On the Latency Performance in Private Blockchain Networks

Xuan Chen, Kien Nguyen, *Senior Member, IEEE*, and Hiroo Sekiya, *Senior Member, IEEE*

***Abstract*—Blockchain technologies have been emerging with the potential to disrupt many fields (e.g., cryptocurrencies replacing the traditional ones, enabling trustworthy voting, etc.). The Internet of Things (IoT) has been predictably strengthened when integrating to the private blockchain, such as Ethereum. In an IoT deployment with private Ethereum, a thorough understanding of the latency is a critical issue that has not been adequately understood in the literature. Motivated by that, this work aims to comprehend the latency performance in the IoT Ethereum with two popular consensus algorithms: Proof of Work (PoW) and Proof of Authority (PoA). Initially, we clarify different latency segments from transaction submission to execution, namely, the transaction lifecycle in a private blockchain. We then consider the three related latency metrics: 1) transaction-oriented latency; 2) mining time; and 3) block-oriented latency in the PoW case. With PoA, the mining time's consideration is omitted since the mining process is not necessary. After that, we construct a realistic private Ethereum IoT network (i.e., using a laptop and seven Raspberry Pi 3b+ nodes) and a large-scale emulated one with 30 nodes. We write and deploy a smart contract to read and write data to the blockchain and measure the latencies in various scenarios. The measurement results reveal the values of transaction-oriented and block-oriented latency with PoW and PoA in both the actual and emulated networks. Moreover, we derive the expected value for the PoW's mining time by fitting the probabilities to an exponential curve.**

***Index Terms*—Ethereum, latency, private blockchain, Proof of Authority (PoA), Proof of Work (PoW).**

I. Introduction

BLOCKCHAIN technology is a verifiable ledger of transactions that distributes the data storage via consensus mechanisms. The mechanisms, integrating with peer-to-peer (P2P) transportation and encryption algorithms, have been widely accepted in many fields. Initially, the blockchain has been distinctive for decentralizing monetary transactions and trade (e.g., Bitcoin [1]). It then brings many revolutionary changes to the traditional finance system [2], voting platform [3], healthcare [4], vehicle communication [5], etc. Among the potential applications, the blockchain promisingly strengthens the Internet of Things (IoT) on various aspects [6], [7]. The blockchain introduces decentralization to the IoT systems, significantly improving the IoT devices cooperation [8] and scalability [9]. The IoT devices can share information without pretrusting each other or trusting a central authority [10]. Moreover, the historical records are reachable for any participators in the network.

Ethereum blockchain is one of the most popular open source blockchain platforms [11]. Ethereum associates with the cryptocurrency named Ether in the public blockchain network (e.g., Mainnet), which is accessible for anyone. The public Ethereum blockchain is considerably heavy for IoT because it requires a huge amount of resource consumption, high transaction fees while providing low throughput and long latency. On the other hand, the private Ethereum can be programmed following the IoT application's performance requirement. Hence, the private Ethereum is more suitable for IoT. The Ethereum blockchain has various consensus algorithms, including Proof of Work (PoW), Proof of Authority (PoA), Proof of Stake (PoS), etc. Moreover, it supports smart contracts, which are automatically executed codes on the blockchain when triggered by transactions [12]. Smart contracts allow users to construct various functionalities or customize them following the needs of different IoT scenarios. For example, in the smart home applications [13], [14], the smart contracts let house appliances store, share, or modify blockchain states cooperatively. In [15] and [16], the smart contracts are constructed to conduct access control and identification on IoT devices.

Despite many salient features, the latency performance is still one of the biggest concerns when deploying blockchain [17], [18]. Several previous works have evaluated the blockchain latency with different definitions in transaction execution processes. Most of them consider the latency from partial perspectives rather than the entire transaction interval involved in the blockchain network. In [19], the latency is understood as the period of reaching a consensus. Additionally, in [20], the latency is exactly the block mining time in PoW. Both works only extract the time cost of the consensus process, hence excluding the information propagation process. In [21], the latency is defined as the interval between the transaction submission and its first confirmation. However, the latency concentrates on the transaction acceptance speed; meanwhile, the transactions are not truly executed at this stage. Notably, Lo *et al.* [22] have conducted a systematic review of 35 published papers on blockchain-IoT solutions. They found that it is necessary to understand blockchain's end-to-end performance from transaction submission to being accepted, aiming to thoroughly and correctly evaluate the blockchain-based IoT

Manuscript received 25 October 2021; revised 13 January 2022; accepted 27 March 2022. Date of publication 7 April 2022; date of current version 23 September 2022. This work was supported in part by the Vingroup Joint Stock Company (Vingroup JSC), Vingroup through Vingroup Innovation Foundation (VINIF) under Project VINIF.2020.DA09, and in part by the Japan Society for the Promotion of Science KAKENHI Grant 20H0417. *(Corresponding author: Kien Nguyen.)*

The authors are with the Graduate School of Engineering, Chiba University, Chiba 263-8522, Japan (e-mail: chenxuan@chiba-u.jp; nguyen@chiba-u.jp; sekiya@faculty.chiba-u.jp).

Digital Object Identifier 10.1109/JIOT.2022.3165666

implementations. This issue motivates us to quantify the latency performance from a complete perspective. To the best of our knowledge, a thorough understanding of blockchain latency has not been available yet in the literature.

This research presents a method of comprehending the entire latency in private Ethereum blockchain-based IoT networks with two popular consensus algorithms (PoW and PoA). We initially define a complete process from transaction submission to acceptance named the transaction lifecycle. We then clarify three latency components following the transaction lifecycle: 1) transaction-oriented latency; 2) mining time; and 3) block-oriented latency. In the PoW case, we consider all three, while in the IoT blockchain with PoA, the mining time is omitted since the blocks are generated at a fixed pace instead of mined for a variable period with PoW. After that, we investigate the defined latency components in an actual testbed and an emulated network. The implemented testbed includes a laptop and seven IoT devices (i.e., Raspberry Pi 3b+), aiming to mimic an IoT application. The emulated network contains 30 nodes that represent a large-scale IoT blockchain. The devices form a private Ethereum blockchain in two networks where a smart contract is preloaded to read and write strings to the blockchain. We measure the latency components with PoW and PoA in a baseline and a realistic scenario. The measurement results show that regardless of the networks, the transaction-oriented and block-oriented latency increases proportionally to the hop number in both PoW and PoA. They have similar transaction propagation, while the PoA network transfers block quicker than the PoW one. Moreover, in the large-scale network, the propagation latency is considerable when disseminated to peers through a large number of hops. We also measure the PoW algorithm's mining time values, which are inputs of a curve fitting method to find the contribution and the expected mining time. The technique works well with the original and customized Ethereum client (i.e., Geth).

The remainder of this article is organized as follows. Section II presents related works. In Section III, we introduce the background of the Ethereum blockchain. In Section IV, we present our methodology in the latency investigation. Section V describes the experiment setting and evaluation results. Finally, Section VI concludes this article.

II. Related Work

Blockchain is an attractive candidate for decentralizing IoT systems. Kshetri [7] discussed the rationality of blockchain-integrated IoT systems theoretically. They predict the IoT systems can be strengthened with the decentralization essence of blockchain, for example, in identity and access management, avoiding the single point failures, reaching a higher level of safety than the centralized method, and expanding their scalability to accommodate more nodes. Accordingly, there are many efforts to bring different blockchain implementations to IoT [23]. For example, the public blockchains with varying consensus algorithms have been in IoT applications, such as Atonomi (PoW) [24], IoT Chain (PBFT) [25], and VechainThor (PoA) [26]. Nevertheless, before public blockchain can fully integrate with IoT applications, it must overcome several difficulties, including scalability, energy efficiency, and security [27]. Recently, the directed acyclic graph (DAG)-based public chains such as IOTA attracted attention since they offer feeless, fast, high throughput transactions, and efficient operations on IoT devices [28]–[31]. However, the public IOTA also needs to be further advanced to be more compatible with IoT (e.g., maturing consensus mechanism, reducing congestion, and preventing spam).

The private blockchains, including consortium blockchain, have also been considered in many practical applications, e.g., energy trading [32], identity management [33], and Industrial IoT [34], [35]. Among those, the private Ethereum blockchain has been found in various applications. Dorri *et al.* [36], [37] presented an overview of the private Ethereum blockchain-based smart home system (SHS). The SHS integrates home appliances and various sensors, which obtain and share information. In the SHS, an elected miner serves as a cluster header (CH) to implement access control and communicate with other CHs. The work in [14] presented a more realistic smart home application that also uses a private Ethereum blockchain. The system comprises four major components (i.e., temperature and humidity sensors, a smartphone-based visualization application, a Raspberry Pi 3b, and a computer). They manage decentralized data sharing over a private blockchain. Mohanta *et al.* [38] proposed a decentralization authentication scheme for IoT systems based on Ethereum blockchain, which achieves identity registration, key management, and digital signatures in a decentralized way. Those works focus more on the feasibility than quantify the blockchain-system performance.

Following the emerging IoT blockchain application, evaluation frameworks are needed to understand and analyze a blockchain-network performance. Dinh *et al.* [39] proposed Blockbench, an overall framework to explore different private blockchain platforms' performance. They consider a blockchain network in a four-layer architecture and evaluate the performance of each layer. However, the latency metric is defined as the response time per transaction, which does not cover the propagation latency between different nodes. The work in [40], an extension of Blockbench, investigated different workloads with varying numbers of transactions to the application layer. Like the Blockbench paper, the authors considered the latency as the submission and reception of a transaction in the blockchain's first node, indicating the transaction is trusted to be accepted. The latency metric cannot reflect the time taken by different processes of how the transaction is embraced into the blockchain. Mikkelsen *et al.* [41] investigated the mining time and provided an average value. They also indicated that the mining time follows the exponential distribution by inspection. In this case, an expected value is better describing the expectation of the mining time than an average value. Different from [41], we have evaluated and confirmed the distribution by measurement and statistical analysis. The works in [42] and [20] provided an average time for block generation. Moreover, they consider the confirmation delay, which refers to the time it takes for a block to be included in the canonical Ethereum chain (e.g., seven

TABLE I
COMPARISON OF PREVIOUS WORKS AND OURS

Previous work	Consensus algorithm	Methods	Latency metrics	Corresponding metrics
[39]	PoW, PBFT	Implementation	Transaction response time	TOL+Mining time+BOL
[40]	PoW, PBFT	Implementation	Transaction response time	TOL+Mining time+BOL
[41]	PoW	Implementation	Mining time	Mining time
[42]	PoW, PoS, DAG	Simulation	Confirmation delay	-
[20]	PoW	Simulation and implementation	Confirmation delay	-
[43]	PoW	Simulation	Propagation delay, block interval	BOL, mining time
This work	PoW, PoA	Emulation and implementation	Mining time, BOL, TOL	Mining time, BOL, TOL

blocks after mined). Aoki *et al.* [43] introduced a blockchain simulator with two latency parameters: 1) block interval and 2) propagation delay. The latter parameter is characterized as propagation delay following Pareto distribution. We summarize and compare those works to ours in Table I regarding the latency depiction, consensus, and method. The table presents the latency metrics, each with a corresponding parameter in our work.

