leader first collects a certain amount of V2V content caching transactions and then calculate a correct hash to create an unverified block. The block verification is a three-phase protocol consisting of block broadcast, block verification, and confirm. In block broadcast phase, the leader broadcasts $\widehat{n} - 1$ broadcasting messages to other delegates. The broadcasting message has the form:

Fig. 3: The PoU consensus of vehicular permissioned blockchain

$Bro = \langle bro_{msg}||PK_{led}||PK_{ver}||ts_{bro}||block\rangle$, where PK_{led} is the source address of the message, PK_{del} is the destination address of the message, ts_{bro} is the time stamp, $block$ is the created block. In the block verification phase, each verifier first verifies the signature of the received broadcasting message. Then verifiers audit the correctness of V2V content caching requests packaged in the newly block and broadcast their audit results with their signatures to each other in a distributed manner. In the confirm phase, each verifier compares its audit result with the received audit results from other verifiers and sends a confirm message to the leader. The confirm message has the form: $Con = \langle con_{msg}||PK_{ver}||PK_{led}||Aud_{self}||Aud_{rec}||Rsu_{comp}\rangle$, where Aud_{self} is the audit result of the verifier own, Aud_{rec} is the records of received audit results of other verifiers, Rsu_{comp} is the comparison result. After receiving all delegates' confirm messages, the leader analyses them and decides the correctness of the block. If more than two third of verifiers agree on the block, the leader will send it to all delegates to store. The BSs which are not in the delegate commission will synchronize the latest blockchain from nearby delegates periodically.

Once the newly produced block has been successfully appended to the blockchain, the BSs participated in block production and verification process will be rewarded to compensate for their resource consumption. If all delegates become the leader once, the order of the delegates are shuffled and then they produce the future blocks in a round-robin manner again. If a delegate fails to create a block during its turn, the block is skipped and the transactions in the skipped block will be transferred to the next one.

B. Utility Evaluation

There are M BSs distributed in vehicular networks, denoted as $\mathcal{M} = \{1, ..., M\}$. All M BSs can communicate with each other at the speed of r via wired line. In the delegate selection process, $\widehat{n}$ of M BSs are elected by vehicles to group as a delegate commission, denoted as $\widehat{\mathcal{N}} = \{1, ..., \widehat{n}\}$.

From the perspective of vehicles, they prefer to vote for the BSs providing fastest block production and verification as delegates. In this paper, we adopt utility function with respect to time consumption to evaluate the performance of BSs.

A short time consumption leads to a good user experience thus the utility function should monotonically decrease with time consumption. To satisfy content delivery constraint, the utility is set as 0 if any content caching time consumption of vehicles is exceeded its maximal content delivery latency τ_{v_i}. We consider that K transactions are collected in the k-th block and $\tau^k = \min\{\tau_1, \tau_2, ..., \tau_K\}$, where $K \leq I$. The utility function of BS m is defined as:

$$U_m^k = \left[e^{1-T_m^k/\tau^k} - 1\right]^+,\tag{24}$$

where T_m^k is the time consumption of block production and verification on BS m and $[x]^+ = \max\{x, 0\}$. T_m^k is consisted of three parts: 1) the delay of transaction collection and hash computing, 2) the delay of block verification, 3) the delay of content transmission, which can be described as

$$T_m^k = T_k^H + T_k^V + T_{ip}.\tag{25}$$

According to DPoS, the delay of transaction collection and hash computing T_k^H is pre-defined, such as 0.5s in EOS. The content transmission T_{ip} is shown in Eq. (8).

