EdgeChain: An Edge-IoT Framework and Prototype Based on Blockchain and Smart Contracts

Jianli Pan, *Member, IEEE*, Jianyu Wang, Austin Hester, Ismail Alqerm, *Member, IEEE*, Yuanni Liu, and Ying Zhao, *Member, IEEE*

***Abstract*—The emerging Internet of Things (IoT) is facing significant scalability and security challenges. On one hand, IoT devices are "weak" and need external assistance. Edge computing provides a promising direction addressing the deficiency of centralized cloud computing in scaling massive number of devices. On the other hand, IoT devices are also relatively "vulnerable" facing malicious hackers due to resource constraints. The emerging blockchain and smart contracts technologies bring a series of new security features for IoT and edge computing. In this paper, to address the challenges, we design and prototype an edge-IoT framework named "EdgeChain" based on blockchain and smart contracts. The core idea is to integrate a permissioned blockchain and the internal currency or "coin" system to link the edge cloud resource pool with each IoT device' account and resource usage, and hence behavior of the IoT devices. EdgeChain uses a credit-based resource management system to control how much resource IoT devices can obtain from edge servers, based on predefined rules on priority, application types, and past behaviors. Smart contracts are used to enforce the rules and policies to regulate the IoT device behavior in a nondeniable and automated manner. All the IoT activities and transactions are recorded into blockchain for secure data logging and auditing. We implement an EdgeChain prototype and conduct extensive experiments to evaluate the ideas. The results show that while gaining the security benefits of blockchain and smart contracts, the cost of integrating them into EdgeChain is within a reasonable and acceptable range.**

***Index Terms*—Blockchain, edge computing, EdgeChain, fog computing, Internet of Things (IoT), scalability, security, smart contracts.**

I. Introduction

IT IS predicted that the emerging Internet of Things (IoT) will connect more than 50 billion smart devices by the year 2025 [1]. It will inevitably change the way we live and work with smart houses, workspaces, transport, and even cities on the horizon. However, such trends create significant scalability and security challenges. First, the IoT devices are relatively weak and most of their data are sent to remote clouds to be processed. Examples include the majority of the smart phones applications and smart home devices, such as Google Home and Amazon Echo. But the existing centralized cloud computing model is very difficult to scale with the projected massive number of devices due to the large amount of generated data and the relatively long distance between IoT devices and clouds. Second, the IoT devices are relatively vulnerable and could be relatively easily controlled by malicious hackers to form "botnet" for various attacks [2], [3]. This is aggravated by the fact that most of the cheap IoT devices are with very limited security capabilities, and very poor or even no technical upgrading or maintenance services, though recently Google's Android Things 1.0 [4] started pushing this.

Edge computing[1] [5]–[9] is an emerging direction to provide solutions for the IoT scalability issue. It pushes more computing, networking, storage, and intelligence resources closer to the IoT devices, and provide various benefits, such as faster response, handling big data, reducing backbone network traffic, and providing edge intelligence. Typical benefited IoT applications include emergency response, augmented reality (AR), video surveillance, speech recognition, computer vision, and self-driving.

Many works have also been devoted to IoT security. Traditional general-purpose security solutions are not suitable to run on the IoT devices due to the capability constraints [32]. A typical compromise is to use lightweight IoT security protocols [13]–[18]. Perimeter-based security through firewall [19], [20] does not require running additional software on IoT devices but cannot prevent internal attacks and has been proved ineffective in securing billions of weak devices. Compared with perimeter-based trust, zero-trust approaches [21]–[23] are proved to be more effective and seem promising. Direct or indirect system-level security approaches, which do not put intensive security-related loads on IoT devices and do not assume the IoT devices being well-maintained, and if enabled with a zero-trust or trustless capabilities, are much needed. Blockchain [24], [25] combined with smart contracts [26], [33] enable a *trustless* environment and are recently attracting more attention due to unique features, such as data/transactions persistence, tampering resistance, validity, traceability, and distributed fault tolerance. Limited efforts have been made applying them into decentralized IoT and edge computing systems, and two typical work are Xiong *et al.* [27], [28] using game theory and

Manuscript received September 18, 2018; revised October 14, 2018; accepted October 22, 2018. Date of publication October 26, 2018; date of current version June 19, 2019. This work was supported in part by the National Security Agency under Grant H98230-17-1-0393 and Grant H98230-17-1-0352 and in part by the National Aeronautics and Space Administration EPSCoR Missouri RID Research under Grant NNX15AK38A. *(Corresponding authors: Jianli Pan; Jianyu Wang.)*

J. Pan, J. Wang, A. Hester, and I. Alqerm are with the Department of Mathematics and Computer Science, University of Missouri–St. Louis, St. Louis, MO 63121 USA (e-mail: pan@umsl.edu; jwgxc@umsl.edu).

Y. Liu is with the Institute of Future Network Technologies, Chongqing University of Posts and Telecommunications, Chongqing 400065, China.

Y. Zhao is with the Department of Information Engineering and Art Design, Shandong Labor Vocational and Technical College, Jinan 250022, China.

Digital Object Identifier 10.1109/JIOT.2018.2878154

[1]Edge computing is also often referred as "fog computing," "Mobile Edge Computing," or "Cloudlet" in different literature, despite slightly different definitions and scopes. We use edge computing or edge cloud in this paper.

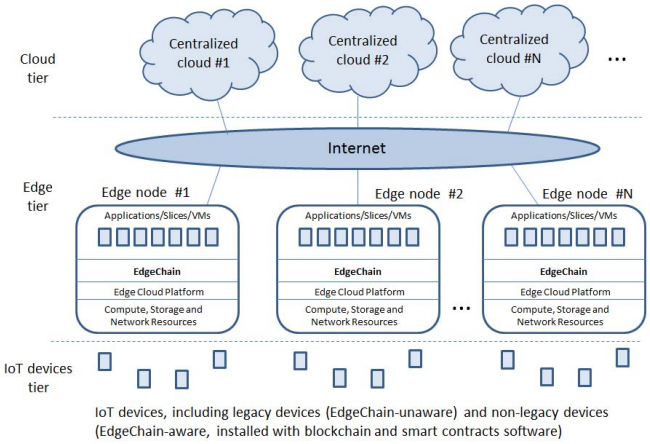


Fig. 1. EdgeChain position in the multitier edge-IoT system network topology.

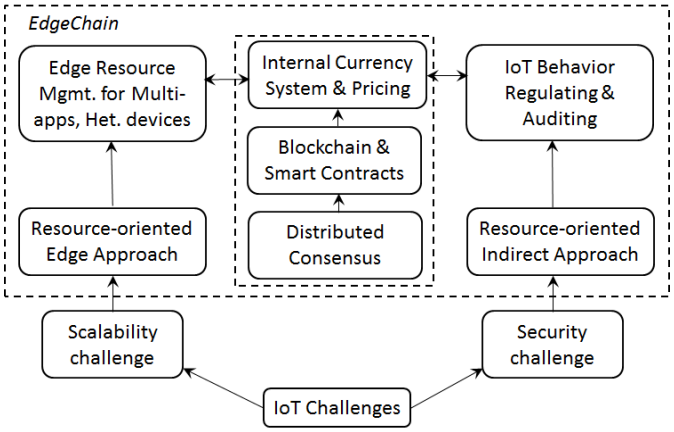


Fig. 2. EdgeChain vision: the problem space and solution space.

Chatzopoulos *et al.* [31] focused on computation offloading. In comparison, our research focus is not on consensus mechanism and mining. Instead, we use permissioned blockchain and smart contracts as carrying vehicle, and our major focus is to provision resources for various IoT applications and control and regulate IoT devices' behavior.

In this paper, we seek a fundamentally different approach to tackle these key challenges collectively through a blockchain-based and resource oriented edge-IoT framework named *EdgeChain*. The EdgeChain's position in the multitier edge-IoT system is illustrated in Fig. 1. As we can see that EdgeChain locates between the edge cloud platforms and the various IoT applications that are launched in the shared infrastructure. It means that EdgeChain can run on different edge cloud platforms, such as HomeCloud [46] or Cloudlet [9].

The core EdgeChain idea is to integrate a permissioned blockchain and the internal currency or coin system to link the edge cloud resource pool with each IoT device' account and resource usage, and hence behavior of the IoT devices. EdgeChain uses a credit-based resource management system to control how much resource IoT devices can obtain from edge servers, based on predefined rules on priority, application types, and past behavior. Smart contracts are used to enforce the rules and policies to regulate the IoT device behavior in a nondeniable and automated manner. All the IoT activities and transactions are recorded into blockchain for secure data logging and auditing. As a short summary, the major contributions of the EdgeChain framework include the following.

1) A new EdgeChain framework integrating permissioned blockchain and smart contracts capabilities.
2) An internal currency or coin system linking the edge cloud resource pool with IoT device accounts and resource usage behavior.
3) A credit-based resource management system to control how much resources IoT devices can obtain from edge servers.
4) A resource-oriented and smart contracts-based policy enforcement method to regulate the IoT device behavior.
5) A prototype implementation and experimentation to validate and evaluate the EdgeChain ideas.

Our latest EdgeChain accomplishments have been included in two provisional patents we recently filed [29], [30]. Note that EdgeChain is still an ongoing project and some of the work are still in progress. We will discuss the status accordingly in the following sections. The rest of this paper is organized as follows. In Section II, we discuss several key approaches and designs of EdgeChain. We present the EdgeChain framework and functional modules in Section III. Section IV is about the key processes and workflows. In Section V, we discuss the prototype and evaluation. Section VI is the related work, while the conclusions and future work follow in Section VII.

II. EdgeChain Key Approaches and Designs

In this section, we discuss some key EdgeChain design considerations. Fig. 2 shows the overall EdgeChain vision including the problem space and the solution space.

