doi:10.3772/j.issn.1006-6748.2021.02.003

Optimizing MDS-coded cache-enable wireless network: a blockchain-based cooperative deep reinforcement learning approach[①]

Zhang Zheng(张 正)[②] [*], Yang Ruizhe[*], Yu Fei Richard[**], Zhang Yanhua[*], Li Meng[*]
([*] Faculty of Information Technology, Beijing University of Technology, Beijing 100124, P. R. China)
([**] Department of Systems and Computer Engineering, Carleton University, Ottawa K1S5B6, Canada)

Abstract

Mobile distributed caching (MDC) as an emerging technology has drawn attentions for its ability to shorten the distance between users and data in the wireless network. However, the DC network state in the existing work is always assumed to be either static or real-time updated. To be more realistic, a periodically updated wireless network using maximum distance separable (MDS)-coded DC is studied, in each period of which the devices may arrive and leave. For the efficient optimization of the system with large scale, this work proposes a blockchain-based cooperative deep reinforcement learning (DRL) approach, which enhances the efficiency of learning by cooperating and guarantees the security in cooperation by the practical Byzantine fault tolerance (PBFT)-based blockchain mechanism. Numerical results are presented, and it illustrates that the proposed scheme can dramatically reduce the total file download delay in DC network under the guarantee of security and efficiency.

Key words: caching technology, blockchain, deep reinforcement learning (DRL)

0 Introduction

Recently, with the explosive growth of mobile devices and the proliferation of multimedia applications, the transmission of files is becoming important in mobile wireless networks. How to avoid too many occupations of expensive storage and bandwidth during the files transmission process has drawn significant interest in academia and industry[1].

To shorten the distance between users and data in the wireless networks, caching has been proposed[2-4]. The key idea is to put some of the most popular content at the network edge during the off-peak periods, which will provide users a better experience in the peak traffic. Therefore, in traditional caching scheme[5], the macro base-stations (MBSs) and small-base-stations (SBSs) are usually deployed as edge caching nodes. To further enhance the system performance, mobile devices can be utilized as a caching node, storing segments of files to facilitate others nearby[6-7]. This plays a noticeable role in device-to-device (D2D) communication, which is referred to as mobile distributed cache. In Ref. [6], by using the mobile distributed cache, users can directly communicate with the nearby users to get the wanted content instead of accessing base-station (BS). Nevertheless, the D2D connections are not always available due to the mobility of the devices.

To overcome the availability issue, a full backup of the cache can be stored at the BS[8]. Coded caching technology is another solution for improving the availability of D2D communication. In Refs[9-11], the separated file contents are coded with maximum distance separable (MDS) code. As a kind of erasure correcting codes, MDS lets users recover the complete file just from a certain number of encoded segments without getting contents from all the caching nodes. Note that the leaving of too many caching nodes may result in the unrecoverable problem of the data in mobile distributed caching (MDC).

Although some works have been done on MDC, it is usually assumed that the state of the DC network is static or real-time detection. However, in the wireless scenarios, especially for the mobile devices caching with high mobility, the real-time detection and maintenance consume lots of resources[12], or worse, the cache placement probably changes after the observa-

① Supported by the National Natural Science Foundation of China (No. 61571021, 61901011) and the Program of China Scholarship Council (No. 201806540039).
② To whom correspondence should be addressed. E-mail: zenzv5@163.com
Received on Apr. 19, 2020

tion[13]. Therefore, updating the network state every period is more rational. The system has high complexity when the mobility and the update-interval delay are jointly considered, which is too difficult to be solved optimally using traditional optimization methods.

In Ref. [14], simulation results showed that deep reinforcement learning (DRL) could effectively solve the complex problems in the wireless environment. Here, the problem is modelled as Markov decision process (MDP) and DRL[15] is exploited as an effective solution, and the contributions are summarized as follows.

(1) Considering the availability and stability as described above, for a D2D wireless network under the controls of SBSs, an MDS-coded caching model is optimized to reduce the total file download latency based on the periodically updated information. The joint consideration of the mobility and time-varying in MDC and the periodical updating is the key point to be solved.

(2) A cooperative deep reinforcement learning approach is proposed. In Refs[14,16], the performance of the traditional DRL may be limited by requirement of a large quantity of memory and computation resources as well as the sufficient interaction with environments. And therefore, to solve the optimization with high complexity, the proposed framework uses multiple agent cooperatively to enhance the efficiency of learning but also separate the resource burden.

(3) Moreover, the cooperation brings the efficiency but also the challenge due to the lack of trust relationships. Exactly, blockchain is introduced, which as a foundational technology that leads to decentralized control can be an appropriate solution to tackle this trust-absence problem. Besides, by using deep neural network loss value as checksum, the authenticity and effectiveness of the shared DRL model parameters can be also verified.

The paper is organized as follows. In Section 1, the system model is presented, where SBSs managing D2D devices employ MDS code to balance the real-time requirement and communication cost redundancy. In Section 2, the latency optimization considering the mobility and the update-interval delay is proposed, which is modelled as MDP, and therefore a DRL problem. Moreover, a novel framework of blockchain-based cooperative DRL is proposed. Simulation results are discussed in Section 3. Finally, conclusions are presented in Section 4.

1 System model

In this section, the system architecture, the moving of users and the distributed caching model are first presented.

1.1 System architecture

The system architecture is shown in Fig. 1, there is one macro base-station (MBS), U small cell base-stations (SBSs) and a maximum of $V_{\max}$ mobile devices served by each SBS. Let u, $\{1, 2, \cdots, U\}$ be the set of SBSs and use u to refer to the u-th SBS. Let v, $\{1, 2, \cdots, V\}$ be the set of mobile devices in single SBS, where the v-th device d in u-th SBS is denoted as d_{u_v}. The mobile device d_{u_v} (user) roams in and out following a Poisson random process, and requests file f at a random time. File f is from a library F, i. e., $F = \{F_1, F_2, \cdots, F_W\}$, where each file is of same size of B bits. Depending on the popularity of the content, some files are encoded and stored in a certain number of mobile devices, which can be referred to as caching nodes. The mobile devices without cache content are referred to normal nodes.