In this work, we present a transaction lifecycle with three latency metrics. Unlike the previous work, the metrics are extracted following the transmission process on the blockchain. Moreover, we measure the latency metrics in the Ethereum IoT blockchain with two popular consensus algorithms PoW and PoA. Our early work has been previously introduced in [44], which investigates only PoW and ignoring mining time. This work advances [44] by an additional consideration of PoA. Moreover, we present a method to fit the exponential curve and measure the mining time's expected value.

III. ETHEREUM BACKGROUND

Ethereum blockchain is a transaction-based state machine, where the state information includes account balances, data of smart contracts, etc. [12]. Any nodes in the blockchain network can submit transactions to modify the state machine. Each node establishes a transaction pool (txpool) in its local memory space to maintain all received transactions. Submitted transactions are propagated to all other nodes among txpools with P2P communication. A mining node can select some transactions from its txpool to form a block, which will be authorized and secured by adding the hash output of the block content and the result of consensus algorithm [45]. The freshly generated block is propagated to other nodes and confirmed by validating the hash-value correctness. When the block is validated, it will be appended to the local blockchain database. Nodes will remove transactions in validate blocks from the txpool. Once most nodes have accepted the block, they will reach a consensus on the state modification.

Geth is the official implementation of Ethereum [46], which propagates transactions and blocks following the P2P networking protocols named DEVp2p [47]. DEVp2p includes a node discovery protocol and an RLPx transport protocol. Geth nodes reach consensuses on dealing with the disseminated information following consensus algorithms.

A. Ethereum DEVp2p Protocols

1) Node Discovery Protocol: In Geth, the node discovery protocol uses a Kademlia-like distributed hash table (DHT) [48] for efficiently locating and storing content in a P2P network. Every node keeps a 256-bit identity or node ID randomly generated from the Secp256k1 elliptic curve [49]. The logical distance between two nodes is defined as the bitwise XOR of two nodes ID (a and b) (distance$(a, b) = a \oplus b$). Each node maintains an Ethereum node record (ENR) containing up-to-date information, including node ID, IP address, TCP and UDP port, etc. The node also keeps the information of its neighborhood nodes in a routing table. According to the integer value of the distance, Ethereum divides the routing table into several k-buckets. For each $(0 \leq i < 256)$, every node keeps a k-bucket for nodes of distance between 2^i and 2^{i+1} from itself. The current protocol uses $k = 16$, which means every k-bucket contains up to 16 node entries.

2) RLPx Transport Protocol: The RLPx transport protocol is a TCP-based transport protocol used for communication among Ethereum nodes. Recursive length prefix (RLP) [50] is a protocol to encode arbitrarily nested arrays of binary data to serialize messages in Ethereum. Based on RLP, RLPx enables nodes to transfer encrypted, serialized data. In RLPx, two nodes need to perform a two-phase handshake to initialize the session before transmitting essential messages. An RLPx connection is established by creating a TCP connection and agreeing on a pair of ephemeral keys for further encrypted and authenticated communication. The process of creating session keys between the initiator and the recipient is the first handshake. After the negotiation, both sides of the connection send a *Hello* message in the second handshake. They may send the *Disconnect* message to inform a disconnection. The sender can append a single byte of reason code in the message on this disconnection. Alternatively, the *Hello* message exchanges their supporting capabilities and the corresponding version.

3) Ethereum Wire Protocol: The Ethereum wire protocol (ETH) is a subprotocol for exchanging blockchain information between full nodes. The light Ethereum subprotocol (LES) is a protocol used by the light nodes, which only download block headers and fetch other parts on demand. It provides full functionalities of safely accessing the blockchain. In the following, we introduce the ETH version *eth64*, which is used in this work. After the nodes agree to use ETH, they need to exchange the *Status* messages, including the total difficulty (TD) and the hash of their latest block. A node with a lower TD after exchanging the *Status* messages will start synchronization immediately. Concurrently, the nodes propagate blocks and exchange transactions according to the protocol. All pending transactions in the local pool are exchanged after the *Status* messages. The nodes then require absent transactions for synchronization. When a new transaction appears, the new *Transactions* message is propagated to the entire network.

Similarly, a *NewBlock* announcement message is disseminated for declaration and PoW validation when producing a new block. After that, the whole block is sent to a small fraction of connected peers (i.e., the square root of the total number of peers). When receiving a new block, the node imports the block to its local database by executing all contained transactions and modifying the state tree. Once the block is fully processed, the node will propagate the block further to peers without previous notification in *NewBlockHashes* message.

B. Ethereum Consensus Algorithms

1) Proof of Work: The PoW algorithm requires nodes to comport with a feasible amount of effort to deter frivolous or malicious behaviors. Specifically, the PoW algorithm involves searching for an appropriate nonce for a mathematical puzzle whose hash function output is below a certain threshold determined by the difficulty value as (1) indicates. The puzzle's solution is difficult to find, while the verification is straightforward. Since the hash algorithm output is evenly distributed when the input nonce is sequential, the Geth client adopts an enumeration strategy by trying with multiple threads. It enumerates nonce one after another until the output is under a required threshold. A new block is then generated by containing the searched nonce in the block header.

Regarding (1), the *Hash* function represents a series of complex computational processes, with the input of the *dataset* of a specific block and an integer *nonce*. The mining devices expend computational power by calculating the *Hash* with input nonce one by one until they find a proper one whose output is lower than the threshold [the right-hand side of (1)]. *BigInt* is fixed (e.g., $2^{256}-1$ in our case), and D is a *difficulty* value. Hence, with a lower D, the threshold becomes lower, the client can easily find a nonce after fewer attempts, leading to a shorter mining time and vice versa. Therefore, the expected amount of effort in solving the problem is controlled by the parameter difficulty value

$$\text{Hash}(\text{dataset}, \text{nonce}) \leq \frac{\text{BigInt}}{D}. \tag{1}$$

Once a proper nonce is approved correctly, a block is generated by packing the nonce into the block header and verified transactions to the block body. Other nodes confirm the correctness of the nonce when receiving the propagated blocks. Subsequently, they reach a consensus on a block with the nonce and transactions inside. With the uniformly distributed output, Ethereum guarantees that the mining time for nonce exploration depends on the difficulty threshold, making it possible to control the block mining time by manipulating the difficulty value.

2) Proof of Authority: PoA is a consensus method in which several nodes are entrusted to validate transactions and generate blocks. In an Ethereum network with PoA, there is a limited number of validators whose identities are normally preapproved by a network constructor or administrator. Periodically, one of those validators is randomly selected to generate the next block, with the time interval is defined in the first block. In a PoA Ethereum network, an account needs to get more than half of the current validators' consent to join the validator pool. Moreover, they could remove a malicious validator by more than half of the current validators reaching an agreement. As a result, the validators are authorized to verify transactions and generate blocks as correctly as PoW works. However, unlike the PoW's miners, which guarantee their correctness by dedicating computational power, the PoA's validators pledge their reputations. The malicious ones will be removed and be rejected to rejoin the validator pool. Thus, there is an incentive to retain the current position they have gained. In general, the PoA-based Ethereum blockchain provides a secure and trustless environment without too much computational power for the mining task. However, as we know, reputation cannot always keep validators from malicious behaviors. Therefore, the PoA algorithm is usually used for private network setup.

IV. Methodology

This research considers the IoT-based application of the private Ethereum, in which the blockchain network can have a linear connected, a partially connected, or a fully connected topology. The public blockchain network usually forms a partially connected blockchain network (i.e., formed by the Node Discovery Protocol). In all cases, the blockchain information (i.e., transactions or blocks) is propagated hop by hop to the entire network. The nodes will declare the receipt of transactions and blocks to avoid redundant transmission. Therefore, the information is always propagated in a linear path. In the following, we name the two latency metrics between a source and destination nodes in a linear topology based on the transaction lifecycle and the blockchain synchronization. Hence, the metrics are applicable to all types of topologies. We then take an in-depth investigation on the mining time period.

A. Transaction Lifecycle

In a blockchain network, not all nodes are able or willing to participate in the block generation processes, while every node has the ability to receive and propagate transactions or blocks. Thus, a proposed transaction needs to be propagated to a mining node's txpool through different hops to be involved in a block. The block then traverses back to the entire network including the submission node. Transactions will be removed from txpools once receive the block to prevent double-mining. Each node will verify and execute received transactions in the Ethereum virtual machine (EVM), causing the Ethereum states to change. Based on the observation of information transmission in a blockchain network, we present a transaction lifecycle that includes the three steps, as shown in Fig. 1. Specifically, the lifecycle indicates the duration from the submitted moment to the time of becoming effective.

1) A transaction is submitted to the txpool and disseminated to a mining node.
2) The mining node executes the consensus algorithm and generates blocks, which contain the submitted transaction.
3) The blocks are broadcast to all nodes and received after multiple hops' propagation.

