Hindawi
Wireless Communications and Mobile Computing
Volume 2022, Article ID 2032165, 19 pages
https://doi.org/10.1155/2022/2032165

Research Article

A Transaction Traffic Control Approach Based on Fuzzy Logic to Improve Hyperledger Fabric Performance

Lei Hang,[1] **BumHwi Kim**,[2] **and DoHyeun Kim**[3]

[1]*Shanghai Normal University Tianhua College, Shanghai 201815, China*
[2]*Daegu-Gyeongbuk Research Center, Electronics and Telecommunications Research Institute, Daegu 2994, Republic of Korea*
[3]*Department of Computer Engineering, Jeju National University, Jeju 63243, Republic of Korea*

Correspondence should be addressed to DoHyeun Kim; kimdh@jejunu.ac.kr

Received 21 October 2021; Revised 9 February 2022; Accepted 1 March 2022; Published 24 March 2022

Academic Editor: Qingqi Pei

Blockchain is a potential technology for migrating the central server's processing burden into a decentralized, secure, and transparent manner. This technology has had significant influence and revolution in various industries. However, the limited transaction processing power is a stumbling block, especially compared to proven alternatives like distributed database systems. This paper proposes a transaction traffic control approach based on fuzzy logic to enhance the blockchain network's transaction processing capacity. The proposed fuzzy controller is implemented in the smart contract to adjust the transaction traffic flow in real-time network conditions automatically. An experiment environment is built on Hyperledger Fabric to demonstrate the design approach's significance. According to the experiment results, the blockchain performance is significantly improved compared to the baseline. Furthermore, the proposed approach is integrated with an existing blockchain performance-enhancing tool, and the results indicate that the proposed approach is flexible enough to integrate with other existing approaches.

1. Introduction

Distributed ledger systems, such as blockchain, allow for secure and verifiable transactions without a trusted third party [1]. Blockchain technology is progressively turning into style, with applications in varied domains like finance, supply chains, healthcare, the Internet of Things (IoT), sharing economy, and vehicular edge computing [2–8]. A blockchain is a distributed ledger of transactions maintained by all of the blockchain network's participating nodes. The transactions are a series of linked blocks that reflect the business logic. Every node in the network maintains a duplicate copy of the ledger and updates it with new blocks when all nodes agree.

As a result, it is widely expected that blockchain will substantially impact various industries, including finance and real estate, government administration, energy, and transportation [9]. However, to be practical, blockchain must handle transaction rates equivalent to those provided by traditional database management systems and provide some of the same transactional guarantees. Performance is one of the biggest problems in implementing blockchain solutions to replace present centralized servers [10]. Limited scalability, throughput bottleneck, transaction latency, and storage limits may hinder blockchain adoption due to inefficient transaction processing capability and a lack of standards [11]. Bitcoin, for example, has a 1 MB block size restriction and creates a new block every 10 minutes. As a result, the Bitcoin network is limited to 7 transactions per second, rendering it unsuitable for trading with a high rate of change [12]. Ethereum transactions take about 15 seconds to complete [13], though the average time would increase exponentially as network conditions change. Anyone can join a permissionless blockchain like Bitcoin or Ethereum, and each member is anonymous. This means that neither the agreements themselves nor the information exchanged can be kept private as a result. Permissionless platforms typically offer tokens to incentivize excessive mining or accelerate transaction processing to compensate for the lack of privacy. A negative link with the adoption of digital

currencies can significantly impact transaction costs and speed. It also makes it challenging to collaborate with other decentralized platforms because the tokens used on each platform must be consistent [14]. Since permissionless blockchains have poor performance and scalability, side chains are used to offload transaction processing from the main chain.

We concentrate on permissioned blockchains, which have known identities of all participating nodes instead of permissionless blockchains that do not restrict network membership. A permissioned blockchain is a technique to safeguard interactions among a set of entities that have a common aim but may not fully trust one another [15]. The resultant throughput is substantially higher since traditional consensus techniques such as crash fault-tolerant (CFT) or byzantine fault-tolerant (BFT) consensus protocols may be utilized, requiring expensive mining to commit transactions to the ledger.

Table 1 depicts the distinctions between permissionless and permissioned blockchains to overview the two briefly. In comparison to permissionless blockchains, permissioned blockchains are faster, more energy-efficient, and easier to implement. However, there is a lack of depth and complexity in the study on analyzing and improving permissioned blockchain performance. Permissioned blockchain's performance can start to compete with traditional databases for small systems, according to the results in [16]. Differences in consistency models and setup, on the other hand, can have a significant effect on the overall performance.

Hyperledger Fabric [17], of the most well-known permissioned blockchain frameworks for enterprise use cases, implements a distributed ledger platform for running smart contracts, leveraging familiar and proven technologies, with a modular architecture and pluggable components aimed at private enterprises. Many studies have investigated the performance modeling of Hyperledger Fabric. In [18], the authors evaluated multiple Hyperledger Fabric versions (v0.6 and v1.0). In terms of throughput and latency, the results show that Fabric v1.0 outperforms Fabric v0.6. To see how the Fabric network behaved, the authors altered block sizes, resource allocation, state database (DB), and endorsement policies in [19]. The authors explored the impact of peer central processing unit (CPU) and disk type on blockchain latency and throughput in [20]. The authors in [21] evaluated the performance of Hyperledger Fabric depending on the indexes such as throughput, latency, number of transactions, and scalability. The results showed that increasing the transaction send rate significantly impacts network performance, particularly latency. However, the throughput approaches zero when the network reaches its maximum capacity. In [22], the authors used seven distinct scenarios to test the performance of Hyperledger Fabric in terms of transaction throughput and network latency. The impact of various parameters such as batch-timeout, batch size, and the number of peers was investigated in these scenarios. Performance bottlenecks differ slightly at each stage depending on the network setup environment. In [23], the authors looked into the performance characteristics of each phase and assessed ordering services like Solo, Kafka, and Raft. The validation stage has been identified as one of the primary bottlenecks affecting transaction processing capability in this research. Because of the continual expansion in network utilization, the performance of Hyperledger Fabric is a significant problem for businesses. Permissioned blockchain platforms, in their current state, cannot meet the performance requirements of massive provenance use cases in the finance industry, such as stock exchanges, credit card companies, and mobile payment platforms. As a result, it is critical to enhance Fabric's performance to keep pace with its rapid expansion. Even with Fabric's present performance, nodes are overprovisioned to meet peak loads because there is no way to scale up or down a network, and unnecessarily high production costs arise.

Recent optimizations include parallel transaction validation, block and identity certificate caching, and bulk reading of slow CouchDB for the validation phase. However, most of these studies modify the Fabric network's original architecture by updating some stages of the transaction process or adding external user-specific modules. Hyperledger Fabric's architecture is still in the works, receiving various changes and bug fixes during development. There are times when new versions outside of the user-specified release cycles may lead to instability and compatibility issues. Besides, these studies present complex configurations which require deep knowledge of blockchain infrastructure, making it difficult to use for ordinary users. These are the main issues that we need to direct our attention to when thinking about improving the performance of the blockchain.

This paper presents a novel transaction traffic control approach based on fuzzy logic to improve blockchain performance. The fuzzy-based transaction traffic controller in the smart contract can automatically control the transaction traffic flow according to network conditions monitored without third-party intervention. Developers can choose the appropriate language to depend on to implement the smart contract without concerning the blockchain's infrastructure. The performance of the designed approach is evaluated in a clinical trial testbed built on Hyperledger Fabric. The evaluation results indicate that the proposed approach can significantly improve the transaction throughput while reducing the transaction latency.

The contributions of this paper are summarized as follows:

(i) Novelty: this paper presents a novel transaction traffic approach based on fuzzy logic to improve blockchain performance. The fuzzy controller resides in the smart contract so as to automatically perform different operations on received transactions according to real-time network conditions.

(ii) Usability: the usability of the proposed approach has been demonstrated in a clinical trial testbed built on Hyperledger Fabric from our previous work. The experiment results indicate that the blockchain performance is improved significantly in terms of transaction throughput and latency compared to the baseline.

Table 1: Comparison of permissionless and permissioned blockchain platforms.

Property	Permissionless blockchain	Permissioned blockchain
Consensus participant	Public ownership	The nodes that have been chosen
Permission	Public	Either public or private
Efficiency	Low	High
Decentralization	Fully decentralized	Partially decentralized
Cost	Not cost-effective	Cost-effective
Consensus mechanism	Proof of stake, proof of work	BFT, raft, CFT
Use case	Bitcoin, Ethereum	Hyperledger Fabric

(iii) Scalability: the scalability of the proposed approach has been validated by integrating with an existing blockchain performance-enhancing tool, and the results show that the network's performance can be improved further with the proposed approach.

The remainder of this paper is organized as follows: Section 2 examines existing research on improving blockchain performance. The designed transaction traffic control approach based on fuzzy logic is presented in Section 3. The execution of the proposed approach is detailed in Section 4. Section 5 presents the experimental setup and evaluates the performance of the designed algorithm in the clinical trial testbed. The security of the proposed approach is discussed in Section 6. Finally, Section 7 concludes the work by outlining some future research directions.

2. Related Work

Because of the technological improvements in the last few years, blockchain technology has shown to be a robust way of ensuring data integrity, particularly in trustless networks [24–27]. Even though blockchain has the property of data integrity, using it to improve distributed systems entails accepting obligatory performance disadvantages such as high latency and low throughput [28]. Blockchain has received much attention as a crucial technology that enables decentralized and incredibly reliable database management. High latency and low throughput in highly concurrent situations, on the other hand, are regarded as the primary performance bottlenecks of blockchain technology and have an impact on its adoption. Due to the blockchain's nature, the consensus process is complex. The operation and maintenance costs are high because the block data is redundantly stored in the nodes constituting the blockchain. Most of the researchers have tried to build a blockchain network based on high-performance hardware to improve throughput. However, it is challenging to use blockchain in small and medium-sized enterprises with insufficient funds due to the high cost.