The delay of block verification consists of three parts: 1) block broadcasting, 2) cross-verification among verifiers, 3) block confirm, which can be described as:

$$T_k^V = T_k^{bb} + T_k^{cv} + T_k^{bc}.\tag{26}$$

BS $j \in \widehat{\mathcal{N}}$ is the leader to dominate block verification process. The other BSs are verifiers to audit the produced block, denoted as $j' \in \widehat{\mathcal{N}}/\{j\}$. Since the leader j broadcasts its produced block to verifiers simultaneously, the block broadcasting time is determined by the longest block transmission time, thus

$$T_k^{bb} = \max_{j' \in \widehat{\mathcal{N}}/\{j\}} \left\{\frac{I_k d_{jj'}}{r}\right\},\tag{27}$$

where $d_{jj'}$ is the distance between the leader j and verifier j', I_k is the size of the k-th block before verification. Cross-verification consists of three steps. Each verifier first performs local verification to verify the raw block from the leader and then broadcasts its local-verified result to other verifiers. After receiving the local-verified result, verifiers performs second audit. We denote O_k as the size of local-verified result and W_k as the size of second-audit result. The k-th block cross-verification time consumption can be written as

$$T_k^{cv} = \max_{j',j'' \in \widehat{\mathcal{N}}/\{j\}, j' \neq j''} \left\{\frac{I_k f_0}{F_{j'}} + \frac{O_k d_{j'j''}}{r} + \frac{O_k f_0}{F_{j''}}\right\}, \tag{28}$$

where f_0 denotes the computation resource for verifying one bit of I_k, $F_{j'}$ and $F_{j'_i}$ denote the computation resource that verifier j' and verifier j'' provide to block verification respectively, and $d_{j'j''}$ denote the distance between verifier j' and verifier j''. Block confirm time is determined by the longest second-audit result transmission time, which is

$$T_k^{bc} = \max_{j' \in \widehat{\mathcal{N}}/\{j\}} \left\{\frac{W_k d_{jj'}}{r}\right\}.\tag{29}$$

Base on Eq. (26), the T_m^k and U_m^k can be obtained, respectively. Then, each vehicle casts it vote to the BS m^*, which satisfies

$$m^* = \arg\max\{U_m^k\},\ m \in \mathcal{M}.\tag{30}$$

The vote weight is equal to content caching transaction fee, i.e., $coin^{v_i->v_j}$.

VI. SECURITY ANALYSIS AND NUMERICAL RESULTS

In this section, we first provide security about our proposed blockchain-based content caching. Then, we evaluate the performance of the proposed V2V content caching scheme based on Uber dataset [37] and analyse the performance of the proposed PoU.

A. Security Analysis

The use of permissioned blockchain establishes a secure content caching for multiple vehicles without mutual trust.

1) *Without reliance on a single trusted third party:* The permissioned blockchain reduces the reliance on a trusted curator. If V2V content caching requires the involvement of a trusted third party, the system security largely depends on the security of the trusted third party. If the centralized security cannot be guaranteed, content in the system faces high risk of leakage. In our schemes, caching requesters and caching providers deliver content a P2P manner without a third party, which makes system robust and scalable.

2) *Privacy protection:* All vehicles and BSs transmit messages about caching request and blockchain in a pseudonymous manner. Specifically, vehicle v_i uses its public key PK_{v_i} as the pseudonym to guarantee the anonymity of its real identity. The messages (i.e., $Req^{v_i \to b_j}$, $Mes^{v_p \to b_j}$, $Resp_{req}^{b_j \to v_i}$, and $Resp_{pro}^{b_j \to v_p}$), and transactions (i.e., $Trans^{v_i \to b_j}$) are signed and can only be accessed by a specified vehicle with the right private key. If a malicious vehicle wants to forge the signature of vehicle v_i to pass the authentication process, the adversary has to forge a signature $Sig_{v_i} = Sign_{SK_{v_i}}(\cdot)$. However, adversaries have no access to the private key SK_{v_i}. The only information that the adversary can obtain is the public key of vehicle v_i. Since there is no feasible solution to obtain the private key from the public key, the adversary cannot forge the signature information of a legal vehicle. The anonymity and digital signature can protect the privacy of vehicles in V2V content caching.

3) *Majority attack:* Majority attack is a noteworthy security issue where an entity or user can take control of the system and use it for self benefit if the attack is performed properly. In the proposed scheme, all blocks and transactions are publicly audited and mutual verified by all elected delegates. The delegate selection process is democratic where delegates are individually selected by vehicles based on the proposed PoU consensus. The selected delegates form a commission to produce block in a round-robin way. Therefore, it is nearly impossible for an adversary to dominate the delegate selection process and make the majority attack.

