Blockchain-Empowered Federated Learning Approach for an Intelligent and Reliable D2D Caching Scheme

Runze Cheng, Yao Sun, *Senior Member, IEEE*, Yijing Liu, Le Xia, Daquan Feng, *Member, IEEE*, and Muhammad Ali Imran, *Senior Member, IEEE*

***Abstract*—Cache-enabled device-to-device (D2D) communication is a potential approach to tackle the resource shortage problem. However, public concerns of data privacy and system security still remain, which thus arises an urgent need for a reliable caching scheme. Fortunately, federated learning (FL) with a distributed paradigm provides an effective way to privacy issue by training a high-quality global model without any raw data exchanges. Besides the privacy issue, blockchain can be further introduced into the FL framework to resist the malicious attacks occurred in D2D caching networks. In this study, we propose a double-layer blockchain-based deep reinforcement FL (BDRFL) scheme to ensure privacy-preserved and caching-efficient D2D networks. In BDRFL, a double-layer blockchain is utilized to further enhance data security. Simulation results first verify the convergence of the BDRFL-based algorithm, and then demonstrate that the download latency of the BDRFL-based caching scheme can be significantly reduced under different types of attacks when compared to some existing caching policies.**

***Index Terms*—Blockchain, device-to-device (D2D) caching, federated learning (FL).**

I. Introduction

CONTENT caching is becoming significantly essential in next-generation wireless networks due to the growing demands of information requests for user devices. This predownload method can mitigate the pressure of backhaul links during peak times and reduce the latency of fetching content for clients [1]–[4]. Meanwhile, device-to-device (D2D) communication technology has been proposed recently, which enables multiple direct transmissions between pairs of nearby devices in cellular networks [5]; thus, the spectrum efficiency can be dramatically improved. Besides, D2D communication is an essential mechanism of establishing an *ad hoc* communication during a disaster scenario. Borrowing the D2D communication technology, a promising and attractive trend is to allow user equipments (UEs) to play an active role as caching servers; thus, to establish caching-enabled D2D networks.

In the caching-enabled D2D networks, a mobile user can get the required contents (e.g., popular videos) from other neighboring UEs via D2D links, besides from base stations (BSs) or the server in the core network. In this way, the network resource (wireless bandwidth and power, etc.) utilization could be greatly improved, and the provisioning quality of service, especially the download latency for UEs should also be upgraded. Moreover, while the traditional content distribution network demands high costs for massive users requiring the same content, the D2D caching is becoming more cost efficient in wireless networks with high user density [6]. Additionally, due to the recent improvement on battery, storage, and GPU/CPU models of UEs, mobile terminals are capable of exploiting machine learning algorithms to achieve a fast and accurate caching policy. Recently, extensive studies have been developed in designing intelligent caching schemes based on deep reinforcement learning (DRL). The DRL-based caching schemes use neural networks to learn the optimal strategy; thus, solving complex multiagent caching problems, when there are sufficient training data [7], [8].

Although traditional DRL-based schemes are capable of addressing the accuracy of decision making, there are still several challenges unsolved. First, users are reluctant in raw data sharing when there is a risk of privacy leakage; hence, it may cause a lack of training data. Furthermore, users tend to be self-interested, which means fewer users willing to participate in the D2D caching network when there is no direct reward or obvious benefit. In order to attract more users to participate in mobile D2D networks and stay active in the caching systems, it is thereby crucial to establish a privacy-preserved and secure caching scheme.

To address the difficulty of designing a privacy-preserved caching scheme, federated learning (FL) emerges as a promising alternative for intelligent decision making in the wireless system without sharing private raw data [9]. As FL only requires the exchange of weight or gradient of the local model, the local training data are merely kept in user local storage, and the terminal and personal privacy can be carefully protected.

Manuscript received March 15, 2021; revised June 14, 2021 and July 24, 2021; accepted August 2, 2021. Date of publication August 6, 2021; date of current version May 23, 2022. This work was supported in part by the Engineering and Physical Sciences Research Council Global Challenges Research Fund through the DARE Project under Grant EP/P028764/1. *(Corresponding author: Yao Sun.)*

Runze Cheng, Yao Sun, Le Xia, and Muhammad Ali Imran are with the James Watt School of Engineering, University of Glasgow, Glasgow G12 8QQ, U.K. (e-mail: yao.sun@glasgow.ac.uk).

Yijing Liu is with the National Key Laboratory on Communications, University of Electronic Science and Technology of China, Chengdu 611731, China.

Daquan Feng is with the Shenzhen Key Laboratory of Digital Creative Technology and the Guangdong Province Engineering Laboratory for Digital Creative Technology, Shenzhen University, Shenzhen 518060, China.

Digital Object Identifier 10.1109/JIOT.2021.3103107

Moreover, the FL is capable of carrying out efficient machine learning among distributed multiagents by sharing training models even under the case with insufficient local training data [10].

Despite these superiorities, one critical challenge faced is the reliability of model updates in the FL process, especially in untrusted D2D networks with faults and malicious attacks. For example, once some D2D nodes do not send the updates or send fault and fake updates, the accuracy and reliability of information exchanges cannot be guaranteed, leading to a heavy degradation of learning performance. Fortunately, a technical solution to jointly maintain reliable databases through decentralization and trustlessness, i.e., blockchain, can be introduced here. Blockchain as a distributed technology is applied to many cryptocurrencies, which has been proved valid of offering a reliable method for information tamper-proof [11]. Theoretically, blockchain has the potential to provide immutable and persistent data records [12] and serve as an information verification and storage tool in FL. Besides, a by-product brought by blockchain is to improve the willingness of users to participate in cache sharing and model training can be improved.

In this study, we propose a privacy-preserved and secure D2D caching scheme by exploiting DRL-based FL under a double-layer blockchain architecture. Simulations are conducted to demonstrate the convergence of blockchain-based deep reinforcement FL (BDRFL), and verify the performance gain of compared with several traditional learning-based caching schemes and a heuristic algorithm-based scheme. Here, the main contributions of our work are listed as follows.

1) We formulate the D2D caching problem as a multiagent Markov decision process (MDP) problem and propose a novel, privacy-preserved, and secure caching scheme named BDRFL.
2) We develop an FL with the double-stage cluster-based framework to establish a reliable learning scheme. This FL method allows users to train models in a distributed way without raw data exchange. In addition, we determine the update method of FL parameters for local models, area models, and the global model.
3) We exploit a double-layer blockchain architecture to underpin the above FL. Specifically, multisubchains based on the Raft consensus mechanism are used to store area models and stimulate users to participate in D2D caching. Meanwhile, a mainchain with the practical Byzantine fault tolerance (PBFT) consensus mechanism verifies area models to resist the Byzantine failures, thus ensuring the accuracy of the global model.

The remainder of this article is organized as follows. In Section II, we overview some related works. The system model is then described in detail in Section III, followed by presenting the problem formulation and BDRFL-based scheme in Section IV. We then illustrate the consensus mechanism of the blockchain and the updating details of the FL model in Section V. After that, we also conduct extensive simulations to further evaluate our scheme in Section VI with some discussions. Finally, we conclude our article in Section VII.

II. Related Work

A. Learning-Based D2D Caching Schemes

In recent years, numerous research exploit learning methods to design effective cache schemes with the aim of assisting traffic offloading and reducing transmission delay. Jiang *et al.* [13] emphasized the influence of content popularity distribution and formulated the D2D caching problem as a multiagent multiarmed bandit learning problem, and solved it via a centralized Q-learning. Chen and Yang [14] analyzed the synthesis of user preferences as well as content popularity. They predict the behavior of each user and update the caching strategy of Q-learning. To tackle the challenge of large action space, a DQN-based caching scheme optimization method is proposed in [15] by learning the preferences of the users and analyzing the similarity between the users and their neighbors. Similarly, Li *et al.* [16] used long short-term memory (LSTM) to learn user behaviors and DQN to update the state value function.

The majority of existing schemes do not consider privacy and security, while only a few related works are privacy friendly of user information. Kumar *et al.* [17] proposed a distributed Q-learning resource reservation framework based on multi D2D controllers. As users only share their privacy with the trusted neighbors, the risk of privacy leakage can be mitigated to some extent. A novel weighted distributed DQN is proposed for edge caching replacement optimization in [18], where each BS trains its own DQN model based on the local data. Besides, they use a reward-based adaptive boosting manner to update the global model. Unfortunately, once failures or attacks (like a malicious node send fake updates) happen, the caching system is easily crashing, if no consensus mechanism adopted in the caching scheme.

B. Blockchain-Enabled D2D Caching Schemes

A few recent works propose to incorporate the blockchain into FL to ensure the D2D caching performance under scenarios with attacks or failures.

