Resource Allocation and Consensus on Edge Blockchain 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, USA
{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 in the blockchain, and propose a recent block storage allocation scheme for quick retrieval of missing blocks. The proposed blockchain system can also reach mining 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 15% less time and consumes 64% less battery power when compared with traditional blockchain systems.**

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.

Consider the following situations where users want to access some valuable data in edge environments and would like to pay for them in peer edge environments. IoT devices produce real-time sensing data and can provide sensing-as-a-services for risk managements [1]. The smart home can offer energy transaction services in the wireless environment [2]. People can also generate personal for-profit "We Media" [3] content like short video clips. Users can subscribe and pay for the desired data. The corresponding subscriptions allow paid users to access data quickly and securely, and deny offering data to unpaid users. Current solutions require a trusted third party or platform to manage contents and subscriptions, similar to YouTube for video creators and Wink for IoT devices. The centralized third party may fail sometimes and may have user privacy concerns. In peer edge environments, micro-access control and micro-payment transactions can provide fast and reliable data accessing. With micro-access control and micro-payment, devices can directly manage subscriptions and payments to deliver for-profit data to the consumer nearby. For example, vehicles can sell road information directly to peer vehicles in edge environments without a trusted cloud backend or other certain entities. The blockchain technology is an applicable secure ledger for sharing 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. A 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 that a user obtains. Second, a blockchain is designed to be resistant to modification of its records. Theoretically, unless malicious users have more than half of the computational power, neither the block nor its contained data can be changed. Third, it is a disintermediation [4] system, meaning that there is no need for a trusted third-party to verify data, and it can also avoid "a central point of 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 onto edge environments directly. Thus, we are facing two design challenges to overcome the limitation of edge devices: 1) how to store data in the network with limited storage optimally, and 2) 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 and fast data accessing when needed. The heterogeneity of different devices requires fair allocation over the resources, store fewer data items onto devices with fewer resources. Battery limitations require energy-saving consensus design in the mining process.

2575-8411/19/$31.00 ©2019 IEEE
DOI 10.1109/ICDCS.2019.00147

In this paper, we first design an optimal resource allocation strategy for applying blockchain to peer 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 also propose a new Proof of Stake (PoS) consensus mechanism to reduce the energy consumption for mining 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 mining.

We make the following contributions in this paper.

- We design a blockchain system targeting resource allocation and mining consensus on edge environments to achieve less storage and energy consumption compared with 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 for quick 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 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. The simulation results show that the proposed blockchain system can achieve fast data access with 15% less time than random store, fair data storage with disparity measurement less than 0.15, and mining consensus with 64% 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 the future work in Section VII.

II. Related Work

The blockchain technology is proposed in 2008 by Satoshi Nakamoto [5] 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 [6], and the 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 [5], Litecoin [7], and Ethereum [8].

Although blockchain technology has some security features, the data transmission and storage remain challenging in distributed systems. The traditional blockchain requires each participating user to store the whole chain for the security and performance, and each new transaction and new block are broadcasted over the internet. Such huge transmission and storage overhead draw some attention on improving the storage consumption and data propagation over blockchain. Ozisik et al. [9] use bloom-filter and IBLT (invertible bloom lookup table) to reduce the transmission overhead in the blockchain. Eyal et al. propose the Bitcoin-NG [10] to reduce the transmission and bandwidth by creating a new micro-block chain to minimize the amount of data transmitted.

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 [11]. 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 [12], [13]. 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 [14], [15].

For resource allocation in edge environments, we propose the resource placement problem. 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, in which the Uncapacitated Facility Location (UFL) problem [16] and the rent-or-buy problem [17] are most popular and well-studied. The more general case for these two problems is the Connected Facility Location (ConFL) problem [18]. 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 will mostly use the UFL problem and its corresponding solutions. The current best solution that we find is proposed by Li et al. [19], where they obtain a 1.488 approximation ratio.

Mining in the blockchain means that nodes, often called miners in this situation, reach consensus on how to add a new block in the blockchain. 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) [5], [20]. Block information from Proof of Work are often hard to obtain but easy to get verified. Another emerging concept is called Proof of Stake (PoS) [21], [22]. PoS, in contrast, achieves the consensus from the publicly owned data of the users such as wealth or age. Since PoS does not rely on exhaustively solving mathematic puzzles, it saves a lot of energy for mining a new block. Instead, the 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 lower compared with PoW.