4) *Transaction traceable:* All broadcasted transactions in blockchain are forever recorded with a timestamp and these

Fig. 4: Spatial distribution of vehicle trace points.

Fig. 5: Comparison of cumulative average reward with respect to number of caching requesters under different scheme.

transactions cannot be modified by a single entity. Since blockchain is a distributed ledger, transactions are synchronized updated and can be easily obtained from any BS. When malicious behavior occurs, any vehicle can easily verifies and traces previous records through accessing a BS. Timestamps in blockchain can be used to keep transactions intact, thus leaving no chance to counterfeits.

B. Permance Analysis of Vehicular Content Caching Scheme

We use Python and TensorFlow to evaluate the performance of the proposed DRL empowered content caching scheme based on a real-world dataset from Uber [37].

The trajectory of Uber dataset is used to simulate the changing locations of vehicles. This dataset has 4.5 million Uber pickups in New York City from April to September 2014, and 14.3 million Uber pickups from January to June 2015, as shown in Fig. 4. We take 100 vehicles as examples from an observation area, whose latitude is from 40.668671 to 40.678719, and the longitude is from -73.930269 to -73.950915. The observed area is approximately 1.52 km^2.

The data size of each content, required caching resource and the maximal content delivery latency of each content are within the range of $[10, 50]$ MB, $[0.5, 2.5]$ GB, and $[5, 10]$s, respectively. The caching capacity of vehicles is 5 GB. The maximal distance of V2V communication is $\gamma = 500$ m. The transmission power of vehicles for content delivery is 24 dBm. The channel bandwidth is 10 MHz. The noise power is $\sigma^2 = 10^{-11}$ mW. The proposed DRL empowered algorithm is deployed on a MacBook Pro laptop, powered by two Intel Core i5 processor (clocked at 2.6Ghz). The activation function is $\frac{tanh(x)+1}{2}$. The maximum episode is 4000 and the maximum number of steps in each episode is set to 20. The size of mini-batch is set up as 32 . The penalty is -100.

To verify the performance of our proposed DRL empowered edge caching and content delivery algorithm, we introduce the following two benchmark schemes,

- *Greedy content caching (GCC)*: In this scheme, each caching requester delivers its content to the caching

provider with the highest wireless communication data rate.
- *Random content caching (RCC)* : This scheme randomly selects a caching provider for a caching requester to perform content caching. Note that the distance between the caching provider and the caching requester should not exceed γ.

We set the number of caching provider as 50. Fig. 5 plots the comparison of cumulative average reward with respect to number of caching requesters under different scheme. From Fig. 5 we can draw several observations. First, the performance of the proposed DRL-empowered algorithm significantly outperforms the two benchmark policies. The reason is that the proposed DRL-empowered algorithm designs the content caching policy based on current network topology and wireless channel condition while GCC and RCC are not able to acquire real-time parameters of the vehicular network, such as available caching resource of caching providers. Second, the performance of GCC is better than that of RCC because of taking wireless communication data rate into account. Third, the cumulative average reward of RCC is the lowest, even lower than 0 (i.e., receiving a penalty). This implies that randomly choosing a caching provider for a caching requester results in unsuccessful content caching.

Fig. 6 plots the comparison of percentage of successful content caching with respect to number of caching requesters under different scheme. We can see that the proposed DRL-empowered algorithm has a good efficiency that over 86% caching requesters can successfully execute content caching within their stringent deadline constraints. The performance of GCC is also well that about 78% caching requesters can successfully execute content caching. The performance of RCC is quite poor that only 5% caching requesters can successfully execute content caching. Thus, we can conclude the proposed algorithm is the most efficient.

Fig. 7 and Fig. 8 show the impact of learning rate and the impact of number of vehicles on the performance of the

Fig. 6: Comparison of percentage of successful content caching with respect to number of caching requesters under different scheme.

Fig. 7: Impact of learning rate on the performance of the proposed DRL-empowered algorithm.