Most of the relevant works focus on how to establish a blockchain-based platform for incentivizing users in caching networks without considering the benefits brings by blockchain in improving security. The partial PBFT (pPBFT)-based blockchains are proposed in the smart contract-based cache delivery markets by Zhang *et al.* [19] and [20]. In these caching networks, UEs get rewards from the cache provider for content delivery. Meanwhile, the blockchains merely work as distributed ledgers to save the content service list. Liu *et al.* [21] developed a multiaccess edge computing (MEC)-enabled blockchain framework, where the UEs play the role of the miners and resort to the nearby edge nodes for performing the computation-intensive Proof of Work (PoW) puzzle and content caching. There is one work [22] that improves the performance of caching policy by using the blockchain to resist failures, where FL and PBFT-based blockchain are combined; thus, the data and local model can be verified, retrieved, and shared securely. Besides, users can get rewards from the blockchain to improve their willingness of sharing content.

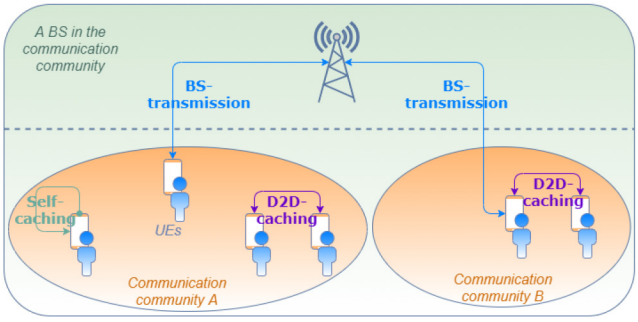


Fig. 1. Three different caching schemes in the communication community, 1) self-caching, 2) D2D-caching, and 3) BS transmission.

However, because of the massive user number, it is exceptionally complex to use the pPBFT or PBFT consensus in a large-scale caching sharing network. Besides, proof-based consensus with high computing and storage consumption is hard to run at the side of mobile equipment. Therefore, when designing an efficient and privacy-preserved FL caching scheme, it is necessary to develop a permissioned blockchain consensus that can incentivize UEs in large-scale networks.

III. System Model

A. D2D Network

We consider that a D2D network consisting of a central BS and multiple UEs with caching capability. Each UE can transmit cached contents to its neighbors via D2D links [13]. Moreover, we assume that the BS can retrieve any content items from the core network, and all the required contents can be obtained from the BS. Let $\mathcal{U} = \{1, 2, \ldots, \kappa\}$ be the set of UEs covered by the BS. For a specific UE u, it requests content from the item library $\mathcal{F} = \{1, 2, \ldots, \beta\}$ with the size of each item $\mathcal{C}_{\mathcal{F}} = \{\varsigma_1, \varsigma_2, \ldots, \varsigma_\beta\}$. Moreover, let $\mathcal{C}_{\mathcal{U}} = \{c_1, c_2, \ldots, c_\kappa\}$ be the set of storage capacity of UEs. When UE u requests for item f, it first broadcasts this request to the nearby UEs. The UEs with the required item f in their storage send a reply. Then, the UE u chooses the nearest UE (the item holder) to obtain this content. If no UE replies to the request, BS should respond to the request.

We do not consider the UE mobility in this work; thus, the locations of UEs are unchangeable. Denote the distance between UE u and UE v as $d_{u,v}$. UE u can connect to UE v only if $d_{u,v} < R_{d2d}$, where R_{d2d} represents the D2D communication range threshold. Table I gives the list of important notations used in this study.

B. Content Caching Schemes

In this work, we consider three possible caching models as shown in Fig. 1, including self-caching, D2D-caching, and BS transmission.

1) *Self-Caching:* When UE u requires content items, it will first check whether the exact cache has been cached in the local storage. The request will be satisfied immediately if the local cache hits.
2) *D2D-Caching:* If the required content items are not cached at the local device, UE u will search the nearby devices that are located within the radius of R_{D2D} trying to get the required item. If all the UEs in this communication coverage do not have the requested content items, the request cannot be met. Note that a UE with needed caching can meet the requests of multiple UEs.
3) *BS Transmission:* BS transmission will be the last method for UE u to obtain the desired content when neither Self-caching nor D2D-caching hits the requirement. The BS will receive the request of UE u and transmit items from the server in the core network.

TABLE I
List of Important Notations

Notation	Definition
$\mathcal{U} = \{1, 2, \ldots, \kappa\}$	Set of UEs
$\mathcal{F} = \{1, 2, \ldots, \beta\}$	Set of contents
ς_β	Size of content β
c_κ	Storage size of UE κ
$d_{u,v}$	Distance between UE u and UE v
R_{d2d}	D2D comunication range threshold
$SNR_{u,v}$	Signal-to-noise ratio between UE u and UE v
P_{d2d}	Transmission power of the user device
B_{d2d}	Bandwidth of the D2D link
$G_{u,v}$	Channel gain of UE u to v transmission
σ_N^2	Gaussian white noise power
$\omega_{u,v}$	Transmission rate of a D2D pair UE u and v
$\tau_{u,v}$	Transmission latency for UE u to fetch item
$\mathcal{A}_u$	Action of UE u
$\mathcal{S}_u$	State of UE u
$\mathcal{P}_u$	Local content popularity
$\mathcal{Q}_u$	Local item hit-rate
Y_u^t	Total transmission latency reduction of UE u
R_t	Total reward
$\hat{R}_t$	Future discounted return
γ	Discount factor of the reward
α	Learning rate
$\boldsymbol{w}$	Weight of the deep Q network
$\boldsymbol{w}^-$	Weight of the target network
L_t	Loss of the training model
δ_t	Target error
$\upsilon_{u,t}$	Training speed of UE u
$\boldsymbol{g}_t$	Stochastic gradient
$\boldsymbol{w}_{u_a}^t$	Weight of area model u_a at time t
$\boldsymbol{w}_g^r$	Weight of the global model at round r

C. Transmission Latency

There are two kinds of transmission links: 1) D2D link and 2) BS-UE link. Therefore, we discuss the two transmission scenarios separately.

We assume that both the wireless bandwidth and transmit power are evenly allocated among the multiple serving devices. For the D2D link, let $SNR_{u,v} = P_{d2d} \cdot G_{u,v}/\sigma_N^2$ represent signal-to-noise ratio (SNR), where P_{d2d} denotes the transmission power of the user device, $G_{u,v}$ represents the channel gain of D2D transmission, and σ_N^2 is the Gaussian white noise power. Furthermore, for the channel gain of D2D transmission, we have $G_{u,v} = k_{d2d} \cdot d_{u,v}^{-\varepsilon_{d2d}}$, where k_{d2d} and ε_{d2d} denote the path-loss constant and exponent of the D2D

link, respectively. The transmission rate of a D2D pair UE u and v is $\omega_{u,v} = B_{d2d} \cdot \log_2(1 + SNR_{u,v})$, where available B_{d2d} is the bandwidth of the D2D link. Therefore, the latency for UE u to fetch item from UE v is $\tau_{uv}^f = \varsigma_f/\omega_{u,v}$. For the BS-UE link, we use a similar way to calculate SNR $SNR_{u,0}$ and transmission rate $\omega_{u,0}$ of UE u. Unlike the D2D link, UE u cannot directly fetch item f from the BS. Before the BS transmits an item to UE, it first retrieves this item from the core network. We assume this retrieve delay as a constant ϵ. Hence, the transmission latency for u getting item f from BS is $\tau_{u0}^f = \varsigma_f/\omega_{u,0} + \epsilon$.

IV. Problem Formulation and DRL-Based Model Training

In this section, to minimize the total transmission latency of all UEs in the D2D caching system, we model this cache scheme design as a multiagent MDP problem. Then, we propose a DRL-based local model training process.

A. Problem Formulation

Based on the system model described in Section II, the caching scheme design should be formulated as a multiagent MDP problem as follows.

1) Action: In this system, UE can cache multiple content items in one period. Denote $\mathcal{A}_u = \{a_{u,1}, a_{u,2}, \hat{a}\breve{A}\c{e}, a_{u,\beta}\}$ as a set of actions for whether UE u caches content items or not. Specifically, $a_{u,f} \in \{0, 1\}$ is a binary decision variable, where UE u caches item f if $a_{u,f} = 1$ and $a_{u,f} = 0$ otherwise. Note that the total size of the cached content items cannot exceed the storage capacity c_u of UE u, i.e., $\sum_{f=1}^{\beta} \varsigma_f \cdot a_{u,f} \le c_u$.

2) State: The state of UE u at the time tth is denoted as $\mathcal{S}_u = \{\mathcal{P}_u, \mathcal{Q}_u\}$. Here, $\mathcal{P}_u = \{p_{u,1}, p_{u,2}, \ldots, p_{u,\beta}\}$ is the local content popularity in the area χ, while $\mathcal{Q}_u = \{q_{u,1}, q_{u,2}, \ldots, q_{u,\beta}\}$ is the local cache hitting rate. The local content popularity and local item hit-rate are determined by UE u and its neighbors. The request rate of content f is $p_{u,f}^t = (n_{u,f}^t/[\sum_{f=1}^{\beta} n_{s,f}^t])$, where $n_{u,f}^t$ is the number of requests for item f. Besides, $q_{u,f}^t = (m_{u,f}^t/n_{ue}^t)$ is the local hit-rate of item f, where $m_{u,f}^t$ is the number of UEs fetch item f from UE u, and n_{ue} is the total number of UE u's neighbors.

