IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 8 July 2022, accepted 1 August 2022, date of publication 10 August 2022, date of current version 18 August 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3197758

RESEARCH ARTICLE

Cell Based Raft Algorithm for Optimized Consensus Process on Blockchain in Smart Data Market

DANA YANG[1], INSHIL DOH[2], AND KIJOON CHAE[1]
[1]Department of Computer Science and Engineering, Ewha Womans University, Seoul 03760, Republic of Korea
[2]Department of Cyber Security, Ewha Womans University, Seoul 03760, Republic of Korea

Corresponding author: Kijoon Chae (kjchae@ewha.ac.kr)

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIP) (No. 2019R1F1A1063194). This work was also supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIP) (No. 2020R1A2C1006497).

ABSTRACT Due to the explosive increase in IoT devices and traffic, big data is developing into smart data that helps the data science experts understand human activities, through the relationship between mobility and resource application of the users in public spaces. For example, smart data markets help to predict crimes or understand the cause of COVID-19 infections. For these smart services, the users agree to the privacy policy so that the personal and sensitive information can be collected by a third party. But the conditions of the privacy policy do not specify whether the information of the users can be tracked. To ensure data transparency, many systems are applying consortium/private blockchains with raft algorithm. The raft algorithm requires nodes to check countless messages for a single transaction. Eventually, as the number of nodes increases, the overall system degradation is derived from the burden of the leader node. This paper proposes a method to process the collected transactions by dividing a certain amount of transactions into cells, without any extra protocol. The proposed scheme also uses the federated learning model with high accuracy and data privacy, in order to determine the optimized cell size in a blockchain system that should lead to consensus on multiple servers. Therefore, the proposed CBR (Cell-based Raft) consensus algorithm proposes a protocol that reduces the number of messages, without interfering with the concept of the existing raft algorithm, in order to maintain stable throughput in the smart data market where massive transactions occur.

INDEX TERMS Smart service, blockchain, consensus algorithm, raft algorithm, federated learning.

I. INTRODUCTION

It is indisputable that we are living in an era of data flooding, which is proven by the enormous amount of data from various environments. For example, IoT devices and traffic are expected to grow in terms of the overall number of users and traffic volume. Over the next 10 years, the number of connected devices is expected to grow to 125 billion [1]. In particular, smart data is growing rapidly and constitutes an important part of smart city development. These new big data sources shed light on the understanding of human activity from an individual and urban perspective, to help discover the relationship between human mobility and resource application in the social and spatial domains in the city [2]. For example, human mobility is an essential attribute of human behavior and acts as a key factor in respiratory infections, including COVID-19. The 4 billion cell phones used worldwide have individual location sensors that can be used to track movement patterns, as well as to verify continued compliance with restrictions [3]. However, the sensors and individual location records collect the sensitive information from the data of the user in the smart mobility data market. Personal data should require personal permission before being shared by the companies for public informativeness. In particular, Data transparency is an important issue in personal privacy. For example, the various services that collect personal data state in the privacy policy "Our services share personal information with affiliated companies

The associate editor coordinating the review of this manuscript and approving it for publication was Ibrar Yaqoob.

or organizations." However, the terms and conditions of the service do not specify whether the user can track the target that is sharing his or her information.

With the introduction of the General Data Protection Regulation (GDPR), European Union (EU) citizens can control the collection and usability of their personal data [4]. The GDPR is only valid in the EU, but multinational corporations are expected to be more transparent about how the companies manage the personal information of customers. Most companies are still currently forced to consider the centralized collection methods vulnerable to personal information violation. Therefore, the multinational companies concentrate on distributed ledger technologies such as blockchain, which have the potential to fully control information and protect personal information, including personal mobility information. The blockchain technology is known to be difficult to modulate, and transactions are transparent to all parties who generated the data [5]. The blockchain is called a distributed data structure or shared ledger, that maintains a list of transaction records and is never changed unless an agreement is reached on the network using consensus algorithms [6].

