Blockchain-Empowered Collaborative Task Offloading for Cloud-Edge-Device Computing

Su Yao, Mu Wang, Qiang Qu, Ziyi Zhang, Yi-Feng Zhang, Ke Xu, *Senior Member, IEEE*, and Mingwei Xu, *Member, IEEE*

***Abstract*— How to enable high-performance task offloading and preserve the trust between participants is imperative yet nontrivial to the Cloud-Edge-Device (CED) computing, mainly because the resources are geo-distributed and operated by different parties. Also, the CED participants are highly dynamic and heterogeneous in resource provision and may conflict in interest. This paper proposes BlockChain-empowered CED (BC-CED), a blockchain-empowered collaborative task offloading for CED computing. In BC-CED, blockchain plays a central role in the main functionality of CED, including task offloading, brokerage of resource usage, and incentives. We distinguish the BC-CED from the existing solutions by modifying the blockchain consensus process, enabling the participants to reach an agreement via solving the task offloading problem. For this purpose, we formulate the offloading problem by considering the computation capabilities of candidate nodes and the network performance. BC-CED allows each participant to apply reinforcement learning-based methods to solve this problem and compete for the right of block output by comparing the offloading policy performance and accepting the best policy as the offloading scheme within the next period. We also propose a truthful incentive mechanism to encourage resource contributions in BC-CED and force them to be honest. Extensive tests by implementing our solutions in a commercialized blockchain platform have shown how BC-CED achieves a superior performance in task offloading and blockchain maintenance.**

***Index Terms*— Cloud-edge-device computing, task offloading, blockchain, reinforcement learning.**

Manuscript received 16 March 2022; revised 16 June 2022; accepted 30 June 2022. Date of publication 31 October 2022; date of current version 22 November 2022. This work was supported in part by the China National Funds for Distinguished Young Scientists under Grant 61825204; in part by the NSFC Project under Grant 61932016, Grant 62101301, Grant 62132011, and Grant 62132009; in part by the Beijing Outstanding Young Scientist Program under Grant BJJWZYJH01201910003011; in part by the China Computer Federation (CCF)-Huawei Populus Euphratica Forest Fund under Grant CCF-HuaweiBC2021005; in part by the Chinese Association for Artificial Intelligence (CAAI)-Huawei MindSpore Open Fund under Grant CAAIXSJLJJ-2020-014A; and in part by the China Postdoctoral Science Foundation under Grant 2021M691787. *(Corresponding author: Mu Wang.)*

Su Yao, Mu Wang, and Yi-Feng Zhang are with the Beijing National Research Center for Information Science and Technology (BNRist), Tsinghua University, Beijing 100190, China (e-mail: yaosu@tsinghua.edu.cn; yifengzhang@tsinghua.edu.cn; muwang@tsinghua.edu.cn).

Qiang Qu and Ziyi Zhang are with the Blockchain Laboratory, Huawei Cloud Tech Company, Shenzhen 518129, China (e-mail: quqiang4@huawei.com; zhangziyi@huawei.com).

Ke Xu and Mingwei Xu are with the Beijing National Research Center for Information Science and Technology (BNRist), Tsinghua University, Beijing 100190, China, and also with the Zhongguancun Laboratory, Beijing 100081, China (e-mail: xuke@tsinghua.edu.cn; xumw@tsinghua.edu.cn).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/JSAC.2022.3213358.

Digital Object Identifier 10.1109/JSAC.2022.3213358

I. Introduction

VARIOUS network applications with delay and computational intensive features, such as multimedia data processing [1], artificial intelligence (AI) [2], Internet-of-things (IoT) [3], quickly exhaust the computation and communication resources at the edge. With the recent fast development in wireless communication technologies and human-carried devices, researchers are now aware of and attracted by exploiting the resource residing in various mobile devices (MDs) [4], [5], [6]. This emerging trend conceptualizes a new computation paradigm known as Cloud-Edge-Device (CED) computation system [1], [7]. The clusters of MDs with elastic resource provision make CED substantially different from the existing centralized computation frameworks and simultaneously overcome the resource shortage, backbone inefficiency, and delay issue. Besides, with stabilized resource provision by Cloud and edge computing servers, CED can also meet task publishers' quality-of-service (QoS) demand.

Despite CED's promise, bringing this novel networked computation paradigm into reality remains problematic. CED is a networked system where computation units are geo-distributed, interconnected via the public Internet, and operated by different parties. Such a character of decentralization makes the CED's performance heavily rely on the task offloading scheme that schedules the tasks among various nodes with different computational capabilities. Several studies [9], [10], [11] attempt to formulate the CED task offloading problems as a non-convex optimization problem and prove its NP-hardness. Solving such a problem requires heuristic algorithms that are suboptimal and difficult to theoretically ensure the performance given the randomness of MD's behaviors. An emerging trend is to apply reinforcement learning (RL) [8], [12] which is tailored for the optimal controls in dynamic system [13], [14]. RL-based methods accommodate the system variation by formulating the task offloading problems as a Markov decision process (MDP) aiming to maximize the long-term cumulative utilities. By observing the interplay between decisions and environments, the RL agent iteratively optimizes the policy towards the optimal long-term utility. Since the system state variation is unpredictable, ensuring the optimality of the output decisions requires the RL method to explore the system state variation as much as possible when training the model, which is time-consuming. Besides, considering the increasing scale of the system and privacy concerns, it is also non-trivial to fully capture the knowledge of the whole

system when relying on a centralized coordinator as in current solutions.

The resource of CED comes from multiple parties which may have conflicts of interest, raising the concern of trustworthiness and data security. Recently, an emerging technology blockchain has been treated as a promising solution for this issue. Every agreement on resource usage/provision in the blockchain will be verified by all the blockchain participants, which is tractable and tamper-resistant. Such a salient feature provides great confidence in CED participants' honesty without a third party's support. Studies such as [15] and [6] attempt to leverage blockchain to build a truthful incentive mechanism when user devices are involved in task offloading. Some efforts such as [16] apply the blockchain to solve the data security problems in distributed learning systems. Several studies such as [17] and [18] use blockchain to build a federated learning structure. Although the increasingly popular blockchain in distributed computation systems, including the CED, the combined usage of the current blockchain and CED can be a challenging task. The major problem is that operating a blockchain in CED can be costly. Each blockchain node requires generating blocks and reaching the agreement via consensus algorithms. Current mechanisms such as proof-of-work (PoW) are computation intensive. For example, the well-known BitCoin using PoW consumes 204TWh of electricity every year, almost comparable to the consumption of the whole Thailand. The transactions per second are only 4.6, which is unacceptable to CED with frequent resource exchange. Thus, consensus mechanisms such as Delegate proof-of-stake (DPoS) [20], or practical byzantine fault tolerance (PBFT) [21] with features of higher throughput and energy-friendly are much more preferred. Still, these methods inevitably introduce extra delay and cost to the CED resource allocations. These concerns demand the design of blockchain-based CED that can reuse the resource consumption on maintaining the blockchain and allow a fast decision on resource allocation.

This paper tackles the above challenges by proposing the **B**lock**C**hain-empowered task offloading scheme for **CED** computing (**BC-CED**). BC-CED distinguishes itself from existing solutions by incorporating the task offloading process into blockchain's consensus mechanism, aiming to improve resource usage and leverage blockchain participants' crowd intelligence to solve the complex CED offloading problem. Specifically, blockchain participants reach an agreement by solving offloading problems rather than only applying conventional consensus mechanisms. The participant with the best results will be elected to output the blocks, including the resources trading information. For this design purpose, we formulate the task offloading problem in BC-CED as a partially observable Markov decision process (POMDP) to capture the system dynamically. Each blockchain participant applies the RL methods to solve this problem. Blockchain participants further compete for the right of block output by comparing the optimality of their output policies. A truthful incentive mechanism to ensure the honesty of computation units in BC-CED is also proposed. Contributions of this paper are multi-folded:

(1) We conceptualize the BC-CED, in which the blockchain plays a central role in maintaining the resource trading information and performing the task offloading scheme. A major difference that distinguishes BC-CED from the existing blockchain-based solution is that the blockchain participants of BC-CED reach a consensus via solving the CED task offloading problem. Each blockchain node outputs the task offloading result and selects the one that maximizes the utility within the given constraint to output blocks of the next time slot. Such design improves the resource utilization of blockchain participants and reduces temporal overhead since the blockchain operations have become necessary steps of the CED system. Also, BC-CED can benefit from the competition between different blockchain participants, given the selected scheme are always among the numerous candidates.

(2) We formulated the CED task offloading problem as a POMDP. The formulated problem not only focuses on the delay or energy consumption on task processing as in most existing solutions but also considers the dynamic of transmission condition because the bandwidth competition at bottleneck links can significantly affect the offloading performance. To solve this problem, we allow each blockchain participant uses an RL-based model trained via its observation of the system dynamic. To further ensure the trustworthy when processing the offloaded tasks, we propose a truthful incentive mechanism integrating the functionality of smart contracts of BC-CED.

(3) By cooperating with the Hangzhou Blockchain Technology Research Institute, we implement BC-CED in their self-developed blockchain platform, BROP [22], and evaluate the performance under various conditions. We first compare BC-CED with several widely used solutions DPoS and PBFT, in terms of the blockchain throughput and delay, showing BC-CED's feasibility. We also compare the task offloading performance with several RL-based methods in terms of loss convergence, total reward, overall latency, and resource usage.

The rest of the paper is organized as follows: The background and related works are reviewed in Section II. Section III presents the system overview of BC-CED. Section IV formulates the CED task offloading problem and discusses how to use RL-based methods to solve this problem. Section V introduces the design and implementation of BC-CED. Section VI evaluates our design through numerical simulations and prototype-level tests and Section VII concludes the paper.

II. Background and Related Work

In this section, we briefly introduce the related studies, including the blockchain-based distributed computation system and task offloading schemes in CED. The main attributes of these solutions are summarized in Table I.

A. Task Offloading in CED System

So far, only a few efforts have been afforded to the designated task offloading strategies in CED. Hong et al. in [9] formulate the resource scheduling problem of the CED system as a multi-task partitioning optimization problem, and an online

algorithm is then proposed to timely allocate computational resources. In [10], the task offloading problem is formulated as a mixed-integer problem and decoupled into subproblems. The first is to minimize the computation resources over a given D2D pair, and the second is to maximize the number of supported devices. Wen [11] et al. aim to improve the energy efficiency of edge computing via a D2D offloading strategy. Xing et al. [23] minimize the computation latency by using D2D connection to offload the task to nearby users and optimize the local users' task assignments. Considering that the formulated problem is mixed-integer and hard to solve, a suboptimal task offloading algorithm is proposed by relaxing the original problem to convex optimization.