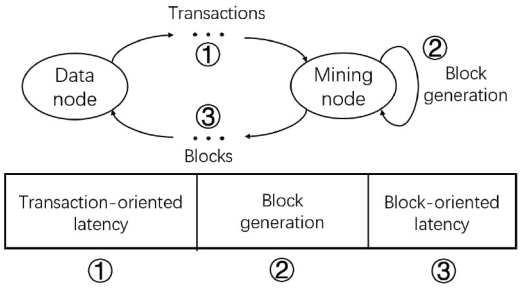


Fig. 1. Transaction lifecycle from submission of the transaction to the reception of the block generated by the mining node.

After the execution, the transaction will finally become effective. As indicated in Fig. 1, we name the leftmost process as TOL, the middle one as block generation, and the rightmost one as BOL. In the Geth implementation, the transaction and block transmissions are triggered by *Transactions* and *NewBlock* messages of the Ethereum wire protocol, respectively. Several other steps then follow them to complete the transmissions. We analyze the Geth log with the highest level of verbosity to record all the detailed information. We use that to clarify the two processes' workflow for TOL and BOL in the private Ethereum network. Both PoW and PoA Ethereum networks have the same workflow because they are only different in the task of block generation. The latency definitions are presented in the next section.

B. Latency Following Transaction Lifecycle

1) Transaction-Oriented Latency: In Ethereum, the workflow of transmitting a transaction between a submission node and a mining node is shown in Fig. 2(a). After a transaction is submitted, it is pushed into a queue, waiting to be verified. When the node finishes the verification (i.e., at the *promoted queued transaction* point), the transaction is submitted and added into the txpool (i.e., at the *submitted transaction* point), in which it will be broadcast to the peers in *Transactions* message. The peer node also first queues the received transaction at the *pooled new future transaction* point. It then verifies the transaction after the promoted queued transaction point. Subsequently, the transaction is added to the txpool of this peer, which is regarded as one transmission hop. This peer repeats the process to propagate the transaction further to other nodes. Therefore, the TOL is defined as the interval between the submission moment in the submission node and the promotion time in a receiving node, that is, one or multiple hops away from the submission one. Hence, the TOL value reflects the time consumption for a transaction to be propagated through several hops. With a lower value of the TOL, a submitted transaction can reach the entire network quicker.

2) Block-Oriented Latency: The mining node selects transactions from txpool and packs them into a block with a consensus algorithm, which will then be propagated to other nodes. Fig. 2(b) shows the workflow of the block propagation. Starting at the *mined potential block* point, the node sends the block to its peers at the *propagated block* point. After the peer receives the block, it is pushed into a queue at the *queued propagated block* point. The peer imports the block at the *importing propagated block* point and performs a quick verification on the block header, which is the end of a transmission. Then, to reach all the nodes as soon as possible, the block is further propagated and executed simultaneously. At the propagated block point, the peer has already finished repeating the block transmission process to the next node. Then the node inserts it in its local database and changes the state at *inserted block* point. The peer announces the ownership of the block to avoid duplicating transmission. The transaction executions are not shown in the Geth log, and they vary on different smart contracts. We name the end of the transaction lifecycle at the *imported new chain segment* point. Consequently, the BOL is defined as the interval from the block generation time to the importing moment on another node (i.e., one or several hops far away). The BOL value describes the time consumption for a block to be propagated through different hops. A block contains several verified transactions. After the block is propagated and appended to a peer, those transactions are formally accepted and come into effect. Thus, this latency describes how fast a block traverses the network.

Our investigation in Fig. 2 shows the fundamental unit (i.e., a single hop) of TOL and BOL propagation workflows. In a multiple-hop scenario, a peer repeats the same workflows when communicating with its neighbors. In the transaction case, each transaction is propagated separately and continuously. The peer does not wait until all transactions in the queue have been received before disseminating them further. Similarly, in the block transmission, the peer will propagate blocks before any deeper processes, aiming to cover the entire network as fast as feasible. Because every node is equivalent to each other and treats messages the same way, we construct a linear model as shown in (2) to understand the propagation process better

$$L_N = T_h \times N + T_m \tag{2}$$

in which L_N denotes the propagation latency (i.e., TOL or BOL) until the Nth-hop node and N is the number of hop. During the transferring process, each node spreads the new messages repeatedly, $T_h \times N$ represents when a peer is aware of the new messages since they are released. Meanwhile, T_m shows the time consumption of receiving the whole message. We will use a fitting method to find T_h and T_m in our linear topology network, in which each node has no more than two peers.

3) Block Mining Time: The block mining time can be defined as the mining task period to generate a block. It is the process of solving a mathematical puzzle whose complexity level is represented by a difficulty value. In the Ethereum blockchain, when mining each block, the Geth client records the associated timestamp. The mining time is, therefore, calculated as the difference of the timestamps in two adjacent blocks. When a mathematical puzzle with the same difficulty value running on different devices, the one with higher computing power tends to solve the puzzle within a shorter time. However, Ethereum includes a dynamical adjustment mechanism to adapt appropriate difficulty value for the next mining block. The mechanism can be shown in the following

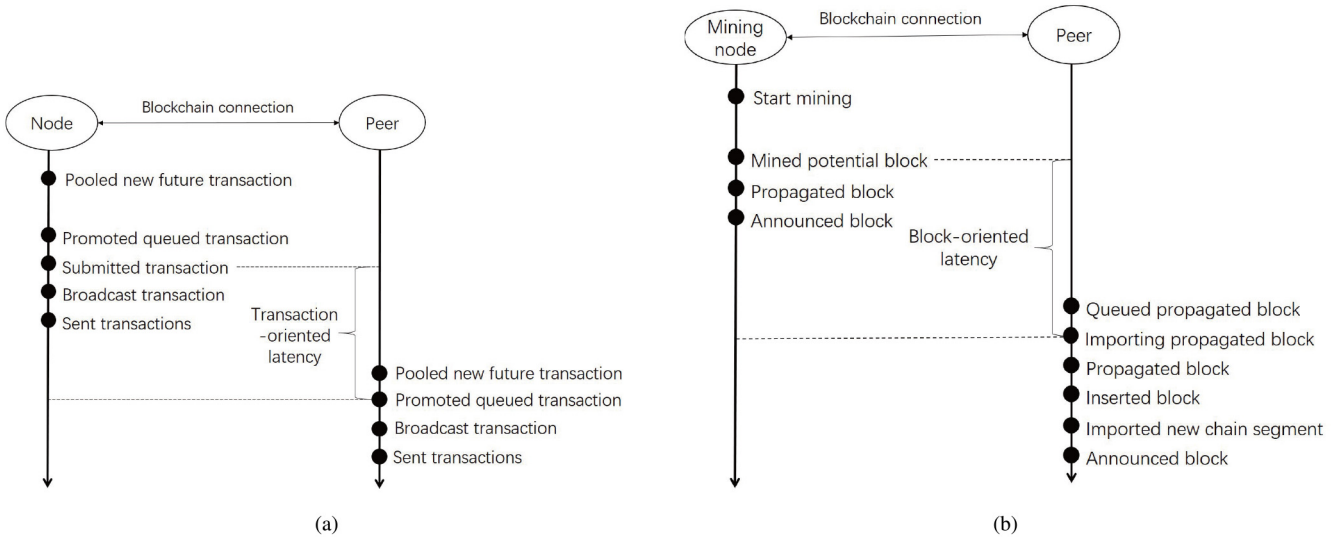


Fig. 2. One-hop workflows of information propagation between nodes. (a) Processes of propagating a transaction and related transaction-oriented latency. (b) Processes of propagating a block and related block-oriented latency.

equation[1] [12]

$$D_N = \begin{cases} \left(D_{N-1} + \frac{D_{N-1}}{2048} \times \max\left[\left(1 - \frac{M}{10}\right), -99\right]\right) \\ (N < 200000) \\ \left(D_{N-1} + \frac{D_{N-1}}{2048} \times \max\left[\left(1 - \frac{M}{10}\right), -99\right]\right) + 2^{\frac{N}{100000}-2} \\ (N \geq 200000) \end{cases} \tag{3}$$

where D_N and N are the current block difficulty and number, respectively. D_{N-1} is the difficulty of the parent block. M is the block mining time, which is calculated by the timestamps' difference in the current block and the parent block. The equation has a corrective term when the block number is over 200 000. While according to the equation under 200 000 blocks, we can know the following.

1) If the mining time is less than 10 s, the difficulty is adjusted upward by $([D_{N-1}]/2048)$.
2) If the mining time is within 10–19 s, the difficulty is left unchanged.
3) If the mining time is greater or equal to 20 s, the difficulty is adjusted downward proportional to the timestamp difference from $([D_{N-1}]/2048)$ to $99 \times ([D_{N-1}]/2048)$.

When the block number is below 200 000, the current block's difficulty value is calculated based on the previous difficulty value D_{N-1} and the mining time M. In a private Ethereum blockchain, the initial difficulty value is defined in the first block (i.e., block number 0). Afterward, as seen, the Geth client dynamically adjusts the difficulty value based on the mining time until it finds an appropriate value. The adjusted difficulty value depends on the computing capability on the mining device, which keeps the mining time at the expected range (i.e., between 10 and 19 s).

In the mining process, Ethereum adopts an enumeration strategy by trying an input nonce per time. Ethereum enumerates nonce one after another until an output meets (1). Each time the miner tries to calculate the output hash, it is an independent event and is supposed to consume a fixed time on the same device. Considering the properties of the hash function, the nonce of correct proofs are randomly distributed in the output space. When executing the PoW algorithm, the miner continuously enumerates nonce and calculates the output hashes. The arrivals of a correct nonce can occur arbitrarily. Each calculation has a constant probability of meeting the mathematical puzzle and resulting in a valid block. According to the probability theory, the occurrence of correct proofs in a specific time is regarded as a 1-D Poisson process [51], [52] and the interval between two correct proofs (e.g., the mining time), therefore, follows an exponential distribution [53] as the following shows:

$$f(x) = \begin{cases} ae^{-\lambda x} + b\ x, > 0 \\ 0, \qquad \text{otherwise.} \end{cases} \tag{4}$$

In this work, we first run various experiments and collect the mining time and difficulty values. By calculating the probabilities of each specific mining time value, we adopt a probabilistic method to analysis the mining time. Then, we fit the probabilities of different mining time values to (4), where λ is the rate parameter related to the time interval of mining blocks, a and b are optimal parameters. When the dynamic difficulty value reaches a balance, the mining time is anticipated to follow exponential distribution, which can then be evaluated with an expected value.