III. Blockchain System Design

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

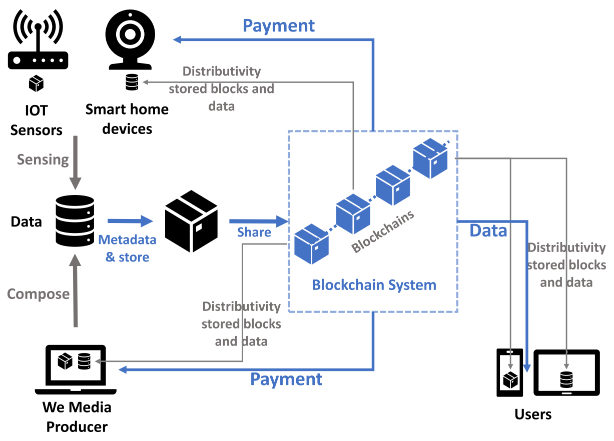

Fig. 1. The application scenario in proposed blockchain system. Nodes generate for-profit data and users pay for them. Real data and blocks are fragmented and stored across the nodes.

Fig. 1 shows an example application scenario 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 deliver to consumers. The corresponding data and payment information 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 conducting data transactions and mining blocks in the blockchain. A node is an edge device participating in the system. Each participating node can generate data, store data and blocks, conduct payment and mine blocks. We assume that all participating nodes have certain computational capabilities like smartphones. Some small IoT nodes may have very constrained storage or computing power thus cannot participate.

Each node has its private and public keys for identification purposes. Keys then generate an account of that node. Each account is unique and associated with each node and has a unique address (hash value) satisfying a certain pattern. The account address can be generated from public keys but not in reverse. Each node tries to mine blocks to get incentives, and such incentives will be store in its account. Since how to assign incentive is not a major concern in this paper, for simplicity, we assign the incentive for mining new block as one “token”. Tokens are used in the Proof of Stake (PoS) processes.

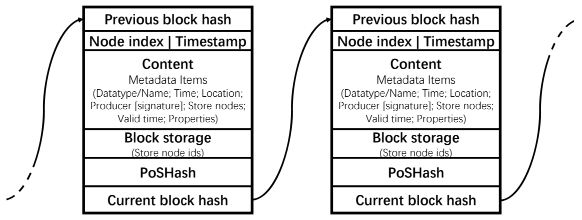

Fig. 2. The proposed blockchain system and components of a block in the blockchain system.

Fig. 2 shows the components of each block 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 corresponds to a data item. In addition, each block records the information about where this block is stored, since only some nodes store this block. The storage allocation is discussed in the next section. Also, the block stores another hash specifically for the PoS mechanism. Each node needs this hash to generate corresponding credentials for mining.

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 proposed metadata design 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 [23]. 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]; Storing nodes; Valid Time; Properties
(AirQuality/PM2.5; 11:00AM06-11-2018;
NewYorkNY/40.72,-74.00; 17,[Signature of producer];
10,11,12,15; 1440; NULL)
(Picture/Traffic; 11:27AM06-11-2018; NassauNY/40.78,
-73.58; 33,[Signature of producer]; 16,17,26,44; 720;
'Camera')
```