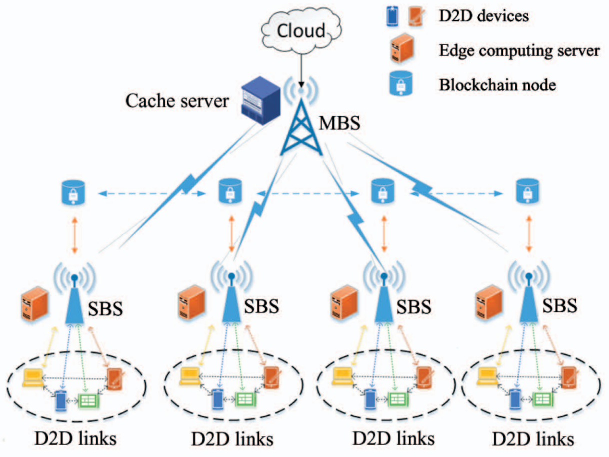

Fig. 1 System architecture

To improve the system efficiency, the D2D links are established between both of normal nodes and caching nodes under the control of the same SBS. When a user requests a file, the SBS will assign the best cache node to provide the download content based on a novel cooperative DRL approach proposed in Section 2. To implement the approach, an edge computing server (ECS) and a full blockchain node are deployed in each SBS. Thanks to ECSs, the SBS could make a proper access strategy for reducing the total file download latency by using DRL arithmetic. At the same time, with the help of blockchain nodes, ECSs can form a trustful cooperative deep learning network.

1.2 Node arrival and departure model

In general, a uniform distribution of nodes[17-18] is usually considered. In this work, among the V devices per SBS on average, there are V_c caching nodes on average. Mobile nodes can move across in the network from one SBS to another one, as shown in Fig. 2. For simplicity, it can be assumed that nodes arrive according to a Poisson process with independent identically distributed (i. i. d.). The probability density function (PDF) with random inter-arrival times T_a can be expressed as

$$f_{T_a}(t) = V\lambda e^{-V\lambda t}, \quad \lambda \geqslant 0, t \geqslant 0 \tag{1}$$

where $V\lambda$ is the expected arrival rate of user and t is measured as time unit (t. u.). Assuming that node stays in the SBS for an i. i. d. exponential random lifetime T_s with PDF:

$$f_{T_s}(t) = \mu e^{-\mu t}, \quad \mu \geqslant 0, t \geqslant 0 \tag{2}$$

where μ is the expected departure rate of the node. This system can be described as an $M\backslash M\backslash\infty$ queue model, and the probability that i nodes in each SBS $\pi_i(V)$ can be described by[19]

$$\pi_i(V) = \frac{(V\lambda/\mu)^i}{i!}e^{-(V\lambda/\mu)} \tag{3}$$

here it can be assumed that $\mu = \lambda$, which implies the average number of node in the SBS is V. Similarly, the arrival of nodes that store cache also can be described as a Poisson random process. Hence the PDF of caching nodes with random inter-arrival times T_c can be expressed as

$$f_{T_c}(t) = V_c\lambda e^{-V_c\lambda t}, \quad \lambda \geqslant 0, t \geqslant 0 \tag{4}$$

where $V_c\lambda$ is the expected arrival rate of caching node. Since the stay lifetime is described by Eq. (2) and $\mu = \lambda$, the expected number of caching nodes keeps constant (equal to V_c).

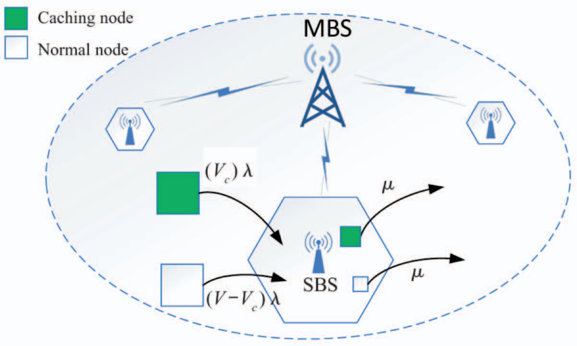

Fig. 2 Node arrival and departure model

1.3 Distributed caching mode

This work considers a MDS-coded distributed caching scenario. As described above, the requested file f is from a library F of W files and each file is the same size of B bits. Assuming that the popularity probability of files $p_\omega \in \{1, \cdots, p_W\}$ follows the time-invariant Zipf distribution, the probability that the ω-th file is requested can be defined as

$$p_\omega = \frac{1/\omega^\sigma}{\sum_{j=1}^{W} 1/j^\sigma}, \quad 1 \leqslant \omega \leqslant W \tag{5}$$

where σ represents the skewness parameter of the distribution, W is the number of files.

The W_N most popular files are stored in caching nodes by using a (n, k)-MDS code. In the MDS scheme, the file which needs caching is partitioned into k pieces, each of size of B/k bits, and from these generate $n > k$ coded packets. After coding, the packets are distributed into n caching nodes, and no two caching nodes store the same symbol. In order to reconstruct the original file, the users only need to download any k out of the n encoded chunks. At the same time, a copy of each encoded file is stored at the BS as the backup. Assuming that there is the coexist of both BS-to-device (B2D) link and D2D link, therefore, when a node requires file content, it can retrieve content from the neighbour caching node through D2D communication, or alternately download the missing content from BS.