Since the initial version of Hyperledger Fabric was launched in 2017, the architecture of Hyperledger Fabric has undergone various changes and bug fixes in development. The Fabric comprises miscellaneous components such as peers, membership service providers (MSPs), clients, and ordering services. The basic transaction workflow goes through the following stages: the endorsement stage, the ordering stage, and the validation stage. Each stage runs independently and does not affect the other stages. The business logic of Fabric is provided by the smart contracts that serve as a trustworthy decentralized program, gaining its trust and security from the blockchain and conjointly the entire agreement across the entire network. Fabric introduces a brand-new approach known as an execute-order-validate to perform transactions in three stages. In simple terms, a submitted transaction will be executed, thus being endorsed, ordered in a block, and before appending to the ledger, these endorsed transactions must be validated against the predefined endorsement policy. The flow of transaction execution across the network is illustrated in Figure 1. The client application should have credentials issued by the certificate authority (CA) to induce approval for submitting dealings proposals. The CA issues credentials to clients who want to submit transaction proposals and authenticates the identities of these clients before they are allowed to participate in the network. Client applications generate transactions, and the application software development kit (SDK) creates connections between the client application and network peers. These peers are the basic units to form the network, and they can be separated into endorser peers or committer peers depending on the type of task. Endorser peers perform on proposals, sign them with their signatures, and then respond with approvals or rejections. Each endorsed transaction is validated against the endorsement policy by committer peers, who then add the block of transactions to the ledger. Endorser peers in a simulated environment handle the received transaction proposal by invoking the smart contract. At this time, the results of transaction execution will not be replicated in the ledger.

Every endorser peer signs the read and write (RW) set and returns proposal responses to the client application for inspection. The client verifies the endorsing signatures to check if the required endorsement policy (e.g., the required number of endorser peers that have to endorse the transaction execution results) has been consummated. These signed transactions are packaged and submitted along with RW sets to the orderer by the client. The batched data is ordered into a block by the orderer and delivered to any or all committer peers. Every committer peer validates the transaction by checking whether the RW sets match the current state or not. Once the committer peer validates the transaction, it writes it to the ledger and updates the state using the Write

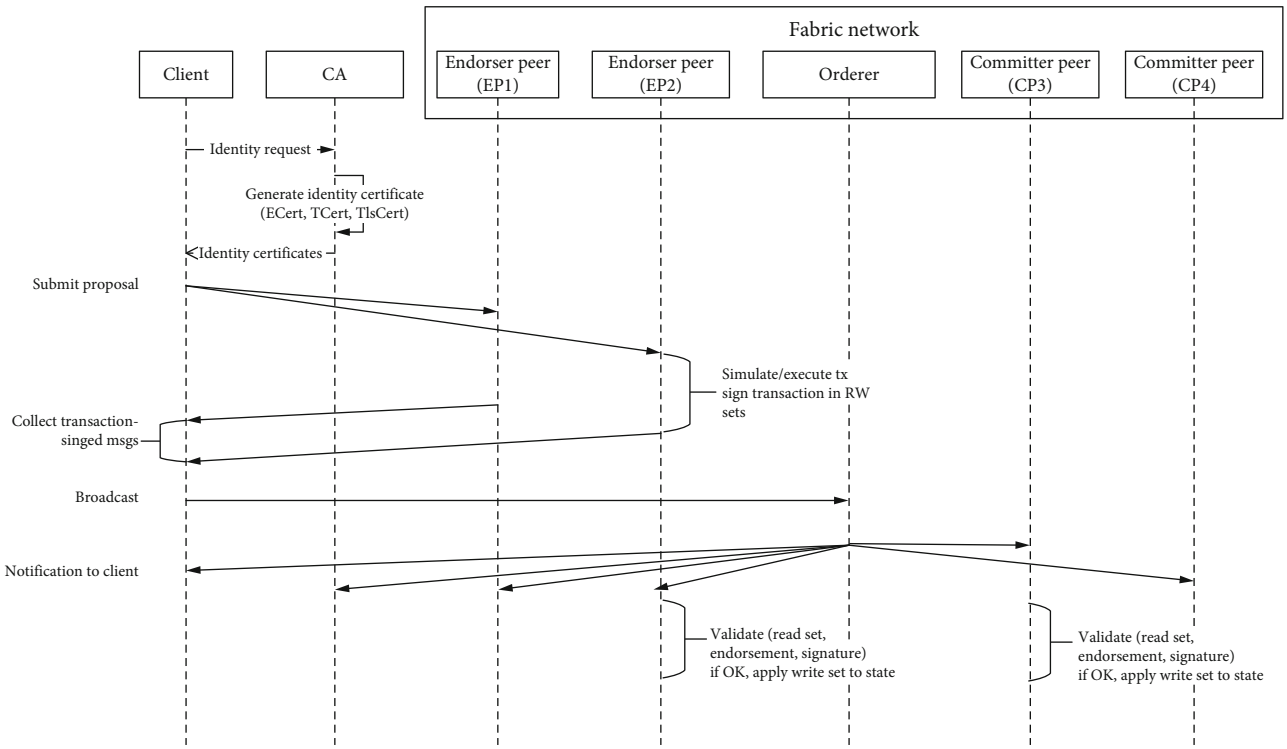

Figure 1: Transaction processing workflow in Hyperledger Fabric.

data from the RW set. Finally, the committer peers send events to the client application to inform whether the submitted transaction succeeded or not. Besides, client applications can subscribe to events to be notified by every committer peer once an event happens.

There are few studies on improving Fabric performance or optimizing its architecture. The authors in [29] propose a configurable blockchain system with a new consensus algorithm to adjust the verification process. The network scale-up/out, space efficiency, and security level are all factors in determining transaction processing capability. As a constraint, the basic authentication procedure is performed within the Hyperledger Fabric network, and general network security is in place. Besides, all peer-to-peer (P2P) data transmission utilizes the public key infrastructure (PKI) encryption module.

Nexledger Accelerator [30] is a novel transaction processing engine with an independent and modular structure that acts as an intermediate between application clients and the blockchain network. Such a feature is appropriate for IoT applications since IoT devices might not have enough computing power to run a novel approach to improve blockchain performance. Nexledger Accelerator provides a straightforward, however robust, transaction processing algorithm using batch scheduling. It classifies the incoming transactions into batched transactions. To this end, Nexledger Accelerator is fastidiously designed to settle on the self-adaptive batch size, counting on the characteristic of requested transactions and the remaining computing resource of the blockchain network. A similar approach is also presented in [31] to reduce the network latency and the number of transfers by batching chaincode execution. Batch processing refers to transferring/handling transactions in a batch by grouping transactions into a set of data rather than processing them sequentially.

Though multiple studies are dedicated to addressing low throughput, these studies only focus on the use cases with giant transaction volumes, neglecting conflicting transactions. Conflicting transactions are those transactions initiated by the client with the constant primary key. The system performance of Hyperledger Fabric may be severely degraded if various conflicting transactions exist. The authors [32] suggest utilizing a cache-enabled endorser to discover conflict transactions before executing the smart contract. When the system is integrated with the cache, it takes fewer steps than the original system to reject conflict transactions. The cache is enforced on the local endorser. The account code is retrieved from the incoming data and saved in the cache as necessary information before each transaction is completed. If the account code is located in the cache, the endorser can refuse to execute such transactions and alert clients, shortening the time between transactions.

Hyperledger Fabric is designed with flexibility and generality in mind, allowing for a wide range of nondeterministic smart contracts and pluggable services to be implemented. Version 1.0, on the other hand, does not include a BFT ordering service implementation, instead offering only a crash fault-tolerant ordering service. The authors in [33] present the design, implementation, and evaluation of a new BFT ordering service, similar to the Practical Byzantine Fault Tolerance (PBFT) protocol. Even

with ordering nodes spread across regions, the BFT-SMART ordering service can achieve up to 10 k representative transactions per second and write a transaction irreversibly within the blockchain in half a second, according to the evaluation conducted on a local cluster and in a geodistributed setting.

In Hyperledger Fabric, every peer node is linked to a distributed repository, so-called state database, to store the latest values of ledger data. LevelDB and CouchDB are the two-state databases that Hyperledger Fabric currently supports. LevelDB is a peer-to-peer database that allows for comparatively speedy access. CouchDB is a client-server database that can be accessed over an HTTP-based representational state transfer (REST) application programming interface (API). The authors in [34] redesign the transaction validation phase of Hyperledger Fabric based on the analysis from a fine-grained breakdown of the validation latency. An optimization approach using chaincode cache is proposed to enable ledger reading and writing operations in parallel. The experiment results reveal performance improvements of 2x for CouchDB and 1.3x for LevelDB.

Similarly, the authors in [35] rearchitect Hyperledger Fabric to increase transaction throughput. They focus on performance bottlenecks beyond the consensus algorithm and present architectural changes that reduce computation and input or output (I/O) overhead during transaction ordering and validation phases. The authors in [36] develop a Hyperledger Fabric GoLevelDB benchmark to characterize database access performance by simulating the transaction behaviors. The characterization results reveal several performance bottlenecks and identify some optimization opportunities to achieve better performance. The authors in [37] propose two other competent APIs between the chaincode and the peer. The first API is called the Differential Update State (DUS), which can reduce reading the state of the key before writing the updated value. As the name implies, the DUS API provides a specified set of operations to compute the updated values via different operations and writes the ledger's commutated value. The second API is called the Compound Request (CR), which supports read, write, and combined functions. It executes all the requests in a specified order and removes the number of requests compared to the DUS API. This feature makes it suitable for use cases requiring frequent parameter read and initialization operations. QiQi, a component-level performance isolation approach for Hyperledger Fabric, is presented by the authors in [38]. By monitoring Fabric's performance, this technique may dynamically adjust the CPU scheduling of Fabric components. The experiment results show that QiQi can support a variety of Fabric ordering services and chaincodes. By rearchitecting Hyperledger Fabric, the authors provide a pipeline execution of validation and commit phases in [39]. They also introduce the sparse peer, a new type of node that may selectively commit transactions to avoid CPU and I/O intensive tasks.

Table 2 describes the comparison between the proposed approaches with some existing studies discussed above. It is evident from the table that most of the existing studies modify the original architecture or process of Hyperledger Fabric. This may result in incompatibility issues, especially when a new version is released. This paper's proposed approach does not change the original system as the fuzzy controller is directly deployed into the smart contract that is flexible enough to be extended. Nexledger Accelerator is a recent blockchain performance-enhancing tool with similar features to the proposed approach. Although it is also built on an independent and modular architecture, however, it is only specified to the Fabric network. Besides, Nexledger Accelerator's configuration is complicated, making it difficult to use, especially for people who know little about blockchain. The proposed approach can be applied to any other blockchain platforms that support smart contracts. Developers can choose the appropriate language to depend on to implement the smart contract without concerning the blockchain's infrastructure.