```
(KeyExchange/PublicKey; NULL; NULL; 15,[]; NULL; 2880;
[Key])
```

Since the metadata item contains 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 might 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 the node which generates the corresponding data item and attaches the 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 blockchain, the metadata information in blocks is difficult to change unless malicious nodes own more than half of the resources 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 and 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 they expire. Further detailed design considering the data validation will be discussed in our future work.

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 has 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 onto 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 Mining Consensus

As previously mentioned, the major disadvantage of Proof of Work (PoW) is that it is too energy consuming. The total amount of energy consumed per year for bitcoin mining is 70 TWh (7×10^{10}kWh) [2]. Thus, the proof of work is impractical for edge devices, for the energy in edge devices is often very limited.

To achieve consensus mining with low energy consumption, we use the PoS concept and develop a new mechanism. Our proposed PoS mechanism gives advantages to nodes that have more contributions to the system, like storing data and mining blocks. We associate a value, called the target value, to each node for mining. It is related to the amount of used storage and the number of token of the node. The larger the value is, the higher the probability will be for the node to mine a block. One major characteristic of our proposed mechanism is that it gives more advantages over mining to such nodes that store more data. This is crucial in peer edge computing environments. The advantages over mining can motivate nodes to participate in the system even when they do not produce or need any data.

[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 receiving block is 2 or more 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 June 2018

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 all nodes, 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 [24] proposed the concept of fair caching in peer edge environments. Each node has different capacities, and the algorithm should store fewer data onto node 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 less remaining storage corresponds to a 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: 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. Note that the range of each node, in reality, will change. Nodes moving outside the original range and new coming nodes will broadcast their 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. The data migration problems will be discussed in future work.

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 above formulation, 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} 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 [19]. 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 [24].

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 mines the next block (the mining 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 every node to store all blocks. Blocks also need to be stored among a fraction of all nodes. Each new block will be assigned to store on some nodes. The storing nodes information are encoded in this block. Then, corresponding nodes receive this information will keep this block in their storages. 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 potential temporary disconnection of nodes. If recent blocks are more pervasive in the network, retrieving them 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 mining 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

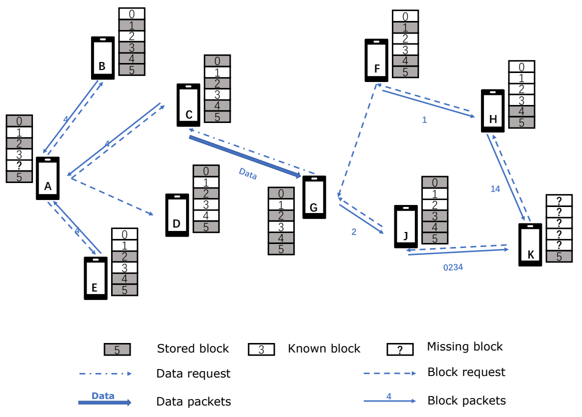

Fig. 3. The data and block access process. Node A reconnects to the network with a missing block. Node G is requesting a data piece. Node K is a new node entering the network.

Fig. 3 shows an example of nodes accessing blocks and data 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. For a disconnected node, it only needs recent blocks. For example, Node A receives block 5 and finds out it misses block 4. It then sends its requests to its neighbor Nodes B, C, D, and E. Since many nodes store recent blocks, the missing blocks can be easily obtained within a few hops.