A. Permissioned Blockchain

Blockchain networks can be generally categorized into permissionless or public blockchain, and permissioned or private blockchain [33]. Permissionless blockchain, such as Bitcoin network is a peer-to-peer decentralized network. It is usually not controlled by any private organization and the whole network runs on broad consensus of all the members in the network. The tradeoff is relatively lower transaction processing throughput and higher latency. Permissioned blockchain, however, is not a pure peer-to-peer network. The stakeholders, such as the application owners of this type of blockchain will have a more controlled and regulated environment, and higher transaction throughput. The consensus mechanisms used for permissionless and permissioned blockchain are also different.

The EdgeChain system uses a permissioned blockchain since the major goal is to support miscellaneous distributed IoT applications that generally have owners and customers. The system stakeholders need more control and higher throughput and performance. For permissioned blockchain, it is also not necessary to run very resource-intensive proof-of-work (PoW) algorithms for consensus because sybil attacks cannot happen. It also removes the necessity of economic incentive for mining, which is usually very resource-consuming in the Bitcoin network. More effective but less resource-intensive consensus protocols are available

and a typical example is practical byzantine fault tolerance (PBFT) [34] for such an environment.

In EdgeChain, the mining work is only done by the edge servers that have more resources than the IoT devices. It is never done by the resource constrained IoT devices. The mining is much less resource intensive compared with permissionless blockchain network. In other words, the edge servers will be in charge of monitoring the transactions, creating, and appending new blocks when new transactions happen. The IoT devices in EdgeChain are only blockchain and smart contracts clients. If they are EdgeChain-aware devices and installed with blockchain and smart contracts software, they are able to interact with the edge servers and get resources and assistance for their tasks through procedures, such as cloud offloading [35]. If they are legacy devices and do not need resources from the edge servers, then they do not even need to install the blockchain and smart contracts software. The EdgeChain is totally transparent to them, but still can create blockchain accounts and manage these IoT devices from the back end.

B. Credit-Based Resource Management

EdgeChain uses an internal currency or coin system enabled by blockchain to link the edge resource pool with the IoT device accounts and resource usage behavior. EdgeChain consists of a novel credit-based resource management system where each IoT device is created a blockchain account and given an initial amount of credit coins. The credit coin balance determines the device's ability to obtain resources from the edge servers. Generally speaking, the device with a larger balance is afforded quicker and faster access. The edge server records credits and debits and provides the necessary resources requested by the IoT device based on a set of rules that takes predefined priority, application types, and past behavior into account. As an ongoing research effort, we are designing detailed intelligent resource provisioning mechanism at the edge clouds for the quality of experience (QoE) of multiple applications and heterogeneous devices.

In fact, we observe that this resource credit management mechanism not necessarily has to be implemented by the internal currency system. The edge server can maintain a traditional credit score system and decide how to grant resources to different devices. However, by utilizing the internal currency system, EdgeChain can gain a series of intrinsic security benefits coming with blockchain. For example, all the coin transactions are automatically logged into the secure and unmodifiable database on blockchain, and it is good for future auditing purposes. Also, it enables smart contracts that could facilitate nondeniable and automated execution of the scheduling rules and policy enforcement in the edge-IoT systems. All these new benefits are not possible without the blockchain and the internal currency system.

C. Resource-Oriented, Smart Contracts-Based Policy Enforcement, and IoT Behavior Regulating

EdgeChain controls the IoT devices based on their behavior and resource use instead of their locations which results in better security control. This overcomes limitations in existing Edge-IoT solutions which are usually "perimeter" based security, i.e., deploying a firewall or a filtering system between

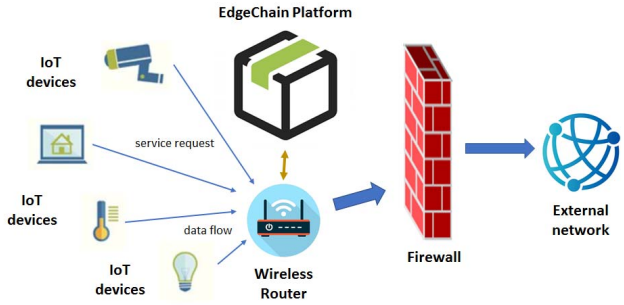


Fig. 3. Simple example of a standalone EdgeChain box deployment in smart home.

the internal and external network and by default trusting the users and nodes "inside" the network. If internal IoT devices were hacked and turned to botnet, it is hard to control them.

EdgeChain uses a resource-oriented, smart contracts-based, and indirect security scheme for IoT behavior regulating and auditing. EdgeChain adopts an indirect system-level security approach, which means that we do not require the IoT devices to run resource-intensive security software. Instead, EdgeChain monitors, controls and regulates the behavior of IoT devices based on their resource usage and activities. Based on the application types, priority, device's past behavior, the preprogrammed smart contracts enforce the resource policy automatically. It means that if some IoT devices were compromised and controlled by hackers for malicious activities, such as behaving erratically, making continuous resource requests that are out-of-line with its profile or application intent, or initiating denial of service attacks, the smart contracts will execute automatically based on the preprogrammed policies. It will be very soon the device's currency account will run out of balance, through which EdgeChain will be able to quickly identify, control, and contain malicious nodes or devices in the network without requiring them actually to be involved in specific security procedures. EdgeChain can easily take further measures, such as putting the devices into the blacklist or blocking the specific devices for further actions. Since smart contracts are based on blockchain, all the activities are recorded into the blockchain. Thus, it is very difficult for any malicious nodes to cause sustained damage or run away with no traces. As an ongoing research effort, we are designing intelligent methods to learn the devices' history behavior pattern based on the data logged in the blockchain to more accurately identify and recognize potential malicious behavior.

D. Evolutionary and Backward Compatible Approach

We realize the fact that there are a large number of cheap IoT devices that may have very limited security capabilities or are being very poorly maintained and barely upgraded. Though the Google's Android Things 1.0 [4] has just been released trying to work on this, it still has a long way to go. There are some extremely incapable IoT devices, such as narrowband IoT devices. It may be infeasible to run even the most lightweight blockchain client software. We classify these devices as legacy devices which are EdgeChain-unaware. The other type of devices are relatively capable enough to install with blockchain and smart contracts software and act as a blockchain client. We classify them as nonlegacy devices.

TABLE I
MODULES IN THE EDGECHAIN FRAMEWORK

Modules	Technologies	Usage
IoT proxy	IoT gateway loading blockchain client	Accommodating the legacy devices in EdgeChain platform
Smart contract interface	Web3 protocol + Javacript API	Interaction channel between smart contracts and IoT devices
Smart contract	Solidity languages on Ethereum platform	Enforcing management rules of IoT device and edge service; maintaining the virtual currency and credit system
Blockchain server	Ethereum private blockchain platform	Execution platform of smart contracts; recording the activities of IoT devices and edge servers
Application interface	Node.js framework	Interaction channels between edge servers and smart contracts or IoT devices
Edge resource provisioning	Virtual machines	Providing edge service to IoT devices

Nonlegacy devices are able to interact with EdgeChain directly and request resources and assistance from the edge servers. Legacy nodes are unaware of the existence of and incapable of working with edge servers.

The EdgeChain framework adopts an evolutionary and backward compatible approach allowing legacy or extremely incapable IoT devices to work in the new paradigm without assuming them to install new blockchain software or to be updated regularly. The EdgeChain system level capability enables measuring, monitoring, and controlling resource usage of both current and previously installed IoT devices. This goal is achieved through a proxy that works between the legacy IoT devices and the blockchain and smart contract modules, through which the blockchain and smart contracts run transparently to the legacy devices. The proxy sniffs the activities of the legacy nodes and creates blockchain accounts for them just as for nonlegacy nodes. In such case, EdgeChain only monitors the behavior and take necessary action if detecting malicious activities. It will not involve allocating edge server resource for the devices. Through the proxy, the legacy IoT devices are not required to know anything about blockchain and smart contracts but they can still be monitored, managed, and controlled by the new Edge-IoT framework. Even if they are compromised by hackers, their malicious behavior can be identified and damages can be contained.

E. Standalone Deployment Versus Distributed Deployment

Another important advantage with EdgeChain is the ability to be tailored to the need of the intended application. This allows it to be deployed in both stand-alone modes, such as in a smart home as well as distributed modes, such as a smart campus or smart city scenario. Fig. 3 shows a simple example of a standalone EdgeChain box that is deployed in a smart home. In larger scale use cases and applications, such as smart campus and smart cities, multiple such EdgeChain boxes could work in a fully distributed environment, in which cases the distributed boxes work together and share the blockchain and smart contracts data. The edge servers are also able to offload and handover workloads with each other in busy situations. The edge servers can also run appropriate incentive or gaming algorithms associated with their resource pool and blockchain coin accounts to optimize specific goals in revenue, cost, or service latency.