The consensus algorithm is a solution to the problem of how individual nodes achieve message consistency in the blockchain network. Consensus algorithms are the most important factor in the entire blockchain system because efficiency directly determines the performance of the blockchain! [7]. Consortium/private blockchain has wise applications and uses more efficient consensus mechanisms other than the PoW (Proof of Work) or PoS (Proof of Stake), to avoid high computational costs, low transaction throughput, and long confirmation delay [8]. These kinds of blockchains such as Paxos and Raft algorithms also have a low-complexity protocol developed for distributed systems based on certified nodes [9]. In particular, the raft algorithm is applied to Hyperledger Fabric, which makes consortium/private blockchain suitable for many business applications. The raft algorithm classifies nodes into one leader and multiple followers and divides into the leader election term and transaction record term. When multiple followers respond to a vote or result messages, and the leader receives the majority of the responses, the overall process indicates that the transaction has been successfully inserted into the blockchain. In this processing process, the following problems exist in the raft consensus algorithm.

- Whenever one log is recorded, the leader checks confirmation messages of all followers.
- The load balancing of the leader node causes system performance to degrade overall as the number of followers increases.

To solve this problem, this paper proposes a method of processing collected transactions by dividing the transactions into cells within an acceptable range for the system without additional protocols. Therefore, the proposed Cell-Based Raft (CBR) consensus algorithm is a protocol that reduces the number of messages without interfering with the concept of the traditional Raft algorithm, to maintain stable throughput in the smart data market which involves a large number of the transactions.

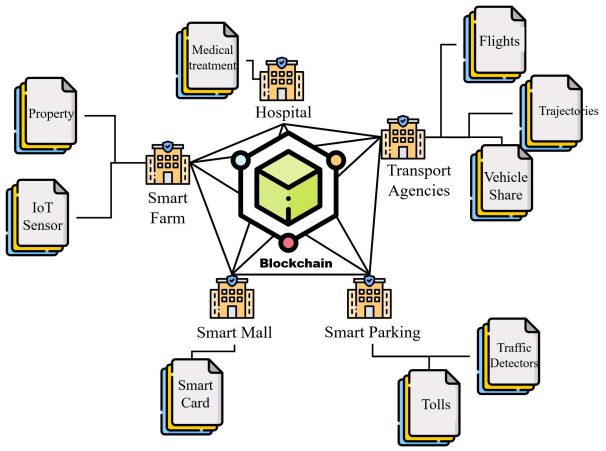

FIGURE 1. Blockchain network for various transaction collectors in the smart data market.

The rest of this paper is organized as follows. The next section discusses traditional research related to the proposed method, such as blockchain systems for the smart data market, the Raft consensus algorithm, and federated learning. Section III introduces the proposed CBR consensus algorithm. Section IV evaluates the proposed mechanism in terms of the number of messages and the Transaction Per Second (TPS) comparison with the traditional raft algorithm. Section V concludes the paper with a summary and future work.

II. RELATED WORKS

A. BLOCKCHAIN SYSTEM FOR SMART DATA MARKET

The smart data are the basis for technological innovation with AI technology in almost all areas and industries [10]. To support this development, the German Federal Ministry for Economic Affairs and Energy (BMWI) launched a project to address important challenges in industrial, mobility, medical, and energy fields called "Smart Data–Innovation from Data" and selected 13 programs with more than 60 participating companies [11]. For smart data to be successful, smart data technology should be developed within projects suitable to various environments. Therefore, successful project results must be achieved through collaboration and effort in terms of entrepreneurship. In addition, smart data markets should also guarantee data transparency because most of the data comes from users. When data owners provide personal information to companies, the companies should provide tracking to show how the personal information is being shared and transformed. To solve this problem, distributed ledger technologies such as blockchain have the potential to give people full control over their information, to keep their personal mobility information secure, and to protect their privacy.