3) Reward and Return: The transmission latency of UE u for caching content item f at time t is given by

$$\Gamma_{u,f}^t = \left(1 - a_{u,f}^t\right) \cdot \left[Z_{u,f}^t + \tau_{u0}^f \cdot \prod_{\mu \in \mathcal{N}(u)} \left(1 - a_{\mu,f}^t\right) \right] \quad (1)$$

where $\mathcal{N}(u)$ denotes the credible potential neighbors of u, μ is the UE with the μth lowest latency for sending content items to UE u, and $Z_{u,f}^t$ denotes the lowest latency of UE u to fetch the content item f from $\mathcal{N}(u)$, which is given by

$$Z_{u,f}^t = \sum_{u=1}^{|\mathcal{N}(u)|} \left(\tau_{u,\mu}^f \cdot \prod_{v=1}^{\mu-1} \left(1 - a_{v,f}^t\right) a_{\mu,f}^t \right) \quad (2)$$

where $\prod_{v=1}^{\mu-1}(1 - a_{v,f}^t)a_{\mu,f}^t$ is the indicator function with the meaning that no UE can fetch content faster than μ. When μ is larger than the total number of neighbors, it means that no UE in $\mathcal{N}(u)$ caches the content item f. In addition, when UE u fetches item f, the transmission latency reduction can be calculated by

$$Y_{u,f}^t = \tau_{u0}^f - \Gamma_{u,f}^t. \quad (3)$$

If $a_{u,f}^t = 1$, UE u caches the content item f. In other words, the request should be satisfied immediately by the self-caching. If $\prod_{v=1}^{\mu-1}(1 - a_{v,f}^t)a_{\mu,f}^t = 1$, UE μ will deliver item f to UE u through the D2D link. Otherwise, UE u will fetch the content item from BS with $\prod_{v=1}^{|\mathcal{N}(u)|}(1 - a_{v,f}^t) = 1$. Therefore, based on (4), we have

$$Y_{u,f}^t = \begin{cases} \tau_{u0}^f, & \text{self} - \text{cache} \\ \tau_{u\mu}^f, & D2D - \text{cache} \\ 0, & \text{otherwise.} \end{cases} \quad (4)$$

The total reduction of the transmission latency for UE u in round t is defined as

$$Y_u^t\left(a_{u,1}^t, \ldots, a_{u,f}^t\right) = \sum_{f=1}^{\beta} b_{u,f}^t \cdot Y_{u,f}^t + \sum_{f=1}^{\beta} \sum_{\mu \in \mathcal{N}(u)} a_{\mu,f}^t \cdot b_{u,f}^t \cdot Y_{u,f}^t \quad (5)$$

where $b_{u,f}^t \in \{0, 1\}$ is a binary decision variable that indicates whether UE u requests for content items f at time t.

Therefore, for all the participated UEs, the total reduction of the transmission latency for self-cache and D2D-cache can be given by

$$Y_t(\boldsymbol{a_1^t}, \ldots, \boldsymbol{a_u^t}) = \sum_{u=1}^{\kappa} \sum_{f=1}^{\beta} b_{u,f}^t \cdot Y_{u,f}^t + \sum_{u=1}^{\kappa} \sum_{f=1}^{\beta} \sum_{\mu \in \mathcal{N}(u)} a_{\mu,f}^t \cdot b_{u,f}^t \cdot Y_{i,f}^t. \quad (6)$$

The total reward that can be cooperatively obtained by UEs is given as

$$\begin{aligned} R_t(\boldsymbol{a_1^t}, \ldots, \boldsymbol{a_u^t}) &= \frac{1}{|\mathcal{N}(u)|} Y_t(\boldsymbol{a_1^t}, \ldots, \boldsymbol{a_u^t}) \\ &= \frac{1}{|\mathcal{N}(u)|} \sum_{u=1}^{\kappa} \sum_{f=1}^{\beta} b_{u,f}^t \cdot Y_{u,f}^t \\ &\quad + \frac{1}{|\mathcal{N}(u)|} \sum_{u=1}^{\kappa} \sum_{f=1}^{\beta} \sum_{\mu \in \mathcal{N}(u)} a_{\mu,f}^t \cdot b_{u,f}^t \cdot Y_{i,f}^t. \end{aligned} \quad (7)$$

In order to ensure the highest overall reward, some UEs may sacrifice their own storage space to meet the needs of others. Hence, in some cases, the content transmission latency of partial UEs could be relatively high. To avoid this problem, we add a constraint that the maximum average latency for each UE to fetch an item cannot exceed the threshold $\tau_{\max}$.

The goal of the UEs is to cooperatively distribute content by selecting actions in a way that maximizes future returns (composed of the short-term rewards and long-term rewards). In this

scheme, we assume that the future returns are discounted by a factor of γ per time step, $0 < \gamma < 1$.

Thus, the future discounted return at time t is denoted as $\hat{R}_t$, and is shown as

$$\begin{aligned}\hat{R}_t &= R_t + \gamma R_{t+1} + \gamma^2 R_{t+2} + \gamma^3 R_{t+3} + \cdots \\ &= \sum_{i=0}^{m-1} \gamma^i R_{t+i} + \gamma^m \hat{R}_{t+m}. \end{aligned} \tag{8}$$

B. DRL-Based Local Model Training

In this study,there are three kinds of models in our proposed BDRFL, which are: 1) the local model; 2) area model; and 3) global model. The parameter of the global model is updated by utilizing the area models, as there are several areas in this D2D network. Each area model is updated by using multiple local models in the area leader's coverage area. Meanwhile, each UE uses the local data to train its local model. In this part, we illustrate the DRL-based local model training process. The area model and the global model update are elaborated in Section V.

The basic idea behind the majority of value-based RL is to estimate the action-value function. As actions $\{\mathcal{A}_{t+1}, \mathcal{A}_{t+2}, \mathcal{A}_{t+3}, \ldots\}$ and state $\{\mathcal{S}_{t+1}, \mathcal{S}_{t+2}, \mathcal{S}_{t+3}, \ldots\}$ are integrated out, only observations $\mathcal{A}_t = \boldsymbol{a}_t$ and $\mathcal{S}_t = \boldsymbol{s}_t$ remain. The action-value function is as follows:

$$Q_\pi(s_t, a_t) = \boldsymbol{E}\Big[\hat{R}_t | \mathcal{S}_t = \boldsymbol{s}_t, \mathcal{A}_t = \boldsymbol{a}_t\Big] \tag{9}$$

where Q_π is the return of taking action $\boldsymbol{a}_t$ in the current state $\boldsymbol{s}_t$, which is related to the policy function π. Moreover, we eliminate the influence of the strategy function on the choice of return action and maximize Q_π. The optimal action-value function is

$$Q^*(\boldsymbol{s}_t, \boldsymbol{a}_t) = \max_\pi Q_\pi(\boldsymbol{s}_t, \boldsymbol{a}_t) \tag{10}$$

where Q^* is independent of the strategy function, which represents the expected return of the best action $\mathcal{A}^\star$ in state $\mathcal{S}$. Whatever policy function π is performed, the result of taking action $\boldsymbol{a}_t$ at state $\boldsymbol{s}_t$ cannot be better than $Q^*(\boldsymbol{s}_t, \boldsymbol{a}_t)$. Moreover, we approximate the optimal action-value function $Q^*(\boldsymbol{s}_t, \boldsymbol{a}_t)$ by $Q(\boldsymbol{s}_t, \boldsymbol{a}_t; \boldsymbol{w})$, where $\boldsymbol{w}$ is a neural network parameter. Note that all actions are scored by the optimal action-value function to select the best action.

