IEEE TRANSACTIONS ON
MOBILE COMPUTING

Resource Allocation and Consensus of Blockchains in Pervasive Edge Computing Environments

Yaodong Huang*, Jiarui Zhang*, Jun Duan†, Bin Xiao‡, Fan Ye*, Yuanyuan Yang*
*Department of Electrical and Computer Engineering, Stony Brook University, Stony Brook, NY 11794
{yaodong.huang, jiarui.zhang.2, fan.ye, yuanyuan.yang}@stonybook.edu
†IBM Thomas J. Watson Research Center, Yorktown Heights, NY 10598, USA
jun.duan@ibm.com
‡Department of Computing, The Hong Kong Polytechnic University, Hong Kong
csbxiao@comp.polyu.edu.hk

***Abstract*—Edge devices with sensing, storage, and communication resources are penetrating our daily lives. These resources make it possible for edge devices to conduct data transactions (e.g., micro-payments, micro-access control). The blockchain technology can be used to ensure transaction unmodifiable and undeniable. In this paper, we propose a blockchain system that adapts to the limitations of edge devices. The new blockchain system can fairly and efficiently allocate storage resources on edge devices, which makes it scalable. We find the optimal peer nodes for transaction data storage and propose a recent block storage allocation scheme for quick retrieval of missing blocks. We develop data migration algorithms to dynamically reallocate data and block storage to adapt topology changes in the network. The proposed blockchain system can also reach consensus with low energy consumption in edge devices with a new Proof of Stake mechanism. Extensive simulations show that our proposed blockchain system works efficiently in edge environments. On average, the new system uses 18.4% less time and consumes 87% less battery power when compared with traditional blockchain systems.**

***Keywords*—Pervasive edge computing, Blockchain, Proof of Stake**

I. Introduction

Innovative edge devices like IoT devices, smartphones, and even vehicles change the way how we connect to the physical world. These edge devices are creating a massive amount of data as they become more and more pervasive and powerful. With the increasing volume of data generated at the edge, sharing data among peer edge devices allows data being processed locally without the involvement of cloud or other centralized authority.

There are many current applications that utilized edge devices to provide valuable data. Nokia is offering IoT devices that produce real-time sensing data and can provide sensing-as-a-services for environment monitoring and risk management [1], [2]. "We Media" content creators can create video clips, fictions, periodicals, compositions, and other types of digital contents and can be directly sold to customers through the platform such as Gumroad [3], [4]. Although such applications are now used worldwide through the Internet, many of them also have great potential in small scale edge scenarios, e.g., local environment monitoring, face-to-face data trading and recommendation.

Traditionally, many of these applications require a third party to manage subscriptions, payments, and data access control for creators and users. However, the third-party model may have availability, dependability, and capabilities issues, especially in edge scenarios. First, when sharing and trading data in edge scenarios, the backend Internet connection is not necessary and sometimes not applicable. Using a well-known online platform is not always possible in such a distributed environment, and may suffer from "a central point of failure". Second, the third-party platform may also face adverse events [5], mostly related to trust and privacy concerns, which may diminish the reliability of such platforms. Third, even we have a trusted entity locally, the capabilities for all transactions may cause huge communication and computation overhead, which is not applicable for edge devices which has much fewer resources. In peer edge environments, we aim at achieving reliable data accessing with managing micro-access control and micro-payment transactions in a distributed manner. The blockchain technology is an applicable, secure, and distributed ledger for sharing and processing such micro-payment and micro-access control information in such distributed environments.

The blockchain technology is now being used widely in cryptocurrencies. A blockchain consists of a chain of blocks, which is a kind of record or ledger. Each block contains the hash result from its previous block to form a chain. The blockchain has many security features over a distributed system. First, a complete history is kept throughout the network. Thus, it is easy to restore and verify the block information a user obtains. Second, a blockchain is designed to be resistant to modification of its records. Theoretically, in Proof of Stake consensus, unless malicious users hold more than half of the stakes [6], neither the block in the chain nor its contained data can be easily modified. Third, it is a disintermediation [7] system, meaning that there is no need for a trusted third-party to verify data, thus it can also avoid central point failure.

Despite the advantages of blockchain technology in such distributed systems, the edge devices have certain constraints on resources, especially storage and battery. The complexity and data duplication of the typical blockchain system make it impossible to deploy that on edge environments directly. Thus, we are facing three design challenges to overcome the

Digital Object Identifier 10.1109/TMC.2021.3053230

limitation of edge devices: 1) how to store data in the network with limited storage optimally, 2) how to adjust the data storage when device moves, and 3) how to reach consensus with low energy consumption in edge devices. Limited storage and communication capabilities require storing data and blocks separately on certain devices for reliable data accessing. The heterogeneity of different devices requires fair allocation over the resources, store fewer data items on devices with fewer resources. The mobility of devices requires data to be migrated to new places to keep reliable access and fair storage. Battery limitations require energy-saving consensus design in the forging process.

In this paper, we first design an optimal resource allocation strategy for applying blockchain to peer edge environments. We focus on building the platform for security but not timely data trading transactions in edge environments. We compress the storage of blocks by storing metadata instead of a large volume of actual data. The metadata items contain information for corresponding data. They also include the signature to keep data from illegitimate modification. We then store data and blocks at optimal places for fast user access, and keep fair resource allocation considering the heterogeneity of different devices. We further propose algorithms to reallocate data and block storages to adapt topology changes of the network due to node mobility. We also propose a new Proof of Stake (PoS) consensus mechanism to reduce the energy consumption for generating new blocks in edge environments. Extensive simulations show that our proposed blockchain system can achieve fast data and block access, fairness in data storage, and low computational overhead for block generating.

We make the following contributions in this paper.

- We design a blockchain system targeting resource allocation and block generating consensus on edge environments to achieve less storage and energy consumption compared to traditional blockchain systems.
- We propose a resource allocation strategy to find optimal places to store data and blocks. We formulate the problem of optimal storage allocation for reliable access and fair storage of data and blocks. We also provide a strategy to store recent blocks for quick retrieval of missing blocks.
- We propose optimal and heuristic data migration algorithms to reallocate data and block storage under network topology changes due to node mobility.
- We propose a new PoS mechanism that is suitable for edge devices with limited battery. The new PoS consensus mechanism can generate new blocks with low energy consumption on edge devices.
- We implement a distributed system and further conduct extensive simulations in peer edge device networks to evaluate the performance. Simulation results show that the proposed blockchain system can achieve fast data access with 18.4% less time, fair data storage with disparity measurement less than 0.2, and consensus with 87% less energy compared with traditional blockchain systems.

The rest of this paper is organized as follows. In Section II we discuss some related work on blockchain systems and peer edge networks. In Section III we discuss the model of our designed blockchain system. In Section IV we propose our block resource allocation strategies on edge environments. In Section V we propose a new Proof of Stake mechanism. We evaluate the proposed blockchain system in Section VI. Finally, we conclude the paper and discuss future work in Section VII.

II. Related Work

In this section, we discuss the related work over blockchain, edge computing and resource allocation.

A. Blockchain

The blockchain technology is proposed in 2008 by Satoshi Nakamoto [8] and has been widely used in cryptocurrencies ever since. It consists of a series of blocks linked using cryptography. Each block typically contains a hash result from the previous block, a timestamp, a hash for the block itself, and some other items based on application scenarios. Together these blocks form a chain. The blockchain can serve as a distributed ledger for storing data among devices [9], and data cannot be easily changed due to the cryptography features. Intuitively, if a malicious user wants to tamper with a piece of data, it has to make up a whole chain. The time and energy used to produce a fake chain are not worth the benefit it can get. The anti-manipulation feature of blockchain lays the foundation for cryptocurrencies, e.g., Bitcoin [8], Litecoin [10], and Ethereum [11].

Although blockchain technology has some security features, data transmission and storage remain challenging in distributed systems. The traditional blockchain requires each participating user to store the whole chain for security and performance, and each new transaction and each new block are broadcasted over the internet. Such huge transmission and storage overhead draw some attention to improving the storage consumption and data propagation over the blockchain. Ozisik et al. [12] use bloom-filter and IBLT (invertible bloom lookup table) to reduce the transmission overhead in the blockchain. Eyal et al. propose the Bitcoin-NG [13] to reduce the transmission and bandwidth by creating a new micro-block chain to minimize the amount of data transmitted.

The security of blockchain is build over the cryptography technologies. The traditional concept of mining is for miners competing with each other solving a mathematics puzzle. Whoever solves the puzzle first has the privilege to write the next block. This concept is called Proof of Work (PoW) [8], [14]. Data from Proof of Work are often hard to obtain but easy to get verified. Another emerging concept is called Proof of Stake (PoS) [15], [16]. The stake is a kind of publicly verifiable virtual resource of a specific user that ties to its historic data such as wealth, age, storage, or combination of those. Proof of Stake achieves the consensus from the publicly-owned data of the stake of users. Since PoS does not rely on exhaustively solving mathematic puzzles, it saves a lot of energy for mining a new block. Instead, history is very crucial for every node to reach a consensus about what a node owns. It can achieve consensus on adding new blocks, but the

computational power is significantly decreased compared with PoW.