3. Designed Architecture of the Transaction Traffic Control Based on Fuzzy Logic in Smart Contract

3.1. System Architecture. Figure 2 illustrates the proposed system's architecture, which comprises the admin, transaction traffic measurement analyzer, blockchain adaptor, benchmark DB, and the Hyperledger Fabric network. The Fabric network consists of various peers who copy the distributed ledger and a smart contract. The admin can configure the benchmark and network files for the performance evaluation. A network configuration file describes the system under test and specifies the network's connection requirements. The performance benchmark workload and user-specified test files are specified in a benchmark configuration file. The blockchain adaptor receives the transactions from the client where the workload happens and sends commands to initialize the blockchain network. Multiple clients can submit transactions to the blockchain network and return the transaction responses by invoking the functions specified in the smart contract. The transaction traffic measurement analyzer reads predefined performance statistics and stores benchmark results into the benchmark DB. The fuzzy controller adjusts the transaction acceptance rate by comparing transaction throughput, transaction latency with the acceptance rate. Transaction throughput and transaction latency are input parameters of the fuzzifier. Rules are evaluated in the inference engine. The defuzzifier converts output data (acceptance rate) into nonfuzzy values. The transaction control module obtains the output value to adjust the transaction acceptance rate. The whole process is repeated, and the transaction processing capability of the Fabric network can be dynamically maintained at a suitable level.

Figure 3 details the block diagram of the network configuration. The admin creates crypto certificates for each network entity and updates the network configuration, which specifies the network's topology. The blockchain adaptor consists of a config validator, Fabric SDK, and network configuration module. The config validator validates each network configuration object. The Fabric SDK provides the interface to connect with the Fabric network. The network

Table 2: Comparison between the proposed approaches with existing studies from the literature.

Name	Change of architecture	Use of smart contract	Interoperability	Compatibility	Configuration difficulty
[29]	Yes	No	No	No	High
[30]	No	No	No	Yes	High
[31]	No	No	No	Yes	High
[32]	Yes	No	No	No	Low
[33]	Yes	No	No	No	High
[34]	Yes	No	No	No	Low
[35]	Yes	No	No	No	High
[36]	No	No	No	Yes	Low
[37]	Yes	No	No	No	High
[38]	No	No	No	Yes	High
[39]	Yes	No	No	No	High
Proposed approach	No	Yes	Yes	Yes	Low

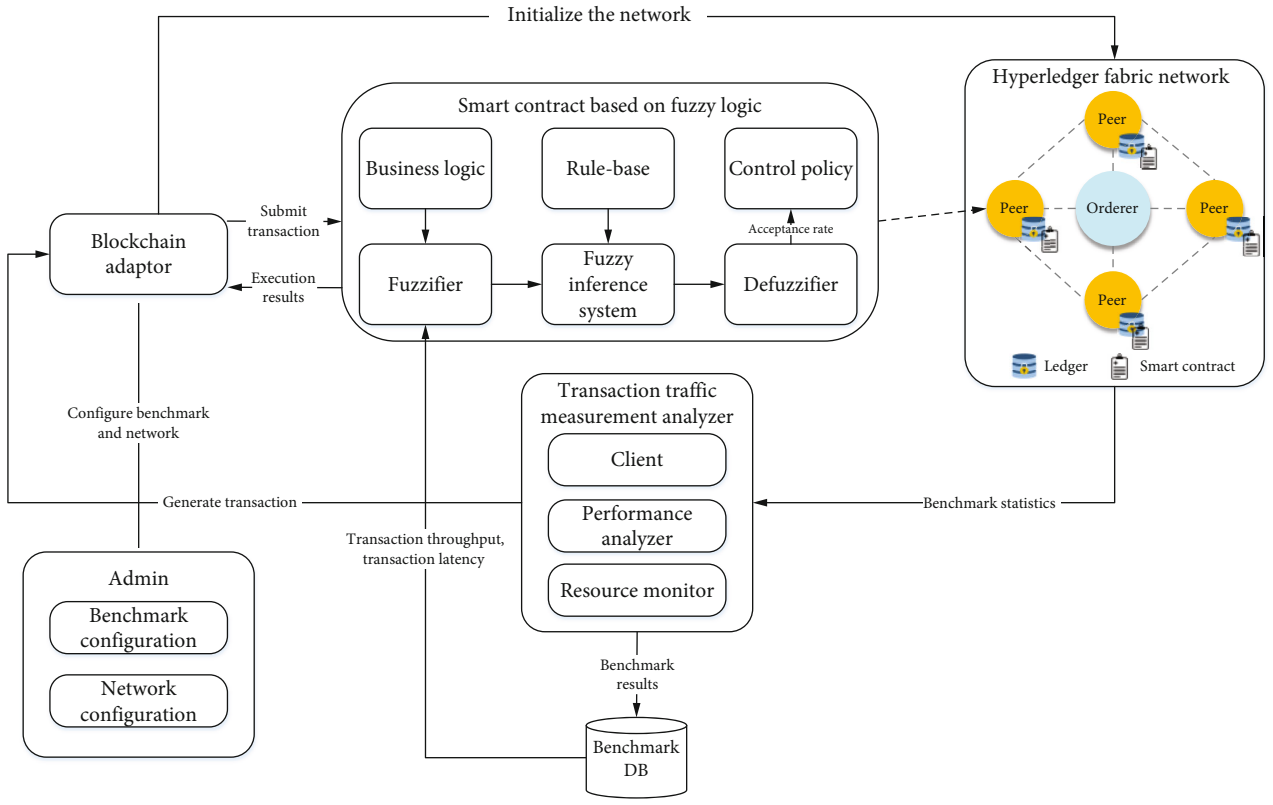

Figure 2: The system architecture of the transaction traffic control based on fuzzy logic.

configuration is used to access information in the connection profile configuration. The blockchain adaptor can send commands to initialize the network (channel, peer) and install the smart contract to the Fabric network.

Figure 4 details the structure of the transaction traffic measurement analyzer. The workload module acts as the brain of the analyzer. When the analyzer schedules transactions for a given round, it is the task of the workload control module to generate the transactions' content and submit it to the adaptor. Multiple local clients are generated according to the benchmark configuration, and each of them is connected with a rate control module. The rate control module regulates the rate of transactions at a fixed rate or follows a specific profile. The resource monitor collects statistics on resource utilization during benchmarking, while the performance analyzer calculates benchmark results according to performance statistics. Benchmark results are collated into a test report by the report generator, and a copy of the report is stored in the benchmark DB.

3.2. Fuzzy Based Transaction Traffic Control. The smart contract based on fuzzy logic is the core component of the proposed transaction traffic control approach that makes decisions based on network conditions from the blockchain.

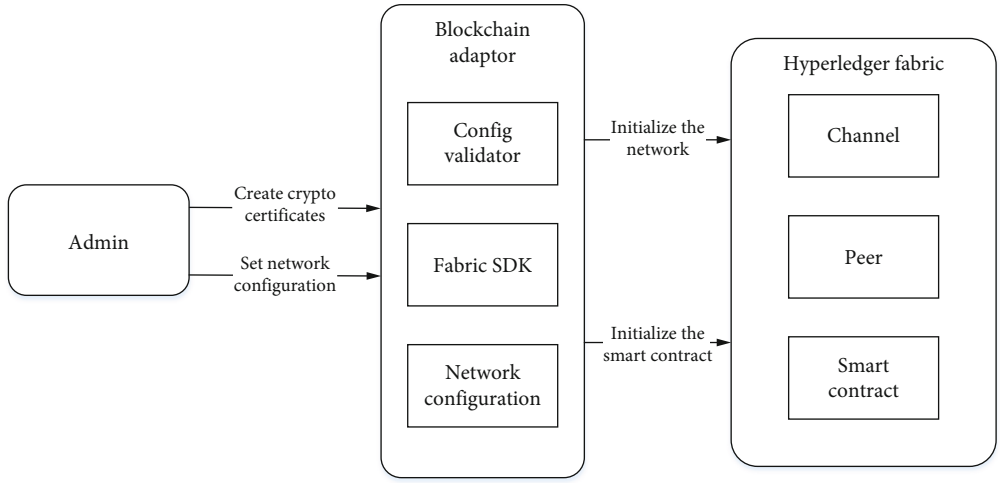

Figure 3: Blockchain adaptor block diagram.

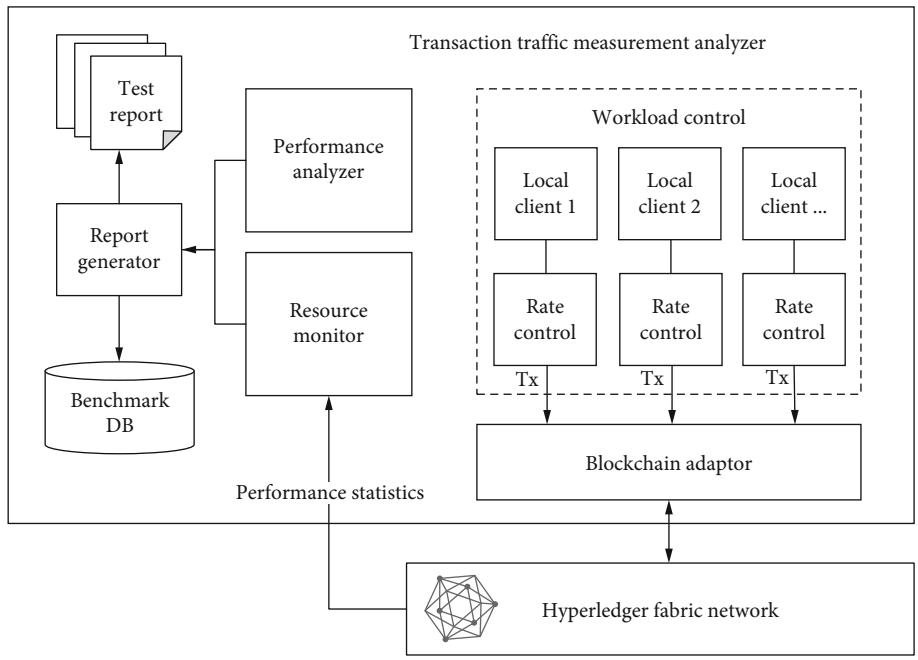

Figure 4: The detailed structure of the transaction traffic measurement analyzer.