Facing the large state space and action space in our D2D caching problems, traditional RL methods are difficult to accurately estimate action value [23]. The value-based DRL uses neural networks to approximate the action value, thus to effectively tackle the large state space and action space. While the majority of value-based methods, such as original DQN and nature DQN, easily lead to overoptimistic value estimates, double DQN (DDQN) eliminates the problem of overestimation by using two different value functions to decouple the selection and the evaluation [24]. Therefore, we use DDQN in this work to reduce the impact of nonuniform overestimation and yield more accurate value estimates. At time t, the state space, action space, reward, and the next state can be packed as a transition $(\boldsymbol{s}_t, \boldsymbol{a}_t, r_t, \boldsymbol{s}_{t+1})$. Moreover, we use a larger buffer with capacity n_b to store these samples. Note that only recent n_t transitions are stored in a replay buffer, and old transitions will be removed. Furthermore, we use the stochastic gradient descent (SGD) scheme to randomly sample a transition from the buffer to compute the TD error and then calculate the stochastic gradient. The expected future return $\boldsymbol{E}[\hat{R}_t]$ parameterized by the weight $\boldsymbol{w}$ is denoted as $Q(\boldsymbol{s}_t, \boldsymbol{a}_t; \boldsymbol{w})$, and is given by

$$Q(\boldsymbol{s}_t, \boldsymbol{a}_t; \boldsymbol{w}) \approx r_t + \gamma Q(\boldsymbol{s}_{t+1}, \boldsymbol{a}_{t+1}; \boldsymbol{w}). \tag{11}$$

In DDQN, the best action selection is given

$$\boldsymbol{a}^* = \operatorname*{argmax}_x Q(\boldsymbol{s}_{t+1}, \boldsymbol{a}; \boldsymbol{w}). \tag{12}$$

The evaluation in DDQN using the TD target network is

$$y_t = r_t + \gamma \cdot \max_{\boldsymbol{a}} Q\big(\boldsymbol{s}_{t+1}, \boldsymbol{a}^*; \boldsymbol{w}^-\big) \tag{13}$$

where $\boldsymbol{w}^-$ is the parameter of the TD target network, the structure of the TD target network is the same as that of the DQN. Moreover, γ helps to balance the short-term reward and long-term reward. DDQN aims to shorten the difference between $\boldsymbol{w}^-$ and $\boldsymbol{w}$ by minimizing the loss function. The loss function is given by

$$L_t(\boldsymbol{w}) = \frac{1}{2}\big[Q(\boldsymbol{s}_t, \boldsymbol{a}_t; \boldsymbol{w}) - y_t\big]^2 = \frac{\delta_t^2}{2} \tag{14}$$

where δ_t is the target error.

We draw samples randomly from the pool of reply buffer. Then, we use the loss to update $\boldsymbol{w}$ as per the following rule:

$$\begin{aligned}\boldsymbol{w}_{u,t+1} &= \boldsymbol{w}_{u,t} - \alpha \cdot \boldsymbol{g}_{u,t} = \boldsymbol{w}_{u,t} - \alpha \cdot \frac{\partial\big(\delta_t^2/2\big)}{\partial \boldsymbol{w}} \\ &= \boldsymbol{w}_{u,t} - \alpha \cdot \delta_t \cdot \frac{\partial Q(\boldsymbol{s}_t, \boldsymbol{a}_t; \boldsymbol{w})}{\partial \boldsymbol{w}} \end{aligned} \tag{15}$$

where $\boldsymbol{g}_t$ is the stochastic gradient and α denotes the learning rate. The DDQN-based algorithm for content delivery optimization is summarized in Algorithm 1.

V. Privacy-Preserved and Secure BDRFL Caching Scheme Design

FL is a good alternative that enables UEs learning without any centralized server [25]. Also, it is a solution for limiting raw data transfer and accelerating learning processes of UEs [26]. However, various failures may occur in these distributed D2D caching systems. Basically, these failures can be divided into three classifications.

1) *Crash Failure:* The crash is termed a fail stop, and it makes no further process. The crash failure is serious and comes without any signal.
2) *Omission Failure:* A node fails to do partial expected processes (like the sending process, receiving process, etc.). Once an omission failure happens, the UE will report the error.
3) *Byzantine Failure:* A node exhibits arbitrary, erratic, and unexpected behaviors. Moreover, these behaviors may be malicious and disruptive.

These failures seriously interfere with the learning process. Fortunately, the blockchain is capable of enabling FL coordination under failures by utilizing the consensus mechanism, and it has been proved validation of offering a reliable method

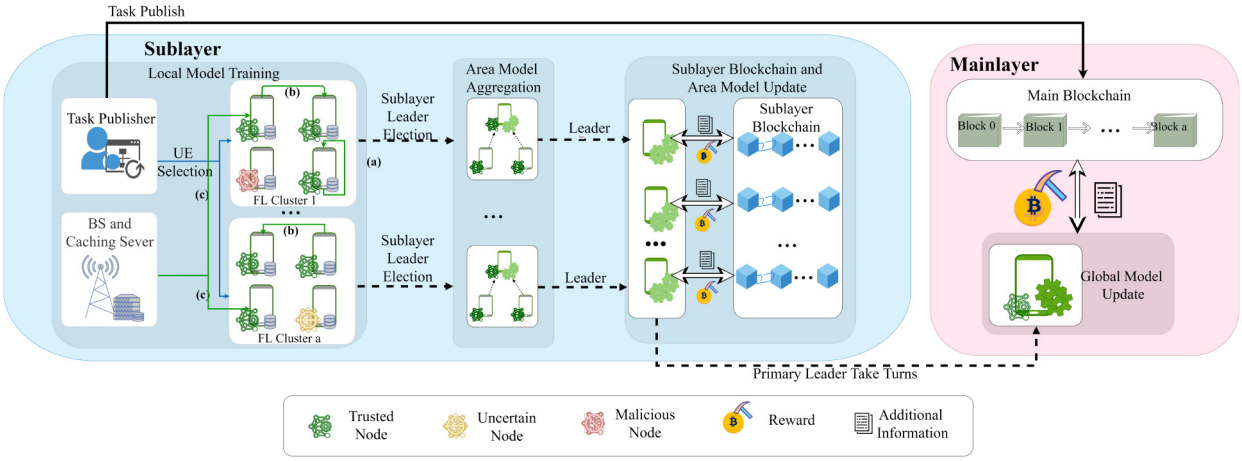


Fig. 2. Blockchain and FL framework of the D2D caching system.

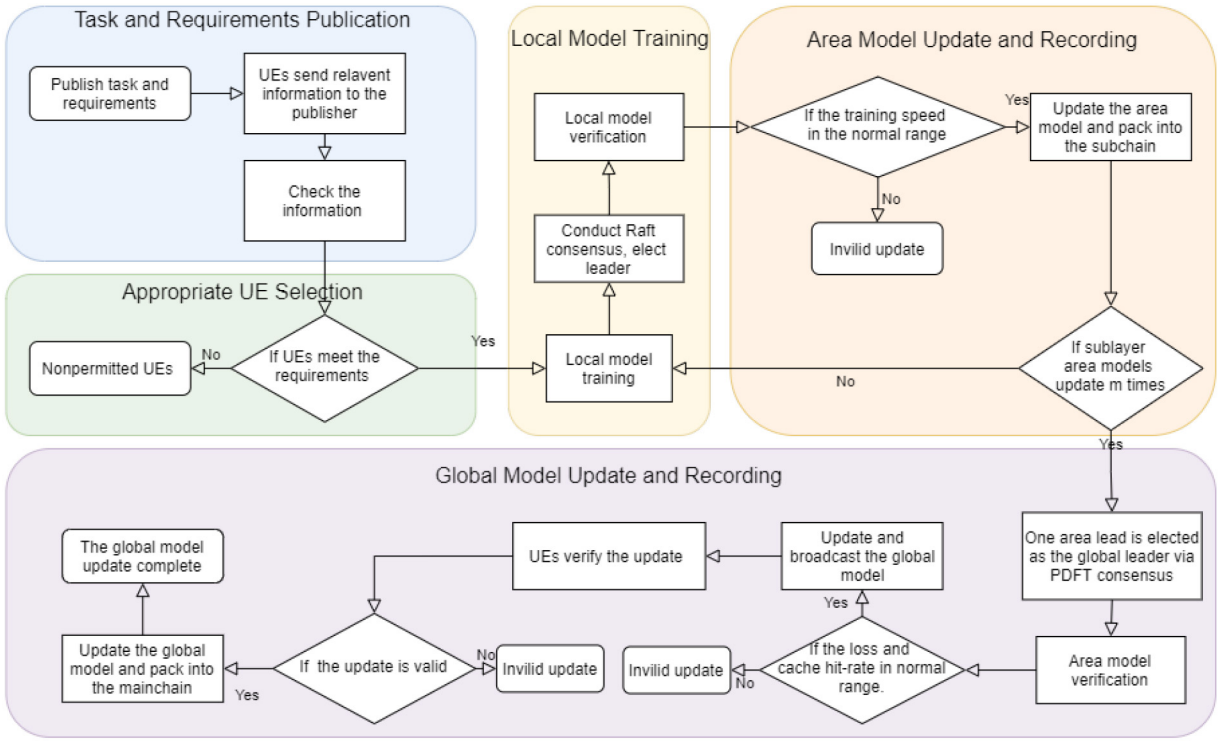


Fig. 3. Blockchain-empowered federated model update flowchart.

for information tamperproof [11], [27]–[29]. Therefore, with the aim to enhance data privacy while improving the transparency and credibility of D2D caching systems, we propose a double-layer blockchain architecture in BDRFL to underpin an efficient D2D caching scheme, as shown in Fig. 2. The sublayer Raft-based blockchains help UEs to reach consensus under crash and omission failures in the large-scale sublayer network, while the PBFT consensus is capable of detecting Byzantine failures in the small-scale mainlayer network. To achieve this secure and intelligent D2D caching framework, five steps are involved in our proposed BDRFL under the double-layer blockchain architecture, including the task and requirements publication, appropriate UE selection, local model training, area model update and recording, and finally global model update and recording, as shown in the flowchart of Fig. 3. In the following, let us illustrate these steps.