The above offloading strategies commonly formulated the problem as a nonconvex optimization problem and solved it via heuristics methods which can only derive a suboptimal solution. The deterministic optimization in these studies also fails to capture the system variation. Consequently, the inconsistency between realism and modeling further impairs the offloading performance. A recent promising trend to tackle the above problems is leveraging the reinforcement learning (RL) methods. Unlike deterministic optimization, RL-based on solving a Markov decision process (MDP) approximates the long-term optimum of a stochastic system. Yi et al. in [24] apply deep reinforcement learning (DRL) to generate the resource allocation algorithm for computational-intensive jobs. In this solution, a deep Q-network is offline trained by maximizing the cumulative time reward. The trained Q-network is forward-propagated to generate the offloading policy to support online decision-making. Chen et al. [25] consider ultra-dense cell scenarios where the client can offload computational tasks to multiple base stations(BSs). The offloading problem is then formulated as an MDP, and a double deep Q-network-based method is applied to learn the optimal offloading strategy. In [26], a multi-access edge computing system with multiple cloud centers is considered. Due to each Cloud center having its interest and competing resources. A distributed task allocation scheme based on multi-agent reinforcement learning is proposed, enabling cloud centers to determine the task offloading by observing others' decisions collaboratively. In [27], Liu et al. apply the RL to minimize the average task completion time when offloading the task to MEC. A distributed algorithm based on the counterfactual MARL approach is presented by further considering the existence of multiple independent users.

B. Blockchain-Enabled Distributed Computation System

Enabling blockchain in computation systems such as Cloud, the edge has begun in recent years. The decentralized and tamper-resist features when recording data make the blockchain perfect for resource management in a distributed system with untrust participants. Numerous studies showcase the promise of blockchain to distributed computation systems from various perspectives. For instance, several studies attempt to leverage blockchain to improve reliability and security. In [28], a blockchain-based federated learning application trading platform called FLEX is given, which allows users to buy and sell computing resources for model training without sacrificing data privacy. In [17], a blockchain-based federated learning for cognitive computing is proposed. Federated learning protects privacy and provides high processing efficiency, while blockchain achieves full decentralization, providing incentives and robustness against poisoning attacks. In [18], Cheng et al. propose a resource pricing and trading scheme to optimize the edge computing resource allocation and further use blockchain to record the entire resource transaction process to protect security and privacy.

Besides the security issues, several studies attempt to facilitate resource utilization's truthfulness via blockchain. To ensure the truthfulness of incentive-based resource trading in edge computing, the study in [29] introduces the blockchain-based incentive scheme that prevents malicious edge servers from tampering with player information by maintaining a continuous tamper-proof ledger database. In [18], Cheng et al. proposes a trusted resource allocation mechanism based on blockchain-driven smart contracts, in which a group-buying pricing mechanism and a reputation evaluation mechanism are proposed to effectively address the problems existing in resources pricing and service quality evaluation of edge servers. In [18], a general framework for blockchain-based edge-computing-enabled IoT scenarios is proposed, which designs smart contracts within a private blockchain network to exploit the asynchronous advantage actor-critic (AC)-based resource allocation methods. In [30], Bai et al. considers the resources scheduling in edge computing scenarios from the perspective of stable and real-time operation, and proposes a multilateral blockchain structure that can contain thousands of edge data, improve the efficiency of on-chain data, and realize cross-chain edge data sharing heterogeneous blockchain system.

Similar to our work, studies [31] and [15] design blockchain-based task offloading for CED. In [15],Zavodovski. et al. apply the blockchain for managing the auction mechanism of edge resource usage. In [31], Wu et al. design a blockchain-based IoT-Edge-Cloud computing structure that leverages blockchain to secure the task upload. In such scenarios, the resource allocation problem is formulated as Lyapunov optimization, whose objective is to minimize energy consumption and task response time jointly. An energy-efficient dynamic task offloading algorithm is given accordingly. Although these studies also focus on Blockchain-based CED systems, our work is different in the following perspectives: (1) Different from these studies that use blockchain to secure the data, BC-CED allows the blockchain to play a central role in data recording, task offloading, and incentives; (2) We also propose a modified consensus algorithm that allows the blockchain members to reach the agreement via solving the task offloading problem, which improves the resource usage and fully exploits the crowd intelligence of blockchain members for refining the offloading performance; (3) To accommodate the complex and dynamic of the environment in CED, each blockchain member applies the RL-based method to solve this problem rather than using conventional optimization methods.

TABLE I
COMPARISON OF RELATED STUDIES

	CED Adoption	RL-enabled	Blockchain-enabled
[9]–[11], [24]	√	×	×
[25]–[28]	√	√	×
[18], [19], [29], [30]–[32], [38]	×	×	√
[31]	×	√	√
[33], [34]	√	×	√
BC-CED	√	√	√

TABLE II
NOTIFICATIONS USED IN PROBLEM FORMULATION

Notation	Definition
$\mathcal{V}, \mathcal{E}$	set of the nodes and links
$\mathcal{T}$	the set of time slot in system
$c_l\,(t)$	available bandwidth of link l at t
b_i	the volume of raw data for processing i
$x_i(t)$	data rate of delivering i
Δt	time length of the time slot
$B_l(t)$	task set using link l during t
$V_l(t)$	the length of virtual queue for l during t
$F_u(t)$	available computation resource of unit u during t
$\overline{x}_i$	the average transmission rate of i
$s_i(t)$	the computation resources required to process i
$Q_u(t)$	task backlog of u during t
$E_i(t)$	the execution delay of i
$R_u(t)$	the unit resource usage of u during t
α_i, β_i	the weight for i
V	penalty parameter
$G(t)$	the set of tasks generated during t
$U_i(t)$	the set of links used by i
$\overline{c}_l$	time average capacity of l

III. SYSTEM OVERVIEW

This section presents the framework of BC-CED, a collaborative CED task offloading scheme empowered by the blockchain. Table II lists the notifications used in the problem formulation. Fig. 1 shows the structure of the proposed BC-CED, which mainly consists of three layers: *Applications layer* includes the various network applications running on top of the BC-CED. These applications acting as BC-CED clients subscribe their resource request and offload the tasks to the computation units in BC-CED; *Blockchain layer* plays a central role in the BC-CED functionality, including task offloading decision, resource management, and incentives. Moreover, the modified consensus mechanism proof-of-optimum that enables blockchain to compete for the right of output blocks via optimizing the task offloading schemes of BC-CED is also implemented in this layer. Together with the blockchain technology, BC-CED aims to provide a truthful, decentralized and high-performance task offloading; *Resource layer* maintains the computation units contributing their resources to support the BC-CED's task offloading. We assume that all clients can access BC-CED resources without loss of generality. Computation units in CED include: (1) the data center or edge computing servers with stabilized and powerful computing capability but pricey in resource usage; (2) mobile or human carried devices with limited and random resource provision.

We define the type of members and basic concepts used in our BC-CED based on the above structure.

Subscriber: The network applications or client offloads their tasks to the BC-CED by specifying their demand on resources and QoS requirement.

Transaction: When the client issues a request of resource in BC-CED, a transaction that includes the resource and task QoS requirement will be generated task brokerage. After deciding on task offloading, the information of computation units allocated to this task will also be recorded into the corresponding transaction. After the computation unit processing the task and returning results to the client, the reward for completing the task offloading will also be written in the transaction.

Blockchain worker: Workers in Blockchain processes the transactions defined above. Similar as current blockchain, worker compete to output the block by achieving a pre-given purpose.

Optimizer: The workers elected from the blockchain members generate the task offloading policy via an RL model. Furthermore, optimizers compare the results and select the optimizer with the best-estimated performance as the leader for the next time slot. The leader outputs the block and gains block reward.

Computation unit: The entities that contribute their resources to process the offloaded tasks.

Brokerage: This entity offers an interface for the subscriber to interact with the blockchain and CUs. It is worth noting that brokerage in BC-CED only acts as a relay to forward the tasks and CUs information to the blockschain. Its main functions include resource management, task offloading, and incentives are deployed on the blockchains. Besides, the brokerage can be either a trust component provided by the CED operator or distributedly operated by multiple entities, for example, blockchain workers.

Reward: The subscriber needs to pay the resource usage fee of task offloading in BC-CED and this fee will be transferred to the entities contributing to the BC-CED. Specifically, entities with contributions in BC-CED consist of blockchain workers and computation units. The former gains the reward by making task offloading decisions and recording transactions, and the latter are rewarded for using their resources to process the offloaded tasks.

IV. TASK OFFLOADING PROBLEM IN BC-CED

In this section, we discuss the problem of task offloading in BC-CED. We first present the system model, including the computation and communication model. We formulate the task offloading problem as a partially observable Markov decision process (POMDP) and analyze how to solve this problem.

A. System Modeling

1) Communication Model: Before introducing the communication model, we make the following assumptions for BC-CED: (1) Subscriber access the computation resources of Cloud via the backbone and wireless links; (2) Edge computing servers directly connect the base station, and its

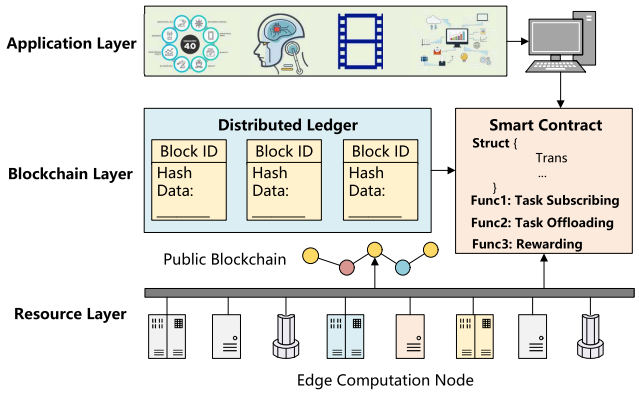


Fig. 1. The framework of blockchain-empowered CED system.

resources can be accessed via only the wireless link; (3) To access the computation resources from mobile devices within the same cell, clients use the base stations as the relay; (4) System time is slotted and denoted by $\mathcal{T} = \{1, 2, \ldots, T\}$. The system remains static within each slot and varies between different slots.

CED topology can be modeled as an undirected graph $G(\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ and $\mathcal{E}$ are the set of nodes and links, respectively. Let $V_u, V_c, V_e, V_m, V_r \subset \mathcal{V}$ denote the universe of subscriber, Clouds, edge servers, MDs and routers, respectively. Let $l(u, v) \in \mathcal{E}$ denotes the network link between node $u, v \in \mathcal{V}$. To ease the description, we simplify $l(u, v)$ to l. For $l \in \mathcal{E}$, $c_l(t)$ denotes its available bandwidth at time slot t. Consider the dynamic of network condition, $c_l(t)$ is defined as a random variable of t within the interval $[0, c_l^{\max}]$, $c_l^{\max}$ is the capacity of l.