III. EDGECHAIN FRAMEWORK AND FUNCTIONAL MODULES

In this section, we discuss the overall EdgeChain framework and functional modules. The overall system framework

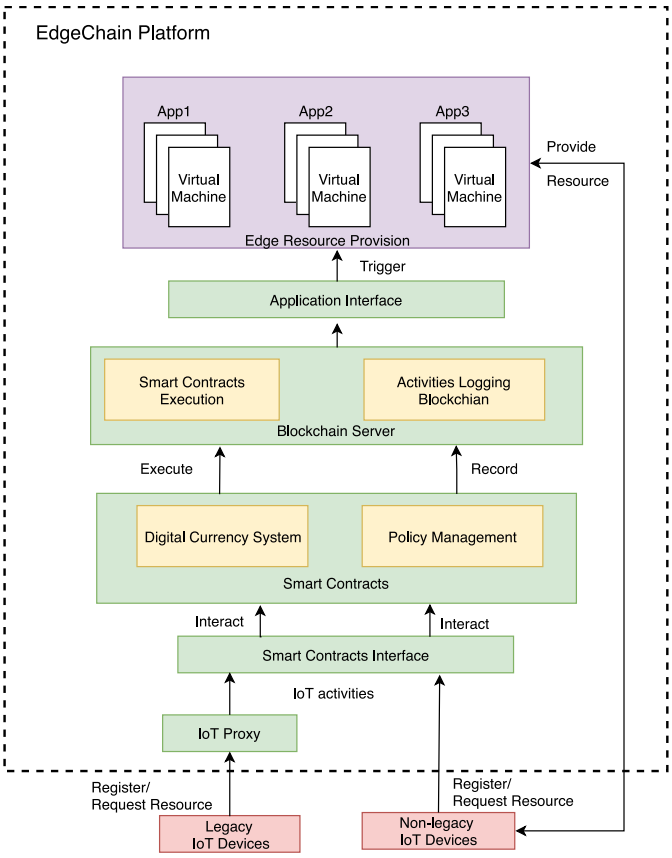


Fig. 4. EdgeChain framework and functional modules.

is shown in Fig. 4. We can see that the EdgeChain sits between the IoT devices and the edge servers listening to messages and performing corresponding tasks which include device registration and device requests processing. Along the message path, the key modules of EdgeChain include IoT Proxy, smart contracts interface, smart contracts, blockchain server, and application interface. We discuss these modules in a bit more details.

The Table I summaries the modules, their technologies and usage in our framework.

A. IoT Proxy

As we discussed in Section II-D, the major function of the IoT Proxy module is to accommodate the legacy devices and facilitate their interactions with the blockchain and smart contracts modules. The proxy listens and sniffs the legacy nodes' activities and creates blockchain accounts for them. Registration is done for them in the same way

as nonlegacy nodes so that the IoT behavior regulating and auditing functions work for them as well. Thus, all their activities are recorded in blockchain as nonlegacy nodes. In contrast, the nonlegacy devices can interact with smart contract directly and get can get accounts created themselves through the smart contracts interface. Implementing this proxy server function requires appropriate sniffing software and we are currently investigating the most effective open-source tools for the EdgeChain project purposes.

B. Smart Contracts Interface

When the IoT activities occur, such as registration, communicating between IoT devices, requesting edge server resources, or sending data to outside servers on the Internet, preprogrammed, and deployed smart contracts will be triggered to automatically perform corresponding operations and enforce the predefined management rules or policies. Smart contracts interface builds a bridge between the IoT applications and the smart contracts. In our implementation, we utilize the Javascript-based APIs, named Web3 protocol, to create the smart contract instances for IoT devices. Smart contract instances can call the functions and perform the rules that were encoded in the contracts on behalf of the specific IoT devices.

C. Smart Contracts

The smart contracts, as the containers of all the rules and policies, consist of two main modules in the EdgeChain system. First, we build a digital currency system whose token are virtual coins to represent the trust levels of IoT devices or their quotas of edge resources they can get. Since every IoT device is bound with a blockchain account, it will be assigned with a certain amount of coins based on its history behavior and resource type. For example, if a device keep behaving well without any malicious actions, it will receive more coins to pay for more service resources. Otherwise, the device may be penalized by being charged more coins to receive the same services or never being rewarded. Second, a module of policy management maintains all the rules that were determined at the time of their creation. The policies can be divided into two types: 1) rules to analyze behavior of IoT devices and handle harmful ones and 2) resource allocation policies to dynamical assign resource to the requests and schedule tasks.

D. Blockchain Server

In our implementation, the smart contracts are deployed and distributed on the blockchain. The blockchain server provides blockchain service where the IoT devices connect to it as clients. Two functions are performed on the blockchain server. First, the server executes the smart contracts by collecting the transactions among devices and generating the new blocks to run the code embedded in the contracts. Seconds, all the activities in our system are recorded on the blockchain by automatically logging device information, requests and other activities on blocks. This process is also called "mining" in the permissionless blockchain. However, as discussed in Section II-A, the EdgeChain mining process is a lot less resource intensive due to the possible usage of more effective consensus mechanism, such as PBFT [34] and no need for PoW mechanism.

E. Application Interface

The application interface bridges the communication between blockchain server and edge cloud servers. After the interaction with smart contracts and blockchain, there are two possible outcomes: 1) the requests are either rejected due to limited balance in their accounts or malicious behavior identified or 2) the requests are accepted and granted with permission to receive extra edge resource from the edge servers. If a request is granted, then the IoT device continues to push up the request to the edge server for corresponding resources. Once confirming that there are still enough resources, the edge servers will interact with the smart contracts of resource allocation policies to permit the requests and schedule the service. With the final permission, the IoT devices can interact with the IoT applications running on the edge servers directly. For example, resource-intensive works, such as face recognition from the video streams can be offloaded to the edge servers for faster processing. In this case, application interface first opens the channels between smart contracts and the edge cloud to trigger resource provision according to the execution results from smart contracts. Then the granted IoT devices receive the edge service and the resource is delivered through the interface. We implemented these functions using the *Node.js* frameworks to listen to the events on the channels and establish communications for IoT devices and edge cloud accordingly.

Note that in terms of delay and time cost, it is true that smart contracts and blockchain operations are not for free and it could take a certain amount of time to finish. The good news is that registration is usually a one-time operation for a specific device. For resource request with the edge servers, after the initial request is granted, the resource provisioning and interactions happen directly between IoT devices and edge servers which will not cause further delay. We conduct very detailed evaluation and experimentation in Section V.

F. Edge Resource Provisioning

Once the IoT devices are granted with resources and their accounts are with enough balance for the requested resources, the edge cloud will provision resources in computation, memory, storage, networking, and intelligence accordingly. Since the application may have various requirements for computer capability, bandwidth, latency, and privacy, individual virtual machines work as the basic units to meet the specific resource requests. For example, for the video stream-based face recognition application example we mentioned, the edge servers could spawn and launch additional virtual machines to process the video stream and get the face recognized. If not sufficient resource available from this edge server, EdgeChain can coordinate with neighbor edge servers to get additional resource. Additional incentive mechanisms and dynamic pricing schemes using game theory or auction can be useful to optimize certain goals in revenue or cost. The IoT devices accounts will be charged accordingly based on the service amount and quality they receive.

IV. EdgeChain Key Processes and Workflows

With all EdgeChain framework and modules, we will discuss the critical processes and workflows in this section.

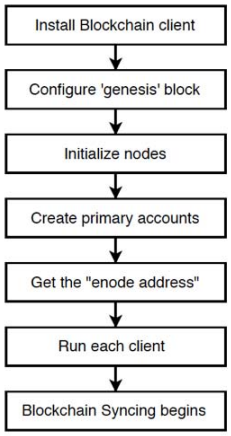


Fig. 5. Blockchain implementation workflow.

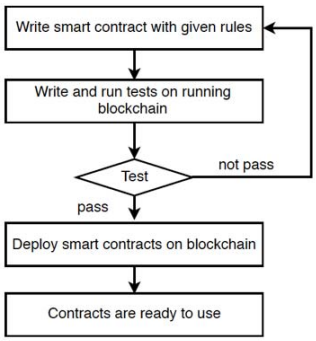


Fig. 6. Smart contracts implementation workflow.

A. Blockchain Deployment

Blockchain implementation can be performed in a distributed way on the edge servers and user devices, and get synchronized across these nodes. We begin by installing blockchain software on the edge server, nonlegacy devices, and the IoT Proxy. Our blockchain is built on the Ethereum platform [47] which is initialized by default to sync with a live public network. However, our EdgeChain system is currently developed for the experimental purpose, so we configure it for use on a private network on campus.

Fig. 5 shows the workflow of blockchain deployment. The blockchain begins with creating a "genesis" block, which holds configuration information, such as the hash value of blockhead, timestamp, and difficulty of block mining. It is worth noting that the amount of difficulty makes a significant influence on the mining speed and then on the global system performance since the mining process is realized by solving a PoW problem with a certain difficulty. Given that only the edge server is permitted to do the mining job, there is no need for a rigorous PoW mechanism to solve the consensus problem. Therefore, our EdgeChain system sets the difficulty to a reasonable low level to balance between over quick mining to avoid storage waste and efficiency of packing transactions. To further reduce the resource consumption of the edge server, we implement an auto-mining function only occurring when there exist unconfirmed transactions.

To sync with one another, all devices must have the same genesis block. The initialization process will provide each node with same genesis configuration. Next, a primary account must be created for each node and public keys are assigned for unique identification. The account gives each node a blockchain address with which it can interact with other nodes and smart contracts. To isolate our system from other public or private blockchains, all nodes are set "no discovery" so they cannot connect to other peers without explicit addresses. Such isolation secures the devices from being hooked by external attackers. Thus, each node maintains a specific whitelist called "enode addresses" which contains the public keys, IP addresses and network ports of the edge server and some dependent IoT devices. Adding the enode addresses to each node's configuration will allow syncing to occur. Upon completion of the above steps, each node is ready to launch. They will begin seeking friend nodes, syncing and shortly be prepared for use.

B. Development and Deployment of Smart Contracts

The proper development of smart contracts guarantees the correct execution of management rules. In our EdgeChain system, the key functional operations including device registration and edge resource allocation are enforced by the corresponding contracts. We deploy smart contracts following the workflow in Fig. 6. When developing a smart contract on the blockchain, it is important to run thorough tests because once deployed, a contract can only be redeployed and lose any data associated with the previous version. Such a redeployment would migrate the contract to the new location and the users may be outdated with an unsupported contract. After deployment, smart contracts are assigned with addresses and treated as normal accounts on blockchain. In order to interact with them, a user must have a copy of the correct address to create an instance as an interface utilizing remote procedure calls protocol. The edge server is the performer to execute the functions in the contracts when the IoT devices are the initiators to trigger them.