Fig. 8: Impact of number of vehicles on the performance of the proposed DRL-empowered algorithm.

Fig. 9: (a) Utility of base station v.s. Block size I_k. (b) Utility of base station v.s. Distance to the leader.

utility improvement.

C. Performance Analysis of PoU Consensus

The size of block before verification I_k, the size of local-verified result O_k, and the size of second-audit result W_k are uniform distributed in $[10, 50]$ MB, $[1, 5]$ MB, and $[100, 500]$ KB, respectively. The computation resources of each BS are within the range of $[5, 10]$ GHz. We evaluate the utility of vehicle v_i towards the BS with various block size in Fig. 9a. From Fig. 9a, we can see that the utility decreases with the increasement of I_k. The reason is that with the increasing in block size, more time consumption is needed to verify the candidate block. When I_k exceeds 30 MB, the utility turns to be 0. This is because the time consumption on blockchain-based content caching exceeds the maximal content delivery latency (i.e., $T_{v_i} > \tau_{v_i}$). Fig. 9b depicts the utility of each vehicle with respect to average distance between the vehicle and the PoU selected leader. In general, the utility decreases with the increasement of distance. This is because a larger

proposed algorithm. From Fig. 7, we can observe that at all learning rates, the cumulative average rewards converge. When the learning rate is 10^{-2}, the proposed algorithm converge slightly faster than the case when the learning rate is 10^{-3} and 10^{-4} . In the proposed algorithm, we set the learning rate as 10^{-2}. From Fig. 8, we can see that the increasement of caching requesters results in a high reward, which means the caching utility is improved. Moreover, the reward is greatly increased as the number of caching requesters changes from 10 to 30 but it is slightly increased as the number of caching requesters changes from 30 to 50. This is reasonable because in all cases the total caching resource of the caching providers is the same, such that when the number is 30, the system utility is already approached to upper bound and the further increasement of caching requesters cannot make a greatly

distance between a vehicle and the PoU selected leader, the more communication time it needed for block verification. We can conclude that the utility decreases with the increasement of block size I_k and communication distance, which is in accordance with our previous analysis in subsection V-B.

VII. Conclusion

In this article, we have proposed a secure and intelligent content caching for vehicles by integrating deep reinforcement learning and permissioned blockchain in vehicular edge computing networks. We first proposed a blockchain empowered distributed and secure content caching framework where vehicles acted as caching requesters and caching providers to perform content caching and BSs acted as verifiers to build and maintain permissioned blockchain. To learn dynamic network topology and time-variant wireless channel condition, we utilized DRL to design an optimal content caching scheme. We exploited permissioned blockchain to maintain the security and privacy of content caching among vehicles and introduced a new verifiers selection metric, PoU to accelerate block verification. Security analysis shows that our proposed blockchain empowered content caching can achieve security and privacy protection. Numerical results based on a real dataset from Uber indicate that the DRL-inspired content caching scheme significantly outperforms two benchmark policies.

References

[1] K. Zhang, S. Leng, Y. He, *et al.*, "Cooperative content caching in 5G networks with mobile edge computing," *IEEE Wirel. Commun.*, vol. 25, no. 3, pp. 80–87, Jun 2018.

[2] Z. Zhou, H. Yu, C. Xu, Y. Zhang, S. Mumtaz, and J. Rodriguez, "Dependable content distribution in d2d-based cooperative vehicular networks: A big data-integrated coalition game approach," *IEEE Trans. Intell. Transp. Syst.*, vol. 19, no. 3, pp. 953–964, March 2018.

[3] Z. Zhou, C. Gao, C. Xu, Y. Zhang, S. Mumtaz, and J. Rodriguez, "Social big-data-based content dissemination in internet of vehicles," *IEEE IEEE Trans. Ind. Informat.*, vol. 14, no. 2, pp. 768–777, Feb 2018.

[4] T. G. Rodrigues, K. Suto, H. Nishiyama, N. Kato, and K. Temma, "Cloudlets activation scheme for scalable mobile edge computing with transmission power control and virtual machine migration," *IEEE Trans. Comput.*, vol. 67, no. 9, pp. 1287–1300, Sep 2018.

