A Novel Proof-of-Reputation Consensus for Storage Allocation in Edge Blockchain Systems

Jiarui Zhang, Yaodong Huang, Fan Ye, Yuanyuan Yang
Department of Electrical and Computer Engineering, Stony Brook University, Stony Brook, NY 11794, USA
{jiarui.zhang.2, yaodong.huang, fan.ye, yuanyuan.yang}@stonybook.edu

Abstract—**Edge computing guides the collaborative work of widely distributed nodes with different sensing, storage, and computing resources. For example, sensor nodes collect data and then store it in storage nodes so that computing nodes can access the data when needed. In this paper, we focus on the quality of service (QoS) in storage allocation in edge networks. We design a reputation mechanism for nodes in edge networks, which enables interactive nodes to evaluate the quality of service for reference. Each node publicly broadcasts a personal reputation list to evaluate all other nodes, and each node can calculate the global reputation of all nodes by aggregating personal reputations. We then propose a storage allocation algorithm that stores data to appropriate locations. The algorithm considers fairness, efficiency, and reliability which is derived from reputations. We build a novel Proof-of-Reputation (PoR) blockchain to support consensus on the reputation mechanism and storage allocation. The PoR blockchain ensures safety performance, saves computing resources, and avoids centralization. Extensive simulation results show our proposed algorithm is fair, efficient, and reliable. The results also show that in the presence of attackers, the success rate of honest nodes accessing data can reach 99.9%.**

I. Introduction

The exponentially growing edge nodes, such as Internet-of-Things (IoT) sensors, smartphones, even vehicles, generate data and create value anytime and anywhere. Under the condition that a single device cannot complete the ever-increasing large-scale tasks, emerging edge computing has promoted the cooperation of edge devices with different capabilities all over the world.

The storage allocation is an important consideration in the research and application of edge computing [1]. Consider the following scenarios that edge devices storage and access data. Many advanced sensors can produce extremely large and continuous data streams [2]. Short video applications such as TikTok allow users to share their daily lives by recording and downloading video clips [3]. Vehicles can access maps and real-time traffic flow through cellular networks to plan driving routes [4]. These scenarios require servers to store large amounts of data generated in the network and quickly access these data when needed by terminals. To resist disagreements and adversaries, some traditional centralized solutions need basic trust like trusted third-party (e.g. certificate authority) or transitive trust assumption. The limitations of these centralized solutions that rely on trust parties are significant and inevitable, such as efficiency problems caused by disconnection and privacy problems caused by the trust. To avoid the issues caused by centralization, decentralized solutions are introduced to edge computing environments.

As a distributed ledger, blockchain is the most prevalent decentralized system in Peer-to-Peer (P2P) networks. Recently, the blockchain is introduced as a safe and effective technology for storage allocation in edge networks. During the storing process, the edge device obtains a piece of new data and stores it in other edge devices or servers in the edge network, and stores the storage location on the blockchain. During the accessing process, the edge device queries the blockchain to obtain storage locations of the data and directly selects the appropriate location to request. Compare to centralized solutions, the blockchain solution guarantees that all the content can be accessed even when some nodes are offline, and the query will only send to the selected location without revealing privacy.

Based on current solutions to resource allocation in edge network environments, we find that two points are often ignored. First, current blockchain consensus mechanisms have limitations. The classic consensus mechanism of blockchain is Proof-of-Work (PoW), which consumes a large number of computing resources. Since most edge devices have limited computing power, it is impossible for them to consume a lot of computing power to generate blocks. Some work [5], [6] apply Proof-of-Stake (PoS) instead. The basic idea of the PoS is that nodes with more tokens generate blocks with higher priority. A problem with the PoS is that it often leads to unavoidable centralization. The reason is that block generators usually have more tokens, and they receive reward tokens as block generators, which makes them more likely to become block generators in the future, leading to centralization. Therefore, we need to design a consensus mechanism to avoid the limitations of PoW and PoS. Second, network participants have different reliability, and they may not provide the services as expected. Generally speaking, if a node provides more incomplete data, the peer interacting with the node considers the node to be less reliable. A reputation mechanism is an appropriate solution to the reliability of network participants [7]. We customize a reputation mechanism according to our environment. We are facing the following challenges to overcome the storage allocation in edge networks, 1) how to reach a decentralized consensus on the blockchain through limited resources, 2) how to build an appropriate reputation

978-0-7381-3207-5/21/$31.00 ©2021 IEEE

mechanism for nodes to evaluate the trust of others, and 3) how to consider fairness, efficiency, and reliability for storage allocation in edge networks.

In this paper, we first design a reputation mechanism in edge networks. Every node in the network shares the personal reputation of others. To reach a consensus, every node obtains the same global reputation from the same personal reputation. We then propose a storage allocation algorithm that considers fairness, efficiency, and reputation. Once a user generates a piece of data, the user stores the data to locations derived by the algorithm. With the help of our reputation mechanism, we build a novel Proof-of-Reputation (PoR) blockchain to maintain the reputation and storage allocation information. The basic idea of PoR is to let the node with the most increase of the global reputation value in each block be the generator. The PoR blockchain avoids limitations of PoW and PoS mechanisms. Extensive simulations show that our proposed system ensures a high success rate of accessing data when keeping fair and efficient storage allocation.

Our main contributions are summarized as follows.

- We design a reputation mechanism for nodes in edge networks to evaluate each other. The personal reputations are generated by each node, and the global reputations are obtained by aggregating personal reputations.
- Considering fairness, efficiency, and reliability, we propose a storage allocation algorithm. Nodes select storage locations through the algorithm to store newly generated data.
- Based on the reputation mechanism, we build a PoR blockchain to maintain the related information of storage allocation. The node with the most increase of global reputation value in each block is selected as the generator. The PoR blockchain satisfies low cost and decentralization.
- We conduct simulations in edge networks to evaluate the performance. The simulation results show that our PoR blockchain ensures decentralization, and most block generators are honest. Compared with previous work, our algorithm improves the success rate of accessing data, reaching close to 99.9% in our simulations. The simulation results also show that our storage allocation algorithm satisfies fairness, efficiency, and reliability requirements.

The rest of this paper is organized as follows. Section II briefly reviews the related work. In Section III, we formulate the problem and propose threat models. In Section IV, we introduce the reputation mechanism. In Section V, we propose the storage allocation algorithm. In Section VI, we propose the PoR blockchain structure. Simulation results are presented in Section VII. The last section concludes this paper.