The smart contracts specify various permissions to different devices where the edge server owns the higher authority to access all the functions but the IoT devices are only limited to some basic functions. Such a setting reduces the impact even if some weak devices are hacked to perform malicious activities. To help engage the legacy nodes into the system, a proxy is deployed in order to fulfill their interaction requests. Other than the direct interaction launched by the nodes, smart contracts are also able to indirectly interface with the outside world by triggering "events" which are watched by application interfaces running on the edge server or other nodes on the network. Upon noticing an event of an application, some smart contract can be automatically triggered to execute the predefined tasks. For example, after the edge server finishes serving one user requests, the related service data like service time would be recorded on blockchain by executing a specific contract.

C. Device Registration on Blockchain

Registration is the first step to engage the IoT devices to be managed and monitored in the EdgeChain system. As

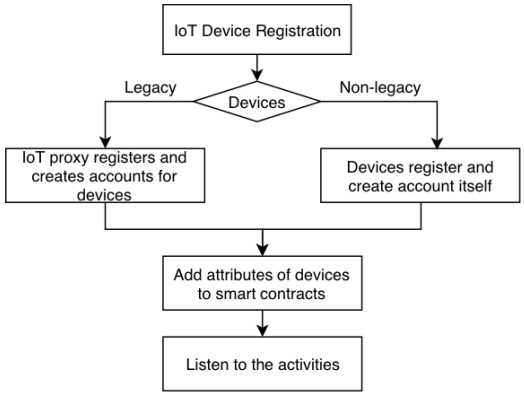


Fig. 7. Devices registration workflow.

TABLE II
REGISTERED DEVICE ATTRIBUTES

Device Specifications	Value Unit	Example
account address	string	0xc968efa8019d (hash value)
network port	int	42024
input/output data	string	video,audio,text
bandwidth request	double	[minValue, maxValue]
CPU request	double	[minValue, maxValue]
memory request	double	[minValue, maxValue]
storage request	double	[minValue, maxValue]
MAC address	string	00-14-22-01-23-45
priority*	int	1 / 2 / 3 / 4
coin balance*	double	200.00
credit*	int	100
isBlocked*	bool	false
isRegistered*	bool	false
last request id*	string	0xcf30613db6a84 (hash value)

illustrated in Fig. 7, the registration starts from determining the type of devices. If there are legacy devices lacking the capability to run blockchain, the proxy can create accounts for each device and register the device specifications stored in the registration smart contracts. If there are nonlegacy devices, they can interact with contracts directly to save their attributes by sending transactions.

The registered information makes decisive effect on the request admission introduced in the next section. Specifically, the device specifications partially reference the manufacturers usage description (MUD) [48] files which list the activities and communications allowed for IoT devices. Such specifications contain input/output data type, requests of edge resources, MAC address, IP address, network port, communication protocol, and indication flags. Besides, each device registers a unique account address to join blockchain. Upon registration, the edge server will verify the above information and take control of the modification rights of registration data. More parameters will be appended, such as priority, coin balance, credit, and requests timestamp to benefit device management. As a summary, Table II represents the key device attributes we defined in the registration database which include all the devices key information, value units, and examples. Edge server and IoT devices have different authorities to modify the registry. The attributes marked with "*" can only be updated by the edge server. The other basic attributes are filled up during the first registration process initialized by IoT devices.

D. IoT Behavior Regulation and Activities Management

The IoT behavior regulation and activities management is the core function of our EdgeChain system for IoT scalability and security. In this section, we explain the critical designs in the following order: detailed workflows, edge resource allocation algorithms and behavior management scheme.

1) IoT Behavior Regulation Workflow: EdgeChain not only regulates the activities among IoT devices but also provides the extra edge computing service to boost the resource-intensive applications. When the activities or the requests from IoT devices are received, they are treated differently based on the type of devices, either legacy or nonlegacy devices. Legacy devices have no request for the support of edge cloud to handle the additional workload. Nonlegacy devices could request to obtain edge resource and services under the enforced rules of smart contracts. The detailed workflow is shown in Fig. 8 and discussed below.

For a legacy device, the blockchain server monitors its data flow to other IoT devices or outside network through a sniffer deployed on the IoT gateway, such as a WiFi router. During the work process, its activities or behaviors, such as network port and data destination, are logged on blockchain. Then the smart contracts start analyzing the behavior of the device by matching the above observation with the registered attributes. Based on the analyzing results, the blockchain server will choose to keep monitoring the normal behavior. Or it will trigger the smart contract to block any malicious legacy devices and update flags in their registration files. Their future activities will be detected and blocked automatically without performing behavior analysis again. Finally, the execution results of the related smart contracts will be stored on blockchain automatically.

For nonlegacy devices, they may send service requests for additional resources for resource-intensive applications, such as virtual reality (VR) gaming. Once received, the requests are recorded on blockchain in the form of transactions. Next, the resource allocation contracts are executed by the edge server to retrieve the attributes of the devices and analyze the resource requirements in the service requests. If the devices are found to attempt malicious behavior, they will be penalized by reducing their coin balance, lowering credit points and even blocking service for all future requests. If the devices behave normally, the edge cloud will first check the remaining available resource before further process the requests. If the resource pool is exhausted, the requests is rejected and logged. Otherwise, smart contracts perform the resource allocation strategy based on the device types, request details and payable coins. After obtaining the decisions, the edge server starts to schedule the service for the devices immediately. In the meantime, coins will be charged from the devices' account when the edge service begins. Again, the decisions and coin exchanges are all recorded on blockchain.

2) Resource Allocation Based on Pricing Mechanism: In this specific instance, our optimization goal of resource allocation is to maximize the acceptance rate of user requests. In this case, the currency system plays as the connector among edge server, IoT devices and blockchain by linking edge resource

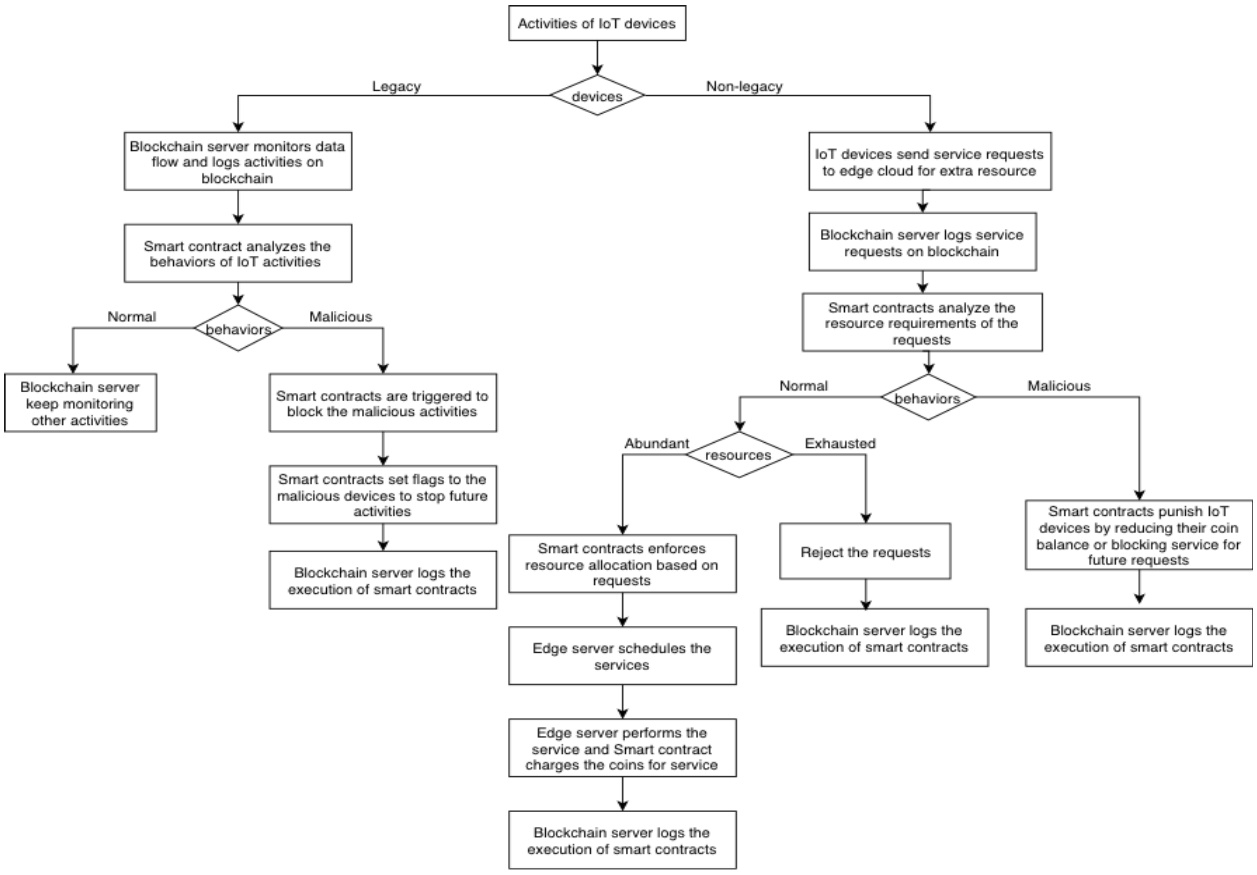


Fig. 8. IoT activities management workflow.

with coins. Our proposed currency system is built on a pricing mechanism to decide: 1) the ordering of the requests may be served and 2) the specific service fee.

The price of a resource request dynamically changes according to the following environmental parameters.