It is considered that the D2D communication is controlled by the SBSs. The SBS records all of the devices belonging to the D2D network in a list, which is called DC library. When the user requests a file, the SBS will choose one of the caching nodes from DC library to assist the user in reconstructing the file content. In order to reduce communication redundancy, DC library should not be updated immediately. The SBS periodically updates in every $\Delta t.\ u.$[18].

Furthermore, It is assumed that a user only can downloads the coded file serially and it takes t_{d2d} t. u. to download a coded chunk in D2D link and t_{bs} t. u. in B2D link. Respectively, according to the nature of the MDS code scheme, the user has to download at least k encoded chunks. Thus the total download delay can be summarized as

$$T_{\mathrm{dw}} = It_{\mathrm{d2d}} + Jt_{\mathrm{bs}}, \quad I + J \geqslant k \tag{6}$$

where I and J represent the download time through D2D link and B2D link respectively. Note that due to the time-varying of network state, not every download behavior succeeds in getting the required cache fragments, the sum of I and J could be greater than k.

2 Problem formulation and proposed approach

In this section, this work focuses on the problem of how to choose the best caching node for users in single SBS. Afterward, this work considers a blockchain-based cooperative DRL approach as a solution.

2.1 Reinforcement learning problem formulation

There is one MBS, a amount of U SBSs and a number of mobile devices managed by SBSs. For relieving the overload of the backhaul link, some of the mobile devices can be utilized as a distributed caching node. Considering the scenario of caching, this work puts the coded-cache content on the moving mobile devices. Hence it can be assumed that the arrival and departure of nodes in SBSs follow an i. i. d. Poisson birth-death process. As the users arrive and leave, the cache state of DC-network in SBS is transformed correspondingly.

Because of the nature of (n, k)-MDS code, when the user requests the file, the SBS has to assign the most suitable caching node continually until k coded symbols are achieved. In order to optimize access policy, this work formulates the total download delay (including at least k times download) as optimization problem, not only minimize the single coded packet download time. Since mobile devices in different SBSs according to Poisson birth-death process with i. i. d, the total download delay optimization problem of the DC network in each SBS can be formulated as the same DRL process. In the following, the system state, action space and reward function are described and deduced.

2.1.1 System state

In this model, the system state S depends on roaming states of probable mobile devices, the cache states and link states of available caching nodes.

In the DRL process, the state space must be set to fixed length so that it can be applied as input to a deep neural network (DNN). However, the number of nodes which (including caching node) is not always equal to V, i. e., V is just the expected value, not an instantaneous value. As analyzed in Section 1, the number of node in DC library can be described as Eq. (3), it means the probability of having a extremely high number of caching devices in one SBS is very low (as derived by Eq. (3), for $V_c = 10$, it has $\pi_{20}(V_c) = 1.8 \times 10^{-3}$, $\pi_{30}(V_c) = 1.7 \times 10^{-7}$ and $\pi_{100}(V_c) = 4.9 \times 10^{-63}$).

The maximum limit of $V_{\max} = 3V$ node is taken as the size constraint of the system state space. Consequently, the system state can be given by

$$S = \begin{bmatrix} H_1, H_2, \cdots\cdots\cdots\cdots, H_{V_{\max}}, \\ \underbrace{C_1, C_3, \cdots, C_7, \cdots}_{v}, \underbrace{\varnothing, \cdots, \varnothing}_{m}, \\ \underbrace{I_1, I_3, \cdots, I_7, \cdots}_{v}, \underbrace{\varnothing, \cdots, \varnothing}_{m} \end{bmatrix}, v + m = V_{\max} \tag{7}$$

where, $\varnothing$ is used to represent unavailable state of node (such as when node roams out from the SBS, the corresponding state will become $\varnothing$), v is the real-time number of mobile node in a SBS, expected value $E(v) = V$, m is the number of $\varnothing$, it will change to meet the constraint requirement of $v + m = V_{\max}$.

The roaming state $H_i \in \{0, 1\}$ means the i-th mobile device whether in the SBS or not, $H_i = 1$ when the i-th device stay in the SBS, $H_i = 0$ when this device roams out from the SBS, and the stored cache is also lost. The cache state $C_i \in \{0, 1\}$, i. e., $C_i = 1$ when i-th mobile device hold requested content, $C_i = 0$ when i-th mobile device does not hold requested content. Cache contents arrive to the SBSs according to a Poisson random process in Eq. (4), i. e., same as caching nodes, expected value of the number of cache state $C_i = 1$ equals V_c. Additionally, due to performance constraints on mobile devices, the D2D caching node cannot provide content to multiple users at the same time. Thus the link state at i-th caching node I_i can be described by two states, idle and busy, i. e., $I_i \in \{0, 1\}$, $I_i = 1$ if i-th caching node is idle when the caching node can provide content to users and $I_i = 0$ in otherwise. The 2×2 link state transition probability matrix $\boldsymbol{\Lambda}_i$ of i-th mobile device can be denoted as

$$\boldsymbol{\Lambda}_i = \begin{bmatrix} 1 - \eta_i & \eta_i \\ \kappa_i & 1 - \kappa_i \end{bmatrix} \tag{8}$$

where $\eta_i = \Pr(I_i = 0 \mid I_i = 1)$, $\kappa_i = \Pr(I_i = 1 \mid I_i = 0)$.

Ideally, the DC library recorded by an SBS should be an instant update, but frequent updates can increase the wireless traffic load on the system. Then this work considers a periodically updated DC network with Δ t. u. of update interval.

The observed system state S_{ob} in the BS is not always equal to real-time state S_r,

$$S_{ob}(t) = \begin{cases} S_r(t) & t \bmod \Delta = 0 \\ S_r(t - \tau) & t \bmod \Delta \neq 0 \end{cases} \tag{9}$$

where $\tau = t \bmod \Delta$.