B. Edge Computing

On the contrary of cloud computing which moves the computing to the centralized cloud, edge computing moves the computing work to the distributed nodes on the edge of the network. The computing mostly or entirely happens on nodes near to or inside the edge devices [17]. Edge computing can offer fast and robust data sharing and processing capabilities for end devices. One major research aspect of edge computing studies the benefit using smaller edge servers (cloudlets) deploying near the network edge (e.g., cellular base stations), serving as the middle layer between edge devices and clouds [18], [19]. These edge servers can offer multiple applications such as caching and resource virtualization. Another research aspect on edge computing studies the innovative functionalities from the collaboration of edge devices. Vehicle network is an example of this topic [20], [21].

C. Resource Allocation

In edge environments, resource allocation is crucial since nodes often have various and limited rearouses. Resource placement problems are often used in such scenarios. The optimal resource placement problem can often be mapped to classic Facility Location (FL) problems. In order to solve such problems, various kinds of FL or modified FL problems are proposed. Uncapacitated Facility Location (UFL) problem [22] and rent-or-buy problem [23] are most popular and well-studied. The more general case for these two problems is the Connected Facility Location (ConFL) problem [24]. Among them, UFL does not consider the content dissemination costs, while the rent-or-buy problem does not consider the facility building costs in the ConFL problem. In this paper, we mostly use the UFL problem and its corresponding solutions. The current best solution that we find is proposed by Li et al. [25], where they obtain a 1.488 approximation ratio.

III. Blockchain System Design

In this section, we introduce our blockchain model and discuss how the resources are allocated in the system.

Fig. 1 shows an application overview of our proposed system. The clients of the blockchain system consist of multiple edge devices. Some devices generate data such as self-generated for-profit data, and other nodes pay for data conducting micro-payment transactions. When payments are successful, data are to be delivered to consumers. The data related information and payment are encoded in the block, which builds the blockchain system. Then, data and blocks are stored among different nodes in the network. We now discuss in detail about the components of the proposed blockchain system below.

A. The Blockchain System Model

Our proposed blockchain system consists of multiple edge devices producing data transactions and forging blocks in the blockchain. A node is an edge device participating in the system. Each node has its private and public keys for identification purposes. The keys further generate an account of that node. Each account is unique and associated with each node and has a unique address satisfying a certain pattern. The account address can be generated from public keys but not in reverse. Each node tries to forge blocks to get incentives, and such incentives will be stored in its account. Since how to assign incentives is not a major concern in this paper, for simplicity, we assign the incentive for forging a new block as one "token" along with some coins. Coins serve as currencies in the system, which can be used to purchase certain data. Tokens are used in the Proof of Stake (PoS) processes.

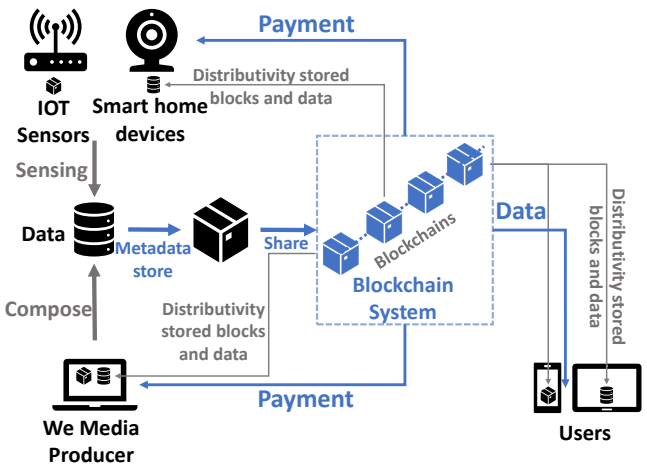


Fig. 1. The overview in the proposed blockchain system. Nodes generate for-profit data and users pay for them. Real data and blocks are fragmented and stored across the nodes.

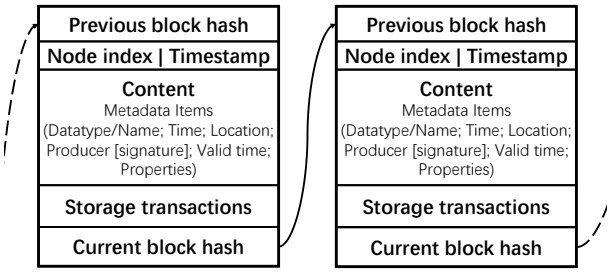


Fig. 2. The proposed blockchain system and components of a block in the blockchain system.

Fig. 2 shows the components of blocks in our proposed blockchain system. It contains the basic information of a typical blockchain, as well as some components that are designed for edge environments. The previous hash, index, timestamp, and current hash are like any regular blockchain system to ensure the connection between blocks. The content in blocks stores the metadata items, each of them is corresponding to a data item. In addition, each block records the information about where this block is stored using storage transactions. The storage allocation is discussed in the next section.

B. Metadata Design and Distributive Storage

In a traditional blockchain system, all data are shared with all nodes in the network and will be stored. Such data can be transactions, smart contracts, and some other forms, and the size of data can be relatively large. Due to the limitation of storage in edge devices, storing all data in every node is impractical. Thus, data items should be stored on a fraction of nodes, and instead, metadata are stored in blocks. The metadata contains basic information about a data item.

1) Metadata design: A metadata item consists of multiple attributes each having a corresponding value of the data. The metadata is proposed for data sharing in peer edges in [26]. A metadata item is generated alongside a data item from the data producer. Metadata items are then broadcasted in the network and nodes will wrap received metadata items into blocks. Here are some metadata item examples in the following.

```
Data type; Time; Location; Producer[Producer
signature]; Valid Time; Properties
(AirQuality/PM2.5; 11:00AM06-11-2018;
NewYorkNY/40.72,-74.00; [Signature of producer];
10,11,12,15; 1440; NULL)
(Picture/Traffic; 11:27AM06-11-2018; NassauNY/40.78,
-73.58; [Signature of producer]; 16,17,26,44; 720;
'Camera')
(KeyExchange/PublicKey; NULL; NULL; []; NULL; 2880;
[Key])
```

Since the metadata item contains the basic information of the corresponding data item, and metadata will be broadcasted along with blocks, users can have all the information about the data items in the network. With the information in the metadata, the user can search what it demands, and request the data item from the nodes that store it.

2) Data integrity: Data integrity is one of the crucial security features in the blockchain system. Since the history of blockchain is difficult to counterfeit, data in blocks are also hard to change. However, in our designed blockchain system, the data items are stored over specific nodes, which might be malicious and modify the data as they wish. Thus, we need to add some security features to make sure the data integrity is kept.

Each metadata item has a record containing the account from which generates the corresponding data item and attaches a signature. The signature embeds the identification information of the producer node and later can be validated through the public key of that node. When broadcasting the new block, it spreads the metadata and public keys in its contents. Nodes can then get data and validate the integrity of the data item through the keys. By the nature of the Proof of Stake blockchain, the metadata information in blocks is difficult to change unless malicious parties own more than half of the stakes to replicate a fake chain.

Note that another malicious behavior is to deny storing or offering data to the demanding user. Since data items are stored in certain nodes, some malicious nodes may deny storing or offering. If a node requests data but does not get any response, it then claims that the data is invalid. Everyone will be informed of this information, and this data storage will be marked as invalid. At the same time, there are always replicas for certain data. Unless all replicas of this piece of data are stored at malicious nodes, there will always be available data pieces before the data expires.

C. Resource Allocation for Optimal Storage

As previously mentioned, the storage in edge devices is too small to store all data items. Meanwhile, when running over time, the total size of all blocks is also too large for all nodes to store. Edge nodes are often of different models and makes, thus varying in resources among all nodes. Thus, data items and blocks should only be stored on nodes with more resources left and can be accessed easily, instead of storing them everywhere.

In order to achieve fair and efficient resource allocation among edge devices, we propose an optimal storage problem finding which node stores which data item or block. Each data item or block is assigned to multiple nodes, and these nodes then proactively cache the corresponding data item or block in case other nodes need it. The optimal assignment can find nodes both near the data demands and have more resources to store data proactively.

Besides, the mobility of nodes in edge environments may cause disconnections and data loss. If a node connection is not stable, it may not get recent blocks. Branches are likely to appear in this situation. A node can detect if it misses some blocks by receiving a blockchain longer than its previous received blockchain (recent blockchain), i.e., receiving a block with index number larger than the index of the recent block plus 1[1]. The node then requests the missing blocks from other nodes. Intuitively, every block is assigned to be stored on multiple nodes, so the node with missing blocks will always get blocks back. However, the disconnection problem caused by mobility is pervasive in edge environments, and recent blocks are requested frequently. Thus, another assignment for caching recent blocks among nodes storage is needed. The more pervasive recent blocks are found, the less time and overhead are used for nodes to get them. The optimal assignment on recent blocks will find how to enlarge the pervasiveness of recent blocks as well as keep the fairness of different nodes.