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 and finds out 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 enters the network and finds out it needs blocks 0 to 4 when it receives block 5. It sends the request to its neighbor Node J and H. Node J and H can satisfy some block but not all. They then request the missing block 1 from 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.)

V. Proof of Stake Mechanism

The proposed Proof of Stake (PoS) algorithm is designed to mine blocks. Mining is a process that nodes compete with each other to get the privilege to generate the next block. The goal of this PoS mechanism is to make sure that, if a node has more token and stores more data and blocks, the node will have more advantages to mine blocks. Inspired by the Nxt [25], we design our own PoS mechanism in the following.

A. Mining Conditions

If a node contributes more, i.e., stores more data and mines more blocks, it will have a higher probability over others to mine the next block. In our design, we give a *target value* R relatively larger for the node which has higher contributions. Each node will calculate a number, called *hit* h, based on the obtained blockchain and the account information. The hit is the same uniform random variable for every node, and the value is determined every time a node starts to mine a block. If a hit from a node is smaller than its target value at a time, the corresponding node is to generate the next block. Thus, the higher target value means the higher probability for mining a block.

To mine the next block, node i first computes the hit h_i. h_i is based on the previous block and the account of this node. We define the hit h_i of node i at specific time (chain length) t as

$$POSHash(t+1,i) = \text{Hash}[POSHash(t) + Account_i],$$
$$h_i = POSHash(t+1,i) \mod M. \tag{7}$$

$POSHash(t)$ is the POSHash in the previous block. It is used for PoS mining process. As mentioned in Section III-A, each node is associated with an account address obtained from its key pairs. Each node uses the previous POSHash appending its account address and hashes them together to get a new POSHash. If this node is to generate the next block, $POSHash(t+1, i)$ will be the new POSHash in the new block. M is the largest possible number of the hit. Mod M is used in (7) because the number that represents the POSHash is often very large (e.g., hash function SHA-256 generates a 256-bit binary number), making the calculation and expression harder. A proper hash function should generate uniform random numbers. Thus, h_i is also a uniform random variable $\mathcal{U}(0, M)$. For each block, h_i is different but unique for each node. Each node can also validate the hit of other nodes, so a node cannot fake a hit to get unfair advantages. Once h_i is obtained, R_i is to determine which node will mine the next block.

Each node has different target value R_i based on the contribution of the node. We define R_i of node i at time t as

$$R_i = S_i Q_i t B. \tag{8}$$

In this target value definition, S_i denotes the token owned by node i. Tokens can be obtained by mining new blocks. In this paper, for simplicity, a new block generates a token, and this token will be added to the account of the node that mines this block. Q_i is the number of data items stored in node n. We assume that each data item is of the same size for simplicity. In real cases, large data can be split into small and same-sized chunks. t is the time passed from the previous block (in seconds, based on the timestamp in the block). Thus, R_i is increasing as time goes until a node meets the criteria and generates a new block. The expectation time for mining a block is t_0, which is preset according to the application scenario. B is a number to adjust R_i to make sure that the average time mining a block throughout the network is as expected. This number is the same for all nodes at a time but will be changing from time to time. R_i is also easily validated because the miner of each block and each data storage information can be accessed from blocks. S and Q of each node can be obtained and validated through the history of the blockchain.