Let b_i and $x_i(t)$ denote the volume of the raw data that needs to be delivered when offloading task i and the data rate of delivering task i within t, respectively. We accordingly have $b_i = \sum_{t=t_0}^{t'} x_i(t)\Delta t$, where t_0 and t' denote the begin and complete time slot of the data transmission. Δt is the time length of slot t. Alternatively, let $\overline{x}_i$ denotes the average transmission rate, then we have the $b_i = (t' - t_0)\Delta t \overline{x}_i$. Hence, the transmission delay $(t' - t_0)\Delta t$ equals to the $b_i/\overline{x}_i$. Namely, a higher average transmission rate reduces the raw data downloading time. The overall data rate over the l should be within bandwidth constraint during t,

$$\sum_{i \in B_l(t)} x_i(t) \leq c_l(t) \tag{1}$$

where $B_l(t)$ denotes the set of tasks using link l. The above constraints can be equally described by a virtual queue model with length $V_l(t)$ shown in Fig 2(a). This virtual queue indicates the backlog data awaiting to be transmitted at t. The arrival rate of the queue is the overall data rate of tasks using l, and dequeue rate is the available bandwidth $c_l(t)$. The dynamic of $V_l(t)$ is

$$V_l(t+1) = V_l(t) + \sum_{i \in B_l(t)} x_i(t) - C_l(t) \tag{2}$$

To avoid the congestion at l, following inequality should hold:

$$\frac{1}{T}\sum_{t=1}^{T}\sum_{i \in B_l(t)} x_i(t) \leq \frac{1}{T}\sum_{t=1}^{T} C_l(t)$$

namely, the $V_l(t)$ is stable.

2) Computation Model: Recalls that $V_k, k = c, e, m$ are the CUs in BC-CED, their available computation resources is denoted by a time-dependent variable $F_u(t)$. We measure the computation capacity by its CPU cycles per second. We assume the computation capacity of Cloud and edge computing servers is stable since these commercialized platforms are dedicated to task offloading. Namely, $F_u(t), u \in V_k, k = c, e$ is a constant value. For MDs, however, the resources provided can be time-varying since they can only contribute their idle resources, which is dynamic. Hence, the $F_u(t), u \in V_m$ is a random variable of t. Normally, we have

$$F_u(t) \gg F_v(t) \gg F_w(t), u \in V_c, v \in V_e, w \in V_m$$

This inequality holds because Cloud computing runs on the powerful data centers with the largest resources compared with the edge and devices. The edge computing servers are resource-limited but still more capable than MDs when performing offloading tasks. For each task i generated at time t, we define a two tuple $\{s_i(t), b_i\}$, where $s_i(t)$ denotes the computational resources required by processing i and we recall b_i is the size of raw data that needs to be delivered the CUs. Assuming computation unit begins to execute i when received all the raw data of i, the task processing delay of i allocated to u can be the summation of delivery latency and execution time. We denote the backlog of tasks waiting to be processed at u as $Q_u(t)$. As Fig 2(a) shows, the queue input is the total computation resource demand of tasks arrived during t, and the dequeue rate equals $F_u(t)$. Hence, the dynamic of $Q_u(t)$ can be described as:

$$Q_u(t+1) = Q_u(t) + \sum_{i \in A_u(t)} s_i(t) - F_u(t) \tag{3}$$

where $A_u(t)$ denotes the set of tasks assigned to u. By the Little's law, we can estimate the total processing time of a task by

$$P_{i,u}(t) = \frac{(Q_u(t) + s_i(t))}{F_u(t)} \tag{4}$$

B. Problem Formulation

The main purpose of task offloading is to minimize the overall delay of task executions and resource usage overhead. For any of the task i, we define the execution delay as the summation of processing time and transmission time:

$$E_i(t) = P_{i,u}(t) + \frac{b_i}{\overline{x}_i} \tag{5}$$

CUs in CED are normally operated by the third party and gain income from processing tasks. For any of the computation unit u, the rate charged for unit resource usage is R_u and the incomes for processing task i at u can be given by

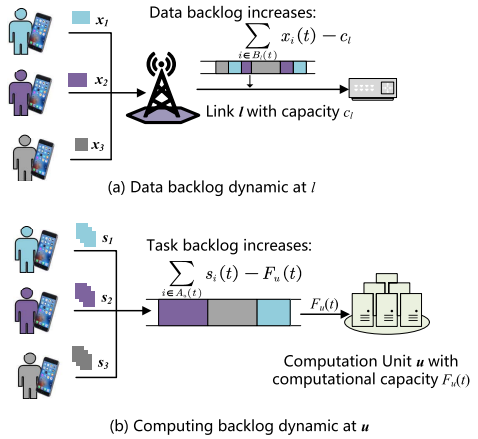


Fig. 2. Queuing models for computation and transmission.

$R_u s_i(t)$. By combining the execution delay and resource usage, we define the cost function of task execution as follows:

$$\alpha_i E_i(t) + \beta_i R_u s_i(t) \tag{6}$$

where the $\alpha_i \in [0,1]$ and $\beta_i \in [0,1]$ are the weight. Give the dimension of the first term execution delay is seconds, we remove the first term dimension by letting the dimension of α_i as $1/seconds$.

Accordingly, we formulate the task offloading problem:

$$\min \sum_{t=1}^{T} \sum_{i \in G(t)} (\alpha_i E_i(t) + \beta_i R_u s_i(t)) \tag{7}$$

$$s.t\ \mathbf{C1}: V_l(t)\ \text{is stable}, \quad \forall l \in \mathcal{E} \tag{8}$$

where the objective of (7) is to minimize the cost function of all tasks generated during the system running, where $G(t)$ denotes the set of the tasks generated at t. The constraint **C1** indicates the decision of task offloading should avoid network congestion. Moreover, an online method for this problem can be derived via solving the following unconstrained optimization problem [36] at each t:

$$\min\ V \sum_{i \in G(t)} (\alpha E_i(t) + \beta R_u s_i(t)) + \sum_{l \in \mathcal{E}} V_l(t) \Big(\sum_{i \in B_l(t)} x_i(t) - c_l(t) \Big) \tag{9}$$

where V is a nonnegative parameter and denotes the penalty to first term. Several features complicate this problem: First, the decisions $x_i(t)$ of computation units using the same link are coupled by the second terms. In such case, optimizing $x_i(t)$ towards the direction of minimizing the overall delay are difficult since it requires to consider not only i's cost function but also the decisions made by other CUs within the same link; Second, due to the complex network status, having full views of the system dynamic when solving this problem is impossible in most cases. For instance, the underlying network of BC-CED may be operated by different network operators. The BC-CED participants cannot directly observe link conditions required in (9). Namely, knowing $V_l(t)$ of each link l is difficult for the coordinator. Instead, we can only estimate the end-to-end delay $D_i(t)$ between any subscriber-CU pair whose approximation.

$$D_i(t) \approx \sum\nolimits_{l \in U_i(t)} \frac{V_l(t)}{c_l(t)} \tag{10}$$

where $U_i(t)$ denotes the set of links used by i.

C. RL-Based Method for Task Offloading Problem

The formulated problem is a time-varying optimization, and its dynamic is unpredictable, which makes the conventional deterministic optimization, such as convex optimization, unsuitable for such cases. Besides, because the system status is unpredictable, it is not easy to model the system variation. Namely, model-based dynamic programming methods are also inapplicable to these problems. RL methods solve the stochastic optimization via interacting with the environment and make decisions by observing the feedback. Unlike conventional optimization methods, RL methods approach the long-term optimum by adapting the decision-making policy according to the system variations. Such explorations make the RL methods efficiently search for optimal solutions within a stochastic environment. Various RL-based methods can be applied to solve this problem. In the following, we discuss how to use an actor-critic (AC) network to solve this method. We first rephrase the problem to a Markov decision process(MDP). The corresponding MDP to **P1** can be defined as follows

*Definition 1: The MDP for **P1** can be denoted by a tuple, $\mathcal{M} = \{\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, b_0\}$, where*

1) $\mathcal{S}$ the state space. We define the $\mathcal{S}$ by a tuple $\mathcal{S} = \{\mathcal{Q}, \mathcal{V}, \mathcal{C}, \mathcal{N}\}$, where $\mathcal{Q}, \mathcal{V}, \mathcal{C}, \mathcal{N}$ are the value set of task backlog at all computing nodes, value set of the virtual queue length of all links in $\mathcal{E}$, the set of available bandwidth of all links in $\mathcal{E}$, the value set of the $|V_k|, k = c, e, m$.
2) $\mathcal{A}$ the action space. BC-CED coordinators need to decide which CU to offload tasks and the data rate of tasks. The action at t can be denoted by a 2-tuple $(\mathbf{w}(t), \mathbf{x}(t))$. $\mathbf{w}(t)$ consists of one-hot vectors with $|V_c \cup V_e \cup V_m|$ dimension and each vector $w_j(t)$ is corresponding to a task i running upon the CU j. The components $a_{ij}(t)$ of the one-hot vector $w_j(t)$ is defined as:
$$a_{ij}(t) = \begin{cases} = 1, & \text{offload } i \text{ to } j \text{ at } t \\ = 0, & \text{offload} \end{cases} \tag{11}$$
$\mathbf{x}(t)$ is a $|G(t)|$ dimension vector defined as $\mathbf{x}(t) = \{x_i(t)\}_{i \in G(t)}$.
3) $\mathcal{R}$ the instant reward. According to the (9), we have
$$r(\mathbf{w}(t), \mathbf{x}(t)) = V \sum_{t=1}^{T} \sum_{i \in G(t)} (\alpha E_i(t) + \beta R_u s_i(t)) + \sum_{l \in \mathcal{E}} V_l(t) \Big(\sum_{i \in B_l(t)} x_i(t) - c_l(t) \Big) \tag{12}$$
Instead of directly observing the $V_l(t)$ in $r(\mathbf{w}(t), \mathbf{x}(t))$ which can be difficult as we discussed in IV.B, we approximate the instant reward by the following:

Let

$$g(\mathbf{w}_i(t), x_i(t)) = f(\mathbf{w}_i(t), x_i(t)) + \sum_{l \in U_i(t)} V_l(t) \Delta x_i(t) \tag{13}$$

where $\Delta x_i(t) = x_i(t) - c_l(t)$, $f(\mathbf{w}_i(t), x_i(t)) = V \sum_{t=1}^{T} (\alpha E_i(t) + \beta R_u s_i(t))$. We recall that the end-to-end delay $D_i(t)$ approximates the $\sum_{l \in U_i(t)} (V_l(t)/c_l(t))$ by (10). Dividing the $g(\mathbf{w}_i(t), \mathbf{x}_i(t))$ by V:

$$\frac{g(\mathbf{w}_i(t), x_i(t))}{V} = f(\mathbf{w}_i(t), x_i(t)) + \sum_{l \in U_i(t)} \frac{V_l(t)}{V} \Delta(x_i(t))$$

For each i, we rephrase the penalty V to a vector $\mathbf{V}_i := \{c_l\}_{l \in U_i(t)}$. We replace the $g(\mathbf{w}_i(t), x_i(t))/V$ by

$$\begin{aligned} r'(\mathbf{w}_i(t), x_i(t)) &= f(\mathbf{w}_i(t), x_i(t)) + \sum_{l \in U_i(t)} \frac{V_l(t)}{c_l(t)} \Delta x_i(t) \\ &\approx f(\mathbf{w}_i(t), x_i(t)) + D_i(t) \Delta x_i(t) \end{aligned} \tag{14}$$

Since the end-to-end delay can be observed via lots of network measurement methods, the above approximated $r'(\mathbf{w}_i(t), x_i(t))$ can be easily derived.