The used fuzzy logic module keeps a favorable position by utilizing the general knowledge or experience to take a network-specific decision. In this paper, the Mamdani fuzzy system is used, one of the most well-known theories in the field of fuzzy logic control (FLC) [40]. The linguistic control strategy is born-again into an automatic control strategy supported by up-to-date data. Linguistic expression labeling information granular, like temperature for the weather or age for persons, is expressed as a linguistic variable. It is familiar and comfortable to convert linguistic values using adverbs or adjectives since natural languages do not continuously contain enough worth to define a fuzzy variable scale. This paper utilizes the Mamdani rule structure to set up linguistic modeling to regulate transaction traffic control. Figure 2 shows that the fuzzy logic module mainly consists of a fuzzification module, fuzzy rules, an inference system, and a defuzzification module. The steps of a fuzzy logic implementation can be summarized as follows:

Step 1. Define a set of fuzzy variables (see Table 3).

Step 2. Define a set of membership functions.

Step 3. Build a set of fuzzy rules for each variable (see Table 4).

Step 4. Get outcomes of the fuzzy rules by combing rule strength and output membership functions.

Table 3: Fuzzy set definition for input and output parameters.

(a)

Input variables	Linguistic terms	Fuzzy sets (a, b, c, d)
Transaction throughput	Very low	0, 0, 20, 60
	Low	20, 60, 100
	Acceptable	60, 100, 140
	High	100, 140, 180
	Very high	140, 180, 200, 200
Transaction latency	Very low	0, 0, 0.15, 0.45
	Low	0.15, 0.45, 0.75
	Acceptable	0.45, 0.75, 1.05
	High	0.75, 1.05, 1.35
	Very high	1.05, 1.35, 1.5, 1.5

(b)

Output variables	Linguistic terms	Fuzzy sets (a, b, c, d)
Acceptance rate	Very low	0, 0, 10, 30
	Low	10, 30, 50
	Medium	30, 50, 70
	High	50, 70, 90
	Very high	70, 90, 100, 100

Table 4: Sample fuzzy rule definition.

Transaction throughput	Transaction latency	Acceptance rate
Very low	Very low	Very high
Low	Low	Very high
Acceptable	Acceptable	Medium
High	High	Low
Very high	Very high	Very low

Step 5. Obtain the output by combing the outcomes.

Step 6. Defuzzify the output membership function.

Rule definition is one of the most complex and essential tasks in fuzzy inference systems. It requires domain expert knowledge; for rule generation, we have used the model evaluated from the previous study [41]. The linguistic terms of the input and output variables and their corresponding fuzzy sets are given in Table 3. In the proposed approach, the fuzzy variables for membership functions are defined as transaction throughput and transaction latency, and acceptance rate is valued as "high," "acceptable," and "low." For every acceptance rate factor, the output values are scaled in [0, 100], where the minimum value 0 expresses the lowest acceptance rate and the value 100 represents the highest acceptance rate.

We use both triangular and trapezoidal membership functions to outline the fuzzy variables within the fuzzy system. The trapezoidal fuzzy set A performs $\mu_A(x)$, assigned by four quantified variables (a, b, c, d). The mathematical illustration of the fuzzy membership function is interpreted, as shown in

$$\mu_A(x)\begin{cases} 0, & x < a, \\ \dfrac{x-a}{b-a}, & a < x < b, \\ 1, & b < x < c\,(a < b \le c \le d), \\ \dfrac{d-x}{d-c}, & c < x < d, \\ 0, & x > d. \end{cases} \tag{1}$$

It is worth noting that the trapezoidal function is triangular when b equals c. Equation (2) defines the fuzzy intersection operation between two fuzzy sets A and B, where $A, B \in U$ and x is an element in the U universe:

$$\mu_{A\cap B}(x) = \min\{\mu_A(x), \mu_B(x)\}, \quad \forall x \in U. \tag{2}$$

Moreover, their union is defined by

$$\mu_{A\cup B}(x) = \max\{\mu_A(x), \mu_B(x)\}, \quad \forall x \in U. \tag{3}$$

The proposed fuzzy controller considers two input variables: transaction throughput and transaction latency. The output variable generated by the fuzzy controller is the acceptance rate of the transaction. For combining, these two fuzzy input parameters are used to obtain rule strength by using fuzzy operators. Afterward, the fuzzy inference system evaluates each rule via the membership functions and concludes with the rule's conditions. The quantitative results of the given fuzzy sets and corresponding membership degrees are calculated by

$$\text{mCoA} = \frac{\int f(x) \bullet x dx}{\int f(x) dx}. \tag{4}$$

The defuzzification approach used is the Modified Center of Area (mCoA). It considers all areas covered by the scaled membership functions, even if they extend beyond the range of the output variable, where (x) is the aggregated membership function and x represents the output value. The integration interval is the distance between the minimum and maximum membership function values.

For rule definition, three different approaches are adopted as follows:

(1) Minimum based: in this scheme, the output value is set to the minimum value of given input, e.g.,

$$\text{If } M_1 \text{ is high and } M_2 \text{ is high, then } y \text{ is minimum.} \tag{5}$$

(2) Average based: in this scheme, the output value is set to the average value of given input, e.g.,

$$\text{If } M_1 \text{ is high and } M_2 \text{ is low, then } y \text{ is medium.} \quad (6)$$

(3) Maximum based: in this scheme, the output value is set to the maximum value of given input, e.g.,

$$\text{If } M_1 \text{ is low and } M_2 \text{ is low, then } y \text{ is maximum.} \quad (7)$$

The proposed fuzzy controller is aimed at holding the acceptance rate of the transaction at an optimum level. For example, the acceptance rate is meager if the transaction throughput is exceedingly high and has incredibly low latency. In a word, the fuzzy controller serves as a regulator of transaction traffic in line with transaction throughput and transaction latency. Table 4 presents a sample of specified fuzzy rules, and in total, twenty-five rules are defined. The total number of rules for the fuzzy controller is counted by considering all possible combinations of input variables.

3.3. Transaction Traffic Control Execution Process. Figure 5 describes the execution process of the transaction traffic control approach based on fuzzy logic. At the beginning of each test, the admin should configure the network and benchmark profiles to fulfill the test scenario's requirements. The benchmark file describes how the evaluation test should be executed, including the number of rounds, send rate of the transaction, and settings about monitoring the test network. The network configuration file describes the topology of the test network, such as the configuration of nodes, number of clients, and smart contracts deployed to the test network. Once the network is set up, the admin can start the script to start the benchmark test. One or more clients generate transactions to the adaptor; in turn, the adaptor submits transactions to the Fabric network.

Meanwhile, the transaction traffic measurement analyzer observes and collects the benchmark results. The analyzer calculates the benchmark statistics and stores the results in the benchmark DB. The fuzzifier retrieves the transaction throughput and network latency as the input parameters of the fuzzy inference system. The inference engine evaluates the input parameters according to the fuzzy rules. The defuzzifier produces the acceptance rate as the output value and sends this value to the transaction control module. The transaction module performs transaction traffic control operations concerning the acceptance rate. The transaction execution response is generated and returned to the client. This process is repeated across the entire benchmark experiment until the user stops the test. Finally, all network entities and the smart contract will be removed.

4. Implementation of the Transaction Traffic Control Based on Fuzzy Logic

4.1. Development Environment. Table 5 presents the technology stack used to implement transaction traffic control based on fuzzy logic. The Hyperledger Fabric (v1.4.1) is used as the blockchain infrastructure deployed in the Ubuntu Linux (18.04 LTS) operating system. All the network elements of Hyperledger Fabric are encapsulated as Docker images in Docker containers, which are running in the virtual machine. The Node SDK enables interactions between external applications and the Fabric blockchain network via APIs to submit transactions to the ledger or query content data. Hyperledger Caliper (v2.0.0) is an open-source blockchain benchmark tool that allows users or developers to measure different performance indexes of blockchain implementation [42]. FuzzyIS is a JavaScript library for building a fuzzy inference system in smart contracts that utilize Node.js. MongoDB is a NoSQL database used to store the benchmark results in the JSON-like document with a schema. Express.js is a Node. Js-based web server framework to build web applications provides various REST APIs to manipulate MongoDB.

4.2. Smart Contract Implementation. Algorithm 1 illustrates the process to initialize the fuzzy inference system in the smart contract. The fuzzy inference system in the smart contract contains three core objects. The linguistic variable initializes and adds input and output linguistic variables into the system. In the proposed fuzzy inference system, the input linguistic values are *transaction throughput* and *transaction latency*, while the output linguistic value is the acceptance rate. The variable term describes fuzzy terms for each variable like high/low and very high/very low. The rule describes the connection between input and output linguistic variables. These are conditions like: "if *transaction throughput* is very low and *transaction latency* is very low, then *acceptance rate* should be very high," which describes how the system works. The fuzzy inference system is created with input and output linguistic variables along with described rules. It calculates precise values for output variables referring to the rules given. We can express the algorithmic complexity by using the Big-O asymptotic notation. Algorithm 1 is a constant-time function that can be expressed as $O(1)$.

Algorithm 2 describes the process of invoking the fuzzy inference system in the smart contract. When the smart contract is invoked, it initializes the *Benchmark Url* and connects with the database. A new database instance is created, and the smart contract retrieves the latest record. *Transaction throughput* and *transaction latency* values are extracted from the result. These two values are used as the fuzzy inference system's input variables to compute the output variable *acceptance rate*. Afterward, the smart contract performs different operations on the transaction according to the *acceptance rate's* value, as described in Algorithm 3. Three operations (drop, delay, and accept) are defined in the smart contract along with three thresholds. The algorithmic complexity of Algorithms 2 and 3 is also a constant-time function that can be expressed as $O(1)$.

5. Performance Evaluation

5.1. Clinical Trial Testbed. In this section, we verify the efficiency and usability of the proposed approach by applying it in the clinical trial testbed from our previous work [43]. The workflow of the proposed system is illustrated in Figure 6.