II. Related Work

Bitcoin was invented in 2008 by Satoshi Nakamoto [8] and has been one of the most popular P2P applications. The related technology blockchain is composed of a chain of blocks, and the blockchain records transactions generated by P2P network participants as a distributed consensus ledger. According to whether a node can join the blockchain freely, blockchains can be in two forms, permissionless blockchain and permissioned blockchain. Swanson et al. [9] discussed the main differences between permissioned and permissionless blockchains. Briefly speaking, blockchain was first designed for anonymous cryptocurrency transactions in untrusted environments, and nodes can join a permissionless network freely. There are various permissionless blockchain systems, such as Bitcoin and Ethereum [10]. Permissioned blockchains have authorized identities, and different participants have different access-control authorizations [11], thus a node cannot join a permissioned blockchains freely since a new node is unauthorized. Hyperledger Fabric [12] and Corda [13] are examples of permissioned blockchain applications.

The advantages of PoR, including low energy consumption and safety performance, make it an emerging blockchain consensus mechanism. PoR blockchain applications covering a variety of scenarios are rising. Gai et al. [14] presented PoR, the reputation serves as the incentive for both good behavior and block publication. Based on PoR, Yu et al. invented RepuCoin [15] with better attack resistance and higher throughput compared to PoW blockchains. Oliveira et al. [16] developed PoR and proposed an advanced consensus for private blockchain systems. Wang et al. [17] implemented a reputation incentive scheme for blockchain consensus of Industrial Internet of Things (IIoT).

Blockchain techniques provide secure and decentralized applications in edge computing environments. To empower resource trading in mobile edge computing networks, Qiao et al. [18] applied blockchain to manage resource trading and task assignment. It combines a third-party trust center server and blockchain ledger to manage activities reliably. Huang et al. [19] proposed a consensus resource allocation system in pervasive edge computing environments. This work fairly and efficiently allocates storage resources and applies PoS to save energy consumption. As practical applications, edge computing and blockchain have become common tools for Internet-of-Vehicles (IoV) [20], [21].

Reputation management in different edge computing scenarios draws attention from researchers in recent years. Some work that applies reputation mechanisms to blockchain is studied based on various application scenarios. Huang et al. [22] discussed reputation management in vehicular edge computing and networks. Liu et al. [23] optimized resource allocation in blockchain-based video streaming systems with mobile edge computing. With the help of reinforcement learning, Xiao et al. [24] utilized a blockchain-based trust mechanism to resist attacks in edge networks. In edge computing and networks, the trust environment, conditional restrictions, and environmental requirements are highly customized, thus we can see a variety of different reputation mechanisms and blockchain designs.

III. Problem Statement and Threat Model

A. Problem Statement

In order to characterize the storage allocation model, we describe the following two operations that nodes in the network can perform. First, a node generates a new piece of data, then stores it in other nodes. Second, a node requests a specific piece of data from another node that owns the data.

Our purpose is to allocate storage resources reasonably to fulfill fairness, efficiency, and reliability requirements. The fairness requirement is to balance the proportion of space consumed by each node. The efficiency requirement is to store each piece of data in the network so that all nodes can efficiently request the data. The reliability requirement is to store data in reliable nodes. The fairness and efficiency requirements are straightforward, and we explain the reliability requirement here. Generally speaking, complete data is considered reliable, and partially or completely missing data is considered unreliable. Multiple reasons may lower the reliability of data, such as network fluctuation or malicious behavior. When accessing data, nodes that provide complete data are more reliable with a higher probability.

To help nodes quantify the reliability of others, we design a reputation mechanism that enables nodes to evaluate the reputations of others. We design a structure to maintain the storage allocation and the reputation mechanism so that nodes in edge networks can query data storage and historical reputation records. We mainly focus on the following three issues.

1) We design a reputation mechanism for the nodes in the edge network, which enables the interacting nodes to evaluate each other for reference.
2) We propose a storage allocation system that considers fairness, efficiency, and reliability.
3) We build a structure to maintain and support consensus on the storage allocation and reputation mechanism.

B. Network Assumptions

The network consists of nodes with different storage resources. A node can generate data, select some nodes to store data, and request data from other nodes. We assume that all nodes have the basic computing power to generate blocks of the blockchain. We assume that nodes pay providers revenue for requested data. This not only prevents nodes from sending a large number of malicious data requests but also incentivizes nodes to provide storage resource services. The specific payment method is beyond the scope of this paper.

C. Threat Model

We list several possible attacks on the mechanism by malicious attackers to discuss the security performance of our system. We assume that attackers are all rational, which means they only focus on maximizing their profits, and do not attack without profit.

1) Bad-mouthing attack: Attackers use false reputation feedback to cause deviations in the reputation evaluation mechanism. Generally speaking, attackers will give high reputation ratings to their own nodes, while giving other nodes low reputation ratings.

2) Denial-of-service attack: In a general Denial-of-Service (DoS) attack, attackers dilute regular data by sending a large amount of false and interfering data. In our environment, attackers may send a large amount of feedback to interfere with reputations.

3) Sybil attack: Sybil attackers register multiple different identities and use expanded capabilities of multiple different identities to attack. In our environment, attackers may create multiple identities to participate in the reputation system to obtain profits.

For the sake of simplicity, in our following discussion, nodes are divided into honest nodes and malicious nodes. Honest nodes reply to all requested data and make true evaluations. Malicious nodes may not reply to requested data and make false evaluations. /bf In order to study the worst-case performance of our system, we assume that all malicious nodes belong to the same attacker, which means that the attacker can control all malicious nodes to maximize their total profits.

IV. Reputation Mechanism

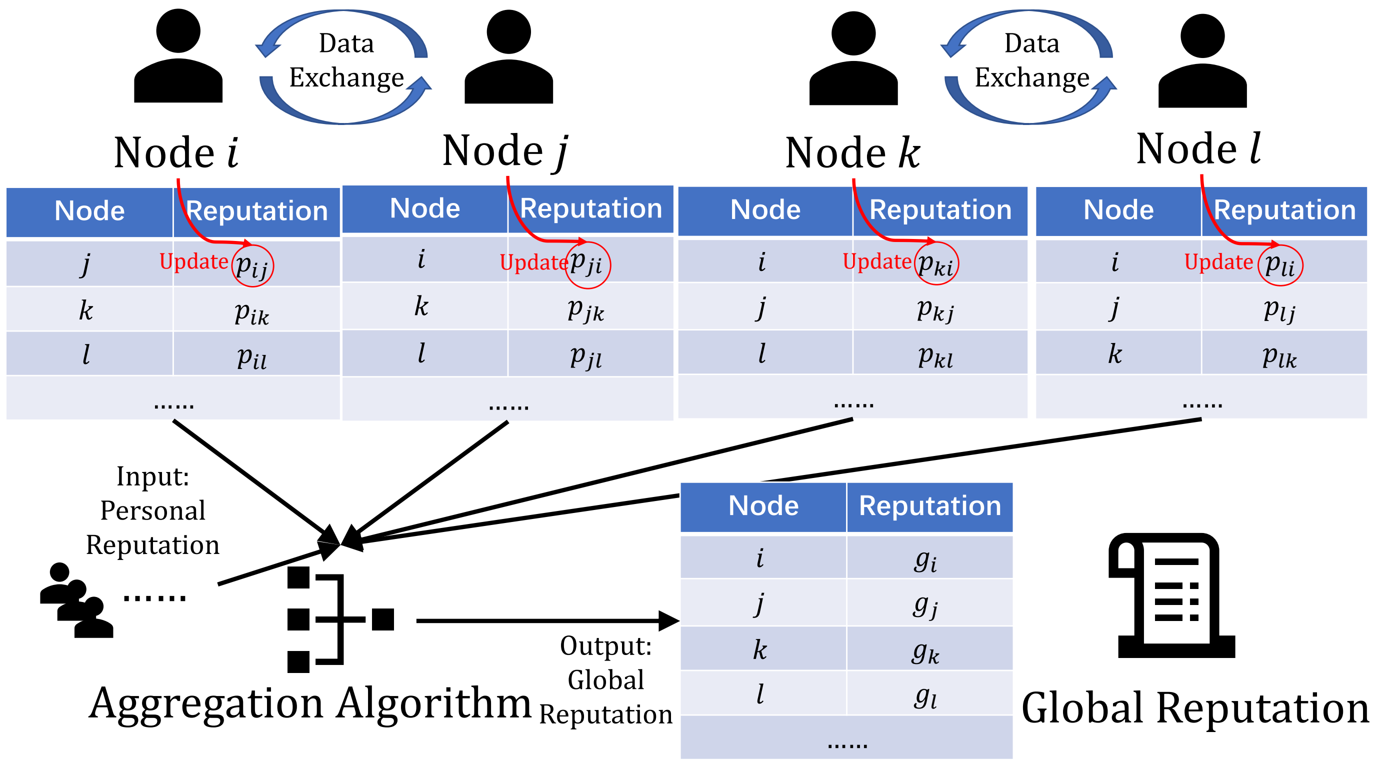
We introduce our reputation mechanism in this section. In edge networks, the reputation of a node is the evaluation of the node by other nodes based on its behavior. Generally speaking, the behavior of nodes with high reputations is more compliant with the rules of the network than nodes with low reputations. According to different sources of evaluators, our reputation mechanism includes the personal reputation and global reputation. We also explain these two types of reputations in detail.

A. Reputation Mechanism Model

In an edge network, the reputation of a node is an evaluation of the node by other nodes based on its behavior. As Fig. 1 shows, our reputation mechanism establishes the reputation of each node in two ways, including personal reputation and global reputation. Personal reputations refer to the evaluations of a single node to another node. Global reputations are overall evaluations obtained by aggregating all personal reputations. Each node maintains a list of personal reputations of other nodes based on its interaction records in edge networks. Nodes share personal reputations in the network as public knowledge. By aggregating the shared personal reputations, all nodes in the network can obtain a list of recognized global reputations. As a result, our reputation mechanism not only gives each node the right to freely evaluate other nodes but also enables nodes to reach a consensus on global evaluations based on public knowledge.

B. Personal Reputation

Each node can make its own personal reputations of other nodes. For the nodes evaluated by node i, we consider that i



(a)

Fig. 1. A simple graph of relationships of global reputation, personal reputation, and data exchange. Each node maintains and shares its personal reputation list. After two nodes exchange data, they will update the reputations each other. Based on the shared reputation list, each node can calculate the same global reputation list using the same aggregation algorithm.

evaluates a personal reputation p_{ij} for node j. This personal reputation is derived from the perspective of i based on the historical behaviors of j. Different nodes may observe different historical behaviors due to different perspectives, and they may have different personal reputation evaluations of the same object. For example, the personal reputation p_{kj} of node j by node k may be different from p_{ij}. Among these different personal reputations, each node maintains its own personal reputation, and each node cannot directly modify the personal reputations of other nodes. In other words, node i maintains its personal reputation p_{ij} for node j, and only i can update the value of p_{ij}.

C. Global Reputation

In practical applications, we often need all nodes to reach a consensus on the reputation of nodes in the network. However, since the personal reputation of the same object is often different, applying the personal reputations of any node cannot make most nodes in the network reach a consensus. In order to obtain a reputation list that most nodes can reach a consensus, we need a method to aggregate a global reputation to evaluate a node based on the personal reputations of all nodes. The global reputation g_i of node i is derived from the personal reputations of all nodes. We aggregate personal reputations by EigenTrust algorithm [25], which is a highly adaptable algorithm for calculating global reputation based on personal reputations in P2P networks. Here, we briefly describe the EigenTrust algorithm.

Since the evaluation criteria and dimensions of each node are different, the EigenTrust algorithm standardizes all personal reputations as shown below,

$$c_{ij} = \frac{\max(p_{ij}, 0)}{\sum_j \max(p_{ij}, 0)}. \tag{1}$$

The personal reputation p_{ij} is standardized to c_{ij} for each $i, j \in \mathcal{V}$, where $\mathcal{V}$ is a node set including all nodes in the network. After the standardization, node i adjusts its views on node j according to the personal reputation provided by other nodes. Using matrix notation to describe, the original standardized personal reputations of node i on all nodes is the vector $\mathbf{c}_i$, the standardized personal reputation matrix of all nodes $[c_{ij}]$ is C, and node i can update the personal reputation vector $\mathbf{c}'_i = C^T\mathbf{c}_i$. Node i can iterate the update process continuously, and the result of iterating n times is $\mathbf{t}_i = (C^T)^n\mathbf{c}_i$. For all nodes i, $\mathbf{t}_i$ often converges to the same vector after less than 10 iterations [25], [26]. To simply compute the converged vector, nodes apply $\mathbf{e}$ instead of any $\mathbf{c}_i$ as the initial reputation vector, $e_i = 1/|\mathcal{V}|$. Assume g_i is used to evaluate the global reputation of node i, nodes can compute the global reputation vector $\mathbf{g}$,

$$\mathbf{g} = (C^T)^n\mathbf{e}. \tag{2}$$

While personal reputations are public knowledge, nodes in the network can independently obtain the same global reputation vector $\mathbf{g}$ and reach a consensus on the vector.

V. Storage Allocation

In edge networks, nodes have different kinds and amounts of resources. For example, edge computing networks take advantage of the widespread of sensor nodes to collect data from many channels. Due to cost issues, these sensor nodes that can be widely distributed often have limited storage and computing resources, and they need to transmit the collected data to other nodes with more resources. Since the cost increases with the growth of storage and computing resources, nodes with a large amount of storage and computing resources cannot be widely distributed. In such edge networks, the storage allocation needs to consider different storage, location, computing power, network bandwidth, and other resources of each node.

Reliability is another concern when considering resource allocation issues. Honest nodes and malicious nodes may not provide complete data, thereby affecting their reliability. Honest nodes may not provide complete data due to reasons such as network disconnection and power outage. Malicious nodes can perform a variety of malicious behaviors, such as providing incomplete or empty data, and not storing data that should be stored. Nodes can use the reputation mechanism to record the completeness of data sent by each node, then determine whether a node is reliable. Therefore, the reputation mechanism helps nodes maintain the reliability of others, thereby helping to adjust resource allocation and improve security.

A. *Fairness and Efficiency of Storage Allocation*

Previous work [19] has introduced the requirements of fair and efficient storage allocation. Based on this work, we briefly introduce the fairness and efficiency requirements in our environments.

We first consider the fairness requirement. Since the total storage resource of nodes is different, in order to keep the storage allocation fair, we consider balancing the proportion of consumed storage resource instead of the absolute amounts of consumed storage resource. Nodes with a smaller proportion of consumed storage resources have a higher priority to store new data. Fairness Degree Cost (FDC) is the resource consumption measurement. The FDC of node i is $f_i = \frac{W(i)}{W_{tol}(i)-W(i)}$, where $W(i)$ is the consumed storage resource of i, and $W_{tol}(i)$ is the total storage resource of i. In particular, when $W(i) = W_{tol}(i)$, node i has no remaining storage resource, $f_i = \infty$.

Then we introduce the efficiency requirement. Generally speaking, the longer the distance between the data sending node and the receiving node, the higher their transmission delay cost. We use hop-count distance in this paper for simplicity. The transmission delay cost of node i transmitting data to the node j is denoted as r_{ij}, which is also called Range-Distance Cost (RDC) in our setting.

Now we discuss the case of storing a piece of data. The following Uncapacitated Facility Location (UFL) problem [27] formula describes the storage allocation based on FDC and RDC measurements [19]:

$$\begin{aligned}
\min \quad & A\sum_{i\in\mathcal{V}} f_i y_i + \sum_{i\in\mathcal{V}}\sum_{j\in\mathcal{V}} r_{ij}x_{ij} && (3)\\
\text{s.t.} \quad & \sum_{i\in\mathcal{V}} x_{ij} \geqslant 1, && \forall j \in \mathcal{V} \quad (4)\\
& y_i - x_{ij} \geqslant 0, && \forall i,j \in \mathcal{V} \quad (5)\\
& x_{ij}, y_i \in \{0,1\}. && \forall i,j \in \mathcal{V} \quad (6)
\end{aligned}$$

x_{ij} and y_i indicate the allocation result. Node i stores the piece of data if $y_i = 1$, and node i would access the piece of data from j if $x_{ij} = 1$. $A = 1000$ in objective function (3) is a parameter to match the value range of FDC and RDC. Constraint (4) ensures each node access data from at least one node, and constraint (5) guarantees a node that receives access requests owns the piece of data.

B. *Reliability of Storage Allocation*

When allocating storage space fairly and effectively, we also need to consider the reliability to ensure the completeness of the requested data. For example, when a node queries a malicious node for a piece of data, the malicious node may not save or provide data to reduce resource consumption. In this case, the node cannot receive the complete data. Therefore, we import the reliability of data provided by nodes when determining the storage allocation to increase the probability of successfully requesting complete data. According to the impact of reliability on fairness and efficiency, we adjust the FDC term and RDC term of the function separately.

1) Reputation coefficient: Since the reputation mechanism in Section IV can maintain the completeness of data provided by nodes, we add a reputation coefficient to the objective function (3) to adjust the storage allocation based on the reputation of nodes. We define a reliability coefficient $h_i = g_i/g_{max}$ represents the ratio of the global reputation of node i to the highest global reputation. g_{max} is the maximum global reputation, i.e., $g_{max} = \max\{g_i\}, \forall i \in \mathcal{V}$. Since h_i shows the relative reliability in standardized global reputation, we adjust FDC and RDC according to this coefficient.

2) Fairness degree cost: Node i may not store the data allocated to it, because it does not want to share storage resources as it claims. Based on storage resources declared by nodes and global reputations, nodes can estimate the storage resources that each node supposes to consume. For node i, it claims total storage resource $W_{tol}(i)$, and its global reputation is g_i, other nodes estimate that it supposes to consume $W'_{tol}(i) = h(i)W_{tol}(i)$. A straightforward method is to replace $W_{tol}(i)$ in f_i with $W'_{tol}(i)$. However, nodes may exaggerate storage resources to reduce FDC, directly replacing FDC that cannot effectively increase FDC in this case. To avoid this case, we regard $W_{tol}(i) - W'_{tol}(i) = (1-h_i)W_{tol}(i)$ as additional consumed storage resources. It can effectively increase the FDC of nodes with exaggerated resources and reduce their priority. Therefore, we use $(1-h_i)W_{tol}(i)+W(i)$ to replace $W(i)$ in f_i. We modify the FDC term as follows,

$$f_i' = \frac{(1-h_i)W_{tol}(i)+W(i)}{W_{tol}(i)-((1-h_i)W_{tol}(i)+W(i))}. \tag{7}$$

Note that it is possible that $W_{tol}(i) - ((1-h_i)W_{tol}(i) + W(i)) \leqslant 0$. In this case, we make $f_i' = \infty$, which means the algorithm will not let node i store new data.

3) Range-distance cost: Each node has reputation evaluation for other nodes based on historical data requests. Since a node with a high success rate of requesting data often have a high reputation, when a node selects targets to request data, it should consider not only the transmission cost but also the reputation. r_{ij} represents the RDC of j accesses data from i. We can adjust it through the related personal reputation p_{ji} or the related global reputation g_i. We choose to adjust RDC with the global reputation for two reasons. First, the standardized global reputation is more appropriate to use in the equation. Second, the global reputation is more difficult to manipulate compared to the personal reputation. Therefore, using the global reputation makes it more difficult for the attacker to manipulate resource allocation. Since $h_i \in [0,1]$, using the reliability coefficient h_i instead of the global reputation g_i keeps the RDC term in scale. We modify the RDC r_{ij} by h_i as follows,

$$r_{ij}' = \frac{r_{ij}}{h_i}. \tag{8}$$

As a result, based on the traditional uncapacitated facility location problem, we additionally considered the role of reliability in both FDC and RDC terms. Our modified UFL model is as follows,

$$\begin{aligned} \min \quad & A\sum_{i\in\mathcal{V}} f_i' y_i + \sum_{i\in\mathcal{V}}\sum_{j\in\mathcal{V}} r_{ij}' x_{ij} && && (9) \\ \text{s.t.} \quad & \sum_{i\in\mathcal{V}} x_{ij} \geqslant 1, && \forall j \in \mathcal{V} && (10) \\ & y_i - x_{ij} \geqslant 0, && \forall i,j \in \mathcal{V} && (11) \\ & x_{ij}, y_i \in \{0,1\}. && \forall i,j \in \mathcal{V} && (12) \end{aligned}$$

The current best solution to the UFL problem is proposed by Li et al. [28]. Since it is a 1.488 approximation algorithm with randomization, nodes may compute different results for the same UFL conditions.

Note that the algorithm is conducted by block generators. The latest block generator uses the results to determine the storing nodes of data pieces generated after the previous block.

VI. PoR Blockchain

In this section, we propose a PoR blockchain to maintain the above reputation mechanism and storage allocation records. The blockchain provides a consensus ledger for a series of transactions made by nodes in the network, and it also adapts to the consensus of reputation and storage allocation. Based on the reputation mechanism, we use PoR as the consensus mechanism in our design instead of classic PoW or PoS mechanisms.

A. Recording Information by Blockchain

We have introduced the reputation mechanism and the storage allocation in previous sections, and we need a structure for nodes in edge networks to apply them. The structure should provide a reliable information recording function so that each node can reach a consensus on information recording. The blockchain is widely used in edge networks for nodes to reach a consensus, and it can provide an immutable ledger for recording information. We then discuss how blockchain records the required information.

1) Reputation mechanism: Global reputations are computed by aggregating personal reputations. All nodes need to reach a consensus on global reputations so that nodes can calculate the algorithm in Section V based on the same global reputations. It faces two challenges. First, personal reputations are constantly changing, it is hard for nodes to keep updating the newest global reputations. Second, the network latency makes the personal reputation records of nodes inevitably different, and the different global reputations calculated by nodes cannot reach a consensus. To solve the challenges, we let block generators record current global reputations when generating blocks. More specifically, based on global reputations recorded in the previous block and personal reputation transactions in the current block, the generator of the current block calculates new global reputations and records them in the current block. Once a node receives a new block, it validates the content in the new block and computes the global reputation by aggregating personal reputations recorded in the new block and previous blocks. The node applies the global reputations recorded in this block when needed and refreshes the reputation records until it accepts the next block. Therefore, all nodes follow the global reputations recorded in the latest block.

2) Storage allocation: Recording storage allocation is for the convenience of nodes to know where it is stored when querying data. When a node is ready to store a piece of new data, it obtains locations to be stored according to the storage allocation algorithm. Then it broadcasts the data summary and storage location in the form of transactions. The blockchain records those transactions, and other nodes can query the storage location of any data.

B. Consensus Mechanisms Overview

The blockchain needs a consensus mechanism so that all nodes can reach a consensus on the content of the block. PoW and PoS are common consensus mechanisms in previous blockchain applications. These consensus mechanisms are mainly used to confirm the generators for new blocks. PoW selects new block generators through the consumption of proven computing power, and PoS selects new block generators based on the assets owned by participants.

PoR is an emerging consensus mechanism. The basic idea of PoR is to let the node with the most increase of global reputation value in each block be the generator. Intuitively speaking, the node with the most increase of global reputation is the node that makes the most profit at the reputation aspect. It needs to be honest to obtain the support of other nodes and obtain the corresponding increase in reputation. As evidence of personal and global reputations, the block generator needs to record changes in both reputations.

Commonly used consensus mechanisms, such as PoW and PoS, have limitations in our environment. The PoW mechanism needs to consume a lot of computing resources. In edge computing environments, the computing resources owned by different devices are unbalanced and predictable. Using the PoW mechanism not only consumes a lot of computing power but also makes it difficult for most edge nodes to compete for generating blocks. The PoS mechanism leads to inevitable centralization, that is, the block generators are always the same group of nodes, which makes it possible for them to manipulate the reputation mechanism. Therefore, in our environment, PoW and PoS mechanisms are inappropriate.

C. PoR Blockchain Design

We describe the specific PoR blockchain in detail. PoR needs a reputation mechanism as a foundation, and we have already proposed a reputation mechanism in Section IV. The reputation mechanism allows nodes to evaluate the personal reputation of others and generates the global reputation of each node. We next describe in detail how to use the existing reputation mechanism to design a PoR blockchain.

1) Restricting reputation updates: The reputation mechanism has no restriction for the frequency and timing of reputation updates. If a large number of nodes send reputation update information at a high frequency, a large amount of reputation update information will flood the network.

To solve the above problem, we limit the frequency and timing of reputation updates. We stipulate that nodes can update the reputation only if they have finished an interaction. In our storage allocation environment, the interaction means that node i requests data from node j, and they can update the reputation of each other once. In order to prove that a reputation update is related to an interaction, the reputation update information points to the corresponding data interaction. Thus, the frequency and timing of reputation updates are related to the frequency of data requests, and we need to restrict the frequency of data requests. The attacker can achieve the purpose of frequent malicious evaluations through frequent data requests. To avoid this situation, we allow one node to ask another node for the same piece of data once in one block. The reason is that for the same piece of data, the node has a cache for its recent access. For different pieces of data, they are generally stored in different nodes. The extra interaction messages will be ignored by other nodes.

2) Selecting block generators: The node with the most increase of global reputation in the current block is selected as the new block generator. The selection is as fair as possible to obtain confirmation from other nodes. However, a malicious node can behave maliciously in the beginning, then behave honestly to make a fluctuation of its global reputation between two different blocks. To avoid this case, we further add a constraint that the block generator must have a global reputation in the top 50%. This constraint prevents a node from becoming a block generator by reputation boost in a short period.

Assume the current block is the t-th block in the blockchain, and the global reputation of node i is $g_i(t)$ after applying all personal reputation changes recorded in the t-th block. Assume $\mathcal{V}'(t)$ contains the top 50% of the nodes in global reputation in the t-th block. If the block generator of the t-th block is node i, it satisfies

$$g_i(t) - g_i(t-1) \geqslant g_j(t) - g_j(t-1), \forall j \in \mathcal{V}'(t). \quad (13)$$

3) Generating new blocks: We describe the block generation process from the perspective of one node. The node continuously exchanges data in the network to generate corresponding transactions and reputation updates. The node broadcasts this information to the entire network after signing and receives such information from other nodes. When the number of data exchange transactions and reputation updates reaches a certain number, a new block is required to package and record these transactions and reputation updates on the blockchain. The node calculates new personal reputations through local personal reputation records and reputation updates that will be recorded in the new block. The node then computes the global reputation and checks whether it is the node with the highest reputation in the new block. If it is, it will broadcast the new block to the entire network. Otherwise, other nodes will generate new blocks, the node will check received blocks, and select a block generated by the correct block generator to continue working on it. To verify the validity of the block and the generator, the node verifies the transactions in the block and calculates the corresponding global reputation based on the content of the block.

4) Permissioned blockchain: Traditional blockchain is permissionless blockchain, which means nodes spread out the Internet can join or leave a permissionless blockchain without permission from any party. On contrary, a permissioned blockchain has restrictions in its memberships, users cannot join a permissioned blockchain freely.

We use permissioned blockchain instead of permissionless blockchain in our design. The reason is that permissioned blockchain improves the security performance of the reputation mechanism. The cost of attacks against reputation mechanisms in a permissionless environment is low, such as whitewashing attacks and Sybil attacks. A permitted blockchain has thresholds for joining nodes, or they have to pass the audit of existing nodes. The attacker needs to spend high costs to obtain identities in the blockchain. With the help of the reputation system, malicious nodes will be exposed, greatly increasing the cost of attack and improving the security performance.

Note that nodes need approval to enter and leave the permissioned blockchain. We give block generators the right to approve nodes to enter the network or remove nodes from the network. As long as several consecutive block generators approve, a node enters or moves out of the permissioned blockchain. The more specific design is beyond the scope of this paper.

D. Security Analysis

We discuss the security performance of PoR.

1) Bad-mouthing attack: In our PoR blockchain, the form of bad-mouthing attack is that the attacker conducts malicious evaluations after data interactions, which improves its own reputation and reduces the reputation of other nodes to gain the advantage of becoming a block generator in the competition. Since it is difficult for a third-party node that is not an interactive participant to obtain the actual interaction process between two nodes, it is difficult to detect malicious evaluations by the attacker. However, this attack is difficult to affect PoR. First, evaluations from a single node have a limited impact on the global reputation. Second, a node that keeps evaluating the personal reputation of others maliciously will be detected by honest nodes, and it will receive negative feedback from honest nodes. This leads to a decline in the global reputation of the node, and the node is at a disadvantage in the competition of block generators. Therefore, an attacker who performs a bad-mouthing attack cannot give an advantage to competing block generators.

2) Denial of Service attack: In our environment, the attacker performs a DoS attack by sending a large amount of spam feedback information to flood the network to block the transmission of normal reputation feedback. Since one node can only ask another node for the same piece of data once in one block, the number of times that any pair of nodes can evaluate each other is limited. Therefore, the attacker cannot send a large amount of spam reputation updates, and our PoR mechanism can effectively prevent DoS attacks.

3) Sybil attack: The attacker has two ways to perform Sybil attacks on our PoR blockchain. The first way is to create a large number of puppet nodes and use these nodes to perform malicious operations. The creation of a large number of puppet nodes is hard in permissioned environment since it is generally easy to find and consumes a lot of costs. Thus, this form of attack is hard to implement in our permissioned environment. The second way is that the attacker lets a node exit after its reputation is low, then the attacker joins the network with a new identity, which is also known as a whitewashing attack. However, re-entering the network requires costs, such as the time cost of waiting to join the permissioned blockchain and the monetary cost of identity mortgages. Moreover, the reputation of the new identity will be lowered if it keeps behaving maliciously. Therefore, it is difficult for Sybil Attack to play a role in the permissioned PoR blockchain.

VII. Performance Evaluation

A. Simulation Process and Settings

Since global reputation is standardized, nodes can use a variety of reputation mechanisms. In our simulations, all nodes use a simple personal reputation mechanism. We describe the mechanism in the perspective of an honest node i evaluates personal reputation p_{ij} of a node j.

- Node i records the number of good evaluations $good_{ij}$ and total evaluations of node tot_{ij}. If node i receives a piece of data from node j, then $tot_{ij} = tot_{ij} + 1$. If the data is complete, $good_{ij} = good_{ij} + 1$.
- If node i gives a piece of data to j, $tot_{ij} = tot_{ij} + 1$. If node j increases p_{ij} after receives the data from i, then $good_{ij} = good_{ij} + 1$.
- $p_{ij} = good_{ij}/tot_{ij}$.

We generate an operation list, representing the operations of the nodes in the network arranged in chronological order. From the perspective of node i, we introduce the following two operations.

- Node i generates a new piece of data. Node i provides the data size, queries the latest block generator for storage locations, and stores the data in the corresponding locations.
- Node i accesses a piece of existing data. Node i accesses the data from node j that owns the data to minimize r_{ij}/p_{ij}.

After each access, nodes related to the access update the personal reputation of each other. When the number of operations reaches a threshold, the node with the highest global reputation becomes a new block generator. It generates a new block and updates the global reputation.

We generate the total storage resource $W_{tol}(i)$ for each node i by a normal distribution, $\mathcal{N}(1000000, 100000)$ KB. We generate the size of each piece of data by $\mathcal{N}(100, 30)$ KB. Each piece of data will be accessed by a uniformly random node multiple times, which are generated by $\mathcal{N}(10, 2)$. We generate an operation list, and the simulation runs according to that list. The number of generating operations divided by the number of access operations is approximately 0.1, and they appear evenly in the list. When the number of operations not on the blockchain reaches 110, the node that satisfies condition (13) generates a new block that records these operations. We terminate simulations when the number of blocks reaches 1000. We generate the test network $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ by the Watts-Strogatz model [29]. This model generates networks that reveal properties of some real communication networks.

We let honest nodes constantly give factual evaluations and provide complete data. To test the security of our design, we make some nodes malicious. Malicious nodes perform bad-mouthing attacks by conducting malicious evaluations after operations. In addition, malicious nodes may refuse to provide data. The attack probability is 0.5, that is, there is a probability of 50% that malicious nodes make wrong evaluations, and malicious nodes refuse to provide data with a probability of 50%. Considering the worst case, we assume that all malicious nodes always make good evaluations with each other. We run simulations under environments that include the different proportions of malicious nodes. We only show measurements of honest nodes, since our system is designed for honest nodes.

We use the algorithm in [19] as the benchmark. In the following simulation, we compare the results of the algorithm and the benchmark.

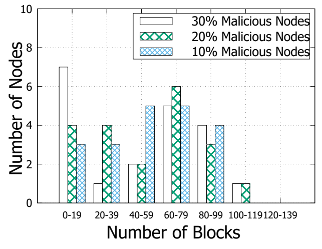


Fig. 2. A histogram representing the distribution of the number of blocks generated by the node.

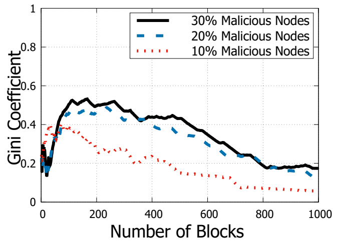


Fig. 3. The Gini coefficients of the used storage ratio for all nodes.

B. Decentralization Test

We count the number of blocks generated by each node, and use a histogram in Fig. 2 to show the distribution of the number under different proportions of malicious nodes. The reason for the distribution of a certain number of nodes in the range of 0 to 19 is that all malicious nodes are distributed in this range. It is worth mentioning that in all simulations, malicious nodes can generate up to 2 blocks out of 1000. Very few honest nodes generate more than 100 blocks or less than 20 blocks, and no node generates more than 120 blocks. Most honest nodes can generate 40 to 100 blocks. These phenomena show that in the PoR blockchain, most honest nodes can generate close to the average number of blocks, and very few nodes generate a small or large number of blocks. Thus, we can conclude that our PoR blockchain is decentralized, and it can ensure that the block generators are honest with a high probability.

C. Fairness Test

We use the Gini coefficient[1] of the used storage ratio to show the fairness of our system. Once a new block is generated, we compute the Gini coefficient of the used storage ratio for all honest nodes. In Fig. 3(b), in different proportions of malicious nodes, the Gini coefficients of all tests decrease with blocks increase. Although the Gini coefficients are unstable at the beginning, it decreases as the block is generated, and finally converges to less than 0.2. The reason for this phenomenon is similar to the reason for the phenomenon Fig. 3(a), reputation gaps in the initial stages also affect the balance of storage allocation results. As blocks increase, the used storage ratio balances. Therefore, we can conclude that our algorithm is fair in the long term.

D. Efficiency Test

To show the efficiency of our algorithm, we count the average hop-count distance to access data. Since the benchmark is an efficient algorithm, we can conclude that our algorithm is efficient if the average hop-count distance of data request computed by these two algorithms is close. In Fig. 4(a), the average hop-count distance computed by our algorithm rapidly reaches 2, then increases slowly and stabilizes at 2.2. Fig. 4(b) shows that the growth rate of the average hop-count distance computed by the benchmark algorithm is relatively slow, and the average eventually tended to around 2.2. The average access distance of our algorithm is unstable at the beginning, and after rapid stabilization, it is close to the result obtained by the benchmark.

[1]The Gini coefficient is widely used to measure the statistical disparities, and previous work [30], [31] used it to measure the fairness properties in edge environments. $Gini = \frac{\sum_i \sum_j |q_i - q_j|}{2\sum_i \sum_j q_j}$.

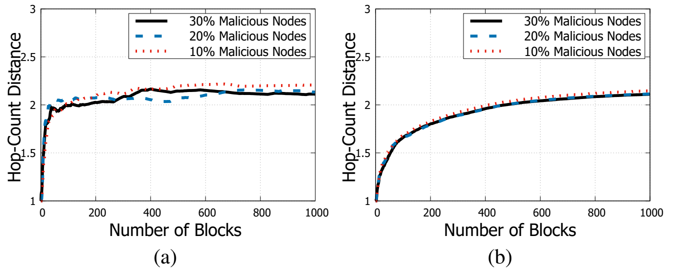


Fig. 4. The average hop-count distance of accessing data. (a) shows the results computed by our algorithm. (b) shows the results computed by the benchmark algorithm.

E. Successful Access Rate Test

The successful access rate test reveals the reliability of our algorithm. The reliability requires the algorithm to store data to honest nodes, and it is supposed to improve the successful access rate. Once a new block is generated, we compute the successful rate of accessing data. In Fig. 5(a), the successful access rate is above 99.9% in all simulations. In Fig. 5(b), the successful access rate decreases significantly with the increase in the proportion of malicious nodes. When the network includes 40% malicious nodes, compared with the benchmark, our algorithm improves the successful access rate by 14.6%. Therefore, our algorithm greatly improves the successful access rate compared with the benchmark and meets the reliability requirements.

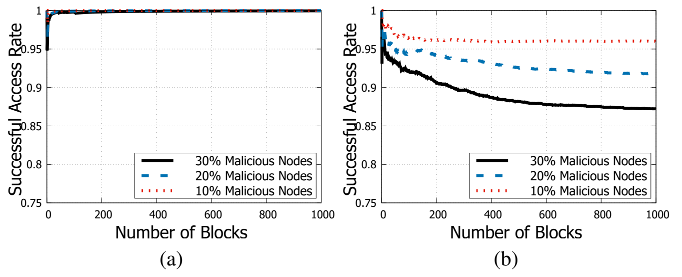


Fig. 5. The average successful access rate. (a) shows the results computed by our algorithm. (b) shows the results computed by the benchmark algorithm.

VIII. Conclusion and Future Work

In this paper, we have proposed a reputation mechanism that consists of personal reputation and global reputation. All nodes evaluate others by personal reputations, and they obtain global reputations by aggregating the same personal reputations of all nodes. We have designed a storage allocation mechanism that satisfies fairness, efficiency, and reliability. We have constructed a PoS blockchain based on our reputation mechanism, which costs low computing power and avoids centralization. Our simulations show that our system meets our expectations from multiple measurements. The storage allocation algorithm improves the access success rate while maintaining fairness and efficiency. In the case of malicious nodes that may not provide data, our system can achieve a 99.9% success rate of accessing data. The simulations also show that the PoR blockchain prevents centralization and ensures that the block generators are honest with a high probability.

In edge networks, new devices and servers will replace old devices and servers. It requires us to manage the joining and leaving of nodes. In addition, we use a permissioned blockchain to manage access to the blockchain to prevent attacks. Based on the above two concerns, we will further improve the permissioned blockchain network protocol to manage nodes joining and leaving the network, and eliminate nodes with lower reputations.

Acknowledgment

The work in this paper was supported in part by the grant from US National Science Foundation under grant numbers 1513719 and 1730291.

References

[1] W. Yu, F. Liang, X. He, W. G. Hatcher, C. Lu, J. Lin, and X. Yang, "A survey on the edge computing for the internet of things," *IEEE access*, vol. 6, pp. 6900–6919, 2017.

[2] P. Beckman, R. Sankaran, C. Catlett, N. Ferrier, R. Jacob, and M. Papka, "Waggle: An open sensor platform for edge computing," in *2016 IEEE SENSORS*. IEEE, 2016, pp. 1–3.

[3] X. Cheng, J. Liu, and C. Dale, "Understanding the characteristics of internet short video sharing: A youtube-based measurement study," *IEEE transactions on multimedia*, vol. 15, no. 5, pp. 1184–1194, 2013.

[4] L. Mendiboure, M. A. Chalouf, and F. Krief, "Survey on blockchain-based applications in internet of vehicles," *Computers & Electrical Engineering*, vol. 84, p. 106646, 2020.

[5] W. Sun, J. Liu, Y. Yue, and P. Wang, "Joint resource allocation and incentive design for blockchain-based mobile edge computing," *IEEE Transactions on Wireless Communications*, vol. 19, no. 9, pp. 6050–6064, 2020.

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

[7] P. Kochovski, S. Gec, V. Stankovski, M. Bajec, and P. D. Drobintsev, "Trust management in a blockchain based fog computing platform with trustless smart oracles," *Future Generation Computer Systems*, vol. 101, pp. 747–759, 2019.

[8] S. Nakamoto *et al.*, "Bitcoin: A peer-to-peer electronic cash system," 2008.

[9] T. Swanson, "Consensus-as-a-service: a brief report on the emergence of permissioned, distributed ledger systems," *Report, available online*, 2015.

[10] G. Wood *et al.*, "Ethereum: A secure decentralised generalised transaction ledger," *Ethereum project yellow paper*, vol. 151, no. 2014, pp. 1–32, 2014.

[11] M. Liu, K. Wu, and J. J. Xu, "How will blockchain technology impact auditing and accounting: Permissionless versus permissioned blockchain," *Current Issues in Auditing*, vol. 13, no. 2, pp. A19–A29, 2019.

[12] C. Cachin *et al.*, "Architecture of the hyperledger blockchain fabric," in *Workshop on distributed cryptocurrencies and consensus ledgers*, vol. 310, no. 4. Chicago, IL, 2016.

[13] M. Hearn, "Corda: A distributed ledger," *Corda Technical White Paper*, vol. 2016, 2016.

[14] F. Gai, B. Wang, W. Deng, and W. Peng, "Proof of reputation: A reputation-based consensus protocol for peer-to-peer network," in *International Conference on Database Systems for Advanced Applications*. Springer, 2018, pp. 666–681.

[15] J. Yu, D. Kozhaya, J. Decouchant, and P. Esteves-Verissimo, "Repucoin: Your reputation is your power," *IEEE Transactions on Computers*, vol. 68, no. 8, pp. 1225–1237, 2019.

[16] M. T. de Oliveira, L. H. Reis, D. S. Medeiros, R. C. Carrano, S. D. Olabarriaga, and D. M. Mattos, "Blockchain reputation-based consensus: A scalable and resilient mechanism for distributed mistrusting applications," *Computer Networks*, vol. 179, p. 107367, 2020.

[17] E. K. Wang, Z. Liang, C.-M. Chen, S. Kumari, and M. K. Khan, "Porx: A reputation incentive scheme for blockchain consensus of iiot," *Future Generation Computer Systems*, vol. 102, pp. 140–151, 2020.

[18] G. Qiao, S. Leng, H. Chai, A. Asadi, and Y. Zhang, "Blockchain empowered resource trading in mobile edge computing and networks," in *ICC 2019-2019 IEEE International Conference on Communications (ICC)*. IEEE, 2019, pp. 1–6.

[19] Y. Huang, J. Zhang, J. Duan, B. Xiao, F. Ye, and Y. Yang, "Resource allocation and consensus on edge blockchain in pervasive edge computing environments," in *2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS)*. IEEE, 2019, pp. 1476–1486.

[20] X. Lin, J. Wu, S. Mumtaz, S. Garg, J. Li, and M. Guizani, "Blockchain-based on-demand computing resource trading in iov-assisted smart city," *IEEE Transactions on Emerging Topics in Computing*, 2020.

[21] F. Ayaz, Z. Sheng, D. Tian, and Y. L. Guan, "A proof-of-quality-factor (poqf) based blockchain and edge computing for vehicular message dissemination," *IEEE Internet of Things Journal*, 2020.

[22] X. Huang, R. Yu, J. Kang, and Y. Zhang, "Distributed reputation management for secure and efficient vehicular edge computing and networks," *IEEE Access*, vol. 5, pp. 25 408–25 420, 2017.

[23] M. Liu, F. R. Yu, Y. Teng, V. C. Leung, and M. Song, "Distributed resource allocation in blockchain-based video streaming systems with mobile edge computing," *IEEE Transactions on Wireless Communications*, vol. 18, no. 1, pp. 695–708, 2018.

[24] L. Xiao, Y. Ding, D. Jiang, J. Huang, D. Wang, J. Li, and H. V. Poor, "A reinforcement learning and blockchain-based trust mechanism for edge networks," *IEEE Transactions on Communications*, 2020.

[25] S. D. Kamvar, M. T. Schlosser, and H. Garcia-Molina, "The eigentrust algorithm for reputation management in p2p networks," in *Proceedings of the 12th international conference on World Wide Web*, 2003, pp. 640–651.

[26] T. Haveliwala and S. Kamvar, "The second eigenvalue of the google matrix," Stanford, Tech. Rep., 2003.

[27] G. Cornuéjols, G. Nemhauser, and L. Wolsey, "The uncapicitated facility location problem," Cornell University Operations Research and Industrial Engineering, Tech. Rep., 1983.

[28] S. Li, "A 1.488 approximation algorithm for the uncapacitated facility location problem," *Information and Computation*, vol. 222, pp. 45–58, 2013.

[29] D. J. Watts and S. H. Strogatz, "Collective dynamics of 'small-world'networks," *nature*, vol. 393, no. 6684, pp. 440–442, 1998.

[30] Y. Huang, X. Song, F. Ye, Y. Yang, and X. Li, "Fair caching algorithms for peer data sharing in pervasive edge computing environments," in *2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS)*. IEEE, 2017, pp. 605–614.

[31] D. Wei, K. Zhu, and X. Wang, "Fairness-aware cooperative caching scheme for mobile social networks," in *2014 IEEE international conference on communications (ICC)*. IEEE, 2014, pp. 2484–2489.