In summary, there are two types of resource allocation problems. First, how we find the best place to store data items and blocks proactively so that data can be quickly accessed. We call this "data and blocks storage allocation". Second, how we cache the recent blocks in the network so that the node can get the missing blocks with less overhead. We call this "recent block allocation".

D. Proof of Stake Forging Consensus

The computing-intensive Proof of Work (PoW) mining requires powerful and expensive hardware setup, or even assembly of these setups, known as mining pools, in order to produce a statistically significant outcome. The major disadvantage of PoW is that it is of high energy consumption. The total amount of energy consumed per year for bitcoin mining is 73 TWh (7.3×10^{10}kWh)[2]. Thus, the proof of work

[1]Receiving block index number equals recent block plus 1 just means there is a new block. It does not necessarily indicate the disconnection. If the received block is 2 or larger than the index of the recent block, there must be blocks that are missing for this node.

[2]Data obtained from https://digiconomist.net/bitcoin-energy-consumption in August 2019.

is impractical for edge devices, as the computing power and energy in edge devices are often very limited.

To achieve consensus with mobile devices, we use the PoS concept and develop a new mechanism having more flexible variations on the stake. Specifically, our proposed PoS mechanism gives advantages to nodes that have more contributions to the system, like storing data and forging blocks. In terms of implementation, we set a value (called the target value) for each node which corresponds to the number of data items and blocks that a node stores, and the number of tokens a node has. The larger the value is, the higher the probability will be for the node to mine a block.

IV. Resource Allocation

In this section, we discuss the resource allocation in the peer edge environments. We first discuss the storage allocation in general and discuss two different situations in applying these allocations. We then discuss the data and blocks accessing process for nodes.

A. Fair and Efficient Storage Allocation

Since edge environments cannot store all data in every node, data and blocks must be stored in certain places that can be easily accessed by demanding users. Meanwhile, the heterogeneity of edge devices also brings problems about fair storage for nodes that have different capabilities.

1) Fair storage: Our previous work [27] propose the concept of fair caching in peer edge environments. Each node has a different capacity, and the algorithm should store fewer data on nodes with fewer resources. Thus, Fairness Degree Cost (FDC) is proposed, which is a measurement for the current resource consumption of a node. The FDC for node i is denoted as

$$f_i = \frac{W(i)}{W_{\text{tol}}(i) - W(i)}, \tag{1}$$

where $W_{\text{tol}}(i)$ is the total storage of node i, and $W(i)$ is the storage used. This FDC definition ensures that the less remaining storage corresponds to the larger value of f_i, thus less possible for storing data in the node. Also, if no remaining resource on the node, (1) will be ∞, and no more data will be stored in this node.

2) Data accessing cost: The mobility of nodes and wireless signal attenuation might cause data loss in the network. Meanwhile, the mobility of nodes also moves stored data around, making the predictions on storage less accurate. To address the problems mentioned above, we propose a new data accessing cost definition, called Range-Distance Cost (RDC), to measure the transmission latency between two nodes. The cost is formulated as

$$c_{ij} = \begin{cases} d(i,j) + range(i) + range(j) & i \neq j \\ 0 & i = j \end{cases}, \tag{2}$$

where $d(i,j)$ is the "distance" between two nodes, and $range(i)$ represents the mobility range of node i. The "distance" is a general term, which can be Euclidean, Manhattan, hop-count distance, etc. The form should be chosen based on the application scenarios. For a scenario that all nodes can connect each other, Euclidean-logarithm distance can represent the signal attenuation; for scenarios that nodes form a multi-hop network, like in many mobile edge environments, hop-count distance, and related variations can represent the data transmission delay through the multi-hop forwarding. For simplicity, we consider using the hop-count distance to measure the RDC in this paper.

In addition, the range of a node represents the maximum replacement trend of a node. The larger the range is, the less steady for a node will be at a certain place, and the mobility caused data transmission delay will be relatively larger. For simplicity, we evaluate the range by the maximum distance of a node from its original location. Note that the range of each node, in reality, will change. Nodes moving outside the original range and new coming nodes will broadcast its new moving ranges to all nodes. This causes topology changes, in which case data may be migrated to fit the new network topology. In our peer edge device environments, we consider that nodes move within such a range in a short period of time. If nodes move above the range, data migration is needed, which we discuss in Section IV-E.

3) Problem formulation: We now propose the formulation for the placement problem in our scenario. We formulate the problem of fair and efficient storage as a facility location problem. The basic idea is to add the FDC and the RDC together in a weighted sum form. After some tests, we use feature scaling to set the weight of FDC and RDC as $1000:1$, which produces the best result. For each data item k and node set $\mathcal{V}$, we formulate the problem in the following:

$$\min \quad A\sum_{i\in\mathcal{V}} f_i y_{ik} + \sum_{i\in\mathcal{V}}\sum_{j\in\mathcal{V}} c_{ij}x_{ijk} \tag{3}$$

$$\text{s.t.} \quad \sum_{i\in\mathcal{V}} x_{ijk} \geqslant 1, (\forall j \in \mathcal{V}) \tag{4}$$

$$y_{ik} - x_{ijk} \geqslant 0, (\forall i,j \in \mathcal{V}) \tag{5}$$

$$x_{ijk}, y_{ik} \in \{0,1\}. \tag{6}$$

In the formulation above, x_{ijk} and y_{ik} are assignment variables. x_{ijk} is the accessing assignment variable. If $x_{ijk} = 1$, node j will access data item k from node i. y_{ik} is the storage assignment variable. $y_{ik} = 1$ means data chunk k will be stored in node i. A is the scaling factor for FDC. After some test, we set $A = 1000$ for better performance. Problem objective function (3) has two terms corresponding to FDC and RDC. Constraint (4) makes sure that at least one data item will be stored to other nodes, and constraint (5) makes sure the data item is stored at specific nodes.

For each data item k, the formulation is an uncapacitated facility location problem. In the problem formulation, f_i corresponds to the facility building cost, and c_{ij} corresponds to the facility accessing cost. The uncapacitated facility location problem is NP-Hard. However, there are many approximation algorithms proposed to solve this question with high efficiency like [25]. For each data item, we use the current network situations (storage used of each node) to solve the problem to determine which nodes store it [27].

B. Data and Block Storage Allocation

Next, we discuss the process of how data items and blocks are stored in selected nodes.

As previously mentioned, when a data item is generated, the producer of the data item also generates the corresponding metadata item and broadcasts it. Each node that receives the metadata item calculates which set of nodes will store the data item. When a node forge the next block (the forging process is discussed in Section V), the node will pack all received metadata items, along with the storing node information, into the block. The block will then be broadcasted over the network. Other nodes in the network will receive the block and check this information. If a node is chosen to be a storing node, it gets the data from the producer and stores them.

When running over time, the blockchain itself becomes a relatively large data structure for all nodes to store all blocks. Blocks also need to be stored among a portion of all nodes. Each new block will be assigned to store on some nodes. This block also has the information which nodes need to store it. Then, corresponding nodes receive this information will keep this block in their storage. The information of the block also contains where the previous block is stored, so that demanding users can obtain the chain starting from the newest block.

C. Recent Block Storage Allocation

Mobility is one of the key characteristics of edge devices. The mobility of nodes might cause unstable connections, which causes data loss. Thus, recent blocks of the blockchain are the most needed for the potential temporary disconnection of nodes. If the recent blocks are more pervasive in the network, retrieving recent blocks will become easier.

Different from data and block storages, nodes are required to cache a certain number of most recent blocks and replace the blocks using FIFO. To start with, all nodes store at least the last block for forging purposes. The node that finds the next block also calculates nodes which need to store one more recent block. The nodes are chosen by solving the same problem, i.e., the fair and efficient storage problem considering the current situations of the network. The chosen nodes will then get the same incentive as the nodes that store a data item or a block.

D. Data Items and Blocks Access Process

Fig. 3 shows an example of nodes accessing blocks and data items in the network. For a node that needs a certain data item (Node G in the example), it first checks the metadata item in the blocks and fetches the data item from nodes that store it. The requesting node sends the data request information to one of the caching nodes (Node C in the example), and this node then sends the data back. If needed, the node can verify the data using the public key and the signature in the corresponding metadata item.

For a node that accidentally disconnects from the network and needs recent blocks (node A in the example), it can first get blocks from any neighbor nodes. (We call nodes within one hop transmission range of a specific node as the neighbors of this node. Each node maintains a neighbor list. When there

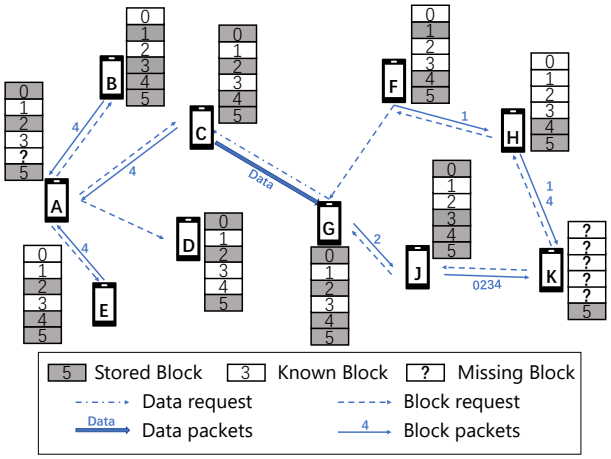


Fig. 3. The data and block access process. Node A reconnects to the network with a missing block. Node G is requesting a data piece. Node K is a new node entering the network.

is a new connection from a new node, it will be marked as one of the neighbors.) For a disconnected node, it only needs recent blocks. For example, node A receives block 5 and finds out it misses block 4. It will broadcast the request message for the missing block and the message is only transmitted in one hop. Its neighbors, in this example, nodes B, C, D, and E will receive this request message and send back corresponding blocks. Since many nodes store recent blocks, the missing blocks can be easily obtained within a few hops, even without receiving from all neighbors.