A node is to mine a block if

$$h_i \leqslant R_i. \tag{9}$$

As resented in (8), R_i grows with each second since the timestamp of the previous block. Since different nodes will have different S_i and Q_i, the time when this inequality holds are different for different nodes. The node which is the first to meet this inequality generates the next block. This node then broadcasts the message of this new block, and other nodes will verify its hit and target value information. If the information is correct, nodes will accept the block and work on the following blocks. Otherwise, nodes will reject this block and continue working on the finding process.

Note that for a new node entering the network, it requires to have at least one token. This token can be obtained from different places in different scenarios, e.g., purchase tokens from other nodes or get from a public pool. This gives the node the initial ability to mine blocks. For storage, since a node will at least store the last block for mining information. Thus, the number of data stored in a new node is also one. Node i can increase S_i and Q_i by mining more blocks or storing more data.

B. Target Value and Expectation Mining Time Settings

Having a stable generating pace for blocks is crucial in a blockchain system. A reasonable time between two blocks makes each block maintain similar size and also helps to keep load balancing for each node. In our designed blockchain system, we use an amendment number B to keep the expected time between two blocks as a given number t_0.

At a specific time, we have

$$t_0 = \mathbb{E}(\min_i\{t_i\}). \tag{10}$$

In this equation, $\mathbb{E}(\cdot)$ denotes the mathematical expectation, and n is the number of nodes in the network. Since S_i, Q_i and B are predetermined and will not change during the time nodes mining a block, according to (8) and (10), the expectation of R_i is as

$$\begin{aligned}\mathbb{E}(S_i Q_i t_0 B) &= \overline{U}\mathbb{E}(\min_i\{t_i\})B \\ &= \frac{1}{n}\mathbb{E}(\min_i\{S_i Q_i t_i B\}).\end{aligned} \tag{11}$$

Since S_i and Q_i are not random variables, $\mathbb{E}(S_i Q_i)$ is represented as the average number. We denote $U_i = S_i Q_i$ and $\overline{U} = \frac{1}{n}\sum_{i=1}^{n} U_i$. According to (8) and (9), we get

$$\mathbb{E}(\min_i\{h_i\}) \leqslant \mathbb{E}(\min_i\{S_i Q_i t_i B\}). \tag{12}$$

Since h_i is a uniform random variable, we let $H = \{h_1, h_2, ..., h_n\}$ and $Z = \min_i\{h_i\}$. We have

$$\begin{aligned}CDF_Z(z) &= Pr(Z \leqslant z) \\ &= 1 - Pr(Z > z) \\ &= 1 - (1 - CDF_H(z))^n, \\ PDF_Z(z) &= \frac{d}{dx}CDF_Z(z) \\ &= PDF_H(z)(1 - CDF_H(z))^{n-1} \\ &= \frac{1}{M}(\frac{M-z}{M})^{n-1}.\end{aligned}$$

The expectation of minimum h_i will be

$$\begin{aligned}\mathbb{E}(z) &= \int_{-\infty}^{\infty} zPDF_Z(z)dz \\ &= \int_0^M \frac{z}{M}(\frac{M-z}{M})^{n-1}dz, \\ \mathbb{E}(Z) &= \frac{M}{n(n+1)}.\end{aligned} \tag{13}$$

From (11), (12) and (13) we get

$$\mathbb{E}(S_iQ_it_0B) \geqslant \frac{M}{n(n+1)},$$
$$B \geqslant \frac{M}{(n+1)t_0\overline{U}}. \quad (14)$$

TABLE I
SYMBOLS USED IN POS MECHANISM

h_i	A hit of node i, $h_i \sim \mathcal{U}(0, M)$
M	the largest possible number for h_i
S_i	The number of tokens of node i
Q_i	The number of data items cached in node i
t_i	Time from previous block that node i can declare to add a new block
t_0	Expectation time between two blocks
B	Expectation time amendment, the value to adjust the time between two blocks of the entire network
U_i	$U_i = S_iQ_i$, $\overline{U} = \frac{1}{n}\sum_{i=1}^{n} U_i$, the average

The expectation time amendment B is adjusted every time before mining a block. Since all the information can be obtained, the number B is easily calculated and verified as well.

Note that B will be decreasing continuously as S_i and Q_i are becoming larger and larger. Over time, B will become very small, which will make the computation hard. A simple solution is to decrease S_i for all nodes simultaneously (by ratio) after a certain number of blocks, and increase B by the same ratio. Since the S_i decreasing ratio is the same for all nodes, the relative mining advantages of each node will remain the same.

C. Mining Process for Nodes

The mining process of node i is shown in the following.