4) $\mathcal{P}$ the probability of observations $Pr(b|s, (\mathbf{w}_j(t), \mathbf{x}_j(t)))$. Instead of having full views of the system dynamic, each CED agent can only observe the dynamic $Q_u(t)$ at Cloud, edge. It is impossible to observe all MDs' $Q_u(t)$ due to its large number and privacy issue. Instead, we assume that the agent can only have the average $Q_u(t)$ of all MDs within the cell. Since the operators maintain the network, knowing $V_l(t)$ of each l is difficult for the agent. Instead, the agent can only observe the end-to-end delay.

We can solve such a problem via various RL-based methods. An example is applying the widely used Actor-Critic (AC) based method.[1]

The AC is a type of RL model that includes two neuro networks: an actor and a critic. The actor is a policy network that outputs the actions at each time t based on the current observed system states, and the critic learns a critic function based on the actions derived by actors and rewards.

1) Actor Network: We define the output policy by $\pi_\theta(\mathbf{w}(t), x(t)|S_t)$, where θ denotes the parameters of the actor network, S_t denotes the state at t. The network iteratively updates the parameter θ by the gradient-based method:

$$\begin{aligned} \theta(t+1) = \theta(t) + \nabla_\theta (\log \pi_\theta(S_t|\mathbf{w}(t), x(t))) \\ \times A^{\pi_\theta}(S_t, \mathbf{w}(t), x(t)) \end{aligned} \tag{15}$$

where $A^{\pi_\theta}(S_t, \mathbf{w}(t), x(t))$ is the advantage function

$$A^{\pi_\theta}(S_t, \mathbf{w}(t), x(t)) = Q^{\pi_\theta}(S_t, \mathbf{w}(t), x(t)) - V^{\pi_\theta}(S_t)$$

given the $Q^{\pi_\theta}(S_t, \mathbf{w}(t), x(t))$ the action-value function and $V^{\pi_\theta}(S_t)$ defined in [34], respectively.

[1]It is worth mentioning that blockchain members in BC-CED can arbitrarily apply any RL-based method to achieve the best performance on task offloading.

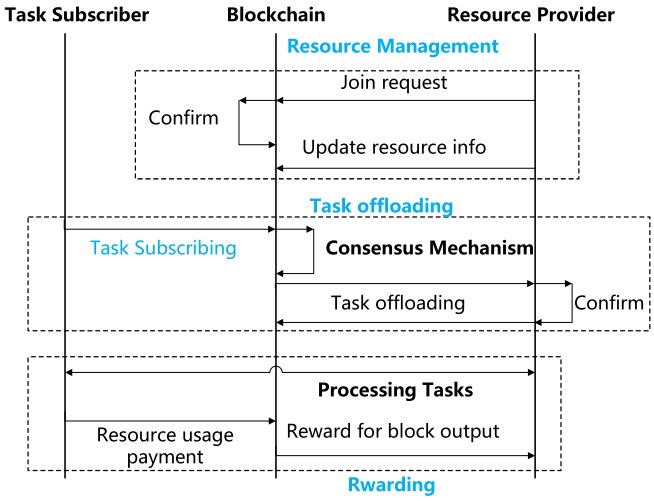


Fig. 3. Workflow of task processing in BC-CED.

2) Critic Network: The critic network uses the TD(λ) method to approximate the advantage function $A^{\pi_{\theta_c(t)}}(S_t, \mathbf{w}(t), x(t))$ in (15), whose network parameter $\theta_{c,t}$ are updated as following:

$$\phi_t = r'(\mathbf{w}_i(t), x_i(t)) + \gamma V^{\pi_\theta}(S_{t+1}) - V^{\pi_\theta}(S_t) \tag{16}$$

$$E_t = \gamma \lambda E_{t-1} + \sigma(s_t) \tag{17}$$

$$\theta_c(t) = \theta_c(t-1) + \alpha \phi_t E_t \tag{18}$$

where γ and α are the learning rate and $\sigma(s_t)$ is a indicator function $\sigma(s_t) = 1$ if $s = s_t$ and 0 otherwise.

According to the above design, at each t, AC optimized the offloading decision by the perform the following iteration until reaching the iteration criterion:

1) The actor network outputs the task offloading action $(\mathbf{w}_i(t), x_i(t))$ according to the observed state S_t and $A^{\pi_{\theta_c(t)}}(S_t, \mathbf{w}(t), x(t))$.
2) The critic derives the reward $r'(\mathbf{w}_i(t), x_i(t))$ by the action $(\mathbf{w}_i(t), x_i(t))$ and estimates the $A^{\pi_{\theta_c(t+1)}}(.)$ via TD(λ) method.

V. Blockchain Design in BC-CED

In this section, we present the details of BC-CED. Fig.3 shows the main procedures in BC-CED, including resource management, task offloading and resource pricing/rewarding. Before introducing these procedures, we first describe the smart contracts deployed upon the blockchain for performing these functions in a self-governed and distributed way.

A. Smart Contracts in BC-CED

In BC-CED, blockchain plays a key role in CED task offloading. Specifically, the blockchain of BC-CED can perform the following functions: 1) Maintain the information about resource providers in BC-CED; 2)Verify and record the task offloading information; 3) Manage the resource allocations in BC-CED; 3) Distribute reward/penalty according to

the task completion status. To fulfill these functions in BC-CED, we design the following smart contracts for BC-CED:

1) **CU registration contract**, this contract creates the "public key" for the CU, which is the unique global identification of CU and can be used to encrypt messages sent to the corresponding MD for secure communication. Once CU invokes this smart contract, it also periodically updates the CU's resource status to the blockchain.
2) **Task offloading contract**, when the client submits its task offloading request to the brokerage of BC-CED, this contract will be invoked to create a transaction that records the task info, including resource and QoS requirements, etc. Specifically, the task info is specified by a 6-tuple $S = \{TID, resReq., duration, reward(TID), Quality(TID)\}$, where TID is the unique ID of the task. $resReq$ specifies the detail of resource requirement such as CPU frequency and storage space, $currency(TID)$ is the reward for accomplishing the tasks, $serQuality(TID)$ specifies the QoS requirement such as delay constraint of task processing. $confirm(TID)$ is used to indicate whether the task is accomplished.
3) **Task offloading contract**, during every t, the BC-CED brokerage allocates the tasks to the CUs according to the task offloading scheme generated by the consensus mechanism. The information about CUs allocated to each task i will be written into i's transaction.
4) **Rewarding contract**, once the CU accomplishes the task or reaches the resource provision deadline, the rewarding contract will decide whether to distribute the reward to the CU according to the BC-CED incentive mechanism.

B. Resource Management in BC-CED

The computation resources of CUs in BC-CED can be geo-distributed and random in provision. Thus, managing the resource that can timely update the resource status of CUs is crucial for the performance of task offloading. During t, when CU j sends the join request to brokerage, the CU registration contract is invoked to generate the ID and record the resource information such as CPU frequency, memory size. The blockchain maintains the information and records in the blocks generated within t to ensure trustworthy resource provision. Once the CU joins the BC-CED, it updates the available resource at each t to ensure that the task offloading scheme can always make decisions based on the latest CED resource status. Thus, any optimizer in BC-CED can access the resource state via accessing the resource CUs's info in the block. When a CU needs to quit the CED system, a quit notification will be sent to the BC-CED brokerage. The CU registration contract will add new information to blocks that set the available resource of CU to zero.

C. Task Offloading Procedure in BC-CED

We propose a task offloading optimization scheme based on the designed smart contracts. We partition the task offloading in BC-CED at every time slot t into three stages, and the pseudo-code of the task offloading is shown in **Algorithm 1**.

1) Observation Stage: optimizers in blockchain observe tasks in arrival queue and resource requirement and raw data volume of each task i. Meanwhile, optimizers access the system states, including the task backlog $Q_u(t)$, computation capability $F_u(t)$ of each CU, and end-to-end delay of each CU-subscriber pair. In practice, CU-subscriber pairs can be large in scale and it is time-consuming to access all the end-to-end delay pairs. Instead, we only ask each CU or subscriber to update their delays to the edge periodically. For example, for a client i, the delay to the nearest based station is $d_i^s(t)$, and CUs j delay to the nearest base station within the same operator $d_j^e(t)$. Then, we use the $d_i^s(t) + d_j^e(t)$ to approximate the $D_{ij}(t)$ since compared with the end-to-end delay, the backhaul latency between two stations is relatively small and stable, which can be omitted.

2) Decision Stage: According to the above state information, optimizers invoke the RL model to output the task offloading actions and reach an agreement on the best actions. The detail of this agreement is accomplished by our proposed consensus mechanism, proof-of-optimality (POO), which will be detailedly described in the next subsection. The output action $\mathbf{w}_i(t), x_i(t)$ will then forward to the brokerage.

3) Offloading Stage: The brokerage allocates the tasks to the CUs according to the $w_i(t)$ and sends $x_i(t)$ to the subscriber. Each subscriber uploads the task's raw data with data rate $x_i(t)$. Once the CUs receive the raw data, they will begin to process the task and return the results once the task is finished.

D. Consensus Mechanism in BC-CED

A major difference between the BC-CED and the existing blockchain-driven computation platform is that BC-CED incorporates the task offloading optimization into the consensus mechanism, which attempts to reuse the computation and communication resources possessed by the blockchain while supporting the high-performance task offloading. To achieve this goal, we propose a consensus mechanism, proof-of-optimality (POO), whose pseudo-code is shown in **Algorithm 2**.

The procedure of POO consists following steps:

Optimizer election Similar to the conventional DPoS, at time slot t, blockchain workers in POO use account balance to elect a set of optimizers $\{o_b\}_{b \in w(t)}$ from the worker set. For the case of deploying BC-CED on the public chain, we only include a small number of workers with a high stake in the optimization process since they can be trusted in behaviors and are more willing to contribute their resources to the system. For private or consortium chain whose workers has no trust issue, all worker can join the task offloading decision. We only need to set a deadline for result submission, ensuring the latency of the decision-making.