For a node that needs the whole blockchain (e.g., new node coming into the network, as Node K in the example), it first requests for blocks and then organizes the received blocks to get the missing blocks. The node then sends new requests for the missing blocks until it gets all blocks. Since a block stores the information about storing nodes for the previous block, a node can recursively request the missing blocks. For example, Node K joins the network and finds out it needs blocks 0 to 4 when it receives block 5. It sends the request and its neighbor J and H will receive the request. Node J and H can satisfy some block but not all. They then request the missing block 1 from its neighbor Node F and block 2 from node G. Finally, the missing blocks will be sent to Node K. (Note that Node F will forward the request to Node G for block 2 since it does not know block 2 is satisfied already. Meanwhile, Node G can ignore the request since it already sends block 2 out.)

E. Data Migration

So far, we have discussed the placement when nodes are moving in a small range and the network topology remains the same. In edge environments, however, mobility is one of the crucial characteristics. Over time, nodes will change locations, carrying data items stored on them, and the information for data accessing will also become non-optimal or obsolete. As more and more nodes move and change the topology of the network, the storage decisions must be adapted continuously. Thus, we propose two online algorithms for data migration to

calculate where the data need to be stored when the topology changes.

1) Optimal migration algoritm: We first propose an online algorithm to migrate stored items to restore the optimality (optimum of the same approximation ratio). The algorithm we proposed is shown in Algorithm 1. It relocates the stored data where the cost of data accessing increases and finds a better place to store data with the lowest cost of accessing. When the topology and the cost of data accessing change, we have the difference between the old and new costs. We denote the difference as $\Delta c_{ij} = c'_{ij} - c_{ij}$, where c'_{ij} is the new cost between node i and j. For the cost that increases for accessing data item k, we collectively search if there exists another node that has a lower accessing cost for all potential requests. If we find the node that already stores the data item k, we only need to compare the RDC with the cost before migration. If the node does not store the data item, we will compare the total new cost (RDC + FDC) with the cost before migration. We then select the node with the lowest cost to store the data item, and remove the data item on the original node if there are no more access requests from that node.

Note that if node i caches multiple data items, these data items will be considered individually to ensure that each item matches the optimality conditions. Thus, we omit the data item subscription k for simplicity as the algorithm needs to run for each data item. Node set K indicates the corresponding nodes that store the data item k.

Algorithm 1 Data Storage Migration Algorithm

Input: Accessing cost matrix $C = \{c_{ij}\}$, node set K which stored data k, moving node i

Output: New accessing variable x'_{ij}, new storage variable y'_i

1: Recalculate C
2: **for all** j **when** $x_{ij} = 1$ **and** $\Delta c_{ij} > 0$ **do**
3: **for all** $i' \in \mathcal{V} \backslash i$ **do**
4: **if** $i' \in K$ **and** $c_{ij} + f_i > c_{i'j}$ **then**
5: $x'_{i'j} \leftarrow 1$, $x'_{ij} \leftarrow 0$
6: **else if** $i' \notin K$ **and** $c_{ij} + f_i > c_{i'j} + f_{i'}$ **then**
7: $x'_{i'j} \leftarrow 1$, $y'_{i'} \leftarrow 1$, $x'_{ij} \leftarrow 0$
8: **end if**
9: **end for**
10: **end for**
11: **if** $\sum_j x'_{ij} = 0$ **then**
12: $y'_i \leftarrow 0$
13: **end if**

The complexity of the algorithm is $\mathcal{O}(nl)$, where n is the number of total nodes and l is the number of affected links. The upper limit of l is n^2, but in reality, l is much smaller than n^2, since usually most links are not affected.

We now prove Algorithm 1 can preserve the optimality of the algorithm if the originally given result is optimal.

Lemma 1. *The proposed Algorithm 1 satisfies the constraints (4)-(6) of problem (3).*

Proof. If $i' \in K$, then node i' has already stored the data k. Thus, $y_{i'} \geqslant x_{i'j}$ and it can offer the data to node j. If $i' \notin K$, we need to put data k on node i' first, and we let $y'_{i'} = 1$ to satisfy the (5) (in line 7). Then node i' can offer data to j thus satisfying (4). If no further node request data item k from node i, we let $y'_i = 0$ to avoid additional cost. Node i then can delete the corresponding cached data item. □

Theorem 1. *Algorithm 1 is the optimal solution in the new condition if the solution for the previous condition is optimal.*

Proof. We consider the following conditions and their corresponding store strategies.

1) For two nodes i and j, if c_{ij} changes to c'_{ij}, and $\Delta c_{ij} \leqslant 0$, then c'_{ij} is the optimal solution. Since the original solution is optimal, i.e., $\forall i' \in \mathcal{V} \backslash i$, $c_{i'j} \geqslant c_{ij} \geqslant c'_{ij}$. Thus, c'_{ij} and corresponding x'_{ij} will remain optimal. In such conditions, y_i remains the same for all node i, since the parameters and the optimal value from FDC are not changed.
2) For two nodes i and j, if c_{ij} changes to c'_{ij}, and $\Delta c_{ij} > 0$, it is possible that $\exists i' \in \mathcal{V} \backslash i$, $c_{i'j} \leqslant c'_{ij}$. Thus, for node $j \in \mathcal{V}$, $x_{ij} = 1$ and $\Delta c_{ij} > 0$, we find the minimal value of any possible new solution. As described in Algorithm 1, if $\exists i'$ and
$$c_{ij} + f_i > c_{i'j} + f_{i'}, \tag{7}$$
then, for a specific j, using node i' to deliver data to j will have less cost. Meanwhile, Algorithm 1 will find node i' with the lowest cost to deliver data to j. Thus, we obtain the lowest cost since $c_{i'j} + f_{i'} < c_{i''j} + f_{i''} (\forall i'' \in \mathcal{V} \backslash i')$. Note that if $i' \in \mathcal{K}$, which indicates node i' has already cached data item k, $f_{i'}$ in (7) is not needed, since $f_{i'}$ is already added in the objective function (3).

□

The original problem is NP-Hard. Thus, it is hard to obtain the optimum value in polynomial time. In the migration process, the original assignment is already obtained, and adjustment is needed to reduce the cost caused by topology change. Under such conditions, only c_{ij} changes in (3). If the previous optimal result is obtained, we do not need to solve the problem from the starting point of the original problem. Algorithm 1 can obtain the optimal under such conditions has a lower computational complexity of $\mathcal{O}(nl)$. This does not show the entire problem is not NP-Hard, but rather the migration process part, when given the result of the original problem, is not NP-Hard.

We now introduce and prove the corollary that the approximation ratio can be preserved following Algorithm 1 given Theorem 1.

Corollary 1. *Algorithm 1 has an approximation ratio k if the solution for the previous condition has an approximation ratio k.*

Proof. From Theorem 1, for the first condition, the original cost remains, which will not change the approximation ratio. In the second condition, since $\Delta c_{ij} > 0$, if there exists node i' that satisfy equation (7), $c_{i'j} + f_{i'} < c_{ij} + f_i (\forall j)$. Since $\sum_{i \in K} f_i + \sum_{i \in K} \sum_j c_{ij} \leqslant m \cdot OPT$ (m is the approximation ratio), we have $\sum_{i \in K \backslash i'} f_i + f'_i \sum_{i \in K \backslash i'} \sum_j c_{ij} + c_{i'j} <$

$m \cdot OPT$, which indicates that the approximation ratio is preserved. □

Note that if there are new nodes coming into the network or nodes leaving the network, the dimension of FDC matrix $F = f_{ik}$ changes. Thus, this algorithm cannot get the optimal result. The original optimal problem and solutions are needed if the number of participating nodes is changed.

2) Heuristic migration algoritm: Algorithm 1 can preserve the approximation ratio when nodes move and change the network topology. However, the algorithm has a high computational complexity when running on multiple nodes. Therefore, we also propose a heuristic algorithm that can run faster and achieve a reasonably good result.

Instead of using multiple nodes to calculate all pairs of costs, we use the information easiest to obtain to reduce the costs. The detailed design is shown in Algorithm 2. When a node moves thus changing the topology, we get the cost change for data accessing between two nodes. For the cost that increases, the node that demands data item k compares three different cases (accesses from the original storage node, stores the data itself, or accesses from another stored node which has less cost) and chooses the option with the lowest total cost. This will decrease or at least remain the current cost, which can improve the performance.