V. Evaluation

A. Experimental Setup

We build our private blockchain using a laptop (or server) as a mining node and seven IoT devices. The server is used only in the evaluation of block mining time. Each IoT device is a single-board raspberry Pi 3b+ (RPi 3b+) (i.e., data node). Table II summarizes the hardware and software configurations on the devices. Moreover, Fig. 3 shows the physical connections the devices. The RPi 3b+ and laptop form an underlying

[1]In Ethereum source code: consensus/ethash/consensus.go.

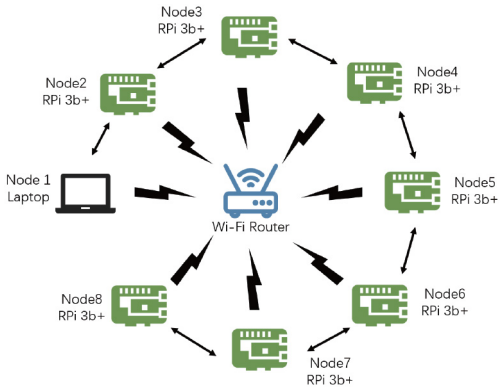


Fig. 3. Private blockchain network (the lightning mark describes a Wi-Fi connection; the solid arrow describes a blockchain connection).

TABLE II
CONFIGURATIONS

Raspberry Pi 3b+	
CPU	1.4 GHz 4 core ARMv8
RAM	1 GB
Storage	16 GB MicroSDHC
OS	Ubuntu Mate 18.04 LTS
Ethereum	Geth 1.9.10
Laptop	
CPU	2.5 GHz 4 core i5-7200U
RAM	8 GB
OS	Ubuntu 18.04 LTS
Ethereum	Geth 1.9.10 (original and modified)
Server	
CPU	2.1 GHz 24 core Xeon Silver 4116
RAM	16 GB
OS	Ubuntu 18.04 LTS
Ethereum	Geth 1.9.10 (original and modified)

TABLE III
NETWORK PERFORMANCE

From	to	RTT (ms)	Bandwidth (Mbits/s)		Lost rate (%)
			TCP	UDP	
Laptop	Node 1	4.291	18.4	27.8	0
Node 1	Node 2	4.003	17.3	25.5	0.02
Node 2	Node 3	3.929	18.2	25.3	0.03
Node 3	Node 4	4.020	17.2	25.4	0.02
Node 4	Node 5	4.133	17.8	25.7	0.04
Node 5	Node 6	3.986	18.1	25.4	0.01
Node 6	Node 7	4.205	17.9	25.6	0.02

network for the blockchain by connecting to the same TPlink Wi-Fi router. All the devices are in the same room. The distance between each device and the router is within 3 m. We use *iperf* and *ping* tools to evaluate the network performance on the testbed. More specifically, we find the TCP, UDP throughput, and the round-trip time (RTT) of the network connection between nodes. The results are shown in Table III. A customized genesis file is created to launch a private Ethereum blockchain client. Depending on the evaluation, we can set an arbitrary value for the *Difficulty* in the genesis file. The critical parameter in the private Ethereum, block gas limit, is set to allow the blocks to contain sufficient transactions. After successful initialization, the private Ethereum blockchain IoT network has a linear structure indicated by the arrows in Fig. 3.

To run our experiments, we have to prepare two important issues as follows.

First, we deploy a smart contract written in Solidity version 0.4.25 [54]. The smart contract, which simulates the IoT-system activities, has two functions: 1) writing a string to the blockchain and 2) reading the current string. Ethereum normally charges a sender some *Ether* based on gas consumption and the transaction's gas price. However, the nodes in a private network suppose to share information without any restriction. In our blockchains, the gas price is set to 0. That means the nodes can submit transactions to write for free. The reading function does not consume any gas either. Second, we synchronize the system time on the nodes to measure the latency on millisecond-level accuracy. We use the *ntpdate* utility to synchronize the blockchain nodes' system time. *ntpdate* sets the local system time by polling the network time protocol (NTP) servers. We selected the NTP server in Japan, which is the closest one to the devices, and guarantee the expected accuracy.

In the evaluations, we use the Web3.js library [55] on all data nodes to send transactions by calling the writing function embedded in a transaction. We preload a JavaScript file to the mining node, enabling a mining process after receiving a transaction. The *verbosity* is set to maximum in all nodes, which allows Geth to output the most detailed information, including all steps [i.e., in Fig. 2(a) and (b)] with a timestamp. We record those outputs from the console to a log file and collect them together. We then extract the timestamp of the claimed steps for each latency type using our self-written bash scripts. We calculate the minimum, average, and maximum values of different latency types in each set of experiments.

B. Latency Evaluation

This section introduces the evaluation of three latencies as described in Fig. 1, which are TOL, BOL, and block generation latency (i.e., mining time). Initially, the first two latency parameters are measured in a baseline and a realistic scenario. We consider both the PoW and PoA consensus mechanisms. Note that, in the PoA case, a miner is not required, while the blocks are generated at a fixed-interval pace by validators. We keep the laptop as the only validator in the PoA related experiments. The time interval of generating a block is 12 s, which is similar to the expected value in the PoW case. In the later part, we present the mining time evaluation.

1) Baseline Scenario: The baseline scenario includes the TOL of transmitting a single transaction and the BOL of transmitting an empty block. The baseline scenario discloses the time interval of transmitting minimal workload in this private blockchain network. Regarding the TOL, we measure the needed period for a transaction to be transferred from each data node (RPi 3b+) to the mining node, validator node (i.e., the laptop) with the PoW and PoA, respectively. The minimum, average, and maximum values of TOL are shown in Fig. 4(a), where the x-axis shows the node number. We observe that the TOL value increases when the number of hops to the validator or miner increases. More specifically, the average increment per hop for PoW and PoA is 11.52 and 10.70 ms, respectively.

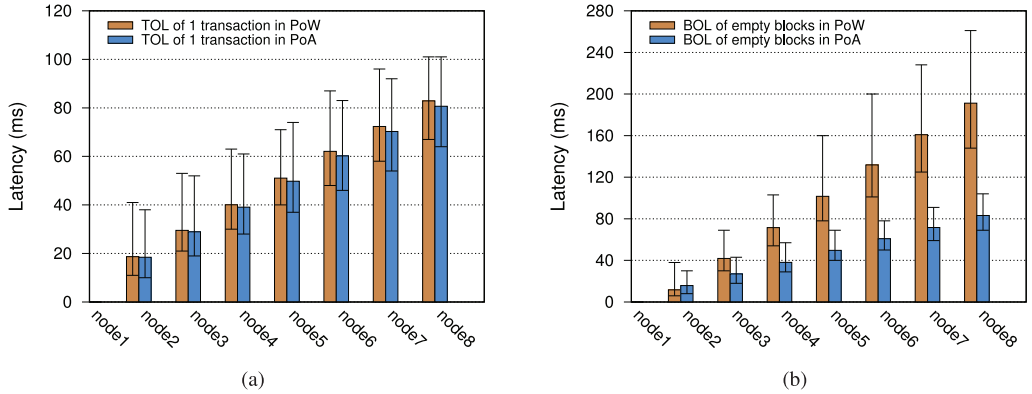


Fig. 4. Baseline scenario with PoW and PoA. (a) TOL results. (b) BOL results.

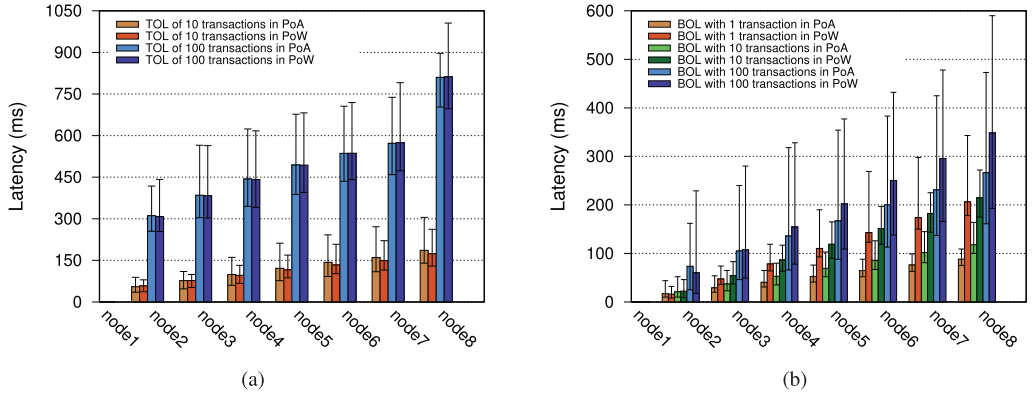


Fig. 5. Realistic scenario with PoW and PoA. (a) TOL results. (b) BOL results.

TABLE IV
BLOCK SIZE WITH DIFFERENT TRANSACTIONS

Number of transactions	Block size (bytes)	
	PoW	PoA
0	540	606
1	750	813
10	2645	2632
100	21544	20204

Regarding the BOL, we measure the latency when transmitting an empty block from the mining node to each data node. In both PoW and PoA cases, the experiment sets are repeated a thousand times. The measurement results are presented in Fig. 4(b), where again the latency increases following the number of passing hops. The PoW blockchain needs 27.31 ms per hop on average. Meanwhile, in the PoA Ethereum network, the average increment of BOL per-hop is 11.52 ms, which is significantly lower than the PoW's one.