```
1: **while** received a new block **do**
2:     Get $h_i = POSHash(i) \mod M$ from current block
3:     Get $U_i = S_iQ_i$ from history of blockchain
4:     Get $B$ from current block
5:     $t = 1$
6:     **repeat** every second
7:         $R_i = U_iB * t$
8:         $t = t + 1$
9:     **until** $h_i \leqslant R_i$ or new block received
10: **end while**
```

First, nodes receiving a new block will check if the block is valid and h_i for certain node i is correct. If the new block is valid, nodes then start to mining the next block. Each node then obtains updated information h_i from (7), U_i, and B from (14). Every second, each node calculates and updates its target value R_i from (8). Once $h_i \leqslant R_i$, the node has the privilege to add a new block in the blockchain. The node then gathers all information, including the metadata it receives during this period, previous hash, the new PoSHash, and the current timestamp. Then, the node calculates which node should store this block, and which node should store more recent blocks (in Section IV-C). It also need to calculate the storing node for each data items in the period (in Section IV-B). The node wraps all information above into a new block (as shown in Fig. 2) and broadcasts it.

D. Discussion on Proof of Stake

As a replacement for PoW algorithms, PoS can have different forms and different metrics. We present a new PoS mechanism for the edge devices scenarios considering the heterogeneity of different devices, and giving the advantages to the node which has real contributions to the network. We also obtain a stable expectation time between generating two blocks.

The key factor of PoS is that the calculation does not consume much energy. For such application scenarios like edge devices, the energy consumption is crucial as battery capacities are limited. However, it also raises a problem due to the low complexity of computation work. In PoW, nodes need a lot of computational power to get the correct hash. Thus, it is hard to work on multiple branches. To maximize its profit, nodes tend to work on the longest chain. In PoS, however, working on different chains has a less computational burden. A node might work on different chains to get more profit. Solutions about inserting checkpoint block are proposed to force nodes working on the chain that has checkpoint blocks.

Another problem for the design is forging accounts. Since there are no central authorities, a node can create multiple different accounts. If each account gets some initial resources, forging a lot of fake accounts will get unfair advantages. A simple solution for a new node is to "rent" some resources from an existing node to get started. Nxt [25] uses this mechanism by having limited tokens for all nodes. We will discuss in our future work about how to adapt the solutions to our PoS mechanism for the problems mentioned above.

VI. PERFORMANCE EVALUATION

In this section, we evaluate our proposed blockchain system. We focus on evaluating the optimal placement effects on different nodes and the power efficiency of the mining consensus algorithms.

We implement the blockchain system using Node.js. The core code of the blockchain is the Naivechain [26], adding the functionality on general information consensus, communication, Proof of Stake (PoS) and optimal data placement. We simulate multiple nodes using Docker [27]. Docker is a container system, in which we can create multiple nodes distributively. Each node runs a blockchain system in the container and communicates with others using standard socket communication. In our simulations, Docker provides a real distributed scenario with no centralized information to nodes. For general consensus over devices, we implement raft algorithm [28] in our blockchain system. We conduct our simulations on a computer with an Intel Core i7-5820K processor and 32 GB RAM. All results are the average of 2 simulations.

In the simulation, we assume all nodes are distributed in an area of 300m $\times$ 300m, and the communication range between two nodes are 70m, based on typical 802.11n communication range. We set the mobility of the nodes is within 30 meters ranges. Each node has the capability to store 250 data items or blocks. Unless otherwise specified, we set the expected average time for mining a block as 60 seconds, and each simulation runs over 500 minutes.

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 the different number of nodes in the network, varying from 10 to 50, and on average 1 to 3 data items are generated throughout the network per minute. In a private blockchain system, the number of nodes may be limited compared to a public blockchain system. The data are requested randomly by 10 percent of nodes. The size of each data item is 1MB, and average block size is less than 10 KB. We simulate the data delay by adding a small delay (10 ms) as propagation delay over one hop. The data is obtained from network simulators as the typical propagation delay over the 802.11. Since the nodes use the socket to transited data, processing, queuing and transmission delay can be simulated through Docker.