Algorithm 2 Data Storage Migration Heuristic Algorithm

Input: Accessing cost matrix $C = \{c_{ij}\}$, node set K which stored data k, moving node i
Output: New accessing variable x'_{ij}, new storage variable y'_i

1: Recalculate C
2: **for all** j **when** $x_{ij} = 1$ **and** $\Delta c_{ij} > 0$ **do**
3: Find $\min_{i'} c_{i'j}$ **where** $i' \in K$
4: **if** $f_j < c_{ij}$ **then**
5: $x'_{jj} \leftarrow 1$, $x'_{ij} \leftarrow 0$
6: **else if** $c_{ij} < c_{i'j}$ **then**
7: $x'_{i'j} \leftarrow 1$, $x'_{ij} \leftarrow 0$
8: **end if**
9: **end for**
10: **if** $\sum_j x'_{ij} = 0$ **then**
11: $y'_i \leftarrow 0$
12: **end if**

In Algorithm 2, the determination of which data item to migrate is the same as in Algorithm 1. The difference is to determine where the data item will be migrated. In Algorithm 1, it finds the best location to store data items, which incurs more complexity. On the contrary, Algorithm 2 only needs to find a better location than the original ones. Lines 3-8 find the existing location with the least cost (RDC) to migrate to, and compare with the FDC with the demanding node. If the smallest RDC is smaller, the data will migrate to the previously stored node; otherwise, the demanding node will store the data. This process only considers one of the two aspects of costs which reduces the complexity and can obtain a smaller overall cost. The complexity of Algorithm 2 is $\mathcal{O}(n)$, where n is the number of the affected nodes. The algorithm has a much less computational complexity compared with Algorithm 1, but it has no guarantee over the optimality. The system can choose which algorithm to apply based on the requirements and resources of the nodes participating in the system.

Note that multiple nodes moving at the same time can also be solved. This can be achieved by applying the algorithm multiple times.

3) Data migration update: So far we have discussed the migration algorithms that can get which data items are needed to migrate to which nodes. When the change of RDC is larger than that of migration cost, the data migration process will be triggered. Participating nodes will use the algorithms mention before and obtain the corresponding data items and caching nodes. The node which is granted the privilege for the next block will append the migration information, as the new storing nodes to the corresponding data item. Upon receiving the data migration instruction from the new block, the corresponding new caching nodes will request the data item from the previous caching node. Note that the information of migration is marked as a caching update. Once the update is published, the old caching information is automatically outdated. Nodes can update the information for its own use without changing data already in previous blocks.

V. Proof of Stake Mechanism

In the Proof of Stake (PoS) algorithm, the creation of new blocks is called forging. Forging is a process that selects a node based on a certain consensus and granted the privilege to generate the next block. A node will have a higher chance to forge a new block if it contributes more in the edge scenarios. The goal of our proposed PoS mechanism is to make sure that, if a node has more tokens and stores more data and blocks, the node will have more advantages to forge blocks. We make our design based on Algorand [28], which is a relatively safe PoS consensus.

A. Goals and Assumptions

1) Security goal: We target three security goals. First, transactions in a block, including money transfers and storage decisions (described in Section V-B), need to be protected from modifying once the block is proposed and accepted. Second, when misbehavior is detected, the privileges and advantages of the corresponding node should be revoked without modifying transaction. Third, a node that participate in the network should not get advantage by creating dummy accounts.

2) Assumptions: We assume that a node does not require a pre-clearance or permission to join the network. Participating nodes can be adversary parities since no check is conducted. We assume that all parties are selfish and rational, thus, adversary parities will behave maliciously if it cannot get unfair advantages. For the storage decisions for data items and blocks, we assume that all nodes with enough information about the network will get the same result following a same algorithm. If the decision in a new block does not match the result a node obtained, the node can refuse to relay or vote against the block.

B. Reward for Edge Blockchain

In peer edge scenarios, node participation is crucial for maintaining a blockchain. A node can participate by getting a coin from an existing node and sharing its storage capacities to other nodes. In the Algorand-based system, a block proposal is accepted when a committee checks the block is valid and vote for this block. If the positive votes reach some threshold, the proposing node will be granted the privilege to create a block and write to the blockchain. In our design, if a node contributes more, i.e., stores more data and forges more blocks, they will have a higher chance to forge new blocks and become committee members to vote for blocks.

To achieve this, we create a new kind of asset, storage token, that can be used for committee selecting and voting process, together with traditional coin assets (called Algos in Algorand). The storage token is given to the node that stores data items of blocks. The storage token can be used in the sortition process. The sortition process chooses how many units of assets can be used for the proposal of blocks or the committee selecting in a certain round. In the sortition process, a user uses a verifiable random function to get a hash and proof. If the hash of a user falls in a range, a certain unit of assets is selected, and the user is ranked by the number of selected units of assets. The proof can be used to validate the hash publicly. Both storage tokens and coins (like Algos) can be assets in the sortition process. Thus, more storage tokens and more coins will have a higher ranking in the sortition, and owners will have a higher probability to be the block proposer or the committee member in a certain round.

The storage token is given to nodes via a special type of transaction, which we call storage transaction. This storage transaction records information including the data item/block to be stored on a specific node, number/value of tokens, senders, receivers, and other necessary transaction items. For each data item and each block, there are several storage transactions associated. The block proposer is responsible to calculate the storage allocation for data items and the block and make these transactions. The sender of these storage transactions is the block proposer, and the receivers are the nodes that will store the data item or the block. When the block is proposed, the proposer needs to include these storage transactions in the block. When the block is later included in the blockchain, these transactions will become effective. The corresponding nodes will get storage tokens, and need to get data from the data producer and store them. Note that the block proposer does not consume any coins or tokens to create these storage transactions. It just needs to sign the transactions to make sure it can be validated and traced whenever needed.

Both storage tokens and coins of a user can be used in the sortition process equally, but there are two fundamental differences between them. First, the storage tokens come from storage transactions. These transactions are not in the traditional transaction pool and there are no transaction fees for these transactions. The transactions are made and signed by the block proposer. If the storage assignment cannot be agreed upon, the block will not pass the soft or certify vote process. Second, storage tokens are not transferable. Unlike coins which can be transferred through payment transactions, nodes cannot transfer their storage tokens to other nodes. However, the storage tokens can be marked invalid if data items are migrated or the node violates its storage commitment. If the data item needs to be stored at another node, new storage transactions and storage tokens are required.

C. Storage Validations

As we discussed in Section III-B2, when a node is assigned to store a data item or a block, a possible malicious behavior is the node does not store the data item. When a consumer needs a data item from this node, it is unable to service the consumer. Thus, the node must follow a certain process to minimize the impact of such malicious behavior.

When a node receives the storage transaction and storage tokens, it needs to proactively get data from the producer. Then, to ensure the data is received and not corrupted, the author must get the hash of the file and sign the hash value. Then, the signed hash value is presented for the data producer to verify. If the producer finds some incorrect hash, the file needs to be sent again. Presenting the hash of the cached data item means that a node acknowledges the storage assignment and can use the corresponding storage tokens in the blockchain. The node commits to store the data until data migration or expiration happens. If the data item is migrated or expired, the corresponding storage tokens also become invalid.

If the data is not migrated or expired, and one consumer cannot receive the file, the consumer will mark the data on the node invalid, and mark the node misbehaving. Misbehaving nodes will be punished by having their corresponding storage tokens invalidated. They may also be banned from future storage assignment and block proposal. However, file delivering failure may be caused by network disconnection. When this happens, the producer can act as an arbitrator. The producer will ask the storing node to give a signed hash about a certain part of the file. If the node can offer correct information, the punishment of the node will not take effect. The producer will tend to be honest in arbitration since it cannot get unfair advantages for lying.

For more complex situations, a reputation system is needed to evaluate how honest a node is. The reputation of a node will decrease if it violates the consensus or commitment. A low reputation will have a specific negative impact on block proposal, sortition rank, and voting power of a node. We will discuss this approach in our future work.

D. Discussion on Proof of Stake

As a replacement for PoW algorithms, PoS can have different forms and different metrics. We present a PoS mechanism for edge environments considering the heterogeneity of different devices. The mechanism gives the advantages to the node which has contributions to the network. The contribution is crucial to maintain the data and block storage in peer edge environments. Thus, we give relatively more advantages for the node that contributes more.

Despite much less energy is required for PoS, it also raises a problem due to the low complexity. In PoS consensus, working

on different chains needs less computation. A node might work on different branches to get more profit. This is often called the nothing-as-skate attack. Also, a node can make a fake fork from a specific point longer than the current chain in order to benefit itself, which is called a history attack. Solutions about inserting checkpoint blocks [29], [30] are proposed to force nodes working on the chain that has checkpoint blocks, which can mitigate the effects of attacks mentioned above. In addition, Algorand also requires nodes to work on the current block reaching "final consensus", which can also help mitigate the number of branches.