Task offloading optimization. Each optimizer o input the $Q_u(t)$, $F_u(t)$, $D_{ij}(t)$ from **Algorithm 1**, and use policy $\pi_\theta(\mathbf{w}(t), x(t)|S_t)$ derived the RL-based method such as AC we discussed to output the offloading action

Algorithm 1 Task Offloading in BC-CED

Input: Task set $G(t)$, Resource requirement $\{s_i(t)\}_{i\in G(t)}$, raw data size $\{b_i\}_{i\in G(t)}$, α,β ;
Output: Task results
Observation Stage:;
foreach *Optimizer* n **do**
  Obtain the task backlog length $Q_u(t)$, computation capacity $F_u(t)$, edge delay $d_j^e(t)$ of CUs;
  Obtain the edge $d_i^s(t)$ of Subscribers;
  ForEachsubscriber-CU pair (i,j) $D_{ij}(t) \leftarrow d_j^e(t) + d_i^s(t)$;
end
Decision Stage:;
foreach *Optimizer* n **do**
  Input the $Q_u(t)$, $F_u(t)$, $D_{ij}(t)$ into the RL model;
  Output the $\mathbf{w}_i(t), x_i(t)$ of i;
end
Derive the offloading action $\mathbf{w}_i^*(t), x_i^*(t)$ via consensus algorithm;
Forward the $\{\mathbf{w}_i^*(t)\}_{i\in G(t)}, \{x_i^*(t)\}_{i\in G(t)}$ to the brokage;
Offloading Stage:;
foreach i *in* $G(t)$ **do**
  The associate subscriber deliver the raw data with $x_i(t)$ to the CU j allocated to i;
  The CU of i process the tasks;
  Return the task results;
end
final ;

$\{\mathbf{w}_i(t)\}_{i\in G(t)}, \{x_i(t)\}_{i\in G(t)}$. It is also worth noting that considering the large scale of the system and stringent constraints on the offload action submission deadline, an optimizer with limited capability may unable to transverse the status of the whole system, and can only use partial state information to optimize the offloading. An optimizer with powerful computation and communication potentially provides a better task offloading scheme since it can explore the environment information as much as possible.

Leader selection. For each optimizer o, once o derives $\{\mathbf{w}_i(t)\}_{i\in G(t)}, \{x_i(t)\}_{i\in G(t)}$ within the given deadline, o broadcasts the $\{\mathbf{w}_i(t)\}_{i\in G(t)}, \{x_i(t)\}_{i\in G(t)}$ to all the optimizers. If the $\{\mathbf{w}_i(t)\}_{i\in G(t)}, \{x_i(t)\}_{i\in G(t)}$ is not derives with the deadline, the result will be discarded for the fairness on competition. For the results $\{\mathbf{w}_i^p(t)\}_{i\in G(t)}, \{x_i^p(t)\}_{i\in G(t)}$ from other optimizer p, o first check the $\{\mathbf{w}_i^p(t)\}_{i\in G(t)}, \{x_i^p(t)\}_{i\in G(t)}$ satisfying the constraints $w_i(t) < |G(t)|$ and $x_i(t) < x_{\max}(t)$. If satisfies, add the $\{\mathbf{w}_i^p(t)\}_{i\in G(t)}, \{x_i^p(t)\}_{i\in G(t)}$ to the feasible list, otherwise, discarded the $\{\mathbf{w}_i^p(t)\}_{i\in G(t)}, \{x_i^p(t)\}_{i\in G(t)}$. o ranks the $\{\mathbf{w}_i^p(t)\}_{i\in G(t)}, \{x_i^p(t)\}_{i\in G(t)}$ in feasible list by the reward function and vote the p in the top rank as the leader. The optimizer with the best performance will be selected as the leader within t.

Block building and agreement. After the optimizer p is selected as the leader, p builds the block containing the transactions generated within t. Each block contains M transaction items, and once it reaches the size limit, a new block containing the subsequent transactions will be generated. The block $block_i$ will be broadcast to the other optimizers who will verify the transactions in the block. If most optimizers vote for admitting the $block_i$, then $block_i$ will be admitted to the blockchain. Otherwise, reject. Specifically, block $block_i$ is accepted in BC-CED if more than 2/3 of the optimizers agree. Once the block $block_i$ is accepted, a reward s will be distributed to the optimizers. Leader p takes a half share of the reward, and the rest are uniformly distributed to the other optimizers.

Fault tolerance analysis. In BC-CED, POO mainly consists of two stages: First, similar to the proof-of-stake (PoS), POO selects some workers with the highest stake to join as the optimizer for task offloading. Thus, to entirely comprise the system in this stage, malicious at least has 51% stake in the system. In the second stage, all selected workers compete for outputting blocks using the RL-based methods to solve the task offloading problem. RL-based methods are machine learning and require computation and storage resources to train the network. Thus, learning performance partially relies on the hardware conditions since large computation resources train a much more complex network with more iterations that approaches the optimum. Ideally, assuming that all the workers are equal in computation resources, the number of malicious nodes must be at least 51% for comprising the system. Besides, the other workers will verify the results and discard them if unsatisfied with the optimal condition. Thus, compared with the existing consensus mechanism, POO can perform at least equal or higher fault tolerance.

Time complexity analysis. To derive the offloading policy in our proposed BC-CED, each blockchain worker first solves the task offloading optimization problem via an RL-based method and then compares the results with each other to elect the next time slot leader. In the first step, all the workers parallelly invoke the RL to solve the problem, and thus the time complexity of this step is upper bounded by the RL method with the highest time complexity. Let the time complexity of worker n's RL algorithm is σ_n, then the time complexity of this stage is $\max_n \sigma_n$. In the second stage, each worker needs to traverse the results of other workers and selects the best. Thus, the time complexity of this step relies on the number of the workers joining the optimization process. Let the number of workers selected to optimize the task offloading is N_o, the total time complexity is $\max_n \sigma_n + N_o$.

E. Incentive Mechanism

Given the BC-CED utilizes the resources from multiple third parities, an incentive scheme can be the driven force to encourage the participants to honestly and actively join the task offloading. By the incentive mechanism, the reward or penalties to the participants of task offloading are measured in their contributions. In BC-CED, the incentive mechanism mainly consists of two components: (1) Reward, to set the prize of using resources of BC-CED. (2) Penalty, penalize the

Algorithm 2 POO Concensus

Optimizer selection:;
foreach *Worker* w *in blockchain* **do**
  Vote for a set of worker as the optimizers;
  Caculate the votes for w;
end
Rank workers by its votes and select the top M workers as the optimizers;
Leader selection:;
foreach *Optimizer* n **do**
  generate the action by policy $\pi_\theta(\mathbf{w}(t), x(t)|Q_u(t), F_u(t), D_{ij}(t))$;
  foreach *Received offloading action* **do**
    if *timestame of* $(\{\mathbf{w}_i(t)\}_{i\in G(t)}, \{x_i(t)\}_{i\in G(t)}$ *from* p*) is within deadline* **then**
      Check the feasiblity;
    end
    if $w_i(t) < |G(t)|$, $x_i(t) < x_{\max}(t)$ **then**
      Add $\{\mathbf{w}_i^p(t)\}_{i\in G(t)}, \{x_i^p(t)\}_{i\in G(t)}$ to feasible list;
    end
    Rank feasible list items by the value of $r'(\mathbf{w}_i(t), x_i(t))$ in a descent order;
    Select p associate with the $r'(\mathbf{w}_i^p(t), x_i^p(t)) > r'(\mathbf{w}_i^q(t), x_i^q(t)), \forall q$ as the leader;
  end
end
Block building and agreement:;
foreach $block_i$ **do**
  Calculate the votes for admitting i;
  if *More than 2/3 optimizers admit* $blocki$ **then**
    Accept the $blocki$;
    Transfer $0.5s$ reward to the leader;
    Transfer $0.5s/(M-1)$ reward to each optimizer;
  end
  else
    Discard $block_i$;
    Replace $block_i$ by $block_{i+1}$
  end
end
final ;

CUs failure to complete the results or provide false results will be punished.

1) Reward: Recall that $w_i^*(t), x_i^*(t)$ denotes the optimal actions derived by POO, for CUs, we define its cost operator $C_{jt}(w_i^*(t), x_i^*(t))$ and $R_j(t)$ as price of offloading i to j at t. $C_{jt}(w_i^*(t), x_i^*(t))$ can be a linear or convex function of $w_i^*(t), x_i^*(t)$. Given the CUs are rational, stimulating resource provision should let $R(t)$ be no less than the cost processing i, i.e., $C_{jt}\left(x_i^*(t), w_i^*(t)\right) < R_i(t)$, because the CUs attempts to obtain the revenue from the task offloading. Let the $C_t(.) = \max_{i\in G(t)} C_i(.)$, we define the total reward during t

$$R(t) = \sum_{i\in G(t)} C_t\left(x_i^*(t), w_i^*(t)\right) \tag{19}$$

Algorithm 3 BC-CED Rewarding and Penalizing

Input: $\{x_i^*(t)\}_{i\in G(t)}$, $\{w_i^*(t)\}_{i\in G(t)}$, t,$C_t(.)$
, **Output**: $\{R_i(t)\}_{i\in G(t)}$,$\{D_j(t)\}_{j\in\mathcal{N}}$
Compute the total reward $R_i(t)$ during t;
while $i \in G(t)$ **do**
  Rewarding: $R_i(t) \leftarrow \frac{(y_i^*)(t)R(t)}{\sum_{i\in\mathcal{G}(t)} y_i^*(t)} - \frac{1}{h_i(t)} y_i^*(t)$;
  Penalizing: $\gamma C_{jt}\left(x_i^*(t), w_i^*(t)\right)/(\hbar(j))$;
end
Subscriber verify the result of i;
while $j \in \mathcal{N}$ **do**
  if j *finish the task* **then**
    transfer reward $R_i(t)$ to j's account;
  else if j *fail* **then**
    confiscate $\sum_{\tau=t_b}^{t} D_i(\tau)$ units of currency from i's account;
  end
end
final ;

CUs of tasks in $G(t)$ share the payoff $R(t)$ based on the volume of their providing resource. We determine the reward for each task i by the following:

$$R_i(t) = \frac{y_i^*(t) R(t)}{\sum_{i\in\mathcal{G}(t)} y_i^*(t)} - \frac{1}{h_i(t)} y_i^*(t) \tag{20}$$

where $(y_i^*)(t) R(t) = \alpha w_i^*(t) + \beta x_i^*(t)$ and α and β are the non-negative pre-defined weight with $\alpha+\beta=1$. According to Eqn. (20), we have $R_i(t) \geq C_{jt}\left(w_i^*(t), x_i^*(t)\right)$ for all i. This indicates that providing resources to the task i is profitable for CUs. Namely, the rewarding scheme is individually rational. In the next section, we will also illustrate the truthfulness of the incentive. Based on the above resource pricing and reward calculation, we further provide a auto reward transferred scheme as following:

Step 1, freezing: Once the subscriber issues the request for task offloading, a payment to the brokerage with $R_i(t)$ will be recorded and frozen.

Step 2, rewarding: Subscriber confirms whether the task is successfully accomplished. If the task is completed, the reward $R_i(t)$ will be automatically unlocked and transferred to the corresponding CU's account.

2) Penalizing: In CED, CUs can be human-carried devices and may break their promise during resource provision, which impairs the system performance and sustainability. Thus, it necessary to penalize the CUs with dishonest behaviors as well. We introduce the concept of `deposit`. After the CU j confirming to process the assigned task, A `deposit` for j according to its provision resource and reliability of j. If the worker completes the task, the `deposit` will be waived and return to the CU's wallet. Otherwise, the `deposit` will be confiscated for compensating the loss of CED. We calculate the `deposit` for j processing i by $D_i(t) = \gamma C_{jt}\left(x_i^*(t), w_i^*(t)\right)/(\hbar(j))$, where α is a constant.