2.1.2 Action space

In this system, an (n, k)-MDS code is used in distributed caching. In order to reconstruct the requested file, user needs to download any k out of the n packets from caching nodes that are assigned by SBS. The SBS also decides whether to choose the BS to finish missing content or not. As shown in the system state, the number of available nodes is not constant. For making the action space feasible as an input to the neural network, this work introduces 'valid action' into action set. Then the action space can be denoted by $a \in \{1, 2, \cdots, V, \text{MBS}, \varnothing\}$, where $a = \text{MBS}$ means user should download this packet from the BS, $a = i$ means download from the i-th available mobile device by D2D link and the invalid action $a = \varnothing$ means that user will not download content from any node. At each time-step, the SBS, which is referred to as a reinforcement learning (RL) agent, has to make a selection from the action set. No matter whether the action is valid or not, time actually proceeds.

2.1.3 Reward function

Since the MDS-coded scheme is adopted in this model, this work focuses on the optimization of the total download-time of the distributed caching system.

When one node requests a file, the following three cases can be distinguished.

Case 1 The encoded packet of requested file can be found in requesting caching node, and the node is idle.

Case 2 The encoded packet of requested file can be found in requesting caching node, but the node is busy.

Case 3 The encoded packet of the requested file can not be found in requesting a caching node.

Aiming to the above cases, the reward function is crafted. The download time of the base station t_{bs} is used to compare the D2D download delay. This work denotes the download time of retrieving one coded chunk in the BS-link as t_{bs} t. u., in D2D-link as t_{d2d} t. u., where t_{bs} is much larger than t_{d2d}. For purposes of analysis, this work sets that t_{bs}/t_{d2d} as an integer N. In order to reduce total download time, the total time-steps in one episode should be minimized. Hence, this work sets the reward of each time step to

$$r(S, a) = \begin{cases} 0 & a = \mathrm{MBS} \\ -t_{d2d} & a = \varnothing \\ H_a C_a I_a t_{bs} - t_{d2d} & \text{otherwise} \end{cases} \tag{10}$$

where $H_a = 1$ if node is staying in the SBS, $C_a = 1$ if the chosen node carries a piece of the requested content, $I_a = 1$ if the chosen caching node is idle, when it meets the above requirements, the coded packet is successfully downloaded by D2D link and saves $t_{bs} - t_{d2d}$. Otherwise, $H_a C_a I_a = 0$, it means this action is 'invalid action', should penalty $-t_{d2d}$. In the latter case, the user downloads file content from the MBS alternately. The total download delay will increase by t_{bs}. The discount factor is set as $\gamma = 1$, thus the cumulative rewards coincide with the total time saved. The episode will end when k coded packets are collected, no matter the packets are collected by the D2D link or by the B2D link.

2.2 Deep AC algorithm

In the previous work[14,20,21], deep Q-learning network (DQN) algorithm is utilized for solving the wireless network problem. Since, in this work, action space is very huge (upper limit $V_{\max} = 3V$), this work uses deep actor-critic (AC) algorithm[22], a kind of state of the art DRL algorithm, which is good at dealing RL problem with the complicated policy.

In traditional actor-critic arithmetic, it involves two important functions, policy function $\pi(S, a)$ (a set of action probability outputs) and value function $\boldsymbol{V}_\pi(S)$ (estimate the value of policy in a certion state). As an extended algorithm from actor-critic, deep AC algorithm using two neural networks, actor network with parameter θ_π and critic network with parameter θ_V, instead of these two functions.

In actor network, the parameter θ_π is updated in the direction suggested by the critic network, which is called policy gradient method. The key idea is to estimate the gradient of the expected discounted cumulative reward $E[\boldsymbol{V}_\pi(S)]$ with respect to (wrt) parameter θ_π by observing the trajectories following that policy $\pi(S, a)$. The process can be described as

$$\nabla_\theta E[\boldsymbol{V}_\pi(S)] = E[\boldsymbol{V}_\pi(S) \cdot \nabla_\theta \log \pi(a \mid S)] \tag{11}$$

where $\pi(a \mid S) = P(a \mid S, \theta_\pi) = \pi(S, a)$.

In order to encourage learning agent to do more exploration, the entropy of the policy $H(\pi(S, a))$ is introduced. $H(\pi(S, a))$ works as the underlying meanings of entropy: following the policy π, if the output actions are with relatively similar probabilities, the entropy will be high. Thus, the updated processing of the policy parameters can be expressed by

$$\theta_\pi \leftarrow \theta_\pi + \xi \nabla_{\theta_\pi} \boldsymbol{V}\pi(S, a) \log \pi(a \mid S) + \varpi \nabla_{\theta_\pi} H(\pi(a \mid S)) \tag{12}$$

where ξ is the step size, ϖ is discounts factor of entropy, it should be set to a large value at the beginning of the training so as to encourage the exploration, whereas it should decrease over time to focus on the exploiting for the purpose of improving the rewards. $\boldsymbol{V}\pi(S, a)$ in Eq. (12) is provided by following critic network.

In critic network, temporal difference (TD) error method[23] based on gradient descent is adopted to improve the accuracy:

$$\nabla \boldsymbol{V}_\pi(S, a) = \sum_{t=0}^{T-1} (r_t(S_t, a) - \boldsymbol{V}_\pi(S_t, a))^2 \tag{13}$$

where $r_t(S_t, a)$ is defined in Eq. (10).

Thus the parameter θ_V for network can be derived as

$$\theta_V \leftarrow \theta_V - \psi \sum_{t=0}^{T-1} (r_t(S_t, a) + \gamma V_\pi(S_{t+1}, a) - V_\pi(S_t, a))^2 \quad (14)$$

where ψ is learning rate for training the critic neural network, γ is discount factor of reward.