1) Total amount of edge resources.
2) Current available edge resources.
3) Requested edge resource.
4) Application priority.

Considering the QoE requirements, we categorize the priority of IoT applications into four levels, from highest to lowest: 1) urgent monitoring: patient monitoring and people crowd sensing; 2) latency sensitive tasks: VR and AR; 3) reliable data transmission: bank transactions and privacy transferring; and 4) tolerant tasks: light control and sensors-based passive monitoring.

Table III shows the symbol notations used to calculate the price. We first define the unit price of resource j for the request i

$$P_{i\_j} = \alpha^{\frac{r_{i\_j}}{c_j}} * \beta^{L_i}. \tag{1}$$

Then the total price for request i is defined as, where $c_j \in [0, w_j]$

$$P_i = \sum_j^M r_{i\_j} * \left[\alpha^{\frac{r_{i\_j}}{c\_j}} * \beta^{L_i}\right] = \beta^{L_i} \sum_j^M r_{i\_j} * \alpha^{\frac{r_{i\_j}}{c\_j}}. \tag{2}$$

TABLE III
PARAMETERS OF PRICING MECHANISM

Symbol	Definition
N	amount of requests at timeslot t
M	number of resource types
R	requested resource $R = \{r_{i\_1}, r_{i\_2}, ..., r_{i\_M}\}$
C	current availabe resource $C = \{c_1, c_2, ..., c_M\}$
W	total resources $W = \{w_1, w_2, ..., w_M\}$
L	priority level
K	amount of accepted requested at timeslot
α	constant basic price value
β	influence factor of priority

The unit price depends on the application priority and available edge resources (1), while the total price also for a specific request includes user's resource requirements (2). Using the dynamic pricing scheme, we propose a heuristic request admission algorithm as illustrated in Algorithm 1. The proposed algorithm proceeds as follows. At the beginning of timeslot t, the number of requests is N and the number of resource types is M. For each request $r_{i\_j} \in R$, judge if any kind of left edge resource r_{i_j} is less c_j. If yes, the request is rejected without consideration in this timeslot. If there still have enough resources, calculate the total price of the requests. After all the requests are estimated, the one with the lowest price value is accepted and added to acceptance queue. Then the amounts of available resources C are updated. The rest of requests are re-estimated in the next iteration. The algorithm continuous

Algorithm 1 Request Admission Algorithm

Require: N requests $\{req_1, req_2, \ldots, req_N\}$ at time t, request queue $Q(t)$, current available amount of resources $C = \{c_1, c_2, \ldots, c_M\}$, requested resource $R = \{r_{i\_1}, r_{i\_2}, \ldots, r_{i\_M}\}$, priority of $req_i L_i$.

Ensure: accept or deny request req_i

```
while there exists resource for at least one request do
    for each req_i in the request queue arrived at timeslot t do
        if r_i_j > c_j then
            deny request req_i;
            continue next iteration;
        else if r_i_j <= c_j then
            calculate the total price P_i;
        end if
    end for
    accept req_i with minimal price P_i;
    remove the accepted request from Q(t);
    update the available edge resources C;
end while
EXIT;
```

until no request can be admitted. Assume the final acceptance number of request is K, we can conclude the time complexity is $O[(N * M + 1 + M) * K] = O(N * M * K)$, where $K < N$. Therefore, the algorithm can be solved in polynomial time.

3) Behavior Management Based on Credit System: Behavior management aims at detecting the potentially malicious activities or requests and taking action to avoid further damage to the system. We propose a credit system to perform the behavior management. Our credit system is distinguished from other similar schemes in the IoT environment because the credit affects resource allocation on the edge server instead of the cooperations between IoT devices. On the other hand, the credit is not directly related to price strategy for edge service but make up the incentive or punishment scheme to restrict the request activities. In this paper, we present our ongoing design and the primary model to show how the behavior management works. We consider the following features.

1) *Resource Requests:* The amount of resource requested by devices indicates whether the devices work in the normal mode. Excessive resource requirements potentially represent abnormal behavior and malicious attempts.
2) *Price Threshold:* Assume each device only runs one kind of application and sends one kind of resource request, a specific price threshold P_{thres} is set for this device which means the maximal reasonable price. And the total price for its current request is represented as P_{total}. If P_{total} exceeds P_{thres}, the request is regard as potential bad behavior so the device credit is reduced. Otherwise, the request is regard as good behavior and credit increases.
3) *Request Frequency:* If a device continuously send requests in an overhigh frequency, it tends to occupy resource than the common use. So we reduce its credit.
4) *Network Port:* A device should communicate with the edge server using the predefined network port in the MUD file. Otherwise, some abnormal behavior happens.
5) *Data Traffic Destination:* A device usually has fixed communication targets, so the strange destination indicates the possibility the device is hacked or under control.

Given the above features, the system monitors the behaviors of IoT devices by comparing their normal activity patterns to the incoming requests. If any feature is observed abnormal, the credit value of the corresponding device will be reduced. Each new registered device owns the same initial credits. With the changes of the real-time credit values, we propose two kinds of management actions: 1) if the credit of a device has already been reduced to 0, it is blocked for any future activities and 2) otherwise, the device will get various coin returned based on the credit changes. The equation is defined as follows:

$$\text{Coins}_{\text{return}} = \text{Coins}_{\text{charged}} + \Delta\text{Credit} * \eta \tag{3}$$

where ΔCredit is the change of credit value and η is the influence factors of changes.

We conclude that the ability to pay for edge service is under the control of the credit system. The better manner receives higher chance to obtain more resources. For better evaluating abnormal behavior, we plan to implement fine-grained machine learning-based behavior profiling and anomaly detection methods in our future work.

V. Prototype and Evaluation

In this section, we first introduce our experimental testbed built as the EdgeChain prototype. Then, we implement the key functions to verify if it is feasible with acceptable performance overhead. In the third part, two typical IoT applications in different service priorities are deployed on the EdgeChain system to show the compatibility between blockchain and applications. Finally, we test the performance of the pricing-based resource allocation system.

A. EdgeChain Prototype Environment Setup

The testbed includes the back-end edge cloud cluster and the front-end IoT devices, proxy, and access point. The edge cloud cluster is an OpenStack deployment including, 4 high-performance Dell PowerEdge R630 rack servers, 1 high-performance Dell PowerEdge C730x rack server, and 1 high-performance Cisco 3850 switch. The front end consists of several Raspberry Pi 3 Model B single board computers, a Google AIY voice kit, a Google AIY vision kit, and a laptop. One desktop is configured as the proxy for legacy IoT devices, and a high-performance Cisco WiFi Access Point, as illustrated in Fig. 9

The detailed hardware and software configurations are as follows. From the aspect of hardware, each OpenStack compute node rack server is equipped with 18 independent CPU cores and 256 GB RAM. The mining environment is set up using one core and the rest of the processor cores are reserved for the edge computing service. The miner can boost up to 3.5 GHz CPU, 8 GB RAM, and 1 TB storage. As the IoT devices, a Raspberry Pi has 1.2 GHz CPU, 1 GB RAM, and 32 GB storage with several accessory modules including cameras, sense hat, microphone and Google bonnet. The laptop has 2.2 GHz CPU, 4 GB RAM, and 256 GB storage. As for the desktop proxy, 3.2 GHz CPU, 16 GB RAM, and 1 TB storage are installed to manage the multiple blockchain accounts of IoT devices.

Regarding the software, the edge server has installed with *CentOS 7* as the operating system, *Go-ethereum* as the

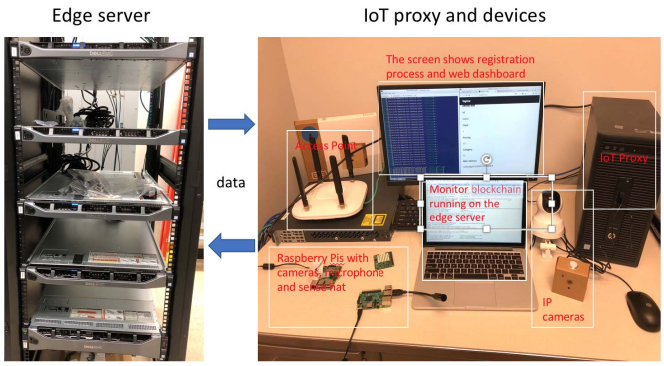


Fig. 9. EdgeChain testbed.

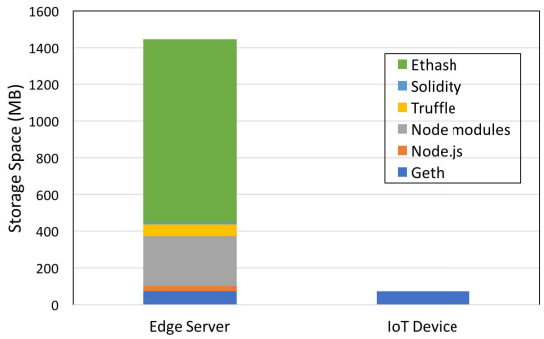


Fig. 10. Storage of prerequisite software.

blockchain running framework, *Solidity* as the smart contract development language, *Truffle* as the contract deployment tool, and *Node.js* as the interface of interactions between IoT applications and blockchain. Except for the blockchain part, the edge computing resources are virtualized using OpenStack cloud platform which helps scale up or down the resource pool flexibly. The edge service is provided in the form of virtual machines to fit the variant specifications of user requests. The Raspberry Pis have been installed with Raspbian operating system and *Go-ethereum* to work in the light mode without block mining function. The laptop is with MacOS and the desktop installs Ubuntu 16.