In this evaluation, we found that a transaction's size is variable according to the attached information. A typical transaction calling the smart contract is shaped within three TCP segments in both the PoW and PoA Ethereum networks. The empty block only contains a block header, whose size is about 540 bytes, as shown in Table IV. We collect the block size information embedded in the block header using the Geth client with the Web3.js library. More specifically, we call the method "*eth.getBlock(block_number).size*" and "*eth.getBlock(block_number).transactions.length*" of each block. We then calculate the size of blocks with different numbers of transactions inside. An empty block is transmitted by using four TCP segments in both algorithms. While, as we observed, both algorithms share the same functionality for transaction processing with txpool. Thus, both algorithms propagate transactions with similar TOL. However, two consensus algorithms (i.e., Ethash[2] for the PoW Ethereum network and Clique[3] for the PoA Ethereum network) have different processes for block verification. The PoA Ethereum network has more simple actions than PoW. For example, the PoA Ethereum network does not need to check uncle blocks or calculate difficulty value. Consequently, the PoA Ethereum network accepts blocks quicker than PoW for empty blocks and for the following realistic scenario.

2) Realistic Scenario: In this evaluation, we add more workloads to simulate latency in a realistic scenario. For the TOL, two workloads of sending 10 and 100 transactions at once have been investigated. We consider three cases of sending blocks containing 1, 10, and 100 transactions for the BOL. A group of transactions is propagated to the mining node through a different number of hops in the former. In the latter, the mining node (or validator) generates blocks containing propagated transactions, traverse to the data nodes in a multihop manner. Each experiment is repeated a thousand times. The evaluation results in the real scenario are shown in Fig. 5.

[2]consensus/ethash/consensus.go
[3]consensus/clique/clique.go

TABLE V
PARAMETERS OF LINEAR MODEL IN RPI-BASED NETWORK

Scenarios	T_h	T_m
TOL_PoW_1	10.722	8.051
TOL_PoW_10	18.897	39.042
TOL_PoW_100	70.976	223.099
BOL_PoW_0	29.893	-18.067
BOL_PoW_1	31.696	-16.016
BOL_PoW_10	32.043	-9.458
BOL_PoW_100	47.735	11.657
TOL_PoA_1	10.376	8.123
TOL_PoA_10	21.589	33.887
TOL_PoA_100	70.196	226.558
BOL_PoA_0	11.223	4.586
BOL_PoA_1	11.816	5.465
BOL_PoA_10	16.161	4.824
BOL_PoA_100	31.934	40.673

Fig. 5(a) shows the TOL results with different workloads in the blockchain using PoW and PoA. Obviously, each blockchain node takes more time when processing more transactions in comparison to the baseline scenario. For both workloads, the TOL values approximately have a linear increase along with the number of hops. In transmitting 100 transactions, the TOL of PoW takes 308.22 ms for the one-hop case. After that, the TOL increases 83.92 ms for each additional hop. On the other hand, the PoA's average TOL is about 310.93 ms for the first hop. Moreover, the increasing gap between each hop is approximately 83.18 ms. Hence, we can conclude that the TOL values of PoA and PoW are on the same level, indicating that the consensus algorithm does not significantly impact the transaction transmission process. Additionally, the transactions' transmission between nodes is continuous. They are hence transferred to the next hop without waiting for all transactions to be received. However, the latency of transferring 10 or 100 transactions is not ten or a 100 times comparing to the baseline.

Fig. 5(b) presents the BOL of the blockchain with PoW, PoA under the varying workloads of 1, 10, and 100 transactions in each block. Each workload results in a different value of block size (in bytes). The relationship of the size and transaction number is shown in Table IV. With the PoW and PoA algorithms, similar to the baseline scenario, the BOL values are proportional to the number of hops from a transmitting node to the miner. With the largest block size (i.e., the block with 100 transactions), the PoW blockchain on average needs 59.96 ms for the first-hop transmission and increases 48.02 ms per hop. However, the PoA Ethereum network takes 73.54 ms for transmitting the same block for the first hop and increases 32.11 ms at each later hop on average. The results also indicate that the PoA Ethereum network propagates blocks quicker than the PoW (i.e., smaller BOL values).

To find the parameters in (2) for each scenario with the RPi-based network, we fit our measured average values to the linear model. The results are shown in Table V, in which each scenario has been abbreviated following the format Latency metrics (TOL or BOL)_consensus mechanism (PoW or PoA)_number of involved transactions. For example, TOL_PoW_1 represents transferring a single transaction with PoW, whereas BOL_PoA_0 refers to propagating blocks containing zero transactions with PoA. We can see that the transactions with both PoW and PoA have similar T_h and T_m values. The block propagation with PoW has negative values of T_m in RPi-based and the later emulated networks. It signifies that the first hop of block propagation with PoW is faster than subsequent steps. The reason is the PoW consensus algorithm requires a miner to share their most recent mining results with other miners to avoid forking.[4] Because the mining feature is activated on all nodes, node1 informs node2 quicker than other block propagation steps.

3) Block Mining Time: We operate the mining function with the original and a modified Geth version on the laptop and a server in the block mining time evaluation. The original client evaluates the difficulty variation, while the modified version is for the case of fixed difficulty value, by which we can customize the expected mining time. During the mining process, the client records the timestamp when each block is generated. In total, we let the client mine 10 000 blocks and collect the timestamp values for the mining time calculation (i.e., the timestamp difference between the previous block and itself). We also track the difficulty value of each block. All tasks have been done with a script accessing the blockchain via the interprocess communication (IPC).

The evaluation results with the original Geth are shown in Fig. 6. We can see the adjustment of the difficulty value and the mining time on the laptop and a server in Fig. 6(a) and (c), respectively. The left y-axis shows the mining time in each figure, while the right y-axis indicates the difficulty. Note that the difficulty keeps increasing on both machines along with the block number (until around the 4000th blocks), considering the initial difficulty value preset in the genesis file is low for the computing power of two devices. Ethereum continuously adjusts the difficulty value until it reaches a balanced value matching the total computational power. In the experiments, we can observe that after mining about 5000 blocks, the difficulty value becomes quite stable. We then take a deeper look into the block mining time in that case. The raw measurement values have big variations and are difficult to evaluate. We take the curve fitting method to exploit the exponential distribution of the mining time in the two machines. The probability of each mining time collected from the last 5000 mining blocks is fitted using a Python optimization package.[5] The fitted results on the laptop and server are presented in Fig. 6(b) and (c). As revealed in the figures, the mining time follows the exponential distribution. Moreover, the fitted equations are derived as in (5) (laptop), (6) (server). From that, we can have the expected mining time of the laptop is 11.89 s. Meanwhile, the value of the server is 11.76 s. Ethereum ultimately regulates the mining time to the same level on devices with different computational power. The difficulty values are adjusted to a balanced value after a sufficient number of blocks, adapting to the devices' computational power. The device with a higher computational power has a higher balanced difficulty value. Consequently, the mining time shows an exponential distribution with distinctive

[4] consensus/ethash/sealer.go
[5] scipy.optimize package in Python3.8

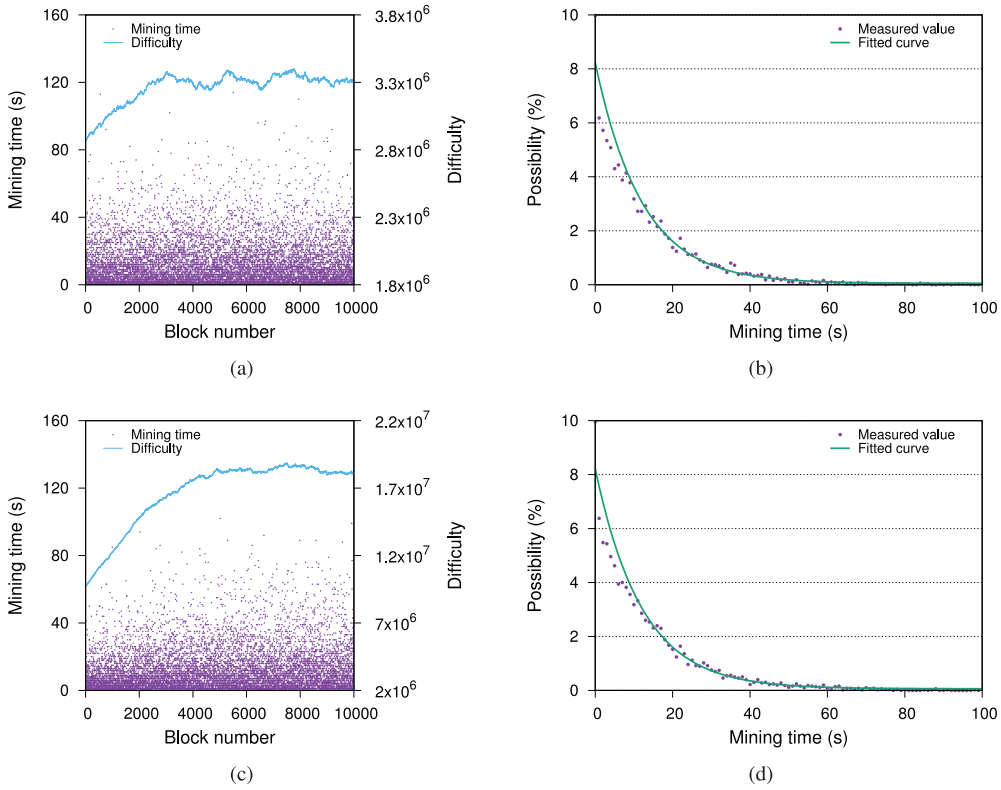


Fig. 6. Evaluation results with original Geth. (a) Mining time and difficulty variation on the laptop. (b) Probability of each mining time and fitted curve of the last 5000 blocks in (a). (c) Mining time and difficulty variation on the server. (d) Probability of each mining time and fitted curve of the last 5000 blocks in (c).