A. Task and Requirements Publication

The task publisher broadcasts FL tasks and determines the key requirements. Specifically, starting from the round 0, the publisher packages the initial weight parameters of global model $\boldsymbol{w}_g$ and the initial state to the first block, i.e., block 0. If UEs intend to join the task training and satisfy the data requirement, they need to send their participation requests, identity code, data resources information, and other relevant information (e.g., CPU/GPU model, storage, and battery).

Algorithm 1 DQN to Implement the Optimization of Content Delivery

Initialize the parameter of models $\boldsymbol{w}_u, u \in \mathcal{U}$, the target networks of UEs with random weights $\boldsymbol{w}_u^-$, reply memories $n_{b,u}$, capacity c_u and discount factor γ.
for epsiode $T = 1, ..., m$ **do**
  for $u = 1, ..., \kappa$ **do**
    Select an action $\boldsymbol{a}_{u,t}$ which is the best Q-value action or random action.
    Input the initial transmit power P_u, the channel gain G_u, the D2D range r_{d2d}, the distance from user u to its neighbors d_{uv}, namely $I_u = \{P_u, G_u, r_{d2d}, duv\}, v \in \mathcal{N}_u$.
    Request for content f.
    Fetch content item and calculate the reduction value of latency $Y_{u,t}$.
    if cache sharing stage end **then**
      Observe a new state $\boldsymbol{s}_u^{'}$.
    end if
  end for
  $Y_t = \frac{1}{\kappa} \sum_{u=1}^{\kappa} Y_{u,t}$.
  Observe the reward r_t according to Y_t, $r_{u,t} = r_t$.
  for $u = 1, ..., \kappa$ **do**
    Store the transition $\{\boldsymbol{s}_u, \boldsymbol{a}_u, r_u, \boldsymbol{s}_u^{'}\}$.
    Calculate the target loss according to (13), perform a gradient descent step on (14), thus to update $\boldsymbol{w}_u$
    Every j step update the $\boldsymbol{w}_u^-$ of target network.
  end for
end for

B. Appropriate UE Selection

Appropriate UEs are chosen by the task publisher. According to the identity code and additional information, such as computing power, storage size, and battery size, the task publisher determines whether UEs have the ability to participate in cache sharing and model training. The approved UEs $\mathcal{U}$ participate in the initial round.

C. Local Model Training

The approved UEs in each area execute the sublayer Raft consensus and elect an area leader from their neighbors. The area leaders are UEs with the best cache hit rate and strong computing power. The leader in each area first downloads the global model from the nearest block of the mainchain and broadcasts it to nearby UEs. Subsequently, UEs $\mathcal{U}$ predict and cache the item that may request by themselves and their neighbors. In the cache sharing slot, UEs fetch required content in the order of self-caching, D2D-caching then BS-transmission. The UEs update the local cache popularity and local cache hit rate after all content requests from UEs are satisfied in the sharing slot. Next, the average latency reduction of users can be calculated, which is used to update the reward. At time t, every UE packs their state space, action space, reward, and the next state as a transition (s_t, a_t, r_t, s_{t+1}). Each UE has a larger buffer with a capacity n_b that is used to store its own transitions. Only recent n samples are stored in the replay buffer, and old transitions will be removed. Each UE randomly samples a transaction from their buffer to compute the TD error, thus to update the weights of its local model. After all local models are updated, the parameter of local models and some publicly available information, such as training data size, training time, and local content popularity, are sent to the area leaders.

D. Area Model Update and Recording

The sublayer area leaders are in charge of the verification of local gradients and area models update. These leaders are elected by UEs in each area by executing the Raft consensus. Each area leader maintains an area model.

Specifically, sublayer leaders receive the local models from UEs in their coverage area. Then, the leaders check whether the training speed $\upsilon_{u,t}$ matches to the training time $\tau_{u,t}$ and training data size $\varsigma_{u,t}$, thus to determine the authenticity of a local model, $\upsilon_{u,t} = (\varsigma_{u,t}/\tau_{u,t})$. If the speed is within the normal training speed range $\upsilon_{\min} < \upsilon_{u,t} < \upsilon_{\max}$, the model will be certified as a reasonable update. Only those verified local models can be used to update the area models. The updated area model and additional information (such as the number of content distributions, training time, and training data size) in each area are recorded in the corresponding sublayer blockchain as reliable and tamper-proof transactions for future verification and updates. According to the number of content distributions for others, UEs get some rewards that can be an amount of virtual currency from the task publisher, which encourages more UEs to participate in this D2D caching system.

E. Global Model Update and Recording

There is one global leader each round takes charge of global model update. The sublayer leaders execute PBFT consensus and take turns to be the mainlayer leader. Once the mainlayer leader received a request for area model verification, it first checks the loss between the previous global model and each area model. Subsequently, the global leader tests the cache hit rate to evaluate the cache prediction performance of these area models. If both the loss $l_{a,r}$ and the hit rate $q_{a,r}$ are in rational ranges, this area model can be utilized in the global model update. Once the global leader completes the model update, it broadcasts the update to area leaders, and area leaders check the update by testing the cache hit rate. Only over half of the leaders confirm the validity of the model, and the updated model can be determined as the new global model. Then, the global leader packs the verified global model and additional information (e.g., area model gradient) into the mainchain. The global leader UE can get some rewards from the blockchain for package data and update blocks.

VI. Consensus Mechanism and Federated Learning Model Update

In this section, we first investigate the consensus mechanism of the double-layer blockchain system, and then discuss how the FL model is updated in this framework.

A. Double-Layer Blockchain Consensus Mechanism

There are three classifications of failures in the D2D caching system, which are the crash failure, omission failure, and

Byzantine failure. Among these failures, the Byzantine failure is most difficult to detect and can cause a vital impact on the caching network security. With the aim of building a reliable and secure D2D caching scheme, we consider the existence of Byzantine failures in this study. However, most of the consensus mechanisms that can detect Byzantine failures are energy intensive and costly, such as PoW and Proof of Stake (PoS). Limited by the low computing capability of user devices, these mechanisms cannot be directly applied to our scheme. Besides, the numerous sublayer nodes also bring exponential growth of consensus complexity when using PBFT or BFT binomial. Therefore, a new blockchain framework and appropriate consensus mechanisms should be explored in this work. In this study, we proposed a double-layer blockchain-empowered FL. The sublayer Raft-based blockchains only consider the crash and omission failures, while the PBFT-based mainchain resolves the Byzantine failure.

In order to allow more users to participate in D2D caching while taking the reliability of the model into account, for the subchains, we require this algorithm has strong consistency and high consensus efficiency, regularly ensuring liveness, observing system status, and distributing machine data. Besides, this algorithm-based system should be maintained, and coordinated independently and rationally. Moreover, it should be easily implemented to allow devices with low computing power to quickly reach a consensus. Fortunately, these standards are consistent with the characteristics of the Raft, which prompts us to choose it.

For the mainchain, we implement the PBFT consensus mechanism to prevent the global model from being severely affected by malicious nodes. The number of nodes in the mainchain can be greatly reduced after supernodes are selected out from several areas. The computational complexity of using the PBFT can thus be decreased due to the few nodes for the consensus process, which makes it applicable for the mainchain.

In this way, the subchains can resolve the crash and omission failures and help UEs reach consensus in a short slot, while the mainchain resists Byzantine failures caused by malicious UEs.

B. FL Area Model Update in Subchain Layer

There are three categories of nodes in Raft consensus, saying leader, candidate, and follower [30]. Each area can only select one leader UE that with the highest cache hit rate and strong computing power from all the UEs within this cluster region, and then the rest of UEs become follower nodes. Candidate nodes are followers at the intermediate state trying to become the next leader node.

The FL model update in sublayer is composed of two stages.

1) *Leader Election Stage:* The leader UE in each cluster sends heartbeats to all followers. If followers cannot receive the heartbeats of the leader within the election timeout period, the leader election will be initiated. These followers transfer to the candidate state and update their term numbers. A follower first votes for itself and then sends RequestVoteRPC, which is a set of data about its information to request other nodes to vote for this follower. When it wins the majority of votes, it will become the new leader and regularly send heartbeats to all followers to maintain its rule.
2) *FL Model Update Stage:* The leader UE starts to receive information from other neighbor UEs, including local models and some extra information. Then, the leader checks whether the training speed matches to the training time and data size, thus to determine the authenticity of a local model. Only those verified local models can be utilized in the area model update. The area model update function is shown as follows:

$$\boldsymbol{w}_{u_a}^{t+1} = \boldsymbol{w}_{u_a}^{t} - \alpha \cdot \sum_{u \in \mathcal{N}(u_a)} \frac{s_{n,u}}{S_a} \cdot \boldsymbol{g}_{u,t} \tag{16}$$

where $\boldsymbol{w}_{u_a}^{t}$ is the weight parameters of area model updated by leader UE u_a, S_a denotes the total size of training data samples from UEs in area a, $S_a = \sum_{u \in \mathcal{N}(u)} s_{n,u}$, and $\boldsymbol{g}_{u,t}$ is the gradient of the local model trained by UE u. The updated area model and the relevant information are packed into a new block in the sublayer blockchain. These data sets are transparent so that can be checked and verified. The model update process in sublayer layer FL with blockchain is summarized in Algorithm 2.