In the testbed, the rack server works as the edge service provider and the block miner solving PoW puzzle. The Raspberry Pis and the laptop act as blockchain clients generating and sending transactions of resource requests to the edge server. The desktop interacts with the blockchain on behalf on the legacy devices as a proxy. Given the above installations, the edge server works as a "full" blockchain node which stores all the transactions, executes the predefined smart contracts and mines new blocks. The IoT devices work as "light" blockchain nodes which only store the transactions data. Fig. 10 shows the storage requirements for the prerequisites software modules, where *Ethash* is the PoW system used to mine blocks. We put most of the computation work occurring on the blockchain to the full node in order to reduce the overhead on the light nodes.

B. Overhead of Blockchain and Smart Contracts Operation

We evaluate the blockchain operation based on the two primary functions: IoT devices registration and edge server resource allocation to illustrate the extra overhead caused by the block mining and the interactions of smart contracts. The source of overhead can be divided into three aspects: 1) computation; 2) communication; and 3) storage.

1) Computation Cost of Mining Process on Edge Server: We first evaluate the overhead of device registration in which device specifications are loaded in the transactions signed by their generators. Then the transactions are broadcasted to all the other devices engaged in our system. Finally, these new transactions are packed in the blocks and verified by the miner. We observe the average usage of computation resource on the edge server during mining and no mining, as illustrated in Fig. 11(a). During the block mining, the edge server consumes much higher CPU and memory resource to commit and packs transactions into new blocks. In contrast, in the idle situation, it only listens to coming transactions, such as mining new block caused by new transactions thus consume much less CPU and memory resource.

2) Communication and Storage Cost for Blocks Synchronization: Given that blockchain is the fully distributed, each device is required to be synchronized with the mainstream chain. The synchronization mechanism relies on the automatic updates and leads to the communication and storage overhead to the system, where the former results from the data transmission and the later from the writing to the local disk. In our system, the edge server maintains the mainstream blockchain and other devices download the chain data from it. In order to evaluate the synchronization delay intuitively, we compare IoT devices to the edge server. Since the edge server as the miner has more computing and bandwidth resource than IoT devices, it completes the validation and transmission of the new blocks faster. As illustrated in Fig. 11(b), we find the average time to synchronize a new block is 4.09 ms for edge server and 35.9 ms for IoT devices. With higher delay, the IoT devices still meet the latency requirements even for the real-time applications that response time is less than 100 ms.

The average size of a block is 128.78 kB and each block can store up to 208 device registrations. Fig. 11(c) presents a sample of 50 blocks which have various sizes ranging from 108 to 223 kB. Thus, the system will generate around 1.8 MB blockchain data on average for 1000 devices' registration.

3) Computation and Communication Cost of Smart Contract Transactions: In addition to block mining and synchronization, blockchain operation relies on the transactions triggered by the smart contracts. Taking the resource request transaction as an example, we evaluate the computation cost and the interaction delay with smart contracts. The CPU and memory usage are compared between the edge server and IoT devices, as illustrated in Fig. 11(d). We observe that the regular transactions take a very low percentage of CPU resource while the memory usage is little higher since the blockchain client occupies 8% even in idle time. We also evaluate the interaction delay of smart contracts which is significant to guarantee system efficiency. Fig. 11(e) shows the completion of one transaction is less than 50 ms. Such delay should satisfy the latency requirement of the real-time applications.

We further discuss the possible impacts to the blockchain overhead when a large number of IoT devices are connected

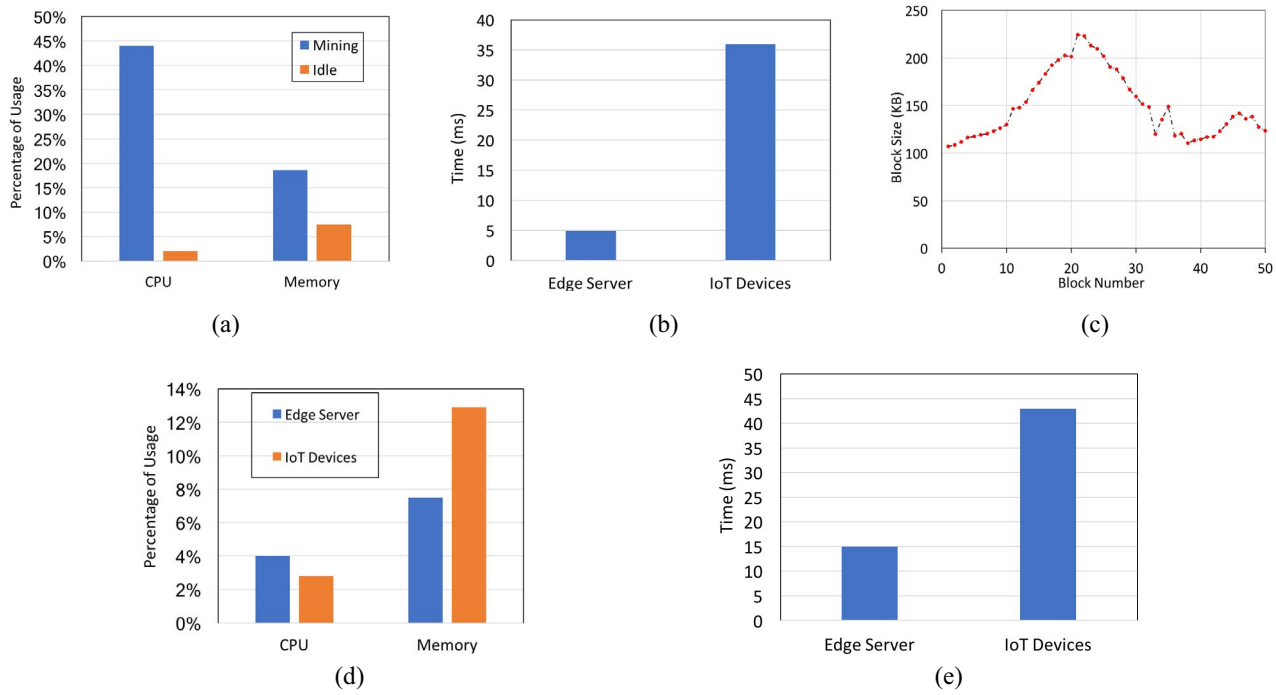


Fig. 11. Overhead of system operation. (a) Computation resource usage of the edge server for mining. (b) Delay to synchronize a block. (c) Block sizes. (d) Computation resource usage for sending transactions. (e) Time to complete one transaction.

in the real production environment. For the miners located at edge servers, it is expected that the computing resources usage for mining, verifying, packaging, and storing new blocks will increase, since many devices may generate more transactions in one unit time. Meanwhile, network traffic between the servers and distributed IoT devices will also increase. To avoid overloading the servers, a specific server may be required to cover only a number of IoT devices, or EdgeChain can launch more VM as the miners to share the load of computation and data traffic. As for the IoT devices, no extra overhead is introduced to them since they work in the client (light) mode where only relevant blocks are synchronized and the transactions generated by one device are steady.

C. Overhead Comparison of Two Typical IoT Applications

To evaluate the feasibility and compatibility of the proposed system, we compare blockchain overhead of two typical Edge-IoT applications. We evaluate the face recognition and the natural-language processing applications by testing the computation and communication cost. Face recognition is widely used in the security monitoring applications, such as city surveillance, crowd control, and door guarding which is latency-sensitive to achieve quick reaction. The typical application of the natural-language processing or voice recognition is the smart home assistant, such as Google Home and Amazon echo.

For the face recognition, the Raspberry Pi captures video frames with camera module in 1080p resolution and 60 Hz frequency, uploads them to the edge server for image processing and waits for the detection results in the form of location coordinates of detected faces. With regard to the natural-language processing, the Raspberry Pi records the human voice with a USB microphone, transfers it to the edge server, and then the translated text is returned.

TABLE IV
COMPARISON OF COMMUNICATION RATE

Applications	Data Rate
Blockchain Transactions	0.54 KB/s
Face Recognition	1.64 MB/s
Natural-language Processing	8.12 KB/s

We first evaluate the computation cost of blockchain comparing with the two applications. Fig. 12(a) shows that the blockchain has the lowest CPU usage compared with the two applications. In addition, Fig. 12(b) shows the blockchain has the highest memory usage but still in a low percentage when working with other applications in parallel. Thus, the IoT devices will not suffer from the overload problem. Second, we evaluate the difference of communication data rate among sending blockchain transactions, video and audio data on a Raspberry Pi, as reported in Table IV. We observe that the regular transactions of resource requests bring very low overhead to the I/O performance and overall network bandwidth.

In summary, we observe that the blockchain can support and collaborate with the IoT application in a distributed and secure way. The overhead is within a reasonable and acceptable range, and the system is feasible to satisfy the requirements to build a multiapplication EdgeChain platform for future demands.

D. Resource Allocation Performance of the Pricing Scheme

At last, we evaluate the resource allocation performance of the proposed pricing scheme. The goal of resource allocation is to improve the acceptance rate of use requests, which mainly depends on the proposed pricing mechanism.

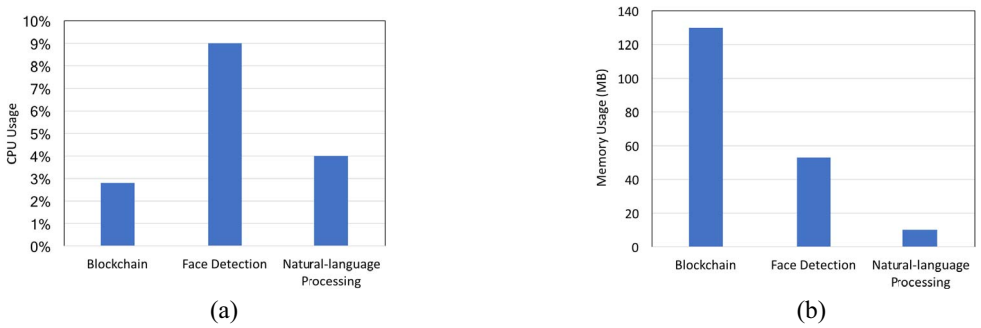


Fig. 12. Overhead comparison with IoT applications. (a) Comparison of CPU usage. (b) Comparison of memory usage.