Algorithm 2 depicts the pesudo code of rewarding and penalizing in BC-CED.

Truthful analysis: According to [15], we define the incentive is truthful if the reward of the participated CUs is maximized. For our proposed task allocation and rewarding algorithms, this indicates that no CU can obtain a higher reward by dishonestly manipulating their costing function. Following theorem ensures the truthfulness of our proposed scheme.

Theorem 1: In each t, the incentive scheme is truthful, namely, for any of CU j, we have $R'_i(t) \leq R_i(t)$, where $R'_i(t)$ is reward derived by changing $C_{jt}(.)$ to $C'_{jt}(.)$, and $C'_{jt}(.) \neq C_{jt}(.)$, $x'_i(t)$ is the volume of contributed resource calculated by using $C'_{jt}(.)$.

Proof: We consider four different cases of each i:

(1) If $x^*_{ij}(t) > 0$, worker j submits $C'_{jt}(.)$ to replace $C'_{jt}(.)$, where $C'_{jt}(x, w) < C_i(x, w)$. According to **Algorithm 3**, if CU j has been selected to provide resource within t, we have $x'_{ij}(t) = x^*_j(t)$, where the $x'_i(t)$ is decision when applying $C'_i(.)$. Because the incentive applying the $C_t(.)$ to calculate the total reward, we have

$$\begin{aligned} R_i(t) &= \frac{y^*_i(t) R(t)}{\sum_{i \in \mathcal{G}(t)} y^*_i(t)} - \frac{1}{h_i(t)} y^*_i(t) \\ &= \frac{y'^*_i(t) R(t)}{\sum_{i \in \mathcal{G}(t)} y'^*_i(t)} - \frac{1}{h_i(t)} y'^*_i(t) \end{aligned} \tag{21}$$

Hence, i cannot gain any benefit from submitting a smaller cost factor $C'_i(t)$.

(2) If $x^*_{ij}(t) = 0$, CU j set its submitted $C'_{jt}(x) > C_{jt}(x)$. Because $C'_{ij}(x, w) \leq C_{jt}(x, w)$, we still have $R_i(t) = R'_i(t)$. Namely, i cannot gain any benefit from submitting a larger cost factor $C'_i(.)$.

(3) If $x^*_{ij}(t) = 0$, CU j set its submitted $C'_{ij}(x) < C_i(x)$. If $x'_{ij}(t) > 0$, namely, i has been selected to processing j by faking the cost function. According to the rewarding scheme, the actual revenue of i can be given by

$$C_t(x^*_i(t)) - C_i(x^*_i(t)) < 0$$

This is because $C_i(.) > C_t(.)$ otherwise the $x^*_i(t) < 0$. Hence, the rewarding of such case is negative.

(4) If $x^*_{ij}(t) = 0$,CU j set its submitted $C'_{jt}(x)! = C_{jt}(x)$. The $x'_i(t) = 0$ the revenue in this case is 0.

Based on above analysis, we can conclude that any CU cannot gain his revenue by manipulating the $C_i(.)$, hence, the theorem holds. ■

VI. Evaluation

We implement our BC-CED prototype. First, we deploy our blockchain design on BROP platform developed by Hangzhou blockchain Technology Research Institute. BROP is a blockchain-based environment for supporting efficient and truthful cooperation among different parties. Besides, BROP also supports various consensus mechanisms, which is ideal to evaluate the performance of POO. In our prototype, the blockchain consists of 4 workers and these workers are implemented on a server (Intel i7-11700k, 8 cores 1.7GHz/32GB). In each block, we specify the block ID, the generated time stamp, number of transactions, and transaction information described in Sec IV. The number of transactions in each block is set to 10000 unless otherwise specified. For every 3s, the block will be agreed and added to the blockchain via the consensus protocol.

Second, we build the RL environment based the a machine learning platform MindSpore Lite tool [37]. The RL model applies a convolution neural network (CNN) as the model. A maxpooling layer follows the first convolutional layer. The out channels of convolutional layers are 25, 48, 256, 64, 16, respectively. The kernel size layers are 9,2,7,3,3,3,4, respectively. The final layer is fully connected and outputs the action with probability distributions. To simulate the difference between the workers, the hyper-parameter, including the learning rate and utility function's weight parameters, are different. We further consider four different RL methods:

(1) Deep Q network (DQN), each client equips with two networks. The $Eval_{net}$ estimates the state-action value, and a $Target_{net}$ maintains the copy of $Eval_{net}$ to calculate the next state value function. A memory buffer caches the former 200 actions. The batch size is set to 32. The delay between $Eval_{net}$ and $Target_{net}$ is 5 iterations.

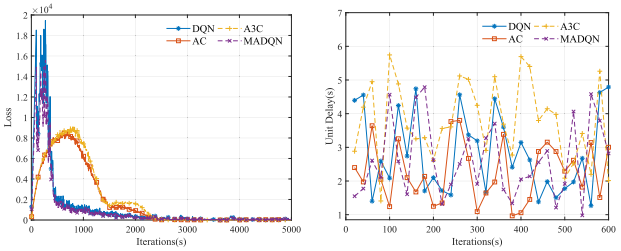
(2) Actor-Critic Network (AC), applies an AC network. The critic outputs the current state value function using the TD algorithm, and the actor outputs the action distribution by minimizing the advantage of the actions based on the critic output.

(3) Asynchronous Advantage Actor-Critic (A3C), extends the AC by introducing paralleled processing. The learning agent evokes four threads to train the model, and they collaboratively update a sharded model integrating the training results.

(4)Multi-agent DQN (MADQN), this method consists of four learning agent and each agent uses the DQN network to generate the policy independently.

Third, to further test the offloading performance, the offloading scheme derived by blockchain will be applied to guide the transcoding task allocation in our previously designed collaborative video transcoding platform. Each joined device can contribute resources for video transcoding on this platform. The mobile devices generate the video transcoding request and decide which CU to offload. Then, the video data will be delivered to the target CU. After finishing the transcoding, the transcoded content will be returned to the MD. This platform consists of one server acting as the Cloud (Intel Xeon Platinum 8163, 8 cores 2.5Ghz/64GB), three edge servers (Intel i7-11700k, 8 cores 1.7GHz/32GB), and four laptops (Intel i7-7700k, 4 cores 4.25Ghz/16GB, AMD Ryzen 5 Quad-Core 3.2GHz/16GB, Intel i7-10750H, 6cores 2.6GHz/16GB, Intel i7-10510U, 4 cores 1.8Ghz/16GB) acting as the MDs and subscriber. The operating system is CentOS 7 and uses FFmpeg as the transcoding tool. Each node installs a resource monitor to record the resources variation. The bandwidth of the wired link is 100Mbps. The length of the transcoding video content is a random variable $U[2, 10]$s. The transcoded bitrate set of the original video includes 1080p(60fps), 1080p(30fps), 720p(60fps), 720p(30fps), 480p(30fps).

The subscriber generates a random number of tasks with different computation resource requirements and video sizes at each time slot. The number of the generated tasks follows a



(a) Loss convergence during the training (b) Reward variation during the test

Fig. 4. RL algorithm performance.

Poisson distribution λ_1. The number of time slots between two consecutive tasks generating follows a uniform distribution with $U[5, 20]$, where $U[a, b]$ denotes a uniform distribution over the range $[a, b]$.

A. RL Methods Performance

To test the generality of our formulated task offloading optimization, we first evaluate the performance of four RL methods (DQN, A3C, AC, MADQN) when solving this problem. The investigated metrics include loss variance during training and unit processing delay during the test.

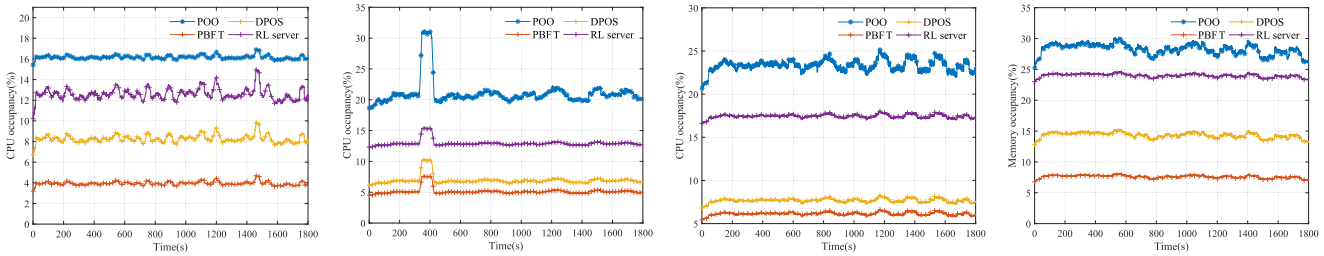
Loss variance: Fig. 4 (a) shows the loss function value during the training. According to the figures, all solutions experienced a fast decreasing trend and then entered the stable phases, indicating that the algorithm converged. The reason causes this phenomenon is that the learning agent has no prior knowledge of the system dynamic at the beginning and improves policies by estimating the values of actions at different states. Once the value function or Q-value is learned, the agent's policy converges, making the curves stable. All these curves converge around 2500s, revealing that they have similar performance for task offloading problems. The difference is that the AC-based method experiences an increasing trend initially and then decreases after 900s. Meanwhile, DQN and MADQN continue falling during the training. Thus, in the case of insufficient training time, DQN and MADQN are preferred for performing the task offloading.

Unit Delay (UD) during the test: We define the UD as the latency of processing a unit raw data of the task. We test the UD of different solutions via datasets of 600s and show the results in Fig. 4 (b). As shown in the figure, A3C outperforms the other solutions in most cases since it has the lowest delay among all solutions. The advantage of A3C is that the AC model always seeks the action better than the average. Besides, A3C uses four different processes to explore the environments for achieving better performance simultaneously. The other three solutions are close in performance. The curves corresponding to DQN are higher than AC and MADQN initially and become very close after the 300s. Based on the above analysis, we can conclude that all these RL methods can be applied for solving the CED task offloading problem. Therefore, blockchain workers can arbitrarily choose the RL methods to output offloading policy since it is difficult to find an RL method that dominates the other in all cases.

1) Blockchain Results: Applying the consensus mechanism to process the blocks can introduce extra resource consumption and delay to the system. Lower resource consumption and faster blocks processing can benefit the overall system performance. We test our proposed POO with two widely used consensus mechanisms DPoS and PBFT, in terms of the CPU and memory occupancy, bandwidth consumption, and block building latency. CPU occupancy: Fig. 5 shows the CPU occupancy with the variant of transactions volumes during the test. Our POO has the highest CPU occupancy under four different cases, i.e., POO is about 500% and 200% higher than that of the DPOS and PBFT when the transaction volume during the simulation is 2.5 million. The high CPU consumption in POO is because each blockchain worker applies an RL model to output the task offloading policy, which consumes most of the computation resource. DPoS maintains relatively low and is stable when transaction volume varies, since a committee with a given number of workers agree and output blocks. PBFT has a similar performance compared with the PBFT since all nodes vote for the right of the block. Although the PBFT and DPoS have a lower CPU usage, these two consensuses cannot provide the function of making task offloading decisions and require an independent optimizer to perform the task offloading. The purple curves in the figures indicate the CPU usage of an RL-based task offloading optimizer whose performance is very close to the POO, especially when in high transaction volumes. Therefore, applying POO for the blockchain-based system will not increase CPU consumption.