Figure 1 shows a blockchain network for various transaction collectors in the smart data market [12]. Various institutions, including hospitals, smart farms, smart malls, and

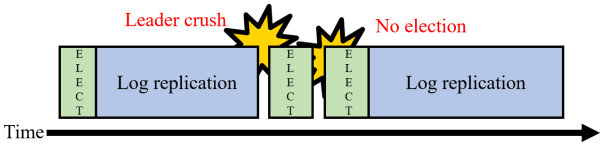

FIGURE 2. Process of the Raft algorithm over time.

smart parking and transport agencies become nodes of the blockchain, and the nodes generate transactions based on the collected data. Each node is the owner of the transactions and can share information including the ID or part of the transaction to another node.

Blockchain network configuration can be divided into public blockchain and private blockchain according to the purpose of the system. There is no limit to the number of anonymous nodes in the public blockchain, but each node can communicate secularly based on encryption. Everyone participating in the public blockchain network is encouraged to act according to the contract to achieve the best outcome of the agreement [13]. In the public blockchain, a long time is needed for anonymous users to mine only one block into the blockchain. Therefore, the public blockchain cannot be used in an environment where explosive transactions are required. However, the consortium/private blockchain reaches a consensus agreement in a shorter time by involving only authenticated node [14].

The system shown in Figure 1 can be classified as a consortium/private blockchain system. Only thoroughly authenticated organizations can participate as blockchain nodes and have the authority to process transactions through a consensus algorithm [15]. In other words, the consortium/private blockchain has higher transaction throughput than the public blockchain system, because the consortium/private blockchain is only a limited number of nodes participating in the consensus scheme [16]. However, if the blockchain system exceeds the transaction threshold, the system can suddenly crash. This study considers the scalability of the blockchain system so that many companies can participate in a successful business project in the smart data market. The proposed CBR algorithm shows that the number of nodes gradually increases and that the nodes process transactions stably, even if the number of nodes is outside the allowable range of the blockchain system.

B. RAFT ALGORITHM

The type of blockchain system depends on the consensus algorithm. Blockchain systems for smart data markets generally use both public blockchain and consortium/private blockchain in a hybrid architecture [17]. In the top layer, a consortium/private blockchain in which only certified institutions or systems participate is implemented [18]. The most commonly used algorithms for consortium/private blockchain are PBFT(Personal Byzantine Fault Tolerance) [19], Paxos [20], and raft algorithms [21]. The Raft algorithm is evolved from Paxos and applied to Hyperledger Fabric as an open platform optimized for business enterprises [22].

The Raft algorithm is a leader-based asymmetric consensus algorithm for maintaining the same log sequence. This consensus scheme is a structure in which one server is a leader responsible for all logs over a certain time, and another server accepts only decisions (accept or fail) [23]. Therefore, even if there are multiple servers, the servers except the leader server do not have data permission and the client only communicates with the leader server [24]. The process of the Raft algorithm is basically divided into a ''Leader election'' period and a ''Log replication'' period, and these two periods are repeated periodically, as shown in Figure 2.

The leader election period, shown as ''elect'' in Figure 2, elects only one of several servers as a leader server through voting. The Raft algorithm generally works stably, but because the leader has all the responsibility, the process returns to the leader election period when the leader crashes or is absence. The follower, who notices absence of the leader firstly, becomes a candidate and receives a vote message from the other followers. If the candidate receives more than half of the votes, the candidate becomes the leader and starts the log replication period. If no leader is elected during the leader election period, the follower servers enter again a new leader election period. The previous leader server become the follower [29]. Whether the leader is alive or not is periodically communicated through heartbeat messages and is checked by followers during each timeout. A summary of the processes performed during the Log replication period is shown in Figure 3. It can be described as follows [26]:

1) The leader server receives commands from the client.
2) The leader appends the command to a new log entry.
3) he leader issues AppendEntries Remote Procedure Call (RPC) messages in parallel to the follower servers to replicate the entry.
4) The followers respond to success (true or false)
5) Commit in state machine: If the command in the log is copied correctly, the state machine on each node executes these commands in the same order and finally obtains a consistent state.
6) The result of the consensus is then returned to the client.