As we mentioned before, a node should not get unfair advantages by forging accounts. This attack is often called Sybil attack. Since there is no central authority, a node can create multiple different accounts. If each account gets some initial resources, forging a lot of fake accounts will get unfair advantages. In our design, an account must get at least one coin from other nodes to participate. If a node forges multiple accounts, it should get multiple coins which gives more burdens on the node, which prevents it to get unfair advantages.

VI. Performance Evaluation

In this section, we evaluate our proposed blockchain system. We focus on evaluating the performance of our proposed strategies over different sizes of networks and the power efficiency of forging consensus strategies.

We conduct simulations and experiments on the blockchain system, data placement, the PoS strategy, and data migration. We implement the simulation using the Python-based SDK offered by Algorand. We use the private network in Algorand as blockchain bases, and implement new transaction types. The official Algorand uses HTTP services to communicate information between nodes. We extend the functionality of the client in Algorand by adding functions HTTP services to support requesting, forwarding, and file-delivering. The client can listen on several ports that serve as the blockchain service, relay service, and file transfer service in the blockchain system. Then, we simulate data migration using different mature mobility patterns. We set the maximum range of a node is 7m. If a node exceeded this range, the node is regarded as moving to a new place. Since changing the connection actively over different containers is difficult, we do not use the Algorand network to test the data migration due to node mobility. Instead, we implement the algorithm with Python and use some mature mobility patterns to test the data migration. Thus, we only simulate mobility and storage. The connection and PoS are not considered in the simulation. Finally, we also implement PoS forging and PoW mining consensus on a real tablet to test the power consumption. The experiments over the blockchain system and simulations on data migration are conducted on a computer with an Intel Core i7-5820K processor and 32GB RAM. The power consumption of two different consensus protocols is conducted on a Microsoft Surface Pro 6 with i5-8250U, 8GB RAM, and a 5940mAh battery.

We assume up to 80 nodes are distributed in an area of 300m $\times$ 300m, and the communication range between two nodes are 70m. The area and number of nodes are typical application scenarios for a pico cell 5G small cell network [31]. Each node has the capability to store 250 data items or blocks. Unless otherwise specified, we set the expected average time for forging a block as 60 seconds, and each simulation runs over 500 minutes. In the Algroand system, every node needs to connect to relay nodes. Thus, we use the greedy vertex-cover solution to find such nodes and set them as relay nodes in the network.

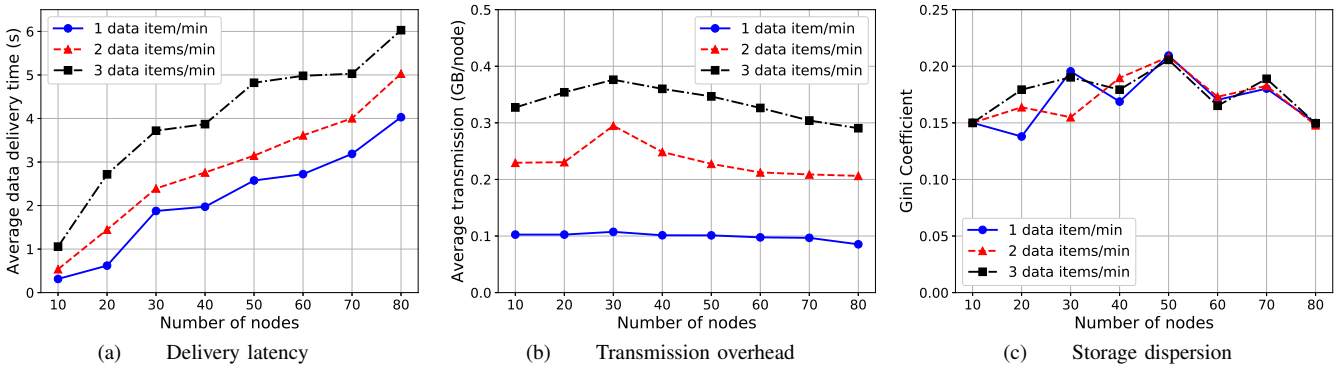
A. Performance under Different Data Amounts

First, we evaluate the overall performance of the proposed blockchain system under different total data amounts. We test it under different numbers of nodes in the network, varying from 10 to 80, and on average 1 to 3 data items are generated throughout the network per minute. We focus on a private blockchain system, in which the number of nodes may be limited compared to a public blockchain system. 80 users in a private blockchain are sufficient to maintain the blockchain system. The data items are requested randomly by 10 percent of nodes. The size of each data item is 1MB. Since the nodes use the socket to transmit data, processing, queuing and transmission delays can be obtained through Docker. We record all data and block transmissions to show the overhead for transmission and the data delivery time. We also use the Gini coefficient [3] to measure the fairness of the proposed system.

Fig. 4 shows the overall performance over different numbers of nodes and different data amounts. Fig. 4(a) shows that the node can get the data in a small amount of time. The time increases with more nodes and more data in the network since more hops will use more time for data delivery. Nevertheless, about 6 seconds in maximum are used for a node to get the desired data. The results show that our proposed system can achieve fair storage and low data delivery latency.

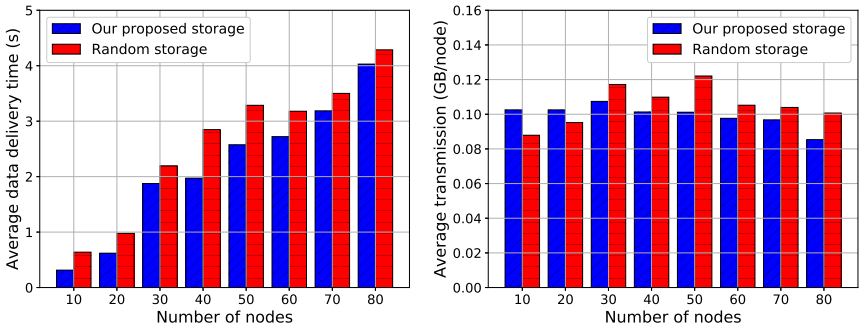
Fig. 4(b) indicates the transmission overhead for the blockchain system. It includes data requests, data dissemination (storing node proactive get data from the producer), and other blockchain broadcasting packets. Overall, the transmission overhead is about two to six times of transmissions, 500MB to 1.5 GB data are generated in over 8 hours, yet the total transmission is less than 20GB. On one hand, more data mean more transmission in the network, which increases the transmission overhead. On the other hand, if the amount of data is the same, having more nodes indicates less transmission is needed per node, which decreases the average overhead. Note that there is a reverse trend between the number of nodes from 30 to 40. This is because as the node increases, the number of neighbors (in one hop transmission range) of a node will also increase. In an ideal case, the average distance from the second closest node is 76.4m when the number of nodes is 30, and it decreases to 66.5m when the number is 40, which is smaller than the communication range. This change increases the connections between nodes in the network, and

[3]Gini coefficient is widely used to depict income disparity and is also used in previous works to measure fair caching [27], [32]. $Gini = \frac{\sum_i \sum_j |t_i - t_j|}{2 \sum_i \sum_j t_j}$.



(a) Delivery latency (b) Transmission overhead (c) Storage dispersion

Fig. 4. The average data delivery time (a), average transmission of each nodes (b), and Gini coefficient (c) under different number of nodes and different data amount. Our proposed algorithm has low transmission overhead and fast data access time, and the storage distribution is fair among nodes.



(a) Delivery latency (b) Transmission overhead

Fig. 5. The average data delivery time from nodes (a) and the average data transmission overhead (b) under different number of nodes and two different data placement strategies. The optimal data placement receives the fast data delivery time and similar transmission overhead compared with no proactive store solution.

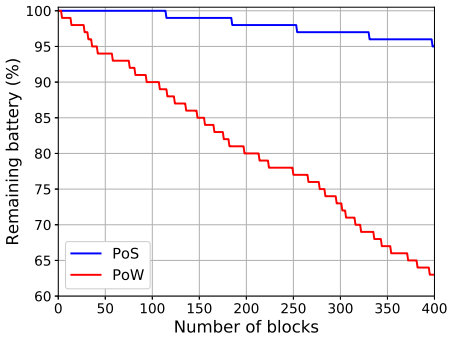


Fig. 6. The remaining battery in forging and mining processes under different consensus algorithm. The PoW consumes much more energy than PoS.

greatly increasing the number of neighbors of a node. Thus, the reverse trend happens when the number of nodes is larger than 40, and the average transmission amount is decreasing.

Meanwhile, the Gini coefficient shown in Fig. 4(c) for all tests is below 0.2, which means the storage disparity is relatively low. Since we set the storage capacity of all nodes the same in the simulation, this shows that the dissemination storage is fair among all nodes.

B. Performance under Different Placement Strategy

Next, we evaluate the performance of the data storage placement strategy we proposed. The system proactively stores data on nodes that can offer quick access for all users that demand data. We compare the proposed strategy with random storage solutions. For a fair comparison, the total number of data and blocks stored is the same for both placement strategies. We test the performance under different numbers of nodes and at a data generation rate of 1 item per minute.