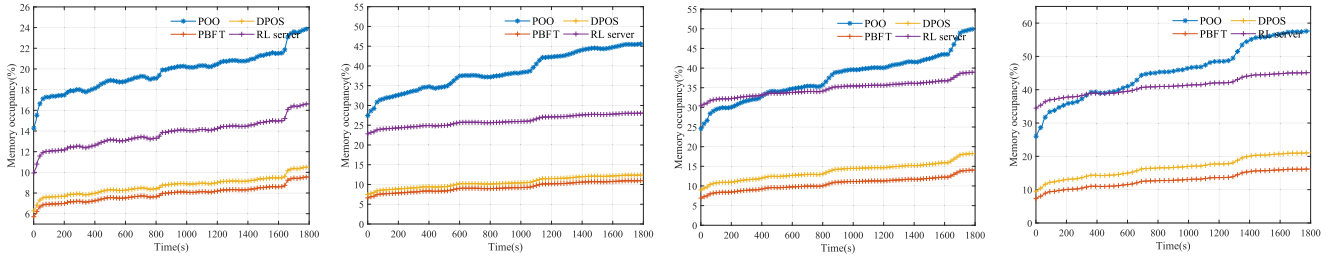
Memory occupancy: Fig. 6 shows the memory occupancy with the variant of transactions volumes during the test. Similar results as the CPU occupancy can be observed that the POO has the highest memory occupancy under all conditions of transaction volumes. Yet the independent optimizer also achieves high memory usage, especially when the transaction volume is large. For instance, the memory usage of the independent optimizer is 14%, 26.1%, 35% and 40% at 1000s, comparing the POO with 20%, 38%, 40% and 46% at 1000s, respectively. Besides, PBFT and DPoS memory occupancy is about to 10% in most cases. The result indicates applying the POO can save memory usage compared with using a consensus mechanism and performing task offloading optimization separately.

Bandwidth consumption: Fig. 7 illustrates the average bandwidth consumption during the test under different transaction volumes. According to the figure, POO has a higher bandwidth occupancy than the other two consensus mechanisms when the transaction volume is 1 million and becomes the lowest when the number of transactions increases. The reason is that workers in POO require to collect the environment information that introduces extra bandwidth. With the transaction volume increasing, the share of the bandwidth consumed by the interaction between workers increases and becomes the main consumer of the communication resources. DPoS has the highest bandwidth consumption when transactions volume equals 1.5 - 2.5 million since DPoS requires frequently elect the witnesses and leaders among the witnesses.



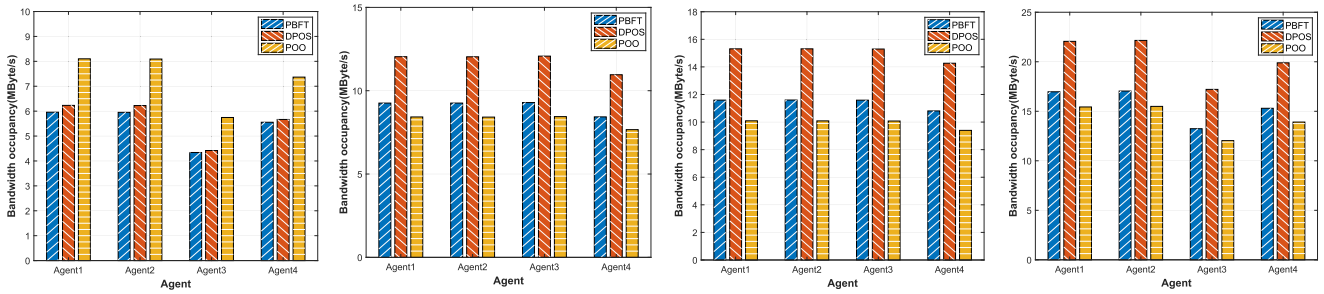
(a) Trans=1 million (b) Trans=1.5 million (c) Trans=2 million (d) Trans=2.5 million

Fig. 5. CPU occupancy during the test.



(a) Trans=1 million (b) Trans=1.5 million (c) Trans=2 million (d) Trans=2.5 million

Fig. 6. Memory occupancy during the test.

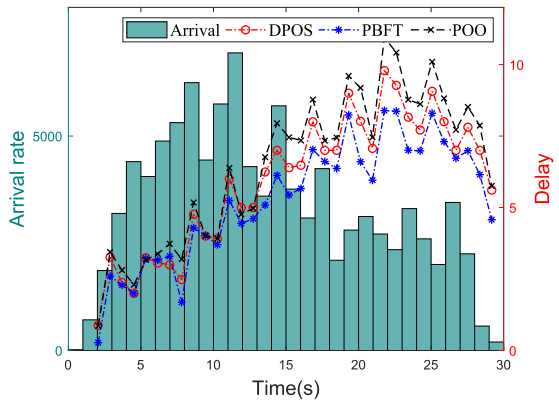


(a) Trans=1 million (b) Trans=1.5 million (c) Trans=2 million (d) Trans=2.5 million

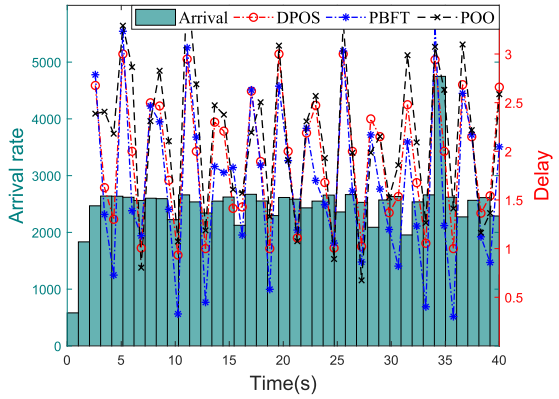
Fig. 7. Average bandwidth consumption.

Transaction verifying delay: We define the transaction verification delay (TVD) as the time interval between issuing and adding transactions to the blockchain. Fig. 8 shows the average TVD under two different arrival patterns: (1) The burst arrival pattern, the arrival rate (green bar in the figure) is relatively high (up to 8000 per seconds) within 5s to 15 seconds and remain stable in the rest of the simulation; (2) Stabilized arrival pattern, the arrival rate is stable at 2800 in most cases during the test. The TVD of three solutions in a burst arrival pattern experiences an increase of up to 10s and then decreases to 5s. The reason is that the arrival rate during the 5s to 15s is high and overwhelms the transaction processing rate. As a result, the number of pending transactions increases, thereby pushing up the TVD. The pending backlog decreases when the transaction arrival rate becomes lower than the processing rate. All curves fluctuate around the 2s during the test for the stabilized pattern. The cause of this fluctuation is that each block is added to the blockchain once the number of contained transactions equals 10000 or reaches the 3s generating deadline. Thus, transactions added to a block almost full or close to the building deadline can have a lower delay. We also observe that three consensus algorithms have similar performance in TVD since the RL network applied by the workers is trained, and the optimization process is relatively fast compared with the block processing latency.

2) Offloading Performance: Based on our designed collaborative transcoding platform, we compare the BC-CED with our previous proposed task offloading algorithm AGO [1] and the random policy. The AGO decides the task offloading policy based on the stochastic network optimization framework, and random policy selects the node to offload transcoding requests by a uniform distribution. Fig. 9 (a) (b) show the average CPU and Memory usage of different types of CUs. According to the figure, BC-CED prefers offloads the task to the MDs while AGO favors the Edge servers. This is because our formulated problem considers the processing delay of the task and the transmission capability. Heavily relying on the edge servers may congest the links between subscribers and edge servers, which increases the overall latency. This conclusion can be derived by observing the Fig. 9 (c) where the BC-CED has

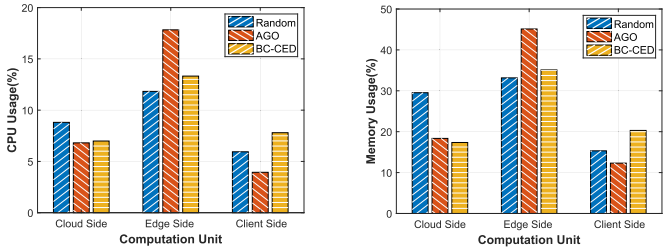


(a) Burst arrival pattern

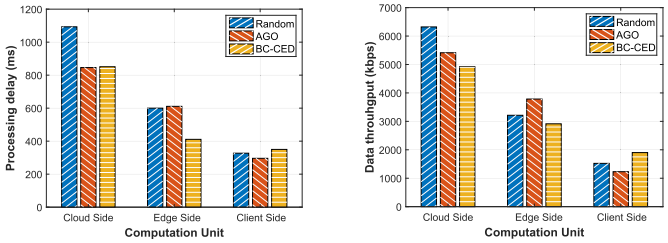


(b) Stablized arrival pattern

Fig. 8. Transaction verifying delay.



(a) CPU Usage (b) Memory Usage



(c) Processing Delay (d) Data Throughput

Fig. 9. Performance of three solutions.

a lower delay at the edge side compared with AGO. Besides, BC-CED also saves the pressure edge bandwidth as shown in Fig. 9 (d). The random policy uniformly distributes the task two to three types of CUs and hence has the highest processing delay and throughput on the Cloud side, which is not preferred by time-sensitive computation tasks.

VII. Conclusion

This paper proposes a blockchain-empowered task offloading for CED computing, BC-CED. We first give the designate of the BC-CED, which mainly consists of an application, blockchain, and resource layers. We then formulate the CED task offloading as a POMDP and illustrate how to use the RL-based method to solve this problem. For functionality of task publishing, task offloading, and resource management in BC-CED, we further design four smart contracts and give the detailed design of POO. This novel consensus mechanism enables each blockchain worker to reach an agreement on output blocks via solving the formulated POMDP. Moreover, we also provide an incentive mechanism that ensures trustworthiness and encourages the CUs to contribute their resources to the BC-CED. To evaluate the performance of BC-CED, we conduct a series of numerical evaluations and simulation tests based on a prototype system. Especially, we implement the design of our POO in a commercialized blockchain platform BROP to test the performance in the realistic. Simulation results show our proposed BC-CED improves resource usage and reduces the overall processing delay when using POO. Our prototype-level results also show that our proposed algorithm outperforms the existing offloading schemes in terms of the execution delay, load rate, and resource usage.

References

[1] X. Chen, C. Xu, M. Wang, Z. Wu, L. Zhong, and L. A. Grieco, "Augmented queue-based transmission and transcoding optimization for livecast services based on cloud-edge-crowd integration," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 31, no. 11, pp. 4470–4484, Nov. 2021.

[2] S. Wang et al., "Adaptive federated learning in resource constrained edge computing systems," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 6, pp. 1205–1221, Jun. 2019.