Algorithm 2 Model Update Process in Sublayer FL With Blockchain

```
Initialize the parameter of global model w_g, the local models of UEs with random weights w_u, u ∈ U, weights of area models w_{u_a}, u_a ∈ U_a, reply memories n_{b,u}, cache capacity c_u and discount factor γ.
for time T=1, ..., m do
    for u = 1, ..., κ do
        Download the global model.
        Select a joint action a_{u,t} = max_a Q{s_{u,t}, a_u, w_u}.
        Input the initial transmit power, the channel gain, the D2D range, the distance from user u to its neighbors d_{uv}, namely S_u = {P_u, Q_u}, v ∈ N(u).
        Observe the environment state s'_u.
        Get reward r_l.
        Update the parameters of the local model.
        Upload information and model to leader u_a ∈ U_a.
    end for
    Implement Raft consensus and elect new leaders.
    for u_a = 1, ...ι do
        Varificate updates.
        Perform model update step on (16).
        Pack updated model and additional information into the new block of sublayer blockchain.
    end for
end for
```

C. FL Global Model Update in Mainchain Layer

In the PBFT-based mainchain, the nodes are divided into two categories: 1) primary node and 2) child node [31]. Only area leaders are involved in the mainlayer global model and

Algorithm 3 Model Update Process in Mainlayer FL With Blockchain

for Round=1, ..., r **do**
  for $u_a = 1, ..., \iota$ **do**
    Fetch area updates of every cluster.
    Verify update model and data.
  end for
  Implement PBFT consensus and a area leader UE become a new global leader u_g.
  for $u = u_g$ **do**
    Perform global model update step on (17).
    Pack updates and additional data to the mainchain.
  end for
end for

mainchain update. The area leaders become the global leader in a round-robin mode. That means, there is merely one primary node each time, and other area leader UEs perform as the child nodes. All nodes are capable of communicating with each other. The ultimate goal is that all sublayer leaders reach a consensus on a principle of the minority obeying the majority.

Once the primary node received a request for area model verification, it first checks the loss between the previous global model and every area model, and then evaluates the cache prediction performance of these models. Only well-performed area models with normal loss and cache hit rate are used to update the global model. Then, the primary node broadcasts the verification request to all the child nodes. Here, we should note that the maximum number of malicious nodes in PBFT that can be tolerated is $f = (n_a - 1)/2$, where n_a is the total number of sublayer leader UEs. Unlike the Raft, child UEs in PBFT have the right to question the reliability and rationality of the primary UE. After the primary UE analyzes all area models and updates parameters of the global model, the child UEs can test the update by using it to predict cache. Though checking the cache hit rate, the child UEs determine whether the model update is valid, reasonable, and effective. The global model update has a longer update time slot, after area models update several times, the global model update once, it is capable of avoiding the waste of communication resources. The update function of global model is given by

$$\boldsymbol{w}_g^{r+1} = \sum_{u_a=1}^{\iota} \frac{\boldsymbol{w}_{u_a}^r}{\iota} \tag{17}$$

where $\boldsymbol{w}_g^{r+1}$ is the weight parameters of global model at round $r+1$. The model update process in mainlayer FL with blockchain is detailed in Algorithm 3.

VII. Simulation Results and Discussions

In this section, we evaluate the performance of our proposed BDRFL scheme through simulations.

We compare our proposed BDRFL caching scheme with the four following caching schemes.

1) *Centralized DQN-Based Scheme:* This scheme collects and utilizes all users' information to train a DQN model by a central server [16].

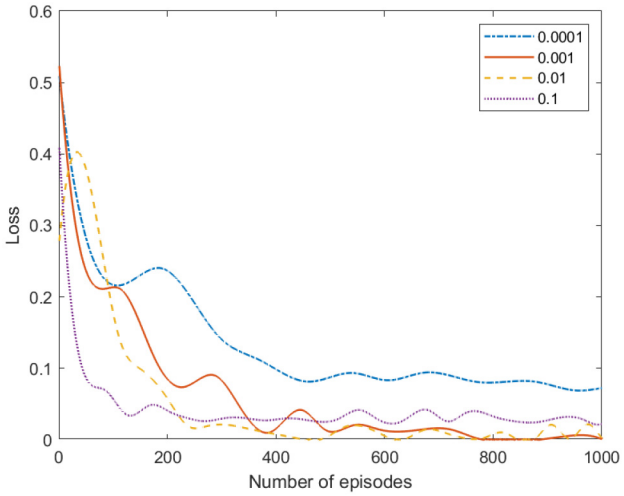


Fig. 4. Training process of BDRFL under different learning rate.

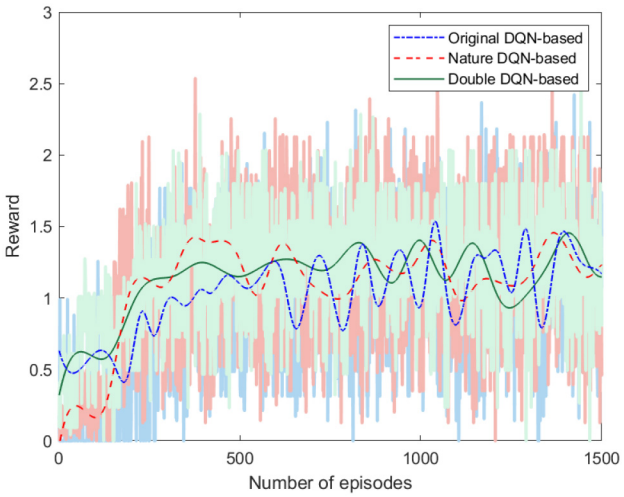


Fig. 5. Training process of BDRFL with different DQN-based methods.

2) *Distributed DQN-Based Scheme:* Cluster leader UEs collect neighbor UEs' information and update models independently, and each leader maintains a learning model [17].
3) *FL-Based Scheme:* This scheme uses FL in model training, and each UE maintains a local model. All the local models are utilized to update the global model [18].
4) *Zipf Random:* UEs randomly cache content with an assumption that the content popularity obeys Zipf distribution.

We consider crash, omission, and Byzantine failures in this D2D caching network. All these failures are caused by attacks. In our simulations, a crash failure is that a UE halts all the activities (such as cache sharing and model training). A UE that stops updating the model or sending training data for ten rounds is the omission failure. The Byzantine failure is that a malicious UE randomly generates fake model updates/training data.

A. Simulation Setting

We consider a cache-enabled D2D network scenario with a total of three clusters covering five UEs for each. All UEs have the same capacity with the size of $c_u = 1000$ MB.