We record all data and block transmission to show the overhead for transmission and the data delivery time. We also use Gini coefficient [3] to show the fairness of the proposed system.

Fig. 4 shows the average transmission overhead for nodes (a) under different data amounts and different numbers of nodes in the system. The transmission overhead includes data request and transmission, data dissemination (storing node proactive get data from the producer), and all blockchain broadcasting information. The transmission overhead is relatively small, with a maximum about 120 MB data are transmitted for a node. 500MB to 1.5 GB data are generated in 8 hours running, yet the total transmission is less than 4GB. Thus, only no more than 8 times of data are transmitted in the network. The total amount of transmission is increasing along with the data amount since more data are transmitted over the network. The same total overhead also indicates the decreasing on average overhead per node when more nodes are presented. The result shows that the system performs well under the larger size of networks.

Meanwhile, the Gini coefficient (b) for all the tests is below 0.15, which means the storage disparity is relatively low. Since we set the nodes are of the same storage capacity in the simulation, this shows that the dissemination storage is fair among all nodes in the system. The average data delivery time (c) 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, but overall 4 seconds in maximum is used for a node to get the desired data.

[3]Gini coefficient is widely used to depict income disparity, and is also used in previous works to measure fair caching [24], [29]. $Gini = \frac{\sum_i \sum_j |t_i - t_j|}{2 \sum_i \sum_j t_j}$.

B. Performance under Different Placement Strategy

Next, we evaluate the performance of the optimal data placement strategy we proposed. We store data onto nodes that can offer quick access for all the users demanding data. We compare the proposed strategy with a naive solution that data are randomly stored. For a fair comparison, the total number of data and blocks stored is the same as the optimal placement. We test the performance under different numbers of nodes and 1 data item generated per minutes.

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 no proactive data storage, on average, uses less than 15% of the time used to access data than random storage. Meanwhile, The message overhead is almost the same between two different strategies, showing that our proposed algorithm does not cost extra communicational overhead. The results show that the proposed optimal caching strategy achieves less time for data access, and it does not increase the message overhead in the proposed blockchain system.

C. Performance under Different Mining Consensus Algorithm

To test the energy consumption of different mining consensus algorithms in real edge environments, we also implement the Proof of Work (PoW) and Proof of Stake (PoS) on a smartphone. We make sure the phone is fully charged before mining, and test different mining algorithms. When a block is mined, we record the remaining battery of the phone. To reduce the affections from other factors, all background processes are closed, and both tests on PoS and PoW are using the same fake data pieces. The experiment platform uses react-native [30]. The mining process is tested on a Samsung Galaxy S8.

Fig. 6 shows the remaining battery with the number of blocks mined. We set the difficulty of PoW as 4 zeros at the beginning of the block hash. The average mining time for each block is 25 seconds at this difficulty. For a fair comparison, we also set the same average mining time for PoS as 25 seconds. The PoW consumes more than half of the battery over 84 minutes, while PoW only consumes less than 20% of the battery. In average, 4 blocks consume about 1% battery of the phone in PoW, while 11 blocks consume 1% battery of the phone. The difference will be even larger if the difficulty for PoW increase. The computational complexity grows exponentially in PoW but remains almost the same for PoS. The result shows the PoS algorithm consumes much less energy compared to traditional PoW algorithm in the mining process in edge environments.

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 onto blocks, and corresponding data items and blocks are

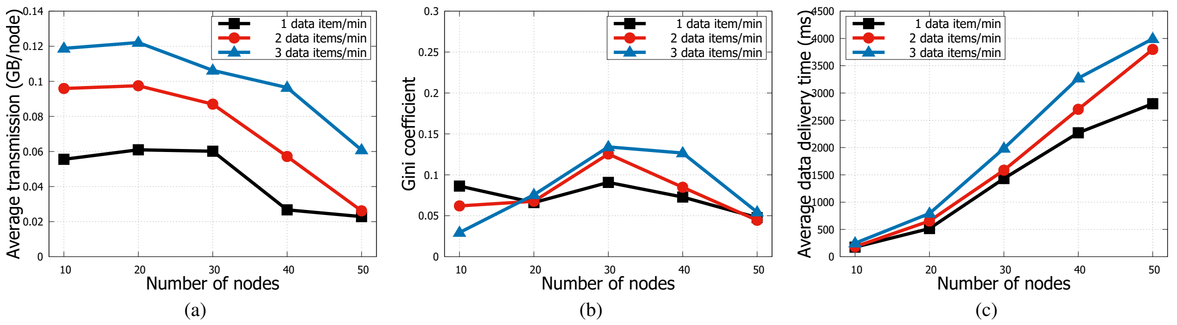

Fig. 4. The average transmission of each nodes (a), Gini coefficient (b), and average data delivery time (c) under different number of nodes and different data amount. Our proposed algorithm has low transmission overhead and fast data access time, and the storage distribution is fair among nodes.