[3] H. Wu, Z. Zhang, C. Guan, K. Wolter, and M. Xu, "Collaborate edge and cloud computing with distributed deep learning for smart city Internet of Things," *IEEE Internet Things J.*, vol. 7, no. 9, pp. 8099–8110, Sep. 2020.

[4] X. Wang, X. Chen, and W. Wu, "Towards truthful auction mechanisms for task assignment in mobile device clouds," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, May 2017, pp. 1–9.

[5] I. Jang, S. Choo, M. Kim, S. Pack, and G. Dan, "The software-defined vehicular cloud: A new level of sharing the road," *IEEE Veh. Technol. Mag.*, vol. 12, no. 2, pp. 78–88, Jun. 2017.

[6] M. Wang, C. Xu, X. Chen, L. Zhong, Z. Wu, and D. O. Wu, "BC-mobile device cloud: A blockchain-based decentralized truthful framework for mobile device cloud," *IEEE Trans. Ind. Informat.*, vol. 17, no. 2, pp. 1208–1219, Feb. 2021.

[7] B. Lin, Y. Huang, J. Zhang, J. Hu, X. Chen, and J. Li, "Cost-driven offloading for DNN-based applications over cloud, edge, and end devices," *IEEE Trans. Ind. Informat.*, vol. 16, no. 8, pp. 5456–5466, Aug. 2020.

[8] K. Zhang, Z. Yang, and T. Basar, "Multi-agent reinforcement learning: A selective overview of theories and algorithms," *Handbook of Reinforcement Learning and Control* (Studies in Systems, Decision and Control). Cham, Switzerland: Springer, 2021, pp. 321–384.

[9] Y. Huang, F. Wang, F. Wang, and J. Liu, "DeePar: A hybrid device-edge-cloud execution framework for mobile deep learning applications," in *Proc. IEEE Conf. Comput. Commun. Workshops (INFOCOM WKSHPS)*, Apr. 2019, pp. 892–897.
[10] Y. He, J. Ren, G. Yu, and Y. Cai, "D2D communications meet mobile edge computing for enhanced computation capacity in cellular networks," *IEEE Trans. Commun.*, vol. 18, no. 3, pp. 1750–1763, Mar. 2019.
[11] J. Wen, C. Ren, and A. K. Sangaiah, "Energy-efficient device-to-device edge computing network: An approach offloading both traffic and computation," *IEEE Commun. Mag.*, vol. 56, no. 9, pp. 96–102, Sep. 2018.
[12] D. Bertsekas, *Reinforcement Learning and Optimal control*. Nashua, NH, USA: Athena Scientific, 2019.
[13] M. Tang and V. W. S. Wong, "Deep reinforcement learning for task offloading in mobile edge computing systems," *IEEE Trans. Mobile Comput.*, vol. 21, no. 6, pp. 1985–1997, Jun. 2022.
[14] L. Ale, N. Zhang, X. Fang, X. Chen, S. Wu, and L. Li, "Delay-aware and energy-efficient computation offloading in mobile-edge computing using deep reinforcement learning," *IEEE Trans. Cognit. Commun. Netw.*, vol. 7, no. 3, pp. 881–892, Sep. 2021.
[15] A. Zavodovski, S. Bayhan, N. Mohan, P. Zhou, W. Wong, and J. Kangasharju, "DeCloud: Truthful decentralized double auction for edge clouds," in *Proc. IEEE 39th Int. Conf. Distrib. Comput. Syst. (ICDCS)*, Dallas, TX, USA, Jul. 2019, pp. 2157–2167.
[16] J. Weng, J. Weng, J. Zhang, M. Li, Y. Zhang, and W. Luo, "DeepChain: Auditable and privacy-preserving deep learning with blockchain-based incentive," *IEEE Trans. Dependable Secur. Comput.*, vol. 18, no. 5, pp. 2438–2455, Nov. 2021, doi: 10.1109/TDSC.2019.2952332.
[17] Y. Qu, S. R. Pokhrel, S. Garg, L. Gao, and Y. Xiang, "A blockchained federated learning framework for cognitive computing in industry 4.0 networks," *IEEE Trans. Ind. Informat.*, vol. 17, no. 4, pp. 2964–2973, Apr. 2021.
[18] H. Cheng, Q. Hu, X. Zhang, Z. Yu, Y. Yang, and N. Xiong, "Trusted resource allocation based on smart contracts for blockchain-enabled Internet of Things," *IEEE Internet Things J.*, vol. 9, no. 11, pp. 7904–7915, Jun. 2022.
[19] Z. Qin, J. Ye, J. Meng, B. Lu, and L. Wang, "Privacy-preserving blockchain-based federated learning for marine Internet of Things," *IEEE Trans. Computat. Social Syst.*, vol. 9, no. 1, pp. 159–173, Feb. 2022.
[20] G. Wood, "Ethereum: A secure decentralised generalised transaction ledger," Ethereum Project, Zug, Switzerland, Yellow Paper 151, 2014, pp. 1–32.
[21] M. Castro and B. Liskov, "Practical Byzantine fault tolerance," in *Proc. OSDI*, vol. 99, 1999, pp. 173–186.
[22] *BROP*. Accessed: Jul. 2022. [Online]. Available: https://www.brop.cn/
[23] H. Xing, L. Liu, J. Xu, and A. Nallanathan, "Joint task assignment and resource allocation for D2D-enabled mobile-edge computing," *IEEE Trans. Commun.*, vol. 67, no. 6, pp. 4193–4207, Jun. 2019.
[24] D. Yi, X. Zhou, Y. Wen, and R. Tan, "Efficient compute-intensive job allocation in data centers via deep reinforcement learning," *IEEE Trans. Parallel Distrib. Syst.*, vol. 31, no. 6, pp. 1474–1485, Jun. 2020.
[25] X. Chen, H. Zhang, C. Wu, S. Mao, Y. Ji, and M. Bennis, "Optimized computation offloading performance in virtual edge computing systems via deep reinforcement learning," *IEEE Internet Things J.*, vol. 6, no. 3, pp. 4005–4018, Jun. 2019.
[26] Y. Zhang, B. Di, Z. Zheng, J. Lin, and L. Song, "Distributed multi-cloud multi-access edge computing by multi-agent reinforcement learning," *IEEE Trans. Wireless Commun.*, vol. 20, no. 4, pp. 2565–2578, Apr. 2021.
[27] C. Liu, F. Tang, Y. Hu, K. Li, Z. Tang, and K. Li, "Distributed task migration optimization in MEC by extending multi-agent deep reinforcement learning approach," *IEEE Trans. Parallel Distrib. Syst.*, vol. 32, no. 7, pp. 1603–1614, Jul. 2021.
[28] Y. Deng, T. Han, and N. Zhang, "FLeX: Trading edge computing resources for federated learning via blockchain," in *Proc. IEEE Conf. Comput. Commun. Workshops (INFOCOM WKSHPS)*, May 2021, pp. 1–2.
[29] W. Sun, J. Liu, Y. Yue, and P. Wang, "Joint resource allocation and incentive design for blockchain-based mobile edge computing," *IEEE Trans. Wireless Commun.*, vol. 19, no. 9, pp. 6050–6064, Sep. 2020.
[30] F. Bai, T. Shen, Z. Yu, K. Zeng, and B. Gong, "Trustworthy blockchain-empowered collaborative edge computing-as-a-service scheduling and data sharing in the IIoE," *IEEE Internet Things J.*, vol. 9, no. 16, pp. 14752–14766, Aug. 2022.
[31] H. Wu, K. Wolter, P. Jiao, Y. Deng, Y. Zhao, and M. Xu, "EEDTO: An energy-efficient dynamic task offloading algorithm for blockchain-enabled IoT-edge-cloud orchestrated computing," *IEEE Internet Things J.*, vol. 8, no. 4, pp. 2163–2176, Feb. 2021.
[32] M. Min, L. Xiao, Y. Chen, P. Cheng, D. Wu, and W. Zhuang, "Learning-based computation offloading for IoT devices with energy harvesting," *IEEE Trans. Veh. Technol.*, vol. 68, no. 2, pp. 1930–1941, Feb. 2019.
[33] M. E. Sudip, A. Mukherjee, A. Roy, N. Saurabh, Y. Rahulamathavan, and M. Rajarajan, "Blockchain at the edge: Performance of resource-constrained IoT networks," *IEEE Trans. Parallel Distrib. Syst.*, vol. 32, no. 1, pp. 174–183, Jan. 2021.
[34] R. S. Sutton, D. McAllester, S. Singh, and Y. Mansour, "Policy gradient methods for reinforcement learning with function approximation," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 12, 1999, pp. 1–7.
[35] Y. He, Y. Wang, C. Qiu, Q. Lin, J. Li, and Z. Ming, "Blockchain-based edge computing resource allocation in IoT: A deep reinforcement learning approach," *IEEE Internet Things J.*, vol. 8, no. 4, pp. 2226–2237, Feb. 2021.
[36] M. Neely, *Stochastic Network Optimization With Application to Communication and Queueing Systems*. San Rafael, CA, USA: Morgan & Claypool, 2010.
[37] (2020). *Mindspore*. [Online]. Available: https://www.mindspore.cn/

Su Yao received the Ph.D. degree from the National Engineering Laboratory for Next Generation Internet Interconnection Devices, Beijing Jiaotong University. Currently, he is with the Beijing National Research Center for Information Science and Technology (BNRist), Tsinghua University, as an Assistant Research Fellow. His research interests include future network architecture, the IoT security, and artificial intelligence for network systems.

Mu Wang received the Ph.D. degree in computer technology from the Beijing University of Posts and Telecommunications (BUPT) in 2020. He is currently serves as a Post-Doctoral Researcher with the Beijing National Research Center for Information Science and Technology (BNRist), Tsinghua University. His research interests include information centric networking, wireless communications, and multimedia sharing over wireless networks.

Qiang Qu is a Professor with the University of Chinese Academy of Sciences, with Shenzhen Institute of Advanced Technology, Chinese Academy of Sciences. He is currently the Director of Blockchain Laboratory of Huawei Cloud Tech Company Ltd. His research interests include blockchain and data intensive systems.

Ziyi Zhang is the Chief Architect of blockchain with Huawei Cloud Tech Company Ltd. His research interests include blockchain and cloud computing.

Yi-Feng Zhang is currently pursuing the D.Eng. degree with Tsinghua University. He is the Director of Zhongchao blockchain Technology Research Institute. He won several prizes of Bank S and T Progress Award from People's Bank of China in last ten years. His current research interests include central bank digital currency, blockchain, and network security.

Ke Xu (Senior Member, IEEE) received the Ph.D. degree from the Department of Computer Science and Technology, Tsinghua University. He is a Full Professor with Tsinghua University. He has published more than 100 technical articles and holds 20 patents in the research areas of next generation internet and P2P systems.

Mingwei Xu (Member, IEEE) received the B.Sc. and Ph.D. degrees from Tsinghua University. He is a Full Professor with the Department of Computer Science, Tsinghua University. His research interests include computer network architecture, high-speed router architecture, and network security.