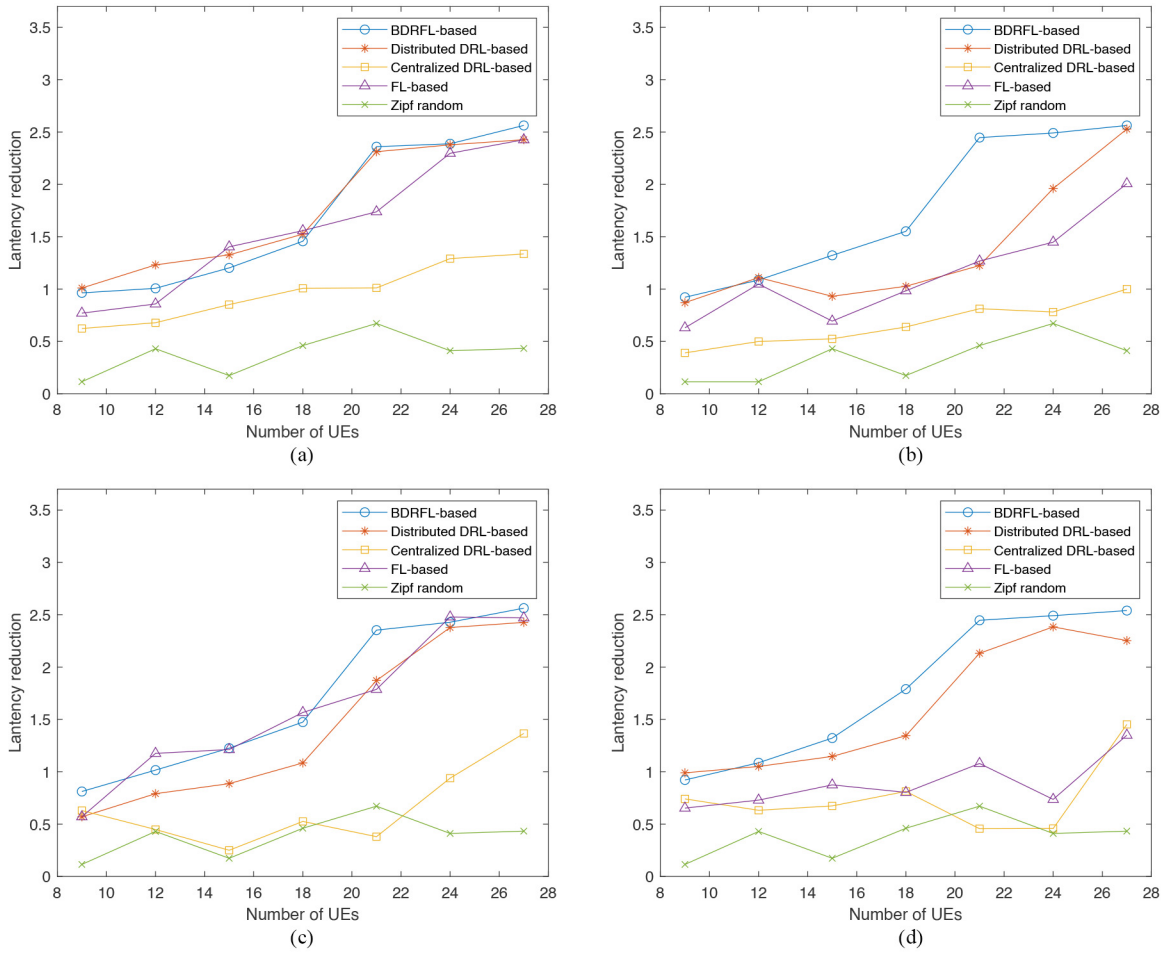


Fig. 6. Reduced value of average latency under different UE numbers. (a) Comparisons of latency reduction without any failure. (b) Comparisons of latency reduction under the crash failure. (c) Comparisons of latency reduction under the omission failure. (d) Comparisons of latency reduction under the Byzantine failure.

There are ten types of contents with the same size. Moreover, we assume that UEs can communicate with each other in the same area via D2D links. The number of requested contents of a UE in a round is randomly generated within $[1, 3]$. The request pattern of a certain UE is modeled by the Zipf distribution. We set different content popularity parameters based on the heterogeneous preferences of UEs.

The details of DQN networks in this simulation are set as follows. The input layer neural number, the hidden layer neural number, and that of the output layer are 20, 10, and 10, respectively. We use *Tanh* as the activation function from the input layer to the hidden layer, and the activation function from the hidden layer to the output layer is *ReLU*. The size of the reply buffer is set to 100 and the number of transitions used to calculate the target loss is set to 5. The training process under different learning rate are shown as Fig. 4, and we fix the learning rate α as 0.001 with the influence factor $\gamma = 0.1$ in the other following simulation.

B. Numerical Results

Then, we compare the average reward of BDRFL when using three different DQN-based methods, which are the original DQN-based, nature DQN-based, and DDQN-based methods in the scenario without any failure, shown as Fig. 5. From this figure, we find that the DDQN-based method always outperforms the other two methods in terms of reward convergence. This is because the original DQN and nature DQN always choose the action with the optimal value at the next decision slot to update the Q function, which leads to an overestimation issue.

We first evaluate the average download latency reduction of fetching items for the five schemes. Fig. 6(a) shows the latency reduction of these five caching schemes under the scenario without any failure. The average latency reduction of these five schemes all increases with the UE number. It is because the cache sharing network performs better under the higher user density. The latency reduction of the BDRFL-based caching scheme shows the fastest increase, while the FL-based scheme and distributed DRL-based scheme show similar performance. Meanwhile, applying the distributed DRL-based caching scheme only brings a bit more latency reduction than applying the worst schemes of Zipf random. The reason is the BDRFL, distributed DRL, and FL can achieve satisfactory model training and caching prediction without raw data sharing. However, the centralized DRL-based scheme requires global information, which cannot be shared in this simulation.

In order to evaluate the performance under the scenario with a crash failure, we compare the reduced transmission latency of these five schemes while there is a UE accidentally fail stop and cannot recover. From Fig. 6(b), we find that our proposed BDRFL caching scheme always outperforms than other four schemes in terms of average latency reduction. That is because the impact of the crash failure is minimized via blockchain consensus by canceling the qualification of that fail-stop UE. Besides, the crashed UE should be replaced by a qualified normal node. However, the other four schemes cannot detect and replace this failure node, thus the crashed UE does not cache and distribute content in the network.

Then, we assess the impact of the omission failure for the five caching schemes. Fig. 6(c) shows the reduced latency while a UE does not update the model or sharing the training data. From this figure, we find that the BDRFL caching scheme and the FL caching scheme achieve a similar latency reduction, which is larger than that of the other three schemes. It is because that FL helps the failure UE to replace the local bad-performance model with an updated global model via area leader broadcasting.

With the aim of evaluating the impact of the Byzantine failure, we compare the reduced transmission latency of the BDRFL scheme with the other four schemes in the scenario that a malicious node in the system randomly generates fake model updates/training data. Fig. 6(d) shows that the latency reduction of the BDRFL scheme increases significantly with the number of UEs, which is almost the same as that in the nonfailure scenario, but much higher than that of the other schemes. The reason is that the introduced blockchain in BDRFL is capable to verify model updates and detect malicious UEs. Only well-performed models are used to update the global model; hence, BDRFL can ensure the valid global model update and achieve satisfied performance. Besides, the qualification of the malicious UE is canceled and a new qualified UE is permitted to participate in, thus to eliminate the impact of malicious UEs on the cache sharing network.

VIII. Conclusion

In this study, we have developed an intelligent and privacy-preserving caching scheme BDRFL in the D2D network. BDRFL is based on a framework of FL underpinned by a double-layer blockchain system. We have illustrated the blockchain consensus of each layers, and streamlined the process of BDRFL including FL model training as well as model data recording on blockchain. We have conducted simulations in scenarios with and without malicious attacks, where numerical demonstrated both the improvements of caching performance and the reliability of resisting attacks. In general, this work can be seen as a pioneer to explore the interplay of blockchain and FL, thus to develop an intelligent and trusted caching scheme under an unreliable wireless network.

References

[1] R. Amer, M. M. Butt, M. Bennis, and N. Marchetti, "Inter-cluster cooperation for wireless D2D caching networks," *IEEE Trans. Wireless Commun.*, vol. 17, no. 9, pp. 6108–6121, Sep. 2018.

[2] D. Feng, L. Lu, Y. Yuan-Wu, G. Y. Li, G. Feng, and S. Li, "Device-to-device communications underlaying cellular networks," *IEEE Trans . Commun.*, vol. 61, no. 8, pp. 3541–3551, Aug. 2013.

[3] D. Feng *et al.*, "Mode switching for energy-efficient device-to-device communications in cellular networks," *IEEE Trans. Wireless Commun.*, vol. 14, no. 12, pp. 6993–7003, Dec. 2015.

[4] W. Zhuang, Q. Ye, F. Lyu, N. Cheng, and J. Ren, "SDN/NFV-empowered future IoV with enhanced communication, computing, and caching," *Proc. IEEE*, vol. 108, no. 2, pp. 274–291, Feb. 2020.

[5] M. Ji, G. Caire, and A. F. Molisch, "Fundamental limits of caching in wireless D2D networks," *IEEE Trans. Inf. Theory*, vol. 62, no. 2, pp. 849–869, Feb. 2016.

[6] Z. Chen, Y. Liu, B. Zhou, and M. Tao, "Caching incentive design in wireless D2D networks: A Stackelberg game approach," in *Proc. IEEE Int. Conf. Commun. (ICC)*, Kuala Lumpur, Malaysia, 2016, pp. 1–6.

[7] Y. Sun *et al.*, "Efficient handover mechanism for radio access network slicing by exploiting distributed learning," *IEEE Trans. Netw. Service Manag.*, vol. 17, no. 4, pp. 2620–2633, Dec. 2020.

[8] Q. Ye, W. Shi, K. Qu, H. He, W. Zhuang, and X. Shen, "Learning-based computing task offloading for autonomous driving: A load balancing perspective," in *Proc. Int. Conf. Commun. (ICC)*, vol. 21, 2021, pp. 1–6.

[9] J. Konečnỳ, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, and D. Bacon, "Federated learning: Strategies for improving communication efficiency," 2016. [Online]. Available: arXiv:1610.05492. 2016.

[10] K. Bonawitz *et al.*, "Towards federated learning at scale: System design," 2019. [Online]. Available: arXiv:1902.01046.

[11] B. Cao *et al.*, "Performance analysis and comparison of PoW, PoS and DAG based blockchains," *Digit. Commun. Netw.*, vol. 6, no. 4, pp. 480–485, 2020.

[12] Y. Sun, L. Zhang, G. Feng, B. Yang, B. Cao, and M. A. Imran, "Blockchain-enabled wireless Internet of Things: Performance analysis and optimal communication node deployment," *IEEE Internet Things J.*, vol. 6, no. 3, pp. 5791–5802, Jun. 2019.