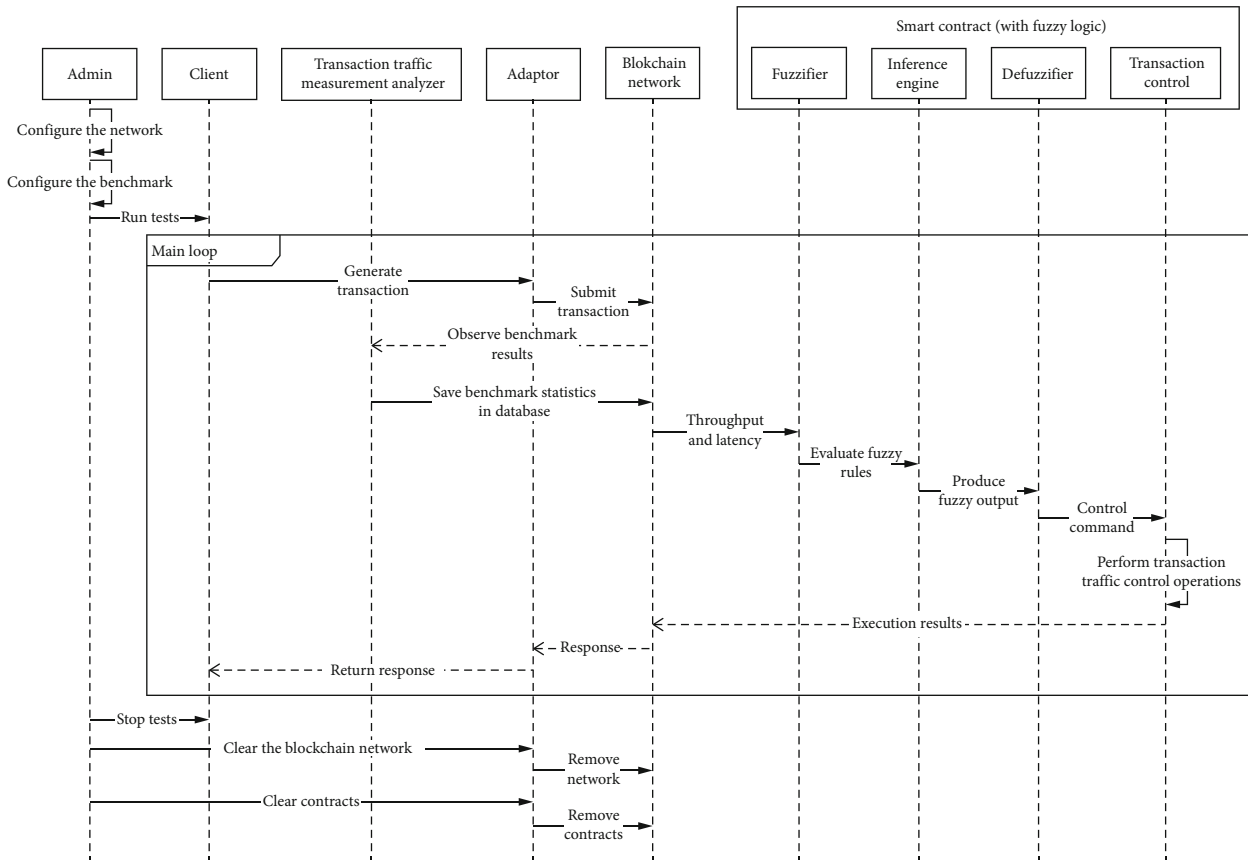

Figure 5: Sequence diagram of the transaction traffic control approach based on fuzzy logic.

Table 5: Development environment of the transaction traffic control based on fuzzy logic.

Component	Description
CPU	Intel Core i5-8500 @ 3.00 GHz
Memory	12 GB
OS	Ubuntu Linux 18.04 LTS
Docker engine	v19.03.8
Docker-composer	v1.24.0
SDK	Node.js v8.17.0
Blockchain infrastructure	Hyperledger fabric v1.4.1
Transaction traffic measurement tool	Hyperledger caliper V2.0.0
FIS library	FuzzyIS
DBMS	MongoDB
Web server	Express.js
Programming language	JavaScript
IDE	VSCode

Each participant must have credentials to get the authorized permission for submitting a transaction to the blockchain network. The principal investigator (PI), clinical research coordinator (CRC), and clinical research associate (CRA) can only read and update their profiles. The PI and CRC can create profiles for new subjects participating in the clinical trial. They can also set profiles for devices (pillbox, bgm), which will update the settings accordingly. The devices collect biomedical data from subjects and generate electronic case report form (eCRF) pillbox/blood glucose meter (bgm) data in the blockchain. The eCRF PI consult data and lab test data are created by the CRC when the subject visits the clinical site. After confirmation by the PI, these data cannot be modified. The CRA can review the data

```
Input (Linguistic Variables, Variable Terms, Rules)
Begin
    Describe a new fuzzy inference system
    Initialize and add Linguistic Variables into the system
    If the input variable is null, then
        Initialize and add transaction throughput variable
        Initialize and add transaction latency variable
    Else
        Throw an error
    If the output variable is null, then
        Initialize acceptance rate variable
    Else
        Throw an error
    Describe Variable Terms for each variable
    IfVariable Term is null
    Describe term for transaction throughput variable
    Describe term for transaction latency variable
    Describe term for acceptance rate variable
    Else
        Throw an error
    Describe Rules for each variable
    IfRule is null
    Describe each rule in the same order as listed in the term description
    Else
        Throw an error
End
```

Algorithm 1: Initialize fuzzy inference system method.

```
Input (Transaction Throughput, Transaction Latency, BenchmarkDB Url)
Output (Acceptance Rate)
Begin
    Initialize the BenchmarkDB Url
    Connect to the database with the BenchmarkDB Url
    If an error occurs, then
        Throw an error
    Else
        Initialize the database instance
        Find the latest record from the collection
    If an error occurs, then
        Throw an error
    Else
        Extract the Transaction Throughput value from the result
        Extract the Transaction Latency value from the result
        Compute the Acceptance Rate value
        Close the database connection
End
```

Algorithm 2: Invoke fuzzy inference system method.

and generate audit queries if there exist errors in data. Afterward, the PI and CRC can access the audit and correct the data accordingly.

The smart contract for the clinical trial testbed contains seven participants, five assets, and nine transactions, as shown in Table 6. The participants are CRC, PI, CRA, subject, pillbox, bgm, and last but not least, the admin of the network. Table 6 gives a list of transactions and describes the transaction structure, which comprises the participant, operation, and resource. Participants are users who can submit the transaction to the business network. The operation specifies the action (e.g., create and read) that the transaction can perform on the resource. ALL represents that the transaction can support all kinds of actions. Resources represent either participant (e.g., CRC and CRA) or assets such as eCRF pillbox data and eCRF bgm data. Transactions submitted by a participant are to perform the specified operation against the resource.

Input (*Acceptance Rate, Drop Tx Threshold, Delay Tx Threshold, Accept Tx Threshold*)
Begin
Set the *Acceptance Rate*
Set the *Drop Tx Threshold*
Set the *Wait Tx Threshold*
Set the *Accept Tx Threshold*
If*Acceptance Rate* is less than the *Drop Tx Threshold* then
Drop the transaction
Else If*Acceptance Rate* is greater than the *Drop Tx Threshold* and less than the *Wait Tx Threshold* then
Delay the transaction
Else If*Acceptance Rate* is greater than the *Acceptance Tx Threshold then*
Accept the transaction
Else
Throw an error
End

Algorithm 3: Transaction control method.

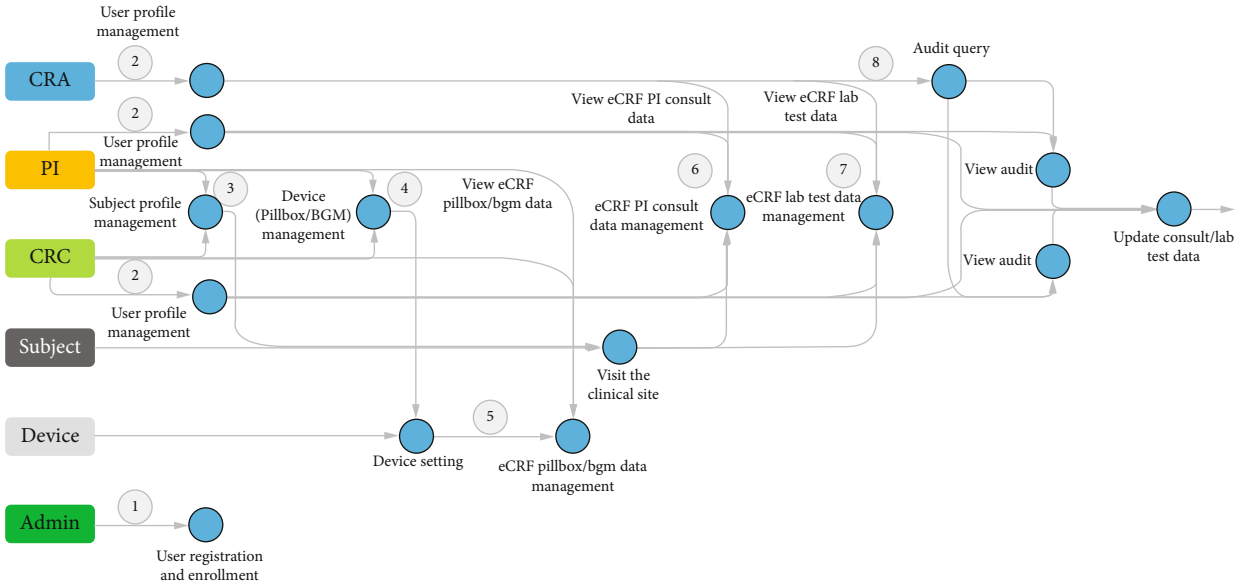

Figure 6: The workflow of the blockchain-based clinical trial service platform.

Table 6: Defined transactions in the smart contract.

Transaction	Participant	Operation	Resource (participant, asset)
User profile management	Admin	ALL	CRC, CRA, PI
Subject management	CRC, PI	ALL	Subject
Device pillbox profile management	CRC, PI	ALL	Pillbox
Device BGM profile management	CRC, PI	ALL	BGM
eCRF pillbox data management	CRC, PI, pillbox	READ, CREATE	eCRF pillbox data
eCRF BGM data management	CRC, PI, BGM	READ, CREATE	eCRF BGM data
eCRF PI consult data management	CRC, PI, CRA	ALL	eCRF PI consult data
eCRF LAB data management	CRC, PI, CRA	ALL	eCRF lab data
eCRF CRA audit	CRC, PI, CRA	ALL	eCRF audit

5.2. Setup for Experiment. This experiment was performed in a single channel of the clinical trial testbed network, consisting of 4 organizations with 6 endorser peer nodes in total. A new block is generated every 250 milliseconds, with a default block size of 10 transactions per block. The default ordering service is solo, with only one ordering node. The default

Table 7: Default experiment setup unless otherwise stated.