2.3 Blockchain-based cooperative DRL approach

This model considers a cooperative reinforcement learning approach maintained by blockchain to derive the optimal policy for the above-formulated problem.

In DQN used in the previous work[14,21], a single agent is represented by a single neural network interacting with a single environment, which makes the samples gathered highly correlated during a run of an agent. In order to overcome this issue, a technique named experience replay can be used. Its key idea is to store the samples in memory and retrieve them in random order from a batch. However, it still brings with some problems, such as much more storage space and quite low efficiency[24]. Asynchronous gradient descent method[25] is an advanced deep learning framework proposed by Google DeepMind to take replace of experience replay. By using multiple agents in parallel on a single multi-core central processing unit (CPU) machine, the asynchronous methods for DRL not only break the correlation in training dataset but also achieve high efficiency.

In this model, the caching nodes in each SBS adapt to a Poisson birth-death process with i. i. d., and this means that all of SBSs can be considered in a similar environment. Thus this work proposes a cooperative learning framework, extended by asynchronous methods principle, where each SBS can be set as a parallel agent having its own neural network in local network, but be the cooperator for other SBSs. As shown in Fig. 3, the process of cooperation learning approach for SBSs in each training episode is as follows.

(1) Copy the newest parameter θ_{Coop} from cooperation network to local network.

(2) Perform actions in separate environments according to local network policy, compute the gradients $\Delta\theta_{\text{Local}}$ after each episode has been done.

(3) Copy the newest parameter θ'_{Coop} to local network.

(4) Update the cooperation network parameter as

$$\theta''_{\text{Coop}} = \theta'_{\text{Coop}} + \Delta\theta_{\text{Local}} \quad (15)$$

Furthermore, the trust issue of cooperative DRL is also considered. If one of SBSs in cooperative RL is attacked or out of ill intentions subjectively, it may upload the wrong parameters to cooperation network, eventually make trouble for all of the users in the DC system. Hence

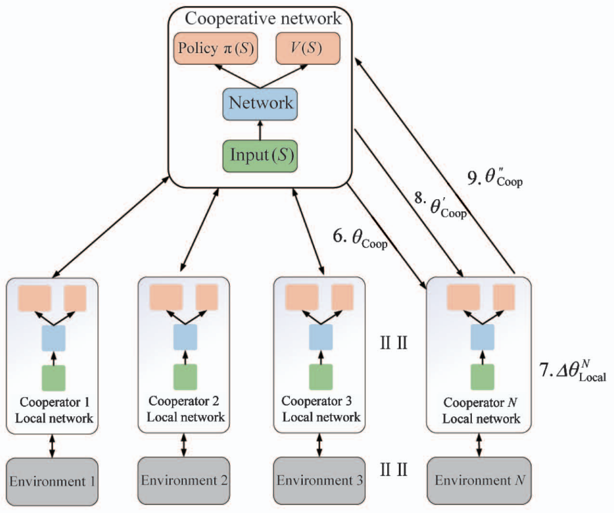

Fig. 3 The procedure of one time cooperative learning

blockchain, as a novel decentralized ledger technology, is introduced for helping build trust and confidence between different SBSs. Specifically, a permissioned BC based on practical Byzantine fault tolerance (PBFT)[26] consensus mechanism is established. According to principle of blockchain, the SBSs in cooperative not only in turn upload the parameters of network model but also its Hash, which can ensure the integrity.

For this framework, the verification of DRL model is also considered. During DRL training process Eq. (12) and Eq. (14), temporal difference (TD) error $\nabla V\pi(s, a)$ Eq. (13) is not only used in back propagation to update the network model weights but also can be seen as a metric to evaluate how well the network model is. Utilize this characteristic, TD error is also introduced as validation loss in cooperative RL, if the fraud node in cooperative DRL shares wrong parameters to other cooperators, the loss error value must be very high, which can be easy to distinguish from regular parameters.

As shown in Fig. 4, the detailed process can be described as follows.

Request ECS in the SBS trains new neural network parameters, and sends them along with corresponding loss value l_{target} in DNN to BC node in same SBS, in which BC node is referred to as primary.

Pre-Prepare Primary relays the DNN weights to other cooperative BC nodes via pre-prepare messages. Other BC nodes are referred to as replicas, replicates record the DNN weights, and apply them to calculate the validation loss value l_{vld} locally. If $l_{\text{target}} - l_{\text{vld}}$ is less than one standard error δ, confirm this update, and send prepare messages.

Prepare Replicas send prepare messages to all

BC nodes (including replicas and the primary node). Once getting $\geqslant 2f+1$ prepare messages, the SBSs update local network and get ready to commit.

Commit SBSs send commit messages to each other. Once getting $2f+1$ commit messages, the SBS starts to update these gradient in cooperative RL network.

Reply Every BC node replies its result to the ECS server.

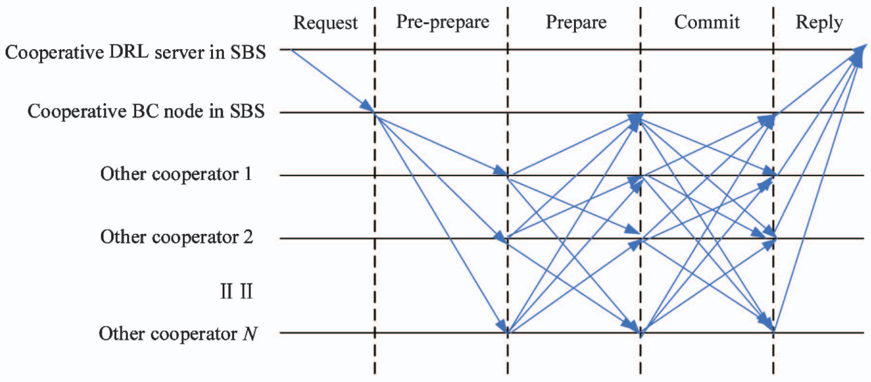

Fig. 4 The detailed procedures inside the cooperate learning block chain

Ideally, all of cooperative networks in SBS should be the same. Otherwise, the ECS should choose the majority result. The consensus process guarantees that the majority of cooperative networks are correct as long as more than two-thirds of cooperators (including primary) are honest, which is the fundamental BFT bound proved in Ref. [26].