If the client sends the ''z←3'' command, the leader server adds a new log entry and sends AppendEntries RPC messages to the followers shown in Figure 4. The ''Term'' number means the period during which a leader is elected and is responsible for multiple logs [25]. Therefore, changing the Term number means that the leader has changed. During the Log replication period, the leader servers and the follower servers exchange some messages and store the same logs sequentially. The AppendEntries RPC message contains the index term of the preceding log entry. Only when the previous term number and previous index in the AppendEntries RPC message match completely can the follower servers append a new log entry and be committed to the State machine [28]. The State machine is a database in which all servers maintain a constant log sequence in a distributed environment [27].

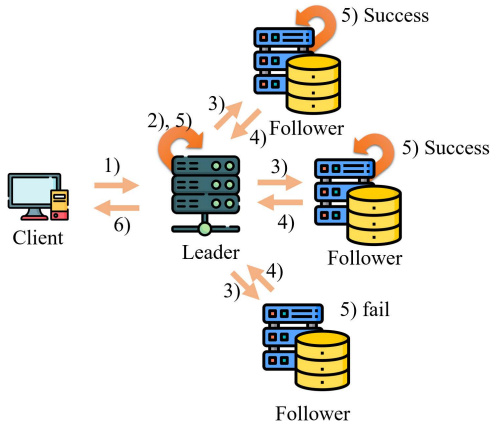

FIGURE 3. Ordering of network messages of the Raft algorithm for log replication process.

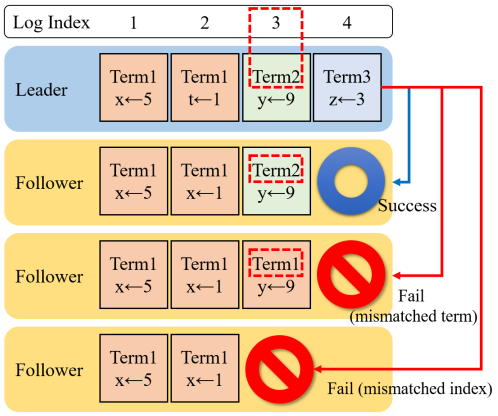

FIGURE 4. Log structure of the leader server and follower server in the case of success or failure of the commit.

If either the terminal number or index does not match, the follower servers send false (fail) in response to the AppendEntries RPC message. When more than half of the responses are received from the follower servers, a complete commit is made to the state machine of all servers. But when the leader receives a fail message from a follower, the leader makes the follower overwrite the previous log entries of the leader [31].

The Raft algorithm was first published in Usenix ATC 14 in 2014 [21], and many researchers have developed the Raft algorithm since then to improve performance of the Raft. The paper by Wang *et al.* [30] maintains log consistency in a separate K-bucket by adding the Kademlia protocol for leader load balancing. However, additional protocols increase actually the number of messages needed for communication relative to the existing Raft algorithm and increase the storage space of the K-bucket. A paper published by Fu *et al.* [32] selects the follower node responsible for each log. However, when the commit fails because the responsible servers for the previous logs are increased, the number of messages needed to determine the responsible server increases, as does the network complexity.

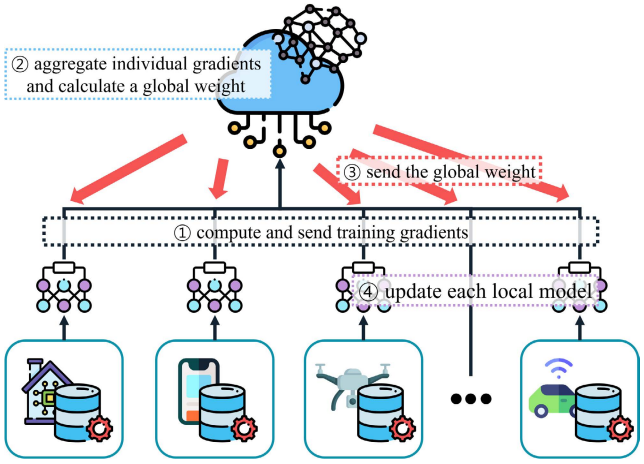

FIGURE 5. Basic architecture and model process of federated learning.