[13] W. Jiang, G. Feng, S. Qin, T. S. P. Yum, and G. Cao, "Multi-agent reinforcement learning for efficient content caching in mobile D2D networks," *IEEE Trans. Wireless Commun.*, vol. 18, no. 3, pp. 1610–1622, Mar. 2019.

[14] B. Chen and C. Yang, "Caching policy for cache-enabled D2D communications by learning user preference," *IEEE Trans. Commun.*, vol. 66, no. 12, pp. 6586–6601, Dec. 2018.

[15] Z. Chen, N. Pappas, and M. Kountouris, "Probabilistic caching in wireless D2D networks: Cache hit optimal versus throughput optimal," *IEEE Commun. Lett.*, vol. 21, no. 3, pp. 584–587, Mar. 2017.

[16] L. Li *et al.*, "Deep reinforcement learning approaches for content caching in cache-enabled D2D networks," *IEEE Internet Things J.*, vol. 7, no. 1, pp. 544–557, Jan. 2020.

[17] N. Kumar, S. N. Swain, and C. S. R. Murthy, "A novel distributed Q-learning based resource reservation framework for facilitating D2D content access requests in LTE-A networks," *IEEE Trans. Netw. Service Manag.*, vol. 15, no. 2, pp. 718–731, Jun. 2018.

[18] R. Li *et al.*, "Edge caching replacement optimization for D2D wireless networks via weighted distributed DQN," in *Proc. IEEE Wireless Commun. Netw. Conf. (WCNC)*, Seoul, South Korea, 2020, pp. 1–6.

[19] R. Zhang, F. R. Yu, J. Liu, R. Xie, and T. Huang, "Blockchain-incentivized D2D and mobile edge caching: A deep reinforcement learning approach," *IEEE Netw.*, vol. 34, no. 4, pp. 150–157, Jul./Aug. 2020.

[20] R. Zhang, F. R. Yu, J. Liu, T. Huang, and Y. Liu, "Deep reinforcement learning (DRL)-based device-to-device (D2D) caching with blockchain and mobile edge computing," *IEEE Trans. Wireless Commun.*, vol. 19, no. 10, pp. 6469–6485, Oct. 2020.

[21] M. Liu, F. R. Yu, Y. Teng, V. C. M. Leung, and M. Song, "Computation offloading and content caching in wireless blockchain networks with mobile edge computing," *IEEE Trans. Veh. Technol.*, vol. 67, no. 11, pp. 11008–11021, Nov. 2018.

[22] Y. Lu, X. Huang, Y. Dai, S. Maharjan, and Y. Zhang, "Blockchain and federated learning for privacy-preserved data sharing in industrial IoT," *IEEE Trans. Ind. Informat.*, vol. 16, no. 6, pp. 4177–4186, Jun. 2020.

[23] Y. Li *et al.*, "Direct acyclic graph-based ledger for Internet of Things: Performance and security analysis," *IEEE/ACM Trans. Netw.*, vol. 28, no. 4, pp. 1643–1656, Aug. 2020.

[24] H. Van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double Q-learning," in *Proc. 30th AAAI Conf. Artif. Intell.*, vol. 30, 2016, pp. 2094–2100.

[25] S. Hosseinalipour *et al.*, "Multi-stage hybrid federated learning over large-scale D2D-enabled fog networks," 2020. [Online]. Available: arXiv:2007.09511.

[26] Y. Zhan, P. Li, Z. Qu, D. Zeng, and S. Guo, "A learning-based incentive mechanism for federated learning," *IEEE Internet Things J.*, vol. 7, no. 7, pp. 6360–6368, Jul. 2020.
[27] B. Cao, S. Xia, J. Han, and Y. Li, "A distributed game methodology for crowdsensing in uncertain wireless scenario," *IEEE Trans. Mobile Comput.*, vol. 19, no. 1, pp. 15–28, Jan. 2020.
[28] B. Cao, L. Zhang, Y. Li, D. Feng, and W. Cao, "Intelligent offloading in multi-access edge computing: A state-of-the-art review and framework," *IEEE Commun. Mag.*, vol. 57, no. 3, pp. 56–62, Mar. 2019.
[29] B. Cao *et al.*, "When Internet of Things meets blockchain: Challenges in distributed consensus," *IEEE Netw.*, vol. 33, no. 6, pp. 133–139, Nov./Dec. 2019.
[30] D. Ongaro and J. Ousterhout, "In search of an understandable consensus algorithm," in *Proc. USENIX Annu. Techn. Conf. (USENIX ATC)*, 2014, pp. 305–319.
[31] M. Castro and B. Liskov, "Practical Byzantine fault tolerance," in *Proc. 3rd Symp. Oper. Syst. Design Implement. (OSDI)*, vol. 99, 1999, pp. 173–186.

Runze Cheng received the B.Eng. degree in electrical engineering and automation from Shanghai University of Engineering Science, Shanghai, China, in 2019, and the M.Sc. degree in electrical and electronic engineering from the University of Nottingham, Nottingham, U.K., in 2020. He is currently pursuing the Ph.D. degree with the James Watt School of Engineering, University of Glasgow, Glasgow, U.K.

His research interests include next-generation mobile networks, machine learning, blockchain system, and resource management in wireless communication.

Yao Sun (Senior Member, IEEE) received the B.S. degree in mathematical sciences, and the Ph.D. degree in communication and information system from University of Electronic Science and Technology of China (UESTC), in 2014 and 2019, respectively.

He is currently a Lecturer with the James Watt School of Engineering, the University of Glasgow, Glasgow, U.K. He has published more than 30 peer-review papers, two book chapters, and three patents. His extensive research experience in wireless communication area. His research interests include intelligent wireless networking, network slicing, blockchain system, Internet of Things, and resource management in mobile networks.

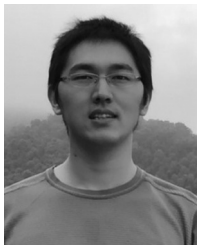
Dr. Sun has won the IEEE Communication Society of TAOS Best Paper Award in 2019 ICC. He has been a guest editor for special issues of several international journals. He has served as the TPC Chair for UCET 2021 and a TPC member for number of international conferences, including GLOBECOM 2020, WCNC 2019, and ICCT 2019.

Yijing Liu received the B.S. degree from the College of Communication and Information Engineering, Chong Qing University of Post and Telecommunications, Chongqing, China, in 2017. She is currently pursuing the Ph.D degree with the National Key Laboratory of Science and Technology on Communications, University of Electronic Science and Technology of China, Chengdu, China.

Her current research interests include next-generation mobile networks, network slicing, and distributed machine learning.

Le Xia received the B.Eng. degree in communication engineering and the M.Eng. degree in electronics and communication engineering from the University of Electronic Science and Technology of China, Chengdu, China, in 2017 and 2020, respectively. He is currently pursuing the Ph.D. degree with the James Watt School of Engineering, University of Glasgow, Glasgow, U.K.

His research interests include driverless vehicular networks, smart wireless communication, blockchain, and semantic communication.



Daquan Feng (Member, IEEE) received the Ph.D. degree in information engineering from the National Key Laboratory of Science and Technology on Communications, University of Electronic Science and Technology of China, Chengdu, China, in 2015.

He was a Research Staff with State Radio Monitoring Center, Beijing, China, and then a Postdoctoral Research Fellow with Singapore University of Technology and Design, Singapore. He was a Visiting Student with the School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, GA, USA, from 2011 to 2014. Since 2016, he has been with the College of Electronics and Information Engineering, Shenzhen University, Shenzhen, China, as an Assistant Professor and then an Associate Professor. His research interests include URLLC communications, MEC, and massive IoT networks.

Dr. Feng is an Associate Editor of IEEE COMMUNICATIONS LETTERS.

Muhammad Ali Imran (Senior Member, IEEE) received the M.Sc. (Distinction) and the Ph.D. degrees from Imperial College London, London, U.K., in 2002 and 2007, respectively.

He is a Professor of Wireless Communication Systems with research interests in self organized networks, wireless networked control systems, and the wireless sensor systems. He heads the Communications, Sensing and Imaging CSI Research Group, University of Glasgow, Glasgow, U.K. He is the Dean of the University of Glasgow, UESTC. He is an Affiliate Professor with the University of Oklahoma, Norman, OK, USA, and a Visiting Professor with the 5G Innovation Center, University of Surrey, Guildford, U.K. He has over 20 years of combined academic and industry experience with several leading roles in multimillion pounds funded projects. He has filed 15 patents; has authored/coauthored over 400 journal and conference publications; was editor of five books and author of more than 20 book chapters; and he has successfully supervised over 40 postgraduate students at Doctoral level. He has been a consultant to international projects and local companies in the area of self-organized networks. He has been interviewed by BBC, Scottish television, and many radio channels on the topic of 5G technology.

Prof. Imran is a Fellow of IET and a Senior Fellow of HEA.