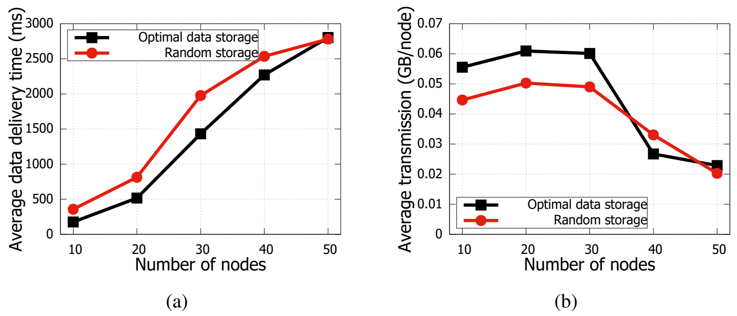

Fig. 5. The average data delivery time from nodes (a) and the average data transmission overhead (b) under different number of nodes and two different data placement strategies. The optimal data placement receives the fast data delivery time and similar message overhead compared with no proactive store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

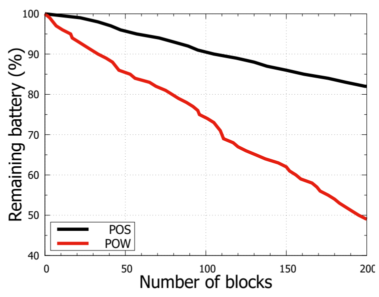

Fig. 6. The remaining battery in mining process under different mining consensus algorithm. The PoW consumes much more energy than PoS.

distributively stored for quick data access. We have proposed the recent block storage for quick access of missing blocks. 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 mining. Simulations show that our blockchain system works well under pervasive edge environments with limited resources.

Over time, data items may become obsolete, and nodes will also change the location. The distributed storage will not remain optimal during that time. Calculating the optimal storage problem is not necessary if the change over the network is small. In the future, we will discuss the data migration problem, which will study how to use less operation to achieve less offset from the optimal result. Besides, recent blocks storage will need the expiration to avoid using up the storage. We will address in future work about the process running over time. Furthermore, general information consensus algorithms are also important in edge environments. We partly use the raft algorithm in our simulation, but the approach transmits a large number of heartbeat messages. In the future, we will develop a new consensus algorithm for edge environments with less message overhead.

Acknowledgment

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

References

[1] A. Zaslavsky, C. Perera, and D. Georgakopoulos, “Sensing as a service and big data,” *arXiv preprint arXiv:1301.0159*, 2013.

[2] M. A. A. Pedrasa, T. D. Spooner, and I. F. MacGill, “Coordinated scheduling of residential distributed energy resources to optimize smart home energy services,” *IEEE Transactions on Smart Grid*, vol. 1, no. 2, pp. 134–143, 2010.

[3] S. Bowman and C. Willis, “We media,” *How audiences are shaping the future of news and information*, 2003.

[4] J. Mattila *et al.*, “The blockchain phenomenon–the disruptive potential of distributed consensus architectures,” The Research Institute of the Finnish Economy, Tech. Rep., 2016.

[5] S. Nakamoto, “Bitcoin: A peer-to-peer electronic cash system,” 2008.

[6] M. Iansiti and K. R. Lakhani, “The truth about blockchain,” *Harvard Business Review*, vol. 95, no. 1, pp. 118–127, 2017.

[7] 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.

[8] 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.

[9] 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.

[10] I. Eyal, A. E. Gencer, E. G. Sirer, and R. Van Renesse, “Bitcoin-ng: A scalable blockchain protocol.” in *NSDI*, 2016, pp. 45–59.

[11] M. Research, “Edge computing,” https://www.microsoft.com/en-us/research/project/edge-computing/, Oct. 2008.

[12] M. Satyanarayanan, V. Bahl, R. Caceres, and N. Davies, "The case for vm-based cloudlets in mobile computing," *IEEE pervasive Computing*, 2009.

[13] 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.

[14] 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.

[15] 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.

[16] G. Cornuéjols, G. L. Nemhauser, and L. A. Wolsey, "The uncapacitated facility location problem," DTIC Document, Tech. Rep., 1983.

[17] 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.

[18] 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.

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

[20] 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.

[21] S. King and S. Nadal, "PPCoin: peer-to-peer crypto-currency with proof-of-stake," https://peercoin.net/assets/paper/peercoin-paper.pdf, 2012.

[22] R. P. d. Santos, "Pow, pos, & hybrid protocols: A matter of complexity?" *arXiv preprint arXiv:1805.08674*, 2018.

[23] 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.

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

[25] Nxt community, "Nxt whitepaper," http://nxtwiki.org/wiki/Whitepaper:Nxt, 2014.

[26] L. Hartikka, "Naivechain," https://github.com/lhartikk/naivechain, 2017.

[27] S. Hykes, "Docker," https://www.docker.com/what-docker, 2015.

[28] D. Ongaro and J. Ousterhout, "In search of an understandable consensus algorithm," in *2014 USENIX Annual Technical Conference (USENIX ATC 14)*, 2014, pp. 305–319.

[29] 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.

[30] T. Occhino, "React native: Bringing modern web techniques to mobile," https://code.facebook.com/posts/1014532261909640/react-nativebringing-modern-web-techniques-to-mobile, 2015.