difficulty values on both devices

$$f_1(x) = 0.0818e^{-0.0829x} + 0.0005 \quad (5)$$

$$f_2(x) = 0.0816^{-0.0832x} + 0.0006. \quad (6)$$

In the following, we show the evaluation with the modified version of Geth. In Ethereum, we can customize the Geth client in a private blockchain network, for example, disabling or modifying the adjustment function.[6] This evaluation uses the modified Geth to mine blocks with fixed difficulty values on the laptop. Referring to the balanced value in Fig. 6(a), we evaluate two fixed values of difficulty (i.e., 3×10^6 and 6×10^6), each with 5000 mined blocks. The measurement values of mining time for each fixed value are shown in Fig. 7(a) and (c). We use the same method to get the fitted curves as in Fig. 7(b) and (d) for the difficulty of 3×10^6 and 6×10^6. Moreover, we derive the equations for two scenarios in (7) and (8), which lead to the expected mining time of 9.73 and 23.20 s, respectively. We can conclude the mining time also follows the exponential distribution with the fixed difficulty value. The client with fixed 3×10^6 difficulty leads to a similar expected mining time with the original Geth client, which proves the adjustment mechanism can keep the difficulty in balance. Moreover, the client with the difficulty of 6×10^6 has an approximately double mining time than the 3×10^6 client. Thus, we can customize the expected mining time referring to the adapted balanced difficulty value

$$f_3(x) = 0.959e^{-0.0993x} + 0.0008 \quad (7)$$

$$f_4(x) = 0.431e^{-0.0431x} + 0.0001. \quad (8)$$

C. Latency Evaluation on Emulated Network

This section shows our evaluation of BOL and TOL on a large-scale network, aiming to indicate the size of an actual IoT blockchain application. We have deployed a network with 30 nodes on the emulator named Mininet-WiFi [56] on a powerful Dell server. Mininet-WiFi is an open-source emulation platform that allows replicating and emulating wireless network environments (e.g., wireless devices, access points, many versions of IEEE 802.11). In our deployed wireless network, all the nodes connect to an access point using IEEE 802.11g as in Fig. 8. The RTT between any two different nodes is about 5 ms, while the bandwidth is set at 10 Mb/s. We then build a private blockchain on top of the wireless networks with 30 nodes in a linear topology. Transactions and blocks are produced in Node 1 and propagated to the other nodes within 29 hops. We implement the same PoW and PoA as in the previous measurement. Furthermore, on Ethereum Mainnet, a typical block contains about 400 transactions.[7] Hence, we investigate more sufficient workloads of 200, 300, 400, and 500 transactions to capture actual situations. It should be noted that the default gas limitation in the private Ethereum network is 8 000 000. A standard transfer transaction requires 21 000

[6]func (ethhash *Ethash) CalcDifficulty.

[7]https://etherscan.io/

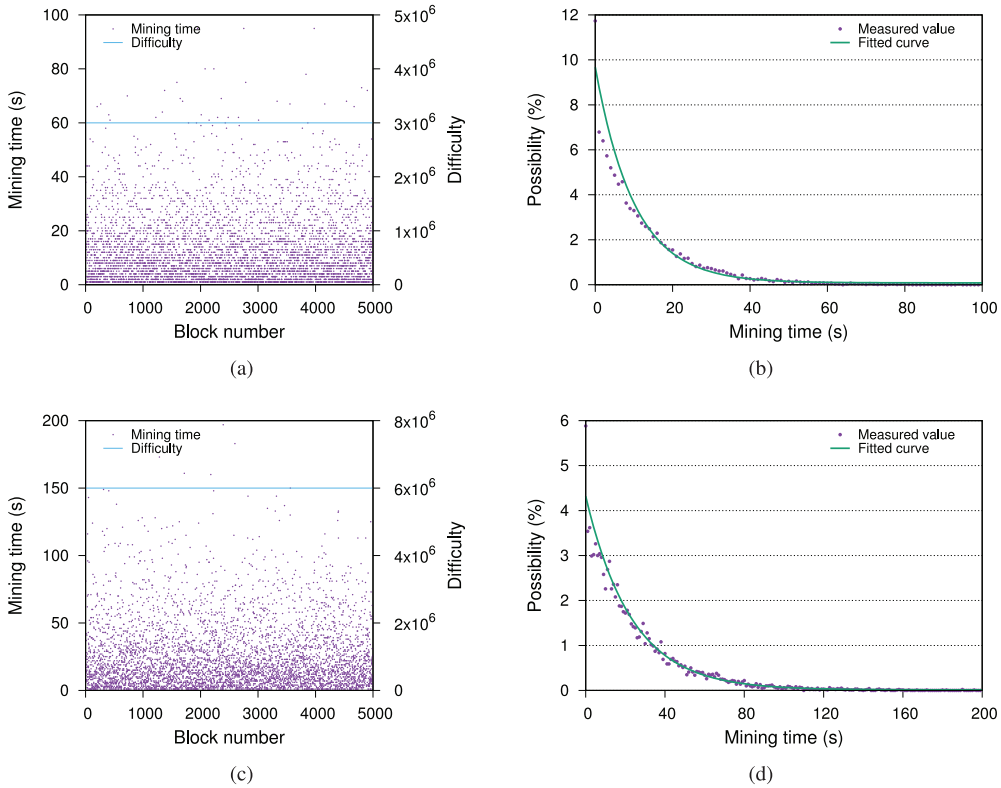


Fig. 7. Evaluation results with modified Geth. (a) Mining time with fixed 3×10^6 difficulty. (b) Probability of each mining time and fitted curve of 5000 blocks in (a). (c) Mining time with fixed 6×10^6 difficulty. (d) Probability of each mining time and fitted curve of 5000 blocks in (c).

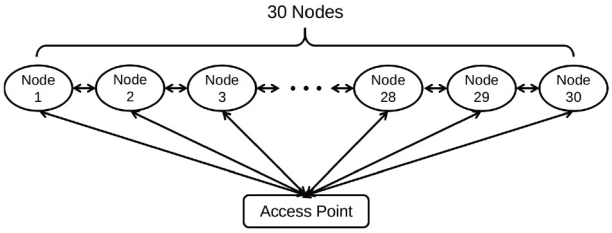


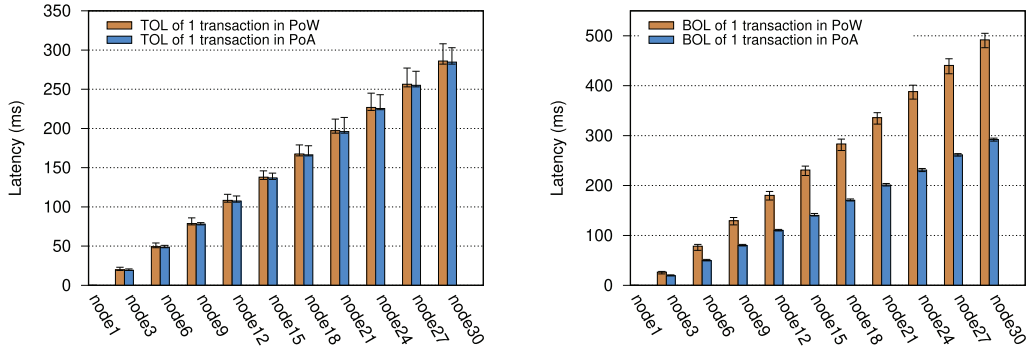
Fig. 8. Topology of private blockchain network in Mininet-WiFi emulation.

units of gas, whereas a transaction calling a smart contract may consume more depending on its complexity. Therefore, to run the investigation properly, we need to expand the default gas limitation with *–miner.gaslimit(gas_limitation)* command when starting the Geth client for a larger transaction capability. We investigate the TOL and BOL with all different workloads in the same method. The results in baseline and realistic scenarios are reported below.

1) Baseline Scenario: In this scenario, Node 1 executes the consensus algorithms to generate new blocks. The node then disseminates single transactions and empty blocks to other nodes. Each experiment is repeated 100 times. The minimum, average, and maximum values of TOL and BOL are shown in Fig. 9, where the *x*-axes indicate the increment of hops. Fig. 9(a) shows that the TOL value strictly increases with the number of hops with both PoW and PoA. On average, the latency gain per hop is 9.86 and 9.82 ms, respectively. Hence, the single transaction transmission latency in the emulated network is close to the one in the real network reported in the previous section. Regarding BOL, we can observe in Fig. 9(b) that the BOL value increases following the number of hops. In the case of PoW, each hop consumes 16.96 ms on average, while PoA requires 10.07 ms per hop on average. Similar to the actual network, the empty blocks are also propagated quicker with PoA than with PoW. Besides, the BOL of empty blocks with PoA in the emulated network is similar to the actual implementation. On the other hand, the BOL values with PoW in the emulated network are shorter. That is, because the block generation in the PoW algorithm requires more processing procedures than the PoA algorithm. Moreover, the server, which hosts the emulation, is much more powerful than the miner in the actual network.

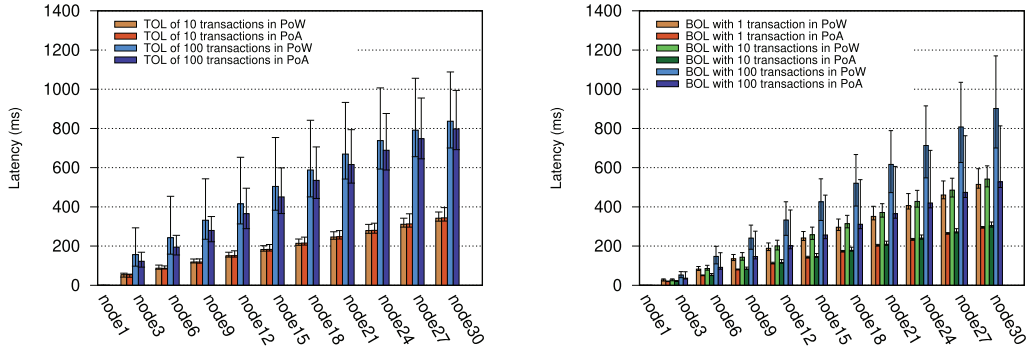
2) Realistic Scenario: In this scenario, we inherit the traffic pattern from the previous evaluation in the actual network. The TOL evaluation will be investigated under the conditions of 10, 100 transactions being transferred at once. In the case of BOL evaluation, we have added different numbers [i.e., (1, 10, and 100) and (200, 300, 400, and 500)] of transactions inside of a block. The results are shown in Figs. 10 and 11.