Finally, the pseudo-code of the BC-based cooperative learning method is shown in Algorithm 1.

Algorithm 1 Blockchain-based cooperative deep reinforcement learning in single cooperator

```
repeat
  Reset gradients: $d\theta_{Local} \leftarrow 0$ and $d\theta_{vLocal} \leftarrow 0$.
  Download the newest cooperative network parameters from blockchain and $\theta'_{Local} = \theta_{Coop}$ and $\theta'_{vLocal} = \theta_{vCoop}$
  Receive target loss $l_{target}$
  Get state $s_t$
  $t_{start} = t$
  repeat
    Perform $a_t$ according to policy $\pi'_{\theta}(s_t, a_t)$
    Receive reward $r_t$ and new state $s_{t+1}$
    $t \leftarrow t+1$
    $T \leftarrow T+1$
  until $t - t_{start} == t_{\Delta}$
  $R = \begin{cases} 0 & \text{for terminal } s_t \\ V(s_t, \theta'_{vLocal}) & \text{for non-terminal } s_t \end{cases}$
  for $i \in \{t-1, \cdots, t_{start}\}$ do
    $R \leftarrow r_i + \gamma R$
    Calculate $l_{vld} = (R - V(s_i, \theta'_{vLocal}))^2$
    if $|l_{vld} - l_{target}| < \delta$ then
      send prepare messages
    end if
    if Receive $\geqslant 2f+1$ prepare messages then
      send commit messages
    end if
    if Receive $\leqslant 2f+1$ commit messages then
      retrieve network parameters $\theta'_{Local} = \theta_{Local}$ and $\theta'_{vLocal} = \theta_{vLocal}$
    else
      Accumulate gradients wrt $\theta'_{Local}$:
      $d\theta_{Local} \leftarrow d\theta_{Local} + \nabla\theta'_{Local} \log\pi_{\theta_{Local}}(s_i, a_i)(R - V(s_i, \theta'_{Local})) + \beta\nabla_{\theta'_{Local}} H(\pi(s_i, \theta'_{Local}))$
      Accumulate gradients wrt $\theta'_{vLocal}$:
      $d\theta_{vLocal} \leftarrow d\theta_{vLocal} + \frac{\partial(R - V(s_i, \theta'_{vLocal}))^2}{\partial\theta'_{vLocal}}$
    end if
  end for
  Download the newest cooperative network parameters from blockchain $\theta''_{Local} = \theta_{Coop}$ and $\theta''_{vLocal} = \theta_{vCoop}$
  Update local network parameters $\theta''_{Local}$ and $\theta''_{vLocal}$ using $d\theta_{Local}$ and $d\theta_{vLocal}$:
    $\theta''_{Local} \leftarrow \theta''_{Local} - \alpha d\theta_{Local}$
    $\theta''_{vLocal} \leftarrow \theta''_{vLocal} - \beta d\theta_{vLocal}$
  Upload the newest cooperative network parameters $\theta_{Coop}$ and $\theta_{vCoop}$ with local network parameter $\theta''_{Local}$ and $\theta''_{vLocal}$
until $T > T_{max}$
```

3 Simulation result and discussions

In this section, the results of dramatical reduction of the total file download delay in MDS-coded DC network by the proposed scheme under the guarantee of security and efficiency are exhibited. Furthermore, the effect of the update interval is discussed.

3.1 Simulation settings

In this simulation, this work uses Python 3.6.8 with TensorFlow 1.0.4[26] and Docker 19.03 with EOS 0.4 RC[27], an open-source blockchain platform on a workstation equipped with Intel Xeon E5-2690 v4 CPUs (3.5 GHz, 56 cores), three NVIDIA Tesla P100 GPUs and 128 GB RAM. By using Docker container technology, it can be run virtual multiple services on a single workstation to establish the blockchain. The scenario illustrated in Fig. 1 is used for the simulations.

The parameters values used for simulation are summarized in Table 1. In this simulation, this work assumes that the file that user requests are always available in MBS, which means that when users can not download the content through D2D communication, the needs of users request can be satisfied by using the cellular link.

Table 1 The parameters values used for simulation

Parameter	Value
Number of SBSs, U	1, 3, 7
Average number of users, V	48
Maximum number of users, V_{max}	144
User arrival rate, λ	1
User departure rate, μ	1
t_{bs}/t_{d2d}, N	10
Total training episodes in single SBS	1000
AC network update frequency	10
AC network learning rate	0.001
Discount factor of reward, γ	1
Discount factor of entropy, β	0.001

The idle state transition probability matrix $\boldsymbol{\Lambda}$ in i-th caching nodes is set as

$$\boldsymbol{\Lambda}_i = \begin{pmatrix} 0.7 & 0.3 \\ 0.3 & 0.7 \end{pmatrix} \tag{16}$$

3.2 Simulation results

Fig. 5 shows the performance of total file download time in MDS-coded DC for different schemes under the conditions that $n = 6$, $k = 2$, update interval $\Delta = 1$ t.u., the number of cooperator $U = 7$, and one of the cooperators in both two cooperative DRL is set as a malicious node. The malicious node will upload error parameters to cooperative network. The BC-based cooperative DRL against cooperative DRL without blockchain, DQN, random strategy, and that only using cellular link strategy is compared.