Fig. 5 shows the average data delivery time (a) and average data transmission overhead (b) for different number nodes in the system. Our proposed optimal data placement saves much more time on data access compared with random placement and uses less than 18.4% of the time used to access data. Meanwhile, the transmission overhead is almost the same between two different strategies, showing that our proposed algorithm does not cost much extra overhead. The results show that the proposed optimal caching strategy achieves less time for data access while keeping the transmission overhead almost the same in the proposed blockchain system.

C. Performance under Different Consensus Algorithms

To test the energy consumption of different consensus mechanisms in real edge environments, we implement the Proof of Work (PoW) and Proof of Stake (PoS) consensus mechanism on a Surface Pro 6 and use Windows subsystem for Linux (WSL) 2 [33] as the experiment environment. We implement the PoS consensus mechanism based on the private network environment offered by Algorand, and PoW uses the most commonly implemented hash (SHA256) which requires a certain number of zeros in the beginning. We set the difficulty of PoW as 5 zeros at the beginning of the block hash. The average mining time for each block is 5 seconds at this difficulty. For a fair comparison, the default time for a block in Algorand networks is about 5 seconds. The storage and connection are not implemented since we only test on one node. The battery consumption for each node with similar computational power is mostly the same. Thus, the performance of one node can represent the battery consumption in general. We make sure that the tablet is fully charged before forging or mining, and test the battery consumption of different strategies. When a

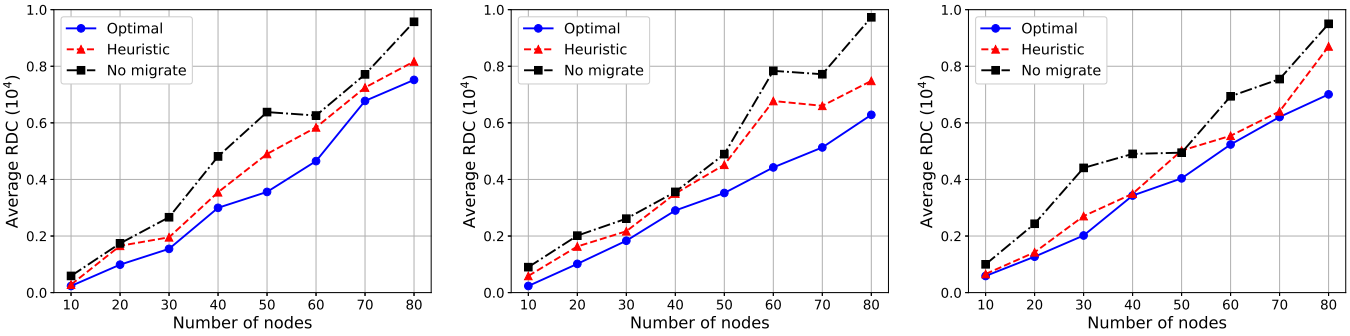


Fig. 7. The average RDC for each data transmitted in the network under different mobility models. We test random way-point (a), truncated Levy walk (b), and time-variant community walk (c) under different numbers of nodes and different algorithms. The proposed optimal and heuristic algorithm can reduce the average RDC for data migration caused by node mobilities.

block is mined or forged, we record the remaining battery of the tablet. To reduce the affections from other factors, background processes are closed if possible and airplane mode is turned on.

Fig. 6 shows the remaining battery with the number of blocks mined or forged. The PoW consensus consumes about 40% of the battery over 62 minutes, while the PoS consensus only consumes 5% of the battery. On average, 10.5 blocks consume about 1% battery of the phone using PoW, while 80 blocks consume 1% battery of the phone using PoS. The PoS consensus can obtain more blocks using the same amount of energy, which indicates less energy consumption. The difference will be even larger if the difficulty of the PoW increases. The computational complexity grows exponentially with the complexity of PoW but remains almost the same for PoS. The result shows that the PoS algorithm consumes about 86.8% less energy in terms of battery consumption compared to the traditional PoW algorithm in the mining process.

To evaluate the energy consumption for the approximation algorithm running on edge devices, we implement the simple algorithm in Section 2 from [34] on a smartphone and running the algorithm only without the consensuses. Overall, energy consumption is very limited. With 1% of the battery, the algorithm can execute 161 times, and on average 2.251 seconds for the algorithm to run one time.

D. Performance under Different Mobility Patterns

Last but not least, we evaluate the data migration algorithms under different mobility patterns. We select three different mobility patterns, random way-point, truncated Levy walk [35] and time-variant community walk [36], and set the speed as human walk speed, which is about 1 to 1.5 meters per second. Since simulating the mobility and connection changes is difficult for Docker containers, we do no use docker here. Instead, we test the average range-distance cost (RDC) defined in Section IV-A of each data delivered to every node. The higher the RDC is, the longer time it will take for data delivery. The simulation stores all data at certain places using the algorithm in [27] initially.

Fig. 7 shows average RDC using different algorithms under different numbers of total nodes. “Optimal” represents Algorithm 1, and “Heuristic” represents Algorithm 2, and “No migration” means data will be stored and do not migrate when nodes move. Under all three mobility patterns, our proposed algorithms reduce the RDC through data migration. Algorithm 1 reduces the RDC by average 57.71% compared with no migration, and heuristic Algorithm 2 reduces by average 20.25%. Note that the heuristic algorithm runs fast but does not have any guarantee on performance. Overall, the simulation shows our proposed algorithm can provide comparable performance under the mobility of nodes.

VII. Conclusion and Future Work

In this paper, we have proposed a blockchain system for edge computing environments. Due to the limitations of edge devices, we have proposed the optimal data and block storage for quick and fair data accessing. We store metadata items on blocks, and corresponding data items and blocks are distributively stored for quick data access. We have proposed the recent block storage for quick access for nodes that miss blocks. We have proposed a data migration scheme under the circumstance of network topology change due to mobility. We have also developed a new Proof-of-Stake mechanism in such edge environments, considering the past contribution a node to the system to give corresponding advantages over forging. Simulations show that our blockchain system works well under pervasive edge environments with limited resources.

Over time, the recent blocks storage will accumulate and occupy more and more spaces. The expiration for data is needed to avoid using up storages. We will address in future work about data expiration when running continuously. The detection of not storing data items assigned also brings challenges to the system if nodes can intentionally misbehave. Previous work offers solutions to verify the integrity of data storage, such as PDP [37] and PoSt [38]. A reputation system can also be used in this situation to evaluate the honesty of a node and can punish it if it does not follow the consensus or breaks its commitments. We will discuss how to implement and enforce some related work about data storage integrity and the reputation system in peer edge environments in our future work.

ACKNOWLEDGMENTS

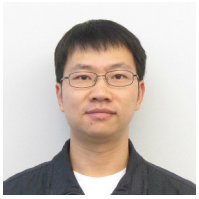
This work is supported in part by US National Science Foundation under grant numbers CNS-1513719 and CNS-1730291.