Fig. 10(a) presents the TOL results with different workloads in the emulated network with PoW and PoA. Compared to the actual implementation, the emulated network's TOL shows a more strict linear increment along with the number of hops. More specifically, in the PoW and PoA networks, transferring ten transactions consumes approximately 11.79 and 11.85 ms, while transporting 100 transactions takes 28.85 and 27.49 ms, respectively. We can say that both networks have a similar level of latency when transferring transactions. Moreover, since the emulation provides a better network environment,



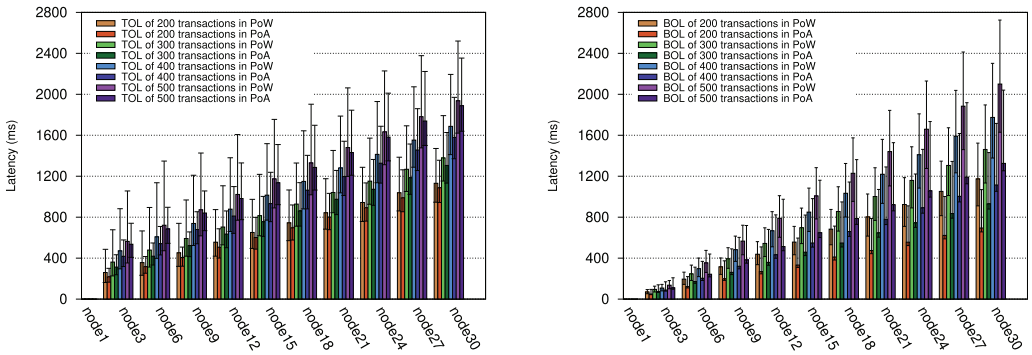
(a) (b)

Fig. 9. Baseline scenario with PoW and PoA in Mininet-WiFi emulation. (a) TOL results. (b) BOL results.



(a) (b)

Fig. 10. Realistic scenario with PoW and PoA in Mininet-WiFi emulation. (a) TOL results. (b) BOL results.



(a) (b)

Fig. 11. Realistic scenario with sufficient workloads in Mininet-WiFi emulation. (a) TOL results. (b) BOL results.

the emulated network's TOL is lower than the actual one.

Fig. 10(b) shows the BOL results in the emulated network with PoW, PoA when the workload varies (i.e., 1, 10, and 100 transactions). In this evaluation, we use the same Ethereum blockchain, the block sizes are similar to those in the actual network. With the PoW algorithm, the transferring blocks with 1, 10, and 100 transactions inside consume 17.74, 18.68, and 31.09 ms per hop, respectively. While, with the PoA algorithm, the BOL per hop results are 10.13, 10.49, and 18.21 ms, respectively. Additionally, we can observe that blocks with plenty of transactions inside need about 1 s to be disseminated to 29 hops. The emulation results show that the blockchain network will produce significant propagation latency when disseminating the information via a large number of hops, leading the blockchain under many risks, such as forks. Therefore, reducing the longest message transmission path is necessary for large-scale blockchain networks.

Fig. 11(a) and (b) show TOL and BOL results for the large workload of 200, 300, 400, and 500 transactions. Those results follow the same tendency as our earlier measurements. As we can observe, dispersing 200 transactions to 24 hops or 500 transactions to 12 hops approximately results in a TOL of 1 s. In terms of the BOL values, the blocks containing 200

TABLE VI
PARAMETERS OF LINEAR MODEL IN EMULATED NETWORK

Scenarios	T_h	T_m
TOL_PoW_1	9.863	0.069
TOL_PoW_10	10.746	31.77
TOL_PoW_100	26.299	121.46
TOL_PoW_200	32.392	195.778
TOL_PoW_300	37.557	291.275
TOL_PoW_400	45.056	381.151
TOL_PoW_500	50.742	465.111
BOL_PoW_0	17.248	-8.884
BOL_PoW_1	18.035	-9.046
BOL_PoW_10	19.016	-9.084
BOL_PoW_100	31.452	-11.718
BOL_PoW_200	40.672	-10.801
BOL_PoW_300	50.743	-10.84
BOL_PoW_400	61.719	-11.83
BOL_PoW_500	72.785	-12.22
TOL_PoA_1	9.823	0.466
TOL_PoA_10	10.837	31.089
TOL_PoA_100	26.176	75.49
TOL_PoA_200	32.345	151.4
TOL_PoA_300	36.638	235.91
TOL_PoA_400	43.281	329.77
TOL_PoA_500	49.912	437.91
BOL_PoA_0	10.094	0.917
BOL_PoA_1	10.146	0.781
BOL_PoA_10	10.517	0.707
BOL_PoA_100	18.215	0.962
BOL_PoA_200	23.739	6.273
BOL_PoA_300	31.877	9.98
BOL_PoA_400	38.032	14.957
BOL_PoA_500	45.142	18.982

transactions take more than 1 s to propagate across 24 hops, and blocks with 500 transactions take over 2 s to be propagated over 30 hops. A node requires approximately half a second to receive 500 transactions. Packing 500 transactions into a block takes 72 and 45 ms in our blockchain with PoW and PoA, respectively. These latency results expose that the propagation latency is significant when transferring huge workloads, emphasizing the need to create appropriate network topologies and lowering propagation latency in blockchain networks.

Regarding the values in the linear model in (2), we adopted the same method as the RPi-based network for the emulated networks' evaluation. The fitted results of T_h and T_m are shown in Table VI. We can find all the parameters in all the 30 scenarios (15 for BOL and 15 for TOL). Moreover, we have the same observation as in the previous fitting evaluation. T_h and T_m of TOL scenarios are similar in PoW and PoA networks. In the case of BOL, the blockchain with PoA has a lower T_h than the one with PoW (i.e., which implies that blocks are transferred faster with PoA).

VI. CONCLUSION AND FUTURE WORK

This article aimed to understand the latency of the private Ethereum IoT network using the PoW and PoA consensus algorithms. First, we have defined three steps according to the transaction lifecycle (i.e., TOL, block generation time, and BOL) and their measurement methodology. We showed the increment of TOL and BOL with both consensus algorithms in the baseline and realistic scenarios. The results gave us an observation of the latencies' relationship in an actual deployment and emulated large-scale network. The change of consensus algorithms does not influence the TOL much, while it affects the BOL. The PoA Ethereum network has a lower BOL than the PoW one due to a simpler block verification process. Moreover, we investigated the PoW algorithm's impact on the adjustment of the difficulty value, which then influences the mining time. Our measurement results and fitting method showed that the mining time follows the exponential distribution when the difficulty is both fixed or stable. With this work's findings, we can thoroughly and correctly understand each latency component from one end to another from a practical perspective. Besides performance benchmarking, we may fastly identify a possible bottleneck or pattern in the IoT-blockchain system and provide appropriate solutions.

In the future, we plan to investigate the latency components when the transaction generation follows a probabilistic distribution. Moreover, we will evaluate the effect of different network topologies on the BOL and TOL performance.

REFERENCES

[1] C. Decker and R. Wattenhofer, "Information propagation in the bitcoin network," in *Proc. IEEE P2P*, 2013, pp. 1–10.

[2] V. Chang, P. Baudier, H. Zhang, Q. Xu, J. Zhang, and M. Arami, "How blockchain can impact financial services—The overview, challenges and recommendations from expert interviewees," *Technol. Forecast. Soc. Change*, vol. 158, Sep. 2020, Art. no. 120166.

[3] N. Kshetri and J. Voas, "Blockchain-enabled e-voting," *IEEE Softw.*, vol. 35, no. 4, pp. 95–99, Jul./Aug. 2018.

[4] C. C. Agbo, Q. H. Mahmoud, and J. M. Eklund, "Blockchain technology in healthcare: A systematic review," *Healthcare*, vol. 7, p. 56, Apr. 2019.

[5] T. Jiang, H. Fang, and H. Wang, "Blockchain-based Internet of Vehicles: Distributed network architecture and performance analysis," *IEEE Internet Things J.*, vol. 6, no. 3, pp. 4640–4649, Jun. 2019.

[6] K. Christidis and M. Devetsikiotis, "Blockchains and smart contracts for the Internet of Things," *IEEE Access*, vol. 4, pp. 2292–2303, 2016.

[7] N. Kshetri, "Can blockchain strengthen the Internet of Things?" *IT Prof.*, vol. 19, no. 4, pp. 68–72, 2017.

[8] B. Yu, J. Wright, S. Nepal, L. Zhu, J. Liu, and R. Ranjan, "IoTChain: Establishing trust in the Internet of Things ecosystem using blockchain," *IEEE Cloud Comput.*, vol. 5, no. 4, pp. 12–23, Jul./Aug. 2018.

[9] O. Novo, "Blockchain meets IoT: An architecture for scalable access management in IoT," *IEEE Internet Things J.*, vol. 5, no. 2, pp. 1184–1195, Apr. 2018.

[10] X. Wang *et al.*, "Survey on blockchain for Internet of Things," *Comput. Commun.*, vol. 136, pp. 10–29, Feb. 2019.

[11] "Ethereum." [Online]. Available: https://ethereum.org/ (Accessed: Mar. 2021).

[12] G. Wood, "Ethereum: A secure decentralised generalised transaction ledger," Ethereum Project, Zug, Switzerland, Yellow Paper, 2014.

[13] Y. N. Aung and T. Tantidham, "Review of Ethereum: Smart home case study," in *Proc. IEEE INCIT*, 2017, pp. 1–4.

[14] Q. Xu, Z. He, Z. Li, and M. Xiao, "Building an Ethereum-based decentralized smart home system," in *Proc. IEEE ICPADS*, 2018, pp. 1004–1009.

[15] A. Z. Ourad, B. Belgacem, and K. Salah, "Using blockchain for IoT access control and authentication management," in *Proc. Int. Conf. Internet Things*, 2018, pp. 150–164.

[16] Y. Zhang, S. Kasahara, Y. Shen, X. Jiang, and J. Wan, "Smart contract-based access control for the Internet of Things," *IEEE Internet Things J.*, vol. 6, no. 2, pp. 1594–1605, Apr. 2019.

[17] O. Ali, A. Jaradat, A. Kulakli, and A. Abuhalimeh, "A comparative study: Blockchain technology utilization benefits, challenges and functionalities," *IEEE Access*, vol. 9, pp. 12730–12749, 2021.