[5] T. G. Rodrigues, K. Suto, H. Nishiyama, and N. Kato, "Hybrid method for minimizing service delay in edge cloud computing through vm migration and transmission power control," *IEEE Trans. Comput.*, vol. 66, no. 5, pp. 810–819, May 2017.

[6] Y. Dai, D. Xu, S. Maharjan, and Y. Zhang, "Joint load balancing and offloading in vehicular edge computing and networks," *IEEE Internet Things J.*, vol. 6, no. 3, pp. 4377–4387, Jun 2019.

[7] Z. Zhou, J. Feng, Z. Chang, and X. Shen, "Energy-efficient edge computing service provisioning for vehicular networks: A consensus admm approach," *IEEE Trans. Veh. Technol.*, vol. 68, no. 5, pp. 5087–5099, May 2019.

[8] I. Sorkhoh, D. Ebrahimi, R. Atallah, and C. Assi, "Workload scheduling in vehicular networks with edge cloud capabilities," *IEEE Trans. Veh. Technol.*, vol. 68, no. 9, pp. 8472–8486, Sep 2019.

[9] K. Shanmugam, N. Golrezaei, A. G. Dimakis *et al.*, "Femtocaching: Wireless content delivery through distributed caching helpers," *IEEE Trans. Inf. Theory*, vol. 59, no. 12, pp. 8402–8413, Dec 2013.

[10] J. Liu, S. Zhang, N. Kato, H. Ujikawa, and K. Suzuki, "Device-to-device communications for enhancing quality of experience in software defined multi-tier lte-a networks," *IEEE Network*, vol. 29, no. 4, pp. 46–52, Jul 2015.

[11] H. Nishiyama, M. Ito, and N. Kato, "Relay-by-smartphone: realizing multihop device-to-device communications," *IEEE Commun. Mag.*, vol. 52, no. 4, pp. 56–65, Apr 2014.

[12] F. Tang, Z. M. Fadlullah, N. Kato, F. Ono, and R. Miura, "Ac-poca: Anticoordination game based partially overlapping channels assignment in combined uav and d2d-based networks," *IEEE Trans. Veh. Technol.*, vol. 67, no. 2, pp. 1672–1683, Feb 2018.

[13] J. Liu, N. Kato, J. Ma, and N. Kadowaki, "Device-to-device communication in lte-advanced networks: A survey," *IEEE Commun. Surv. Tut.*, vol. 17, no. 4, pp. 1923–1940, Fourthquarter 2015.

[14] L. T. Tan and R. Q. Hu, "Mobility-aware edge caching and computing in vehicle networks: A deep reinforcement learning," *IEEE Trans. Veh. Technol.*, vol. 67, no. 11, pp. 10190–10203, Nov 2018.

[15] Y. Dai, D. Xu, S. Maharjan *et al.*, "Artificial intelligence empowered edge computing and caching for Internet of Vehicle," *IEEE Wirel. Commun.*, vol. 26, no. 3, pp. 12–18, Jun 2019.

[16] Y. Wei, F. R. Yu, M. Song, and Z. Han, "Joint optimization of caching, computing, and radio resources for fog-enabled iot using natural actor-critic deep reinforcement learning," *IEEE Internet Things J.*, vol. 6, no. 2, pp. 2061–2073, Apr, 2019.

[17] K. Zhang, Y. Zhu, S. Leng, Y. He, S. Maharjan, and Y. Zhang, "Deep learning empowered task offloading for mobile edge computing in urban informatics," *IEEE Internet Things J.*, vol. 6, no. 5, pp. 7635–7647, 2019.

[18] Y. Dai, D. Xu, S. Maharjan, *et al.*, "Blockchain and deep reinforcement learning empowered intelligent 5G beyond," *IEEE Network*, vol. 33, no. 3, pp. 10–17, May 2019.

[19] P. K. Sharma, N. Kumar, and J. H. Park, "Blockchain-based distributed framework for automotive industry in a smart city," *IEEE Trans. Ind. Informat.*, vol. 15, no. 7, pp. 4197–4205, Jul 2019.