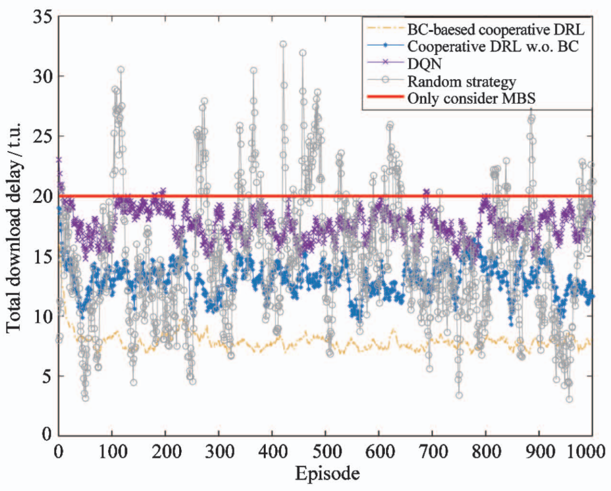

Fig. 5 Performance of different schemes in (6,2)-MDS

It can be observed that with the increase of the number of episodes for BC-based cooperative DRL, the total file download delay T_{dw} is decreased until it converges to a relatively stable value, i.e. 8 t.u., which is much lower than the file download delay without D2D communication T_{bs}, i.e., only using cellular link strategy, the ratio $T_{bs}/\bar{T}_{dw} \approx 2.5$. It can be also observed that compared with the cooperative DRL scheme without blockchain, the blockchain-based scheme has successfully protected the security of cooperative network parameters from attack. Fig. 5 also shows that the performance of the DQN scheme has much worse performance than cooperative DRL (based on actor-critic algorithm), which indicates the AC algorithm is suitable for dealing RL problem with the complicated policy. Additionally, the curve tracking for random strategy has poor performance and significant fluctuation, which indicates that the random strategy is very difficult to cope with the high complexity of this system.

Fig. 6 shows the convergence performance under different cooperator number. When the number of cooperator $U = 1$, the scheme becomes common deep AC network. And it can be seen that with the increase of the cooperator number, the convergence speed and performance of DNN are dramatically improved. This is attributed to not only parallel training but also multiple cooperators in the framework, which makes the experience available for training becomes more diverse.

Fig. 7 shows the effect of update interval in coded DC system. In this simulation, it can be assumed that each caching node only carries one encoded segment, i.e., $n = v_c$. it can be seen that compared with the scenario only using the cellular link and DC without MDS-coded, different (v_c, k) MDS-coded DC networks can all save much time for downloading file content. With the increase of ratio k/v_c, the delay ratio between the average file downloaded without D2D communication and that of the scenario using MDS coded DC is decreased. This is because when the ratio k/v_c is high, users should receive more coded-chunks to complete the download. It also can be seen from Fig. 7 that the performance of DC improves with increasing v_c, because more caching-nodes may supply more choices. It is noteworthy that the performance seems to improve significantly with the decrease of update interval. This is because small update interval will make the observed states approximate to real-time.

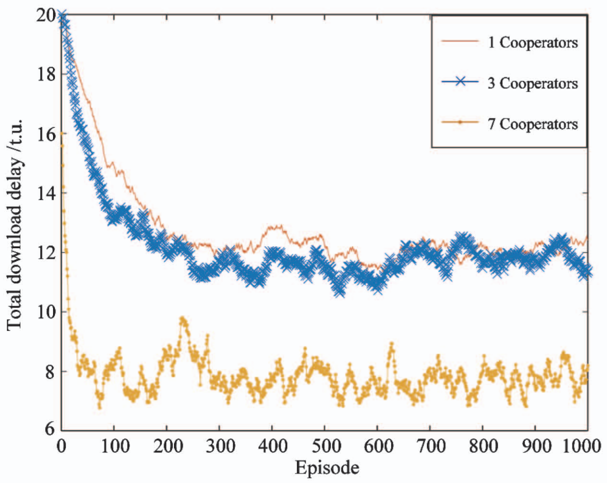

Fig. 6 Convergence performance under different cooperator numbers in single cooperator

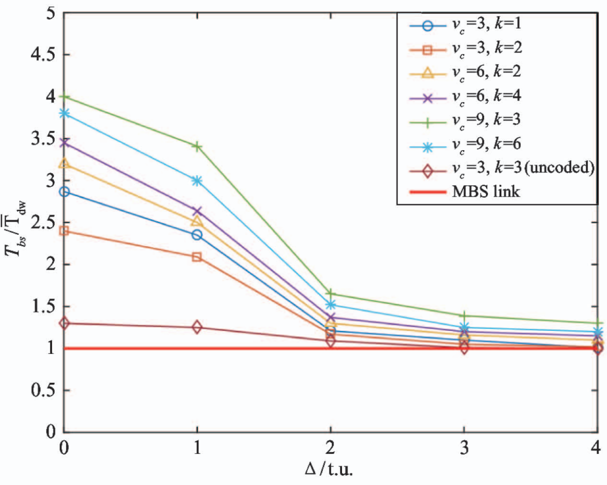

Fig. 7 Effect of update interval

4 Conclusion

In this paper, a new wireless scenario of a periodically update coded DC network is studied, where D2D arrives and leaves at any time and the mobility and time-varying of mobile device are considered. For this system with high complexity, to obtain a proper access strategy for reducing the total file download latency, the access choice can be formulated as a DRL problem. Furthermore, a cooperative learning framework is crafted to enhance the efficiency of learning, and a PBFT-based blockchain is used to protect cooperative learning security. The performance of several schemes is illustrated, which demonstrates that compared with other schemes, the proposed approach can effectively reduce the total download delay in MDS DC, even in face of the faulty node.