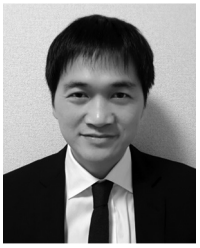
[18] A. A. Monrat, O. Schelén, and K. Andersson, "A survey of blockchain from the perspectives of applications, challenges, and opportunities," *IEEE Access*, vol. 7, pp. 117134–117151, 2019.

[19] S. M. H. Bamakan, A. Motavali, and A. B. Bondarti, "A survey of blockchain consensus algorithms performance evaluation criteria," *Expert Syst. Appl.*, vol. 154, Sep. 2020, Art. no. 113385.

[20] R. Yasaweerasinghelage, M. Staples, and I. Weber, "Predicting latency of blockchain-based systems using architectural modelling and simulation," in *Proc. IEEE ICSA*, 2017, pp. 253–256.
[21] S. K. Kim, Z. Ma, S. Murali, J. Mason, A. Miller, and M. Bailey, "Measuring Ethereum network peers," in *Proc. ACM IMC*, 2018, pp. 91–104.
[22] S. K. Lo *et al.*, "Analysis of blockchain solutions for IoT: A systematic literature review," *IEEE Access*, vol. 7, pp. 58822–58835, 2019.
[23] B. Cao *et al.*, "When Internet of Things meets blockchain: Challenges in distributed consensus," *IEEE Netw.*, vol. 33, no. 6, pp. 133–139, Nov./Dec. 2019.
[24] V. Emery, D. Fragale, A. Zamovsky, and P. Kinnaird, "Atonomi, the secure ledger of things," Atonomi, Seattle, WA, USA, Rep., 2018. [Online]. Available: https://uploads-ssl.webflow.com/5a9f110b6e90d20001b2307d/5b31671d85704919ac1d3838_Atonomi-Network-White-Paper.pdf
[25] "IoT Chain," IoT Chain, Rep. Accessed: Jan. 2022. [Online]. Available: https://iotchain.io/whitepaper/itcwhitepaper.pdf
[26] "Vechain White Paper." Vechain Team. [Online]. Available: https://github.com/ethereum/EIPs/issues/225 (Accessed: Jan. 2022).
[27] J. H. Khor, M. Sidorov, and P. Y. Woon, "Public blockchains for resource-constrained IoT devices—A state of the art survey," *IEEE Internet Things J.*, vol. 8, no. 15, pp. 11960–11982, Aug. 2021.
[28] S. Popov, "The tangle," Rep., vol. 1, no. 3, 2018. [Online]. Available: https://assets.ctfassets.net/r1dr6vzfxhev/2t4uxvsIqk0EUau6g2sw0g/45eae33637ca92f85dd9f4a3a218e1ec/iota1_4_3.pdf
[29] B. Shabandri and P. Maheshwari, "Enhancing IoT security and privacy using distributed ledgers with IOTA and the tangle," in *Proc. IEEE SPIN*, 2019, pp. 1069–1075.
[30] Y. Li *et al.*, "Direct acyclic graph-based ledger for Internet of Things: Performance and security analysis," *IEEE/ACM Trans. Netw.*, vol. 28, no. 4, pp. 1643–1656, Aug. 2020.
[31] M. Cao, L. Zhang, and B. Cao, "Towards on-device federated learning: A direct acyclic graph-based blockchain approach," 2021, *arXiv:2104.13092*.
[32] Z. Li, J. Kang, R. Yu, D. Ye, Q. Deng, and Y. Zhang, "Consortium blockchain for secure energy trading in industrial Internet of Things," *IEEE Trans. Ind. Informat.*, vol. 14, no. 8, pp. 3690–3700, Aug. 2018.
[33] M. A. Bouras, Q. Lu, S. Dhelim, and H. Ning, "A lightweight blockchain-based IoT identity management approach," *Future Internet*, vol. 13, no. 2, p. 24, 2021.
[34] Y. Miao, M. Zhou, and A. Ghoneim, "Blockchain and AI-based natural gas industrial IoT system: Architecture and design issues," *IEEE Netw.*, vol. 34, no. 5, pp. 84–90, Sep./Oct. 2020.
[35] M. I. S. Assaqty *et al.*, "Private-blockchain-based industrial IoT for material and product tracking in smart manufacturing," *IEEE Netw.*, vol. 34, no. 5, pp. 91–97, Sep./Oct. 2020.
[36] A. Dorri, S. S. Kanhere, and R. Jurdak, "Blockchain in Internet of Things: Challenges and solutions," 2016, *arXiv:1608.05187*.
[37] 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 PerCom Workshops*, 2017, pp. 618–623.
[38] B. K. Mohanta, A. Sahoo, S. Patel, S. S. Panda, D. Jena, and D. Gountia, "DecAuth: Decentralized authentication scheme for IoT device using Ethereum blockchain," in *Proc. TENCON Region 10 Conf.*, 2019, pp. 558–563.
[39] T. T. A. Dinh, J. Wang, G. Chen, R. Liu, B. C. Ooi, and K.-L. Tan, "BLOCKBENCH: A framework for analyzing private blockchains," in *Proc. ACM ICMD*, 2017, pp. 1085–1100.
[40] S. Pongnumkul, C. Siripanpornchana, and S. Thajchayapong, "Performance analysis of private blockchain platforms in varying workloads," in *Proc. IEEE ICCCN*, 2017, pp. 1–6.
[41] L. Mikkelsen, K. Mortensen, H. Rasmussen, H.-P. Schwefel, and T. Madsen, "Realization and evaluation of marketplace functionalities using Ethereum blockchain," in *Proc. IEEE IINTEC*, 2018, pp. 47–52.
[42] B. Cao *et al.*, "Performance analysis and comparison of PoW, PoS and DAG based blockchains," *Digit. Commun. Netw.*, vol. 6, no. 4, pp. 480–485, 2020.
[43] Y. Aoki, K. Otsuki, T. Kaneko, R. Banno, and K. Shudo, "SimBlock: A blockchain network simulator," in *Proc. IEEE INFOCOM WKSHPS*, 2019, pp. 325–329.
[44] X. Chen, K. Nguyen, and H. Sekiya, "Characterizing latency performance in private blockchain network," in *Proc. EAI MONAMI*, 2020, pp. 238–255.
[45] Z. Zheng, S. Xie, H. Dai, X. Chen, and H. Wang, "An overview of blockchain technology: Architecture, consensus, and future trends," in *Proc. IEEE BigData Congr.*, 2017, pp. 557–564.
[46] "Go Ethereum." [Online]. Available: https://geth.ethereum.org/ (Accessed: Mar. 2021).
[47] "DevP2P." [Online]. Available: https://github.com/ethereum/devp2p (Accessed: Mar. 2021).
[48] P. Maymounkov and D. Mazieres, "Kademlia: A peer-to-peer information system based on the XOR metric," in *Proc. Int. Workshop Peer-to-Peer Syst.*, 2002, pp. 53–65.
[49] "Secp256k1." [Online]. Available: https://en.bitcoin.it/wiki/Secp256k1 (Accessed: Mar. 2021).
[50] "RLP." [Online]. Available: https://github.com/ethereum/wiki/wiki/RLP (Accessed: Mar. 2021).
[51] N. Houy, *The Bitcoin Mining Game*, vol. 1, Ledger, Paris, France, Dec. 2016, pp. 53–68.
[52] Y. Jiao, P. Wang, D. Niyato, and Z. Xiong, "Social welfare maximization auction in edge computing resource allocation for mobile blockchain," in *Proc. IEEE ICC*, 2018, pp. 1–6.
[53] D. J. Daley and D. Vere-Jones, "Basic properties of the poisson process," in *An Introduction to the Theory of Point Processes: Volume I: Elementary Theory and Methods*. New York, NY, USA: Springer, 2003, pp. 19–40.
[54] "Solidity." [Online]. Available: https://github.com/ethereum/solidity (Accessed: Mar. 2021).
[55] "Web3.js." [Online]. Available: https://github.com/ethereum/web3.js (Accessed: Mar. 2021).
[56] R. R. Fontes, S. Afzal, S. H. B. Brito, M. A. S. Santos, and C. E. Rothenberg, "Mininet-WiFi: Emulating software-defined wireless networks," in *Proc. IEEE CNSM*, 2015, pp. 384–389.

Xuan Chen received the B.S. degree in information engineering from Xi'an Jiaotong University, Xi'an, China, in 2018. He is currently pursuing the M.S. degree with the Graduate School of Science and Engineering, Chiba University, Chiba, Japan.

His current research interests include the blockchain as a service and the underlying P2P networks.



Kien Nguyen (Senior Member, IEEE) received the B.E. degree in electronics and telecommunication from the Hanoi University of Science and Technology, Hanoi, Vietnam, in 2004, and the Ph.D. degree in informatics from the Graduate University for Advanced Studies, Hayama, Japan, in 2012.

He was a Researcher with the National Institute of Information and Communication Technology, Tokyo, Japan, from 2014 to 2018. Since 2018, he has been with the Graduate School of Engineering, Chiba University, Chiba, Japan, where he is currently an Associate Professor. His research achievements have been disseminated in three patents, several IETF Internet drafts, and more than 130 publications in peer-reviewed journals and conferences. His research covers a wide range of topics in networking and distributed systems, including the Internet, the Internet of Things technologies, and distributed ledger technologies.

Dr. Nguyen is a member of IEICE. He also participates in IETF activities.

Hiroo Sekiya (Senior Member, IEEE) was born in Tokyo, Japan, in July 1973. He received the B.E., M.E., and Ph.D. degrees in electrical engineering from Keio University, Yokohama, Japan, in 1996, 1998, and 2001, respectively.

Since April 2001, he has been with Chiba University, Chiba, Japan, where he is currently a Professor with the Graduate School of Science and Engineering. His research interests include high-frequency high-efficiency tuned power amplifiers, resonant dc/dc power converters, dc/ac inverters, and digital signal processing for wireless communications.

Prof. Sekiya is a Senior Member of the Institute of Electronics, Information and Communication Engineers, Japan, and a member of the Institute of Electronics, Information Processing Society of Japan, and the Research Institute of Signal Processing, Japan.