In conclusion, the Raft algorithm relies on the time mechanism to send and receive messages to keep the log consistent, and the leader has the responsibility for all logs. To reduce the burden on the leader node, there have been many studies that added other schemes or transfer responsibility, but these studies did not fundamentally reduce the number of messages. To improve the performance of the leader, this study aims to increase throughput by reducing the AppendEntries RPC message while maintaining the basic concept of Raft. In addition, the proposed mechanism allows the system to reliably process a large number of transactions that occur in the smart data market by finding the optimal network conditions that the system can afford.

C. FEDERATED LEARNING

With the advent of the Artificial Intelligence (AI) era with AlphaGo [35], we have truly witnessed the tremendous potential of AI and have begun to expect more complex and advanced AI technologies in many applications, including financial and medical applications, and unmanned vehicles [36]. Today, AI technology is showing strength in almost every industry and field ([37], [41]). Specifically, in the existing data processing model of AI, it is common for central agencies to integrate various types of data and transmit the results obtained through the built model. However, most applications, except those in some industries, have limited data or poor data quality, making the data more difficult to apply to AI technology than expected [40]. In fact, breaking the barriers between data sources is quite difficult, because all AI projects contain multiple types of data. In addition, the management procedures for privacy and complex data are facing great resistance to data integration between different departments in the same company. Furthermore, it is almost impossible to integrate data that is scattered across countries and institutions, and in some cases, it is not permitted.

Solving these data fragmentation and isolation problems is a major challenge for AI researchers and practitioners today. As a related study, the recently proposed federated

learning by Google builds a machine-learning model based on data sets from many distributed devices to prevent data leakage [38]. The federated learning model focuses on overcoming statistical problems that arise from data integration and on protecting data privacy. In addition, there have been researched efforts on personalization that apply federated learning in a more distributed environment, as shown in Figure 5 [39]. In particular, Figure 5 is a learning method that helps distributed devices collect data and learn from each device, but collects the learning results and updates the modeling of the device. Therefore, it is possible to generate more accurate learning results than to derive results by learning individually in an environment composed of distributed devices such as blockchain. The architecture of federated learning is a model based on data collected from various devices, including smart homes, mobile devices, drones, and smart cars. Based on the collected data, a process that includes the following four steps is performed:

1) Individual devices such as a smartphone, drone, smart car, or hub in a smart home calculate a training gradient through each local model and send the gradient to the central cloud.
2) Since data are collected in different environments, non-uniform gradients arrive at the central cloud. Thus, the individual gradients are aggregated first, and the global weight value is calculated. The basic global weight equation is the average of the gradient.
3) The global weight is transmitted to each device, and each device receives the same value rather than different global weights.
4) The individual devices update each local model through the received global weight.

All the above research focuses on the on-device federated learning that considers distributed mobile user interaction, communication costs, unbalanced data distribution, and the device stability of large-scale distributions. Therefore, this study uses the federated learning model to obtain an optimized cell size in the distributed blockchain-based smart data market. The cell size is generally determined as a device specification considering the communication overhead of the leader node in the blockchain system of the CBR algorithm. However, if a node unilaterally determines the cell size through basic AI modeling without knowing who the leader will be, lower processing performance than the existing Raft may be presented. Therefore, the final goal of the CBR algorithm is to calculate the optimal cell size for the entire blockchain system with the consensus capability of the distributed servers. Where the consensus is achieved by arbitrarily determining a cell size, the number of messages could be lowered to solve the network overhead problem, but it is observed that the specification of the system is low in terms of performance, such as TPS, because the Raft algorithm arbitrarily determining a cell size was not considered at all. Specifically, consensus nodes in the blockchain generate the local weight, and a third-party agency, such as a trust organization, generates the global weight and distributes the global weight to the nodes. The learning data is configured based on TPS information measured by variously changing cell sizes at each node. Finally, it is to obtain the optimal cell size that can lead to the highest TPS according to the number of nodes in the blockchain.

III. PROPOSED MECHANISM

A. PROBLEMS IN TRADITIONAL RAFT ALGORITHM

The Raft algorithm is a mature and efficient consistency algorithm that has