Parameters	Values
Number of orgs	4
Number of endorser peers	6
Endorsement policy	AND (a, b, c)
Ordering service	Solo
Block size	10 transactions per block
Block frequency (maximum timeout to create a block)	250 ms
State database	LevelDB
Programming language	Node.js
Use of TLS	No
Number of clients	5
Smart contract	Clinical trial network

state database in this experiment was LevelDB. Table 7 lists the remaining experiment parameters. The experiment's scripts were specified to target one function of our prototype: eCRF lab data generation, since the user most frequently invokes this transaction. The evaluation tests in this section were averaged over numerous rounds to reduce errors caused by system overload and network congestion. Figure 7 shows the snapshot of the command in the console to install the smart contract onto all of the peer nodes for each organization.

5.3. Performance Metrics. The two main performance metrics used to assess the blockchain network's performance are throughput and latency. The throughput can be further divided into two divisions in terms of the processes to be handled. Read throughput is a metric that counts how many read operations are accomplished in a given amount of time, expressed in reads per second (rps). Most of the systems are often deployed adjacent to the blockchain to obtain significant reading query efficiency. As a result, the read throughput of the blockchain is not utilized as a core performance metric. The rate at which the blockchain commits valid transactions in a given period, represented in transactions per second (tps), is known as transaction throughput. Transaction throughput is measured across all nodes in a network, not only at a single node.

$$\begin{aligned}\text{Read throughput} &= \frac{\text{Total read operations}}{\text{Total time in seconds}},\\ \text{Transaction throughput} &= \frac{\text{Total valid transactions}}{\text{Total time in seconds}}.\end{aligned} \tag{8}$$

Regarding the types of operations, latency can be divided into two sections. The whole time it takes to send a read request and obtain a response is read latency. Transaction latency costs the entire network to validate a transaction, including broadcasting time and consensus algorithm allocation time.

$$\begin{aligned}&\text{Read latency} = \text{response received time} - \text{submission time},\\ &\text{Transaction latency} = \text{confirmation time} - \text{submission time}.\end{aligned} \tag{9}$$

5.4. Throughput and Network Latency Evaluation. This section evaluates the performance of the proposed approach using the clinical trial testbed in terms of transaction throughput and transaction latency. This experiment is performed by scaling the transaction send rate and the number of clients to obtain a comprehensive result.

Figure 8 plots the experimental results of the baseline and the fuzzy logic scheme in terms of average transaction throughput with 1 client. The transaction throughput increased linearly with the increase in send rate until it reached around 100 tps. The transaction throughput growth slowed drastically and eventually came to a halt when the send rate exceeded this point. When the send rate was 125 tps, the transaction throughput was 95 tps, and 106.4 tps, respectively, resulting in a 12% improvement in transaction throughput. Figure 9 plots the experimental results of the baseline and the fuzzy logic scheme in terms of average transaction latency with 1 client. It is observed that the baseline network generated more transaction latency than the fuzzy logic scheme when the send rate was above the saturation point. When the send rate was 200 tps, there is a 57.3% reduction in transaction latency.

Figure 10 plots the experimental results of the baseline and the fuzzy logic scheme in terms of average transaction throughput with 5 clients. The transaction throughput increased linearly with the increase in send rate until it reached around 150 tps. The growth of transaction throughput decreased significantly and approached to a flat when the send rate was above this point. When the send rate was 150 tps, the transaction throughput was 108.6 tps, and 125.4 tps, respectively. When compared to the baseline, the fuzzy-based technique can increase transaction throughput by 15.5% at this point. Figure 11 plots the experimental

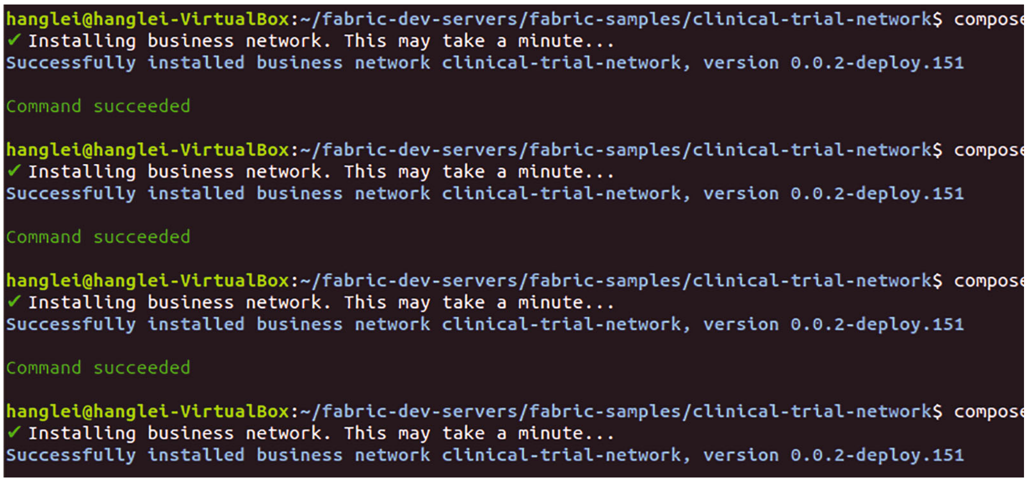

Figure 7: Snapshot of smart contract installation in the network.

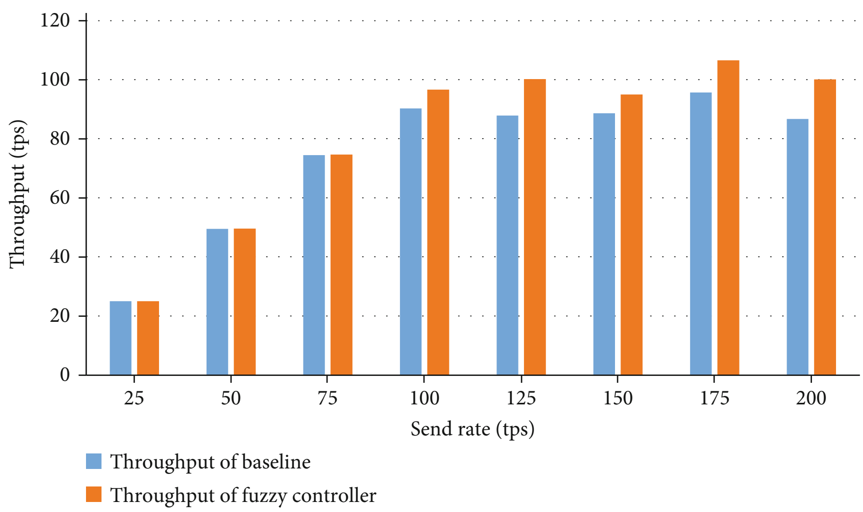

Figure 8: Evaluation of transaction throughput with one client.

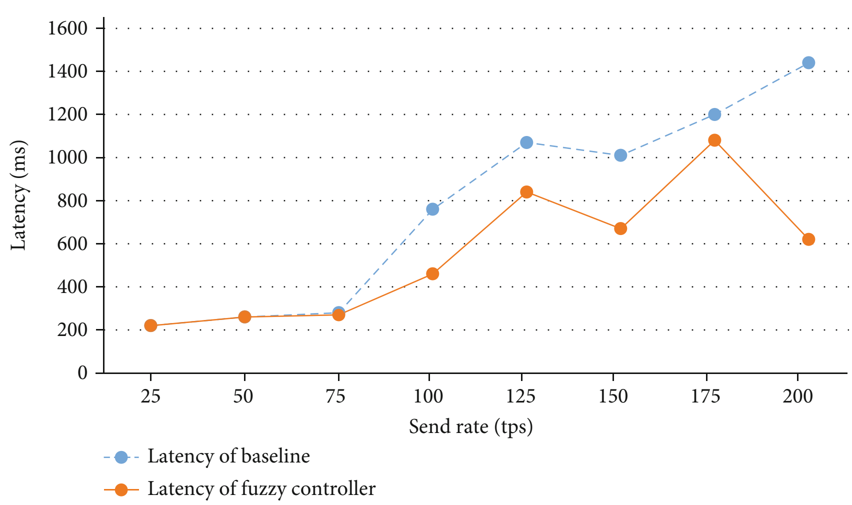

Figure 9: Evaluation of transaction latency with one client.

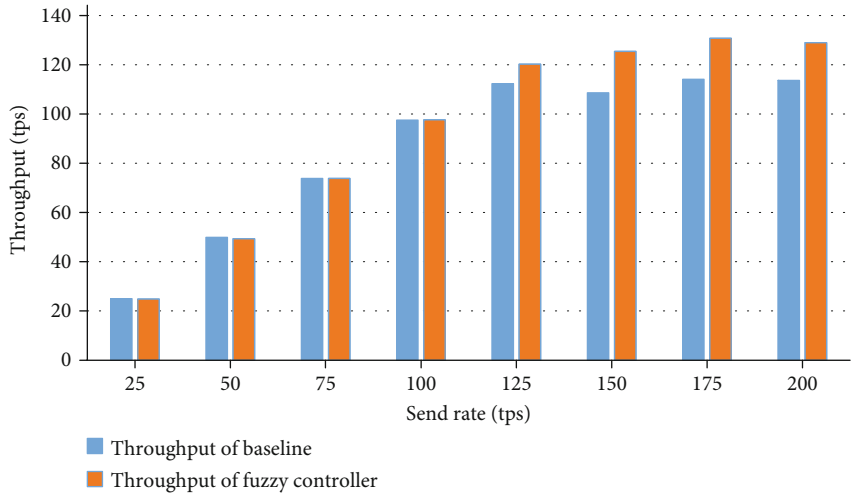

Figure 10: Evaluation of transaction throughput with five clients.