[20] Y. Lu, X. Huang, Y. Dai, S. Maharjan, and Y. Zhang, "Blockchain and federated learning for privacy-preserved data sharing in industrial iot," *IEEE IEEE Trans. Ind. Informat.*, pp. 1–1, 2019.

[21] V. Sharma, I. You, D. N. K. Jayakody, D. G. Reina, and K. R. Choo, "Neural-blockchain-based ultrareliable caching for edge-enabled uav networks," *IEEE IEEE Trans. Ind. Informat.*, vol. 15, no. 10, pp. 5723–5736, Oct 2019.

[22] Y. Lu, X. Huang, K. Zhang, S. Maharjan, and Y. Zhang, "Blockchain empowered asynchronous federated learning for secure data sharing in internet of vehicles," *IEEE Trans. Veh. Technol., accepted*, pp. 1–1, 2020.

[23] J. Kang, Z. Xiong, D. Niyato, Y. Zou, Y. Zhang, and M. Guizani, "Reliable federated learning for mobile networks," *arXiv preprint arXiv:1910.06837*, 2019.

[24] W. Wang, D. Niyato, P. Wang, and A. Leshem, "Decentralized caching for content delivery based on blockchain: A game theoretic perspective," in *2018 IEEE International Conference on Communications (ICC)*, May 2018, pp. 1–6.

[25] Z. Yang, K. Yang, L. Lei, K. Zheng, and V. C. M. Leung, "Blockchain-based decentralized trust management in vehicular networks," *IEEE Internet Things J.*, vol. 6, no. 2, pp. 1495–1505, April 2019.

[26] Y. Yuan and F. Wang, "Towards blockchain-based intelligent transportation systems," in *2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)*, Nov 2016, pp. 2663–2668.

[27] J. Kang, Z. Xiong, D. Niyato, D. Ye, D. I. Kim, and J. Zhao, "Toward secure blockchain-enabled internet of vehicles: Optimizing consensus management using reputation and contract theory," *IEEE Trans. Veh. Technol.*, vol. 68, no. 3, pp. 2906–2920, Mar 2019.

[28] J. Kang, Z. Xiong, D. Niyato, S. Xie, and J. Zhang, "Incentive mechanism for reliable federated learning: A joint optimization approach to combining reputation and contract theory," *IEEE Internet Things J.*, pp. 1–1, 2019.

[29] M. Singh and S. Kim, "Blockchain based intelligent vehicle data sharing framework," *arXiv preprint arXiv:1708.09721*, 2017.

[30] M. Cebe, E. Erdin, K. Akkaya, H. Aksu, and S. Uluagac, "Block4forensic: An integrated lightweight blockchain framework for forensics applications of connected vehicles," *IEEE Communications Magazine*, vol. 56, no. 10, pp. 50–57, OCTOBER 2018.

[31] J. Camenisch, S. Hohenberger, and M. Ø. Pedersen, "Batch verification of short signatures," *Journal of cryptology*, vol. 25, no. 4, pp. 723–747, 2012.

[32] T. UMTS, "101 112 v3. 2.0 (1998-04),"universal mobile telecommunications system; selection procedures for the choice of radio transmission technologies of the umts (umts 30.03 version 3.2. 0)","," *European Telecommunications Standards Institute, Tech. Rep*, 1998.

[33] J. Liu, H. Nishiyama, N. Kato, and J. Guo, "On the outage probability of device-to-device-communication-enabled multichannel cellular networks: An rss-threshold-based perspective," *IEEE J. Sel. Area. Comm.*, vol. 34, no. 1, pp. 163–175, Jan 2016.

[34] T. P. Lillicrap, J. J. Hunt, A. Pritzel *et al.*, "Continuous control with deep reinforcement learning," *arXiv preprint arXiv:1509.02971*, 2015.

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

[36] D. Larimer, "Delegated proof-of-stake (dpos)," *Bitshare whitepaper*, 2014.

[37] "Uber TLC FOIL Response." [Online]. Available: https://github.com/fivethirtyeight/uber-tlc-foil-response