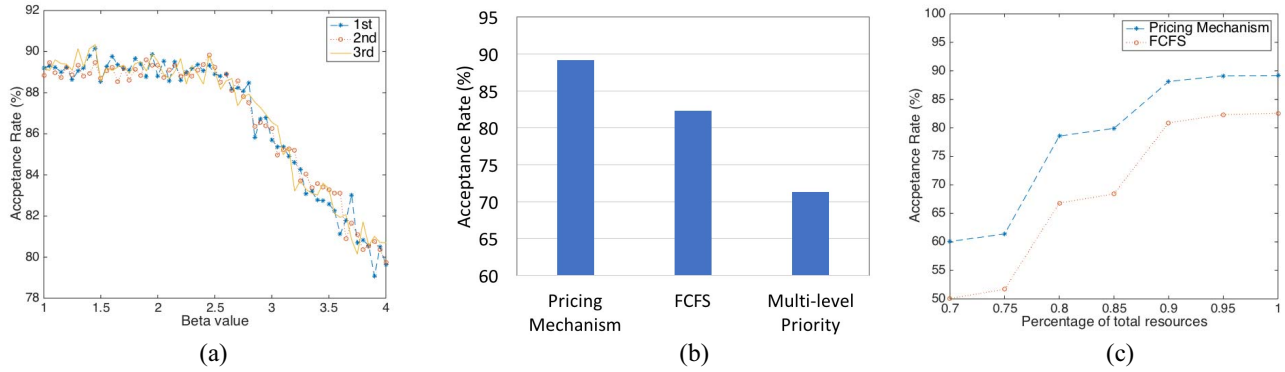


Fig. 13. Resource allocation performance. (a) Influence of β value. (b) Acceptance comparison with constant β. (c) Acceptance comparison with resource change.

TABLE V
SYSTEM PARAMETERS

α	100
CPU capacity	300
Memory capacity	250
Storage capacity	250
Bandwidth capacity	250

TABLE VI
REQUESTS PARAMETERS

Priority	CPU	Memory	Storage	Bandwidth	Lifetime
Level 1	[1,5]	[1,5]	[1,5]	[1,5]	[1,5]
Level 2	[10,15]	[5,10]	[5,10]	[1,10]	[1,5]
Level 3	[1,5]	[1,5]	[1,5]	[1,5]	[1,5]
Level 4	[1,3]	[1,3]	[1,3]	[1,3]	[1,3]

We first evaluate the influence of α and β. α has no effect on the performance since it determines the range of $\alpha^{[(r_{i\_j})/c\_j]}$ is located in [1, α]. In contrast, β adjusts the impact of application priority where the high-priority requests are more likely to be served. We do three random simulations and each one contains 2000 iteration of random numbers of user requests with different resource requirements. The system parameters are set in Table V and the request parameters are set in Table VI. Fig. 13(a) shows the best range of β is in [1.3, 1.4] and the too large β will lead to decrement of acceptance rate since the requests admission simply depends on the priority.

Second, we compare the acceptance rates among the pricing mechanism, first-come-first-serve and multilevel scheduling based on priority, where $\beta = 1.35$. Fig. 13(b) shows that our proposed algorithm performs best. Then, we evaluate the performance with the change of total edge resources, as illustrated in Fig. 13(c). Starting from the configuration in Table V, the amount of resources gradually decreases to lower percentages. Our pricing algorithm performs better.

VI. RELATED WORK

Due to the interdisciplinary essence of EdgeChain, related work comes from different aspects, such as IoT, edge computing, blockchain, and smart contracts. A great amount of efforts have been focused on these individual topics, thus, limited by the space, we will not enumerate all the separate efforts. Instead, we will focus on those directly or closely related work.

The most closely related work are Xiong *et al.* [27], [28] in which the authors proposed a pricing scheme to maximize the profit of the edge or cloud service providers by exploring the Stackelberg equilibrium between the blockchain miners and the service providers. Different to our objective to support IoT applications, such a pricing scheme aims to optimize the resource management on the cloud servers to support the block mining of the distributed miners for blockchain networks. It focuses on the blockchain running costs. Chatzopoulos *et al.* [31] focuses on computation offloading between devices themselves by using some incentive and reputation schemes. They designed a truthful auction strategy and a mutual reputation scheme for bidding the edge service with a fix pricing mechanism for the unit resource.

Sharma *et al.* [36] proposes a conceptual software-defined edge nodes scheme using multilayer blockchain. Different from these work, our research focus is not on blockchain itself. Instead, we use blockchain as carrying vehicle to provision resources for various IoT applications and control and regulate IoT devices' behavior. More reviewing articles [33], [37], [38], [40], [45] presented the overall future prospects in combining blockchain and IoT.

Blockchain and smart contracts are being used to secure many different areas and we will not enumerate them here, but a few example efforts include securing smart home [39], securing 5G fog network handover [41], securing virtual machine orchestration [42], securing access control in IoT [43], and secure data provenance management [44].

Another thrust of related work is about edge computing research. A large amount of existing work are either on specific applications, such as video analytics, vehicular network, cognitive assistance, and emergency response, or very heavily focused on optimizing specific targets, such as revenue, cost, delay, or energy consumption associated with operations, such as mobile edge offloading, service migration, virtual machines chaining, placement, and orchestration. We will not list all of these works but two good start reading points are [5], [6].

VII. Conclusion and Future Work

In this paper, we discussed the design and prototype of the EdgeChain framework which is a novel Edge-IoT framework based on blockchain and smart contracts. EdgeChain integrates a permissioned blockchain to link the edge cloud resources with each IoT device's account, resource usage and hence behavior of the IoT device. EdgeChain uses a credit-based resource management system to control the IoT deivces' resource that can be obtained from the edge server. Smart contracts are used to regulate IoT devices' behavior and enforce policies. With these new designs, all the IoT activities and transactions are recorded into blockchain for secure data logging and auditing. We implemented an EdgeChain prototype and conducted extensive experiments which showed that the cost for EdgeChain to integrate blockchain and smart contracts are within reasonable range while gaining various intrinsic benefits from blockchain and smart contracts. To the best of our knowledge, EdgeChain is the first of its kind of incorporating blockchain in edge computing to provision resources for various IoT applications and to control and regulate the IoT devices' behavior without overloading the devices with full-scale security-related burdens. EdgeChain is still an ongoing project and we are currently working on various issues within the framework, such as IoT Proxy, intelligent resource provisioning for multiple heterogeneous applications, and better IoT device behavior regulations.

References

[1] *CEO to Shareholders: 50 Billion Connections 2020*, Ericsson Inc., Stockholm, Sweden, 2010. [Online]. Available: http://www.ericsson.com/thecompany/press/releases/2010/04/1403231

[2] N. Woolf, *DDoS Attack That Disrupted Internet Was Largest of Its Kind in History, Experts Say*, Guardian, London, U.K., 2016. [Online]. Available: https://www.theguardian.com/technology/2016/oct/26/ddos-attack-dyn-mirai-botnet

[3] Motherboard. (2016). *How 1.5 Million Connected Cameras Were Hijacked to Make an Unprecedented Botnet*. [Online]. Available: https://motherboard.vice.com/en_us/article/8q8dab/15-million-connected-cameras-ddos-botnet-brian-krebs

[4] Ars Technica. (May 2018). *Android Things 1.0 Launches, Google Promises 3 Years of Updates for Every Device*. [Online]. Available: https://arstechnica.com/gadgets/2018/05/android-things-hits-version-1-0-with-centralized-google-update-system/?amp=1

[5] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: Vision and challenges," *IEEE Internet Things J.*, vol. 3, no. 5, pp. 637–646, Oct. 2016.

[6] J. Pan and J. McElhannon, "Future edge cloud and edge computing for Internet of Things applications," *IEEE Internet Things J.*, vol. 5, no. 1, pp. 439–449, Feb. 2018, doi: 10.1109/JIOT.2017.2767608.

[7] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, "Fog computing and its role in the Internet of Things," in *Proc. 1st Ed. ACM MCC Workshop Mobile Cloud Comput.*, 2012, pp. 13–16.

[8] *Mobile-Edge Computing Initiative*, Eur. Telecommun. Stand. Inst., Sophia Antipolis, France, 2016. [Online]. Available: http://www.etsi.org/technologies-clusters/technologies/mobile-edge-computing

[9] M. Satyanarayanan, P. Bahl, R. Caceres, and N. Davies, "The case for VM-based cloudlets in mobile computing," *IEEE Pervasive Comput.*, vol. 8, no. 4, pp. 14–23, Oct./Dec. 2009.

[10] A. Mosenia and N. K. Jha, "A comprehensive study of security of Internet-of-Things," *IEEE Trans. Emerg. Topics Comput.*, vol. 5, no. 4, pp. 586–602, Oct./Dec. 2017.

[11] Q. Jing, A. V. Vasilakos, J. Wan, J. Lu, and D. Qiu, "Security of the Internet of Things: Perspectives and challenges," *Wireless Netw.*, vol. 20, no. 8, pp. 2481–2501, 2014.

[12] D. Puthal, S. Nepal, R. Ranjan, and J. Chen, "Threats to networking cloud and edge datacenters in the Internet of Things," *IEEE Cloud Comput.*, vol. 3, no. 3, pp. 64–71, May/Jun. 2016.