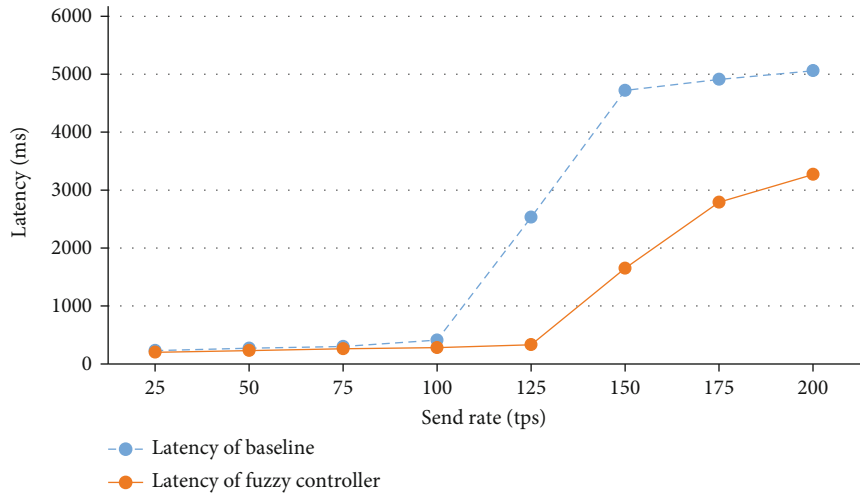

Figure 11: Evaluation of transaction latency with five clients.

results of the baseline and the fuzzy logic scheme in terms of average transaction latency with 5 clients. It is observed that the baseline network generated more transaction latency than the fuzzy logic scheme when the send rate was above the saturation point. This experiment results indicate that the fuzzy logic scheme performs better than the baseline concerning transaction latency and throughput. When the transmit rate was 200 tps, there is a 37% reduction in transaction latency.

The proposed approach is effective, especially when the send rate reaches 150 tps and more. According to the experiment results in the previous work [41], 150 tps is a saturation point of the network, and higher arrival rates can cause lower transaction throughput. The reality is that scaling the rate of transactions sent to the network will likely reach a point where transaction latency becomes untenable because the peers become saturated, consuming all the available CPU and memory resources allocated to the host server. In this case, the fuzzy controller regulates the network's acceptance rate to maintain the transaction processing capability just as we expected.

Furthermore, we perform a collaboration experiment by integrating the proposed approach with one of the existing performance-enhancing tools called Accelerator, overviewed in Related Work. Accelerator retrieves transactions from clients on behalf of blockchain nodes and routes the transactions to the blockchain network. The proposed fuzzy logic controller is applied to the nodes connected with Accelerator nodes. The proposed approach was evaluated thoroughly by comparing it to the baseline network and the baseline network with Accelerator. The experiment was carried out by altering the number of clients while maintaining a fixed send rate of 200 tps, as shown in Table 8. Compared to the baseline, the network with Accelerator boosts transaction throughput by 67.2% and lowers transaction latency by 35.4% for one client. With Accelerator and the proposed technique, transaction throughput is enhanced by 77.7%, and transaction latency is reduced by 52.3%. Compared to

Table 8: Comparison analysis of the performance evaluation with Accelerator.

Number of clients	Send rate (tps)	Performance indexes	Baseline	Accelerator	Accelerator (with proposed approach)
1	200	Transaction throughput (tps)	98.7	165	175.4
	200	Transaction latency	650	420	280
5	200	Transaction throughput (tps)	135.5	246.7	266.8
	200	Transaction latency (ms)	2350	1880	1460

Table 9: Resource utilization of the baseline network.

Name	Memory (max)	Memory (avg)	CPU (max)	CPU (avg)	Traffic in	Traffic out
local-client.js	103.2 MB	88.5 MB	12.64%	9.76%	—	—
http://peer1.company.com	115.0 MB	106.2 MB	12.36%	6.73%	5.6 MB	5.2 MB
http://peer2.home.com	114.2 MB	106.0 MB	16.92%	7.75%	5.7 MB	13.6 MB
http://peer3.home.com	94.6 MB	86.7 MB	12.95%	7.75%	5.2 MB	5.4 MB
http://peer4.hospital.com	133.4 MB	115.5 MB	11.20%	4.74%	3.9 MB	29.0 KB
http://peer5.hospital.com	109.0 MB	96.2 MB	11.46%	5.53%	4.6 MB	4.7 MB
http://peer6.cro.com	112.3 MB	107.0 MB	13.92%	5.24%	5.4 MB	11.1 MB
http://orderer.com	40.6 MB	25.4 MB	6.16%	2.85%	4.3 MB	1472 MB
ca_nodeCompany	6.5 MB	6.5 MB	0.00%	0.00%	536 B	0 B
ca_nodeHome	6.5 MB	6.5 MB	0.00%	0.00%	486 B	0 B
ca_nodeHospital	6.5 MB	6.5 MB	0.00%	0.00%	516 B	0 B
ca_nodeCRO	6.5 MB	6.5 MB	0.00%	0.00%	426 B	0 B

Table 10: Resource utilization of the prototype with the designed approach.

Name	Memory (max)	Memory (avg)	CPU (max)	CPU (avg)	Traffic in	Traffic out
local-client.js	103.7 MB	88.7 MB	12.84%	9.96%	—	—
http://peer1.company.com	115.4 MB	106.7 MB	12.46%	6.93%	5.8 MB	5.4 MB
http://peer2.home.com	114.7 MB	106.5 MB	17.22%	8.05%	5.8 MB	13.8 MB
http://peer3.home.com	95.2 MB	87.1 MB	13.05%	8.15%	5.3 MB	5.6 MB
http://peer4.hospital.com	133.9 MB	115.8 MB	11.35%	4.94%	4.1 MB	31.0 KB
http://peer5.hospital.com	112.0 MB	96.6 MB	11.67%	5.73%	4.7 MB	4.9 MB
http://peer6.cro.com	112.7 MB	107.3 MB	14.02%	5.54%	5.6 MB	11.5 MB
http://orderer.com	41.6 MB	25.6 MB	6.46%	3.05%	4.6 MB	1502 MB
ca_nodeCompany	6.8 MB	6.8 MB	0.00%	0.00%	566 B	0 B
ca_nodeHome	6.8 MB	6.8 MB	0.00%	0.00%	496 B	0 B
ca_nodeHospital	6.8 MB	6.8 MB	0.00%	0.00%	536 B	0 B
ca_nodeCRO	6.8 MB	6.8 MB	0.00%	0.00%	446 B	0 B

the baseline, the network with Accelerator increases transaction throughput by 82.1 percent and reduces transaction latency by 20% for five clients. With Accelerator, the transaction throughput is enhanced by 96.9%, and the latency is reduced by 37.9% using the fuzzy-based approach. The results show that the proposed method is interoperable enough to collaborate with other methodologies.

The performance of the proposed approach is also evaluated in terms of resource utilization, such as memory usage, CPU utilization rate, and traffic in and traffic out. Table 9 and Table 10 represents the results of the baseline and the prototype with the designed approach, respectively. From these two tables, it is observed that there is no significant burden on the network by utilizing the fuzzy-based approach. The results indicate that the proposed algorithm can efficiently utilize system resources.

This paper presents a transaction traffic control approach based on fuzzy logic to improve blockchain platforms' performance that supports smart contracts. A clinical trial testbed implemented on Hyperledger Fabric is used as part of the experimental test to demonstrate the proposed approach's usability. This technique is based on smart contracts rather than the underlying blockchain architecture, accommodating various blockchain implementations easily. Furthermore, it can work with some of the existing blockchain performance enhancement tools to get even more

significant benefits; for example, the proposed approach improves Accelerator's performance.

The experimental results in this section indicate that the proposed approach can be used in business scenarios that require high throughput and concurrency. Some other business disciplines, including supply chains and energy trading, can also benefit from the significance of this work. Due to the absence of transaction processing capabilities caused by a time-consuming consensus procedure that remains a restriction in deploying blockchain on IoT applications, the proposed approach can be expanded into current decentralized food supply chain systems to increase efficiency. IoT sensors can be attached to any product, such as fish entrusted for transportation, and transmit temperature, humidity, and location data. The fuzzy controller is placed on a smart contract that each supply chain business network party can access. The transaction traffic flow can be controlled based on network conditions in real time.

6. Security Verification

This paper introduces an adaptor that acts as a link between the user and the blockchain network. This adaptor is used to set up the network and benchmark profiles that the user has created. It can also receive transactions from numerous clients and distribute them to each network endorser peer. The CA of Hyperledger Fabric secures data transfer between the adapter and the blockchain network. The CA is in charge of digital certificate registration and issuing. By the name subject of the certificate, the digital certificate identifies the owner of a public key. The adapter can consume services using the issued certificate in X.509 to certify itself in network messages. This method allows all participants to trust the digital signature associated with the issued certificate's private key. Recipients of digitally signed messages can check if the signature is valid with the sender's public key to verify the message's origin and integrity.

7. Conclusion and Future Work

Blockchain networks provide a decentralized mechanism for peers to collaborate and create trust through business networks. Each peer node must perform operations and communicate with peers to confirm transactions, reach consensus, and update the shared ledger's status. Many well-known blockchain platforms such as Bitcoin and Ethereum have been widely adopted into different application domains. So far, there has been much confusion about whether or not the blockchain can scale, as well as a paucity of information regarding best practices for improving performance and scaling. Besides, more analysis and evaluation of the performance of these platforms are urgent.

This paper proposes a novel transaction traffic control approach using fuzzy logic to improve blockchain performance. Real-time network feedback is used as input parameters, and the fuzzy controller adjusts the transaction traffic across the whole network accordingly. A clinical trial testbed built on the Hyperledger Fabric is used as the experiment environment to evaluate the proposed approach's performance. The experiment results indicate that the designed approach can significantly improve the transaction throughput while reducing transaction latency. Furthermore, the proposed approach is applied with an existing blockchain performance-enhancing tool called Nexledger Accelerator. The results indicate that the proposed approaches integrate with the existing performance-enhancing approach and improve blockchain performance.

One of this study's limitations is that all benchmark trials are done on a single-host virtual system. The blockchain network is also running on a local network that is inappropriate for production. A future work will refine the prototype system. To test the impact of the proposed approach in the production environment, we will duplicate the results using a cloud service such as Amazon Web Services (AWS) or IBM Blockchain. Furthermore, we will verify the applicability of the proposed technique by deploying it on current smart contract-enabled blockchain platforms such as Ethereum and Corda.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this article.