References

[1] Wu J, Guo S, Huang H, et al. Information and communications technologies for sustainable development goals: state-of-the-art, needs and perspectives[J]. *IEEE Communications Surveys and Tutorials*, 2018, 20(3): 2389-2406

[2] Zhang X, Zhu Q. P2P caching schemes for jointly minimizing memory cost and transmission delay over information-centric networks[C] // 2016 IEEE Global Communications Conference, Washington, USA, 2016: 1-6

[3] Liu D, Chen B, Yang C, et al. Caching at the wireless edge: design aspects, challenges, and future directions [J]. *IEEE Communications Magazine*, 2016, 54(9): 22-28

[4] Paschos G S, Iosifidis G, Tao M, et al. The role of caching in future communication systems and networks[J]. *IEEE Journal on Selected Areas in Communications*, 2018, 36(6):1111-1125

[5] Shanmugam K, Golrezaei N, Dimakis A G, et al. Femtocaching: wireless content delivery through distributed caching helpers[J]. *IEEE Transactions on Information Theory*, 2013, 59(12): 8402-8413

[6] PÄÄkkÖnen J, Hollanti C, Tirkkonen O. Device-to-device data storage for mobile cellular systems[C] // 2013 IEEE Globecom Workshops, Atlanta, USA, 2013: 671-676

[7] Amer R, Butt M M, Bennis M, et al. Delay analysis for wireless D2D caching with inter-cluster cooperation[C] // IEEE Global Communications Conference, 2018: 1-7

[8] Bioglio V, Gabry F, Land I. Optimizing MDS codes for caching at the edge[C] // 2015 IEEE Global Communications Conference, San Diego, USA, 2015: 1-6

[9] Liao J, Wong K K, Zhang Y, et al. Coding, multicast, and cooperation for cache-enabled heterogeneous small cell networks[J]. *IEEE Transactions on Wireless Communications*, 2017, 16(10): 6838-6853

[10] Zhang T, Xiao L, Yang D, et al. Optimal pricing for MDS-coded caching in wireless D2D networks[J]. *Mobile Information Systems*, 2018, 2018: 1-9

[11] Yang K, Martin S, Xing C, et al. Energy-efficient power control for device-to-device communications [J]. *IEEE Journal on Selected Areas in Communications*, 2016, 34 (12): 3208-3220

[12] Sathiamoorthy M, Dimakis A G, Krishnamachari B, et al. Distributed storage codes reduce latency in vehicular networks[J]. *IEEE Transactions on Mobile Computing*, 2014, 13(9): 2016-2027

[13] He Y, Zhang Z, Zhang Y. A big data deep reinforcement learning approach to next generation green wireless networks[C] // IEEE Global Communications Conference, Singapore, 2018: 1-6

[14] He Y, Zhao N, Yin H. Integrated networking, caching, and computing for connected vehicles: a deep reinforcement learning approach[J]. *IEEE Transactions on Vehicular Technology*, 2018, 67(1): 44-55

[15] He Y, Zhang Z, Yu F R, et al. Integrated networking, caching and computing in green wireless networks: a big data deep RL approach[J]. *IEEE Transactions on Vehicular Technology*, 2018, 67(1): 44-55

[16] Piemontese A, Amat A G I. MDS-coded distributed storage for low delay wireless content delivery[C] // International Symposium on Turbo Codes and Iterative Information Processing, Brest, France, 2016: 320-324

[17] Piemontese A, Graellamat A, Graell A, et al. MDS-coded distributed caching for low delay wireless content delivery[J]. *IEEE Transactions on Communications*, 2019, 67(2): 1600-1612

[18] Miller S L, Childers D. Probability and random processes [J]. *Technometrics*, 2010, 40(2): 160

[19] He Y, Zhang Z, Yu F R, et al. Deep-reinforcement-learning based optimization for cache-enabled opportunistic interference alignment wireless networks[J]. *IEEE Transactions on Vehicular Technology*, 2017, 66(11): 10433-10445

[20] He Y, Liang C, Zhang Z, et al. Resource allocation in software-defined and information-centric vehicular networks with mobile edge computing[C] // IEEE 86th Vehicular Technology Conference, Toronto, Canada, 2017: 1-5

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

[22] Mnih V, Kavukcuoglu K, Silver D, et al. Playing atari with deep reinforcement learning[J]. *arXiv*:1312.5602, 2013

[23] Arulkumaran K, Deisenroth M P, Brundage M, et al. Deep reinforcement learning: a brief survey[J]. *IEEE Signal Processing Magazine*, 2017, 34(6): 26-38

[24] Mnih V, Badia A P, Mirza M, et al. Asynchronous methods for deep reinforcement learning[C] // The 33rd International Conference on Machine Learning, New York, USA, 2016: 1-28

[25] Castro M, Liskov B. Practical Byzantine fault tolerance and proactive recovery[J]. *ACM Transactions on Computer Systems*, 2002, 20(4): 398-461

[26] Abadi M, Agarwal A, Barham P, et al. TensorFlow: large-scale machine learning on heterogeneous distributed systems[J]. *arXiv*: 1603.04467, 2015

[27] IO EOS. EOS. IO Technical White Paper[R]. EOS. IO, 2017

Zhang Zheng, born in 1993. He is currently working toward the Ph. D degree at Beijing University of Technology, Beijing. He received the B. S. degree in electrical and information engineering from Beijing Information Science and Technology University, Beijing, China. His research interests include blockchain, deep reinforcement learning, and radio resource management.