[13] J.-Y. Lee, W.-C. Lin, and Y.-H. Huang, "A lightweight authentication protocol for Internet of Things," in *Proc. IEEE Int. Symp. Next Gener. Electron. (ISNE)*, 2014, pp. 1–2.

[14] Z. Shelby, K. Hartke, and C. Bormann, "The constrained application protocol (CoAP)," Internet Eng. Task Force, Fremont, CA, USA, RFC 7252, Jun. 2014. [Online]. Available: https://www.rfc-editor.org/info/rfc7252, doi: 10.17487/RFC7252.

[15] T. Kivinen, "Minimal Internet key exchange version 2 (IKEv2) initiator implementation," Internet Eng. Task Force, Fremont, CA, USA, RFC 7815, Mar. 2016. [Online]. Available: https://www.rfc-editor.org/info/rfc7815, doi: 10.17487/RFC7815.

[16] S. Raza, T. Voigt, and V. Jutvik, "Lightweight IKEv2: A key management solution for both the compressed IPsec and the IEEE 802.15.4 security," in *Proc. IETF Workshop Smart Object Security*, vol. 23, 2012, Art. no. 1.

[17] X. Yao, Z. Chen, and Y. Tian, "A lightweight attribute-based encryption scheme for the Internet of Things," *Future Gener. Comput. Syst.*, vol. 49, pp. 104–112, Apr. 2015.

[18] Internet Engineering Task Force (IETF). *IETF Light-Weight Implementation Guidance (LWIG) Working Group*. Accessed: Nov. 6, 2018. [Online]. Available: https://datatracker.ietf.org/wg/lwig/charter/

[19] R. Oppliger, "Internet security: Firewalls and beyond," *Commun. ACM*, vol. 40, no. 5, pp. 92–102, 1997.

[20] S. Chen and Q. Song, "Perimeter-based defense against high bandwidth DDoS attacks," *IEEE Trans. Parallel Distrib. Syst.*, vol. 16, no. 6, pp. 526–537, Jun. 2005.

[21] B. Osborn, J. McWilliams, B. Beyer, and M. Saltonstall, "BeyondCorp: Design to deployment at Google," in *Proc. USENIX*, 2016, pp. 28–35.

[22] R. Ward and B. Beyer, "BeyondCorp: A new approach to enterprise security," *login*, vol. 39, no. 6, pp. 6–11, 2014.

[23] "Getting started with a zero trust approach to network security," Santa Clara, CA, USA, Palo Alto Netw., White Paper, 2016.

[24] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," White Paper, 2008.

[25] A. Narayanan *et al.*, *Bitcoin and Cryptocurrency Technologies*. Oxford, U.K.: Princeton Press, 2016.

[26] N. Szabo, "Smart contracts: Building blocks for digital markets," *EXTROPY J. Transhumanist*, vol. 16, 1996.

[27] Z. Xiong, Y. Zhang, D. Niyato, P. Wang, and Z. Han, "When mobile blockchain meets edge computing," *IEEE Commun. Mag.*, vol. 56, no. 8, pp. 33–39, Aug. 2018.

[28] Z. Xiong *et al.*, "Cloud/fog computing resource management and pricing for blockchain networks," *IEEE Internet Things J.*, to be published.

[29] J. Pan, J. Wang, and A. Hester, "Edge-IoT framework based on blockchain and smart contracts and associated method of use," U.S. Provisional Patent 62 681 936, Jun. 7, 2018.
[30] J. Wang, A. Hester, and J. Pan, "Method and system for secure resource management of IoT utilizing blockchain and smart contracts," U.S. Provisional Patent 62 657 387, Apr. 13, 2018.
[31] D. Chatzopoulos, M. Ahmadi, S. Kosta, and P. Hui, "FlopCoin: A cryptocurrency for computation offloading," *IEEE Trans. Mobile Comput.*, vol. 17, no. 5, pp. 1062–1075, May 2018.
[32] O. Garcia-Morchon, S. Kumar, and M. Sethi, "State-of-the-art and challenges for the Internet of Things security," IETF draft, Fremont, CA, USA, draft-irtf-t2trg-iot-seccons-15, May 2018.
[33] K. Christidis and M. Devetsikiotis, "Blockchains and smart contracts for the Internet of Things," *IEEE Access*, vol. 4, pp. 2292–2303, 2016.
[34] M. Castro and B. Liskov, "Practical Byzantine fault tolerance," in *Proc. OSDI*, vol. 99, 1999, pp. 173–186.
[35] M. Satyanarayanan, "A brief history of cloud offload: A personal journey from Odyssey through cyber foraging to cloudlets," *GetMobile Mobile Comput. Commun.*, vol. 18, no. 4, pp. 19–23, Jan. 2015.
[36] P. K. Sharma, M.-Y. Chen, and J. H. Park, "A software defined fog node based distributed blockchain cloud architecture for IoT," *IEEE Access*, vol. 6, pp. 115–124, 2017.
[37] K. Yeow *et al.*, "Decentralized consensus for edge-centric Internet of Things: A review, taxonomy, and research issues," *IEEE Access*, vol. 6, pp. 1513–1524, 2017.
[38] A. Dorri, S. S. Kanhere, and R. Jurdak, "Blockchain in Internet of Things: Challenges and solutions," *arXiv preprint arXiv:1608.05187*, 2016.
[39] A. Dorri, S. S. Kanhere, R. Jurdak, and P. Gauravaram, "Blockchain for IoT security and privacy: The case study of a smart home," in *Proc. IEEE Int. Conf. Pervasive Comput. Commun. Workshops (PerCom Workshops)*, 2017, pp. 618–623.
[40] N. Kshetri, "Can blockchain strengthen the Internet of Things?" *IT Prof.*, vol. 19, no. 4, pp. 68–72, 2017.
[41] V. Sharma *et al.*, "Secure and energy-efficient handover in fog networks using blockchain-based DMM," *IEEE Commun. Mag.*, vol. 56, no. 5, pp. 22–31, May 2018.
[42] N. Bozic, G. Pujolle, and S. Secci, "Securing virtual machine orchestration with blockchains," in *Proc. 1st IEEE Cyber Security Netw. Conf. (CSNet)*, 2017, pp. 1–8.
[43] Y. Zhang *et al.*, "Smart contract-based access control for the Internet of Things," *arXiv preprint arXiv:1802.04410*, 2018.
[44] A. Ramachandran and D. Kantarcioglu, "Using blockchain and smart contracts for secure data provenance management," *arXiv preprint arXiv:1709.10000*, 2017.
[45] H. Subramanian, "Decentralized blockchain-based electronic marketplaces," *Commun. ACM*, vol. 61, no. 1, pp. 78–84, 2017.
[46] J. Pan, L. Ma, R. Ravindran, and P. TalebiFard, "HomeCloud: An edge cloud framework and testbed for new application delivery," in *Proc. IEEE 23rd Int. Conf. Telecommun. (ICT)*, 2016, pp. 1–6.
[47] G. Wood, "Ethereum: A secure decentralised generalised transaction ledger," Ethereum Project, Yellow Paper, 2014.
[48] L. Eliot, D. Ralph, and R. Danm, "Manufacturer usage description specification," IETF, Fremont, CA, USA, IETF OPSAWG MUD 22, 2018. [Online]. Available: https://www.ietf.org/id/draft-ietf-opsawg-mud-22.txt

Jianli Pan (GS'08–M'16) received the M.S. degree in computer engineering from Washington University in St. Louis, St. Louis, MO, USA, the M.S. degree in information engineering from the Beijing University of Posts and Telecommunications, Beijing, China, and the Ph.D. degree from the Department of Computer Science and Engineering, Washington University in St. Louis.

He is currently an Assistant Professor with the Department of Mathematics and Computer Science, University of Missouri–St. Louis, St. Louis. His current research interests include edge clouds, Internet of Things, cybersecurity, network function virtualization, and smart energy.

Dr. Pan is an Associate Editor of *IEEE Communication Magazine* and IEEE ACCESS.

Jianyu Wang received the M.S. degree in electrical and computer engineering from Rutgers University, New Brunswick, NJ, USA. He is currently pursuing the Ph.D. degree at the Department of Mathematics and Computer Science, University of Missouri–St. Louis, St. Louis, MO, USA.

His current research interests include edge cloud and mobile cloud computing.

Austin Hester is currently pursuing the undergraduation degree at the Department of Mathematics and Computer Science, University of Missouri–St. Louis, St. Louis, MO, USA.

His current research interests include Internet of Things and blockchain.

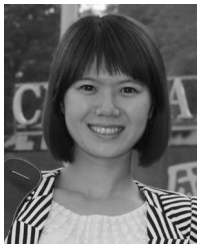
Ismail Alqerm (GS'15–M'16) received the Ph.D. degree in computer science from the King Abdullah University of Science and Technology (KAUSnT), Thuwal, Saudi Arabia, in 2017.

He is a Post-Doctoral Research Associate with the Department of Mathematics and Computer Science, University of Missouri–St. Louis, St. Louis, MO, USA. His current research interests include edge computing, resource allocation in IoT networks, developing machine learning techniques for resource allocation in wireless networks, and software defined radio prototypes.

Dr. Alqerm was a recipient of the KAUST Provost Award. He is a member of the ACM.

Yuanni Liu received the Ph.D. degree from the Department of Network Technology Institute, Beijing University of Posts and Telecommunications, Beijing, China, in 2011.

She is an Associate Professor with the Institute of Future Network Technologies, Chongqing University of Posts and Telecommunications, Chongqing, China. Her current research interests include mobile crowd sensing, IoT security, and data virtualization.



Ying Zhao (M'18) received the master's degree from Shandong University, Jinan, China.

She is an Associate Professor with the Department of Computer Science and Technology, Shandong Labor Vocational and Technical College, Jinan. Her current research interests include data mining and machine learning.