Acknowledgments

This research was supported by Energy Cloud R&D Program through the Ministry of Science ICT (2019M3F2A1073387), and this work was also supported by Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korea Government (MSIT) (2021-0-00188, Open Source Development and Standardization for AI Enabled IoT Platforms and Interworking).

References

[1] R. Maull, P. Godsiff, C. Mulligan, A. Brown, and B. Kewell, "Distributed ledger technology: applications and implications," *Strategic Change*, vol. 26, no. 5, pp. 481–489, 2017.

[2] L. Hang and D.-H. Kim, "SLA-based sharing economy service with smart contract for resource integrity in the Internet of Things," *Applied Sciences*, vol. 9, no. 17, p. 3602, 2019.

[3] F. Jamil, L. Hang, K. Kim, and D. Kim, "A novel medical blockchain model for drug supply chain integrity management in a smart hospital," *Electronics*, vol. 8, no. 5, p. 505, 2019.

[4] L. Hang, E. Choi, and D.-H. Kim, "A novel EMR integrity management based on a medical blockchain platform in hospital," *Electronics*, vol. 8, no. 4, p. 467, 2019.

[5] L. Hang and D.-H. Kim, "Design and implementation of an integrated IoT blockchain platform for sensing data integrity," *Sensors*, vol. 19, no. 10, p. 2228, 2019.

[6] Y. Lu, Y. Qi, S. Qi, Y. Li, H. Song, and Y. Liu, "Say no to price discrimination: decentralized and automated incentives for

price auditing in ride-hailing services," *IEEE transactions on Mobile computing*, vol. 21, no. 2, 2022.

[7] Y. Lu, J. Zhang, Y. Qi et al., "Accelerating at the edge: a storage-elastic blockchain for latency-sensitive vehicular edge computing," *IEEE transactions on intelligent transportation systems*, pp. 1–15, 2021.

[8] Y. Lu, J. Zhang, Y. Qi et al., "Safety warning! Decentralised and automated incentives for disqualified drivers auditing in ride-hailing services," *IEEE Transactions on Mobile Computing*, vol. 1233, no. c, p. 1, 2021.

[9] U. Bodkhe, S. Tanwar, K. Parekh et al., "Blockchain for Industry 4.0: a comprehensive review," *IEEE Access*, vol. 8, pp. 79764–79800, 2020.

[10] W. Gao, W. G. Hatcher, and W. Yu, "A survey of blockchain: techniques, applications, and challenges," in *2018 27th International Conference on Computer Communication and Networks (ICCCN)*, pp. 1–11, Hangzhou, China, 2018.

[11] S. Wang, L. Ouyang, Y. Yuan, X. Ni, X. Han, and F. Wang, "Blockchain-enabled smart contracts: architecture, applications, and future trends," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 49, no. 11, pp. 2266–2277, 2019.

[12] S. Nakamoto and A. Bitcoin, *A Peer-To-Peer Electronic Cash System*, Bitcoin, 2008, https://bitcoin.org/bitcoin.pdf.

[13] S. Rouhani and R. Deters, "Performance analysis of Ethereum transactions in private blockchain," in *2017 8th IEEE International Conference on Software Engineering and Service Science (ICSESS)*, pp. 70–74, Beijing, China, 2017.

[14] R. Henry, A. Herzberg, and A. Kate, "Blockchain access privacy: challenges and directions," *IEEE Security & Privacy*, vol. 16, no. 4, pp. 38–45, 2018.

[15] C. V. Helliar, L. Crawford, L. Rocca, C. Teodori, and M. Veneziani, "Permissionless and permissioned blockchain diffusion," *International Journal of Information Management*, vol. 54, article 102136, 2020.

[16] S. Bergman, M. Asplund, and S. Nadjm-Tehrani, "Permissioned blockchains and distributed databases: a performance study," *Concurrency and Computation: Practice and Experience*, vol. 32, no. 12, article e5227, 2020.

[17] C. Cachin, "Architecture of the hyperledger blockchain fabric," *Workshop on distributed cryptocurrencies and consensus ledgers*, vol. 310, no. 4, pp. 1–4, 2016, https://allquantor.at/blockchainbib/pdf/cachin2016architecture.pdf.

[18] Q. Nasir, I. A. Qasse, M. Abu Talib, and A. B. Nassif, "Performance analysis of hyperledger fabric platforms," *Networks*, vol. 2018, pp. 1–14, 2018.

[19] E. Androulaki, A. Barger, V. Bortnikov et al., "Hyperledger fabric: a distributed operating system for permissioned blockchains," in *Proceedings of the thirteenth EuroSys conference*, Porto, Portugal, 2018.

[20] J. Garay, A. Kiayias, and N. Leonardos, "The bitcoin backbone protocol: analysis and applications," in *Advances in Cryptology-Eurocrypt 2015. Eurocrypt 2015*, E. Oswald and M. Fischlin, Eds., Springer, Berlin, Heidelberg, 2015.

[21] M. Kuzlu, M. Pipattanasomporn, L. Gurses, and S. Rahman, "Performance analysis of a hyperledger fabric blockchain framework: throughput, latency and scalability," in *2019 IEEE International Conference on Blockchain (Blockchain)*, pp. 536–540, Atlanta, GA, USA, 2019.

[22] S. Shalaby, A. A. Abdellatif, A. Al-Ali, A. Mohamed, A. Erbad, and M. Guizani, "Performance evaluation of hyperledger fabric," in *2020 IEEE International Conference on Informatics, IoT, and Enabling Technologies (ICIoT)*, pp. 608–613, Doha, Qatar, 2020.

[23] C. Wang and X. Chu, "Performance characterization and bottleneck analysis of hyperledger fabric," in *2020 IEEE 40th International Conference on Distributed Computing Systems (ICDCS)*, pp. 1281–1286, Singapore, Singapore, 2020.

[24] L. Liu, J. Feng, Q. Pei et al., "Blockchain-enabled secure data sharing scheme in mobile-edge computing: an asynchronous advantage actor–critic learning approach," *IEEE Internet of Things Journal*, vol. 8, no. 4, pp. 2342–2353, 2021.

[25] J. Yin, Y. Xiao, Q. Pei et al., "SmartDID: a novel privacy-preserving identity based on blockchain for IoT," *IEEE Internet of Things Journal*, 2022.

[26] J. Feng, F. R. Yu, Q. Pei, J. Du, and L. Zhu, "Joint optimization of radio and computational resources allocation in blockchain-enabled mobile edge computing systems," *IEEE Transactions on Wireless Communications*, vol. 19, no. 6, pp. 4321–4334, 2020.

[27] C. Chen, C. Wang, T. Qiu, N. Lv, and Q. Pei, "A secure content Sharing scheme based on blockchain in vehicular named data networks," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 5, pp. 3278–3289, 2020.

[28] C. Fan, S. Ghaemi, H. Khazaei, and P. Musilek, "Performance evaluation of blockchain systems: a systematic survey," *IEEE Access*, vol. 8, pp. 126927–126950, 2020.

[29] P. J. Hee and C. S. Hong, *Blockchain Architecture Using Consensus Algorithm with Adjustable Validation and Its Performance Improvement*, The Korean Institute of Information Scientists and Engineers, 2019.

[30] H. Lee, C. Yoon, S. Bae et al., "Multi-batch scheduling for improving performance of hyperledger fabric based IoT applications," in *2019 IEEE Global Communications Conference (GLOBECOM)*, pp. 1–6, Waikoloa, HI, USA, 2019.

[31] J.-W. Lee and S. Park, "A study on performance improvement of hyperledger fabric through batched chaincode message," in *2020 21st Asia-Pacific Network Operations and Management Symposium (APNOMS)*, pp. 259–262, Daegu, Korea (South), 2020.

[32] F. Lu, L. Gan, Z. Dong, W. Li, H. Jin, and A. Y. Zomaya, "A cache enhanced endorser design for mitigating performance degradation in hyperledger fabric," in *2018 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)*, pp. 1001–1006, Halifax, NS, Canada, 2018.

[33] J. Sousa, A. Bessani, and M. Vukolic, "A byzantine fault-tolerant ordering service for the hyperledger fabric blockchain platform," in *2018 48th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, pp. 51–58, Luxembourg, Luxembourg, 2018.

[34] H. Javaid, C. Hu, and G. Brebner, "Optimizing validation phase of hyperledger fabric," in *2019 IEEE 27th International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS)*, pp. 269–275, Rennes, France, 2019.

[35] C. Gorenflo, S. Lee, L. Golab, and S. Keshav, "FastFabric: scaling hyperledger fabric to 20 000 transactions per second," *International Journal of Network Management*, vol. 30, no. 5, article e2099, 2020.

[36] T. Nakaike, Q. Zhang, Y. Ueda, T. Inagaki, and M. Ohara, "Hyperledger fabric performance characterization and optimization using GoLevelDB benchmark," in *2020 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*, pp. 1–9, Toronto, ON, Canada, 2020.

[37] T. Miyamae, T. Honda, M. Tamura, and M. Kawaba, "Performance improvement of the consortium blockchain for financial business applications," *Journal of Digital Banking*, vol. 2, no. 4, pp. 369–378, 2018.

[38] J. Kim, K. Lee, G. Yang, K. Lee, J. Im, and C. Yoo, "QiOi: performance isolation for hyperledger fabric," *Applied Sciences*, vol. 11, no. 9, p. 3870, 2021.

[39] P. Thakkar and S. Natarajan, "Scaling hyperledger fabric using pipelined execution and sparse peers," 2020, https://arxiv.org/abs/2003.05113.

[40] E. H. Mamdani and S. Assilian, "An experiment in linguistic synthesis with a fuzzy logic controller," *International Journal of Man-Machine Studies*, vol. 7, no. 1, pp. 1–13, 1975.

[41] L. Hang and D.-H. Kim, "Optimal blockchain network construction methodology based on analysis of configurable components for enhancing hyperledger fabric performance," *Blockchain: Research and Applications*, vol. 2, no. 1, p. 100009, 2021.

[42] Hyperldger Caliper Documentationhttps://github.com/hyperledger/caliper.

[43] L. Hang, B. H. Kim, K. H. Kim, and D. H. Kim, "A permissioned blockchain-based clinical trial service platform to improve trial data transparency," *BioMed Research International*, vol. 2021, Article ID 5554487, 22 pages, 2021.