REFERENCES

[1] Nokia, "Sensing as a service," https://www.nokia.com/networks/services/sensing-as-a-service/, 2018.
[2] A. Zaslavsky, C. Perera, and D. Georgakopoulos, "Sensing as a service and big data," *arXiv preprint arXiv:1301.0159*, 2013.
[3] S. Bowman and C. Willis, "We media," *How audiences are shaping the future of news and information*, 2003.
[4] T. Kelsey, "Fun with e-commerce analytics Part III: Gumroad," in *Introduction to Google Analytics*. Springer, 2017, pp. 107–128.
[5] "Is gumroad a scam?" https://www.quora.com/Is-Gumroad-a-scam, 2016.
[6] S. Andreina, J.-M. Bohli, G. O. Karame, W. Li, and G. A. Marson, "Pots-a secure proof of tee-stake for permissionless blockchains." *IACR Cryptology ePrint Archive*, vol. 2018, p. 1135, 2018.
[7] J. Mattila *et al.*, "The blockchain phenomenon–the disruptive potential of distributed consensus architectures," The Research Institute of the Finnish Economy, Tech. Rep., 2016.
[8] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," 2008.
[9] M. Iansiti and K. R. Lakhani, "The truth about blockchain," *Harvard Business Review*, vol. 95, no. 1, pp. 118–127, 2017.
[10] K. Fanning and D. P. Centers, "Blockchain and its coming impact on financial services," *Journal of Corporate Accounting & Finance*, vol. 27, no. 5, pp. 53–57, 2016.
[11] T. Chen, Y. Zhu, Z. Li, J. Chen, X. Li, X. Luo, X. Lin, and X. Zhang, "Understanding ethereum via graph analysis," in *INFOCOM 2018-IEEE Conference on Computer Communications, IEEE*. IEEE, 2018, pp. 1484–1492.
[12] A. P. Ozisik, G. Andresen, G. Bissias, A. Houmansadr, and B. Levine, "Graphene: A new protocol for block propagation using set reconciliation," in *Data Privacy Management, Cryptocurrencies and Blockchain Technology*. Springer, 2017, pp. 420–428.
[13] I. Eyal, A. E. Gencer, E. G. Sirer, and R. Van Renesse, "Bitcoin-ng: A scalable blockchain protocol." in *NSDI*, 2016, pp. 45–59.
[14] J. Bonneau, A. Miller, J. Clark, A. Narayanan, J. A. Kroll, and E. W. Felten, "Sok: Research perspectives and challenges for bitcoin and cryptocurrencies," in *Security and Privacy (SP), 2015 IEEE Symposium on*. IEEE, 2015, pp. 104–121.
[15] S. King and S. Nadal, "PPCoin: peer-to-peer crypto-currency with proof-of-stake," https://peercoin.net/assets/paper/peercoin-paper.pdf, 2012.
[16] R. P. d. Santos, "Pow, pos, & hybrid protocols: A matter of complexity?" *arXiv preprint arXiv:1805.08674*, 2018.
[17] M. Research, "Edge computing," https://www.microsoft.com/en-us/research/project/edge-computing/, Oct. 2008.
[18] M. Satyanarayanan, V. Bahl, R. Caceres, and N. Davies, "The case for vm-based cloudlets in mobile computing," *IEEE pervasive Computing*, 2009.
[19] X. Chen, L. Jiao, W. Li, and X. Fu, "Efficient multi-user computation offloading for mobile-edge cloud computing," *IEEE/ACM Transactions on Networking*, no. 5, pp. 2795–2808, 2016.
[20] D. Liu, B. Chen, C. Yang, and A. F. Molisch, "Caching at the wireless edge: design aspects, challenges, and future directions," *IEEE Communications Magazine*, vol. 54, no. 9, pp. 22–28, 2016.
[21] P. Garcia Lopez, A. Montresor, D. Epema, A. Datta, T. Higashino, A. Iamnitchi, M. Barcellos, P. Felber, and E. Riviere, "Edge-centric computing: Vision and challenges," *ACM SIGCOMM Computer Communication Review*, vol. 45, no. 5, pp. 37–42, 2015.
[22] G. Cornuéjols, G. L. Nemhauser, and L. A. Wolsey, "The uncapacitated facility location problem," DTIC Document, Tech. Rep., 1983.
[23] A. Gupta, A. Kumar, and T. Roughgarden, "Simpler and better approximation algorithms for network design," in *Proceedings of the thirty-fifth annual ACM symposium on Theory of computing*. ACM, 2003, pp. 365–372.
[24] C. Swamy and A. Kumar, "Primal-dual algorithms for connected facility location problems," in *International Workshop on Approximation Algorithms for Combinatorial Optimization*. Springer, 2002, pp. 256–270.
[25] S. Li, "A 1.488 approximation algorithm for the uncapacitated facility location problem," *Information and Computation*, vol. 222, pp. 45–58, 2013.
[26] X. Song, Y. Huang, Q. Zhou, F. Ye, Y. Yang, and X. Li, "Content centric peer data sharing in pervasive edge computing environments," in *Distributed Computing Systems (ICDCS), 2017 IEEE 37th International Conference on*. IEEE, 2017, pp. 287–297.
[27] Y. Huang, X. Song, F. Ye, Y. Yang, and X. Li, "Fair and efficient caching algorithms and strategies for peer data sharing in pervasive edge computing environments," *IEEE Transactions on Mobile Computing*, 2019.
[28] Y. Gilad, R. Hemo, S. Micali, G. Vlachos, and N. Zeldovich, "Algorand: Scaling byzantine agreements for cryptocurrencies," in *Proceedings of the 26th Symposium on Operating Systems Principles*, 2017, pp. 51–68.
[29] S. Azouvi, G. Danezis, and V. Nikolaenko, "Winkle: Foiling long-range attacks in proof-of-stake systems," in *Proceedings of the 2nd ACM Conference on Advances in Financial Technologies*, 2020, pp. 189–201.
[30] I. A. I. AlMallohi, A. S. M. Alotaibi, R. Alghafees, F. Azam, and Z. S. Khan, "Multivariable based checkpoints to mitigate the long range attack in proof-of-stake based blockchains," in *Proceedings of the 3rd International Conference on High Performance Compilation, Computing and Communications*, 2019, pp. 118–122.
[31] T. Nguyen, "Small cell networks and the evolution of 5G," https://www.qorvo.com/design-hub/blog/small-cell-networks-and-the-evolution-of-5g, 2017.
[32] 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.
[33] P. Singh, "Linux development on wsl," in *Learn Windows Subsystem for Linux*. Springer, 2020, pp. 131–168.
[34] F. A. Chudak and D. B. Shmoys, "Improved approximation algorithms for the uncapacitated facility location problem," *SIAM Journal on Computing*, vol. 33, no. 1, pp. 1–25, 2003.
[35] I. Rhee, M. Shin, S. Hong, K. Lee, S. J. Kim, and S. Chong, "On the levy-walk nature of human mobility," *IEEE/ACM transactions on networking (TON)*, vol. 19, no. 3, pp. 630–643, 2011.
[36] W.-J. Hsu, T. Spyropoulos, K. Psounis, and A. Helmy, "Modeling time-variant user mobility in wireless mobile networks," in *IEEE INFOCOM 2007-26th IEEE International Conference on Computer Communications*. IEEE, 2007, pp. 758–766.
[37] G. Ateniese, R. Burns, R. Curtmola, J. Herring, L. Kissner, Z. Peterson, and D. Song, "Provable data possession at untrusted stores," in *Proceedings of the 14th ACM conference on Computer and communications security*, 2007, pp. 598–609.
[38] G. Ateniese, L. Chen, M. Etemad, and Q. Tang, "Proof of storage-time: Efficiently checking continuous data availability," 2020, pp. 1–15.

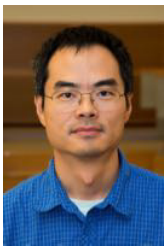
Yaodong Huang received his B.E. in Computer Science and Technology in 2015 from University of Electronic Science and Technology of China. He is now pursuing his Ph.D. degree in Computer Engineering at Stony Brook University. His research interests are in mobile edge computing, with focus on data caching, storage, and blockchain technology over edge environments.

Jiarui Zhang received the BEng degree in computer science and technology from Shanghai Jiaotong University, in 2017. He is currently working towards the PhD degree in computer engineering at Stony Brook University. His research interests include blockchain and mobile edge computing.

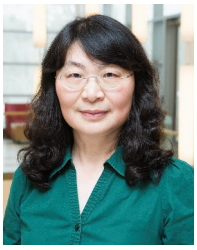


Jun Duan received his B.S. and M.S. degrees from School of Electronics Engineering and Computer Science, Peking University, China, in 2008 and 2011. He received his Ph.D. degree in Computer Engineering from Stony Brook University, US, in 2018. He is a Research Staff Member at IBM Thomas J. Watson Research Center. His research interests include hybrid cloud, blockchain networks, and data center networks. Jun is a member of IEEE.

Bin Xiao (S'01-M'04-SM'11) received the B.Sc. and M.Sc. degrees in electronics engineering from Fudan University, China, and the Ph.D. degree in computer science from The University of Texas at Dallas, USA. He is currently an Associate Professor with the Department of Computing, The Hong Kong Polytechnic University. Dr. Xiao has over ten years research experience in the cyber security, and currently focuses on the blockchain technology and AI security. He published more than 100 technical papers in international top journals and conferences. Currently, he is the associate editor of the Journal of Parallel and Distributed Computing (JPDC) and the vice chair of IEEE ComSoc CISTC committee. He has been the symposium co-chair of IEEE ICC2020, ICC 2018 and Globecom 2017, and the general chair of IEEE SECON 2018.



Fan Ye is an Associate Professor in the ECE department of Stony Brook University, before that he was a Research Staff Member at IBM T. J. Watson Research after getting his Ph.D. from UCLA CS department in 2004. His research interests include mobile sensing platforms, systems and applications in healthcare and location based services, edge computing, Internet-of-Things, and data-centric wireless communication. He has published over 100 papers with 12,000+ citations according to Google Scholar, and 30 granted/pending patents/applications. He has received NSF CAREER award, Google Faculty Research Award, IBM Research Division Award, 5 Invention Achievement Plateau awards, Best Paper Award for IEEE ICCP 2008. He has been a panelist for NSF and Canada, Hong Kong government funding agencies, on program/organizing committees for conferences including IEEE Infocom, IEEE ICDCS, ACM Mobicom, ACM Sensys.



Yuanyuan Yang received the BEng and MS degrees in computer science and engineering from Tsinghua University, Beijing, China, and the MSE and PhD degrees in computer science from Johns Hopkins University, Baltimore, Maryland. She is a SUNY Distinguished Professor of computer engineering and computer science at Stony Brook University, New York, and is currently on leave at the National Science Foundation as a Program Director. Her research interests include edge computing, data center networks, cloud computing and wireless networks. She has published over 460 papers in major journals and refereed conference proceedings and holds seven US patents in these areas. She is currently the Editor-in-Chief for IEEE Transactions on Cloud Computing and an Associate Editor for ACM Computing Surveys. She has served as the Associate Editor-in-Chief for IEEE Transactions on Computers and IEEE Transactions on Cloud Computing and Associate Editor for IEEE Transactions on Parallel and Distributed Systems and IEEE Transactions on Computers. She has also served as a general chair, program chair, or vice chair for several major conferences and a program committee member for numerous conferences. She is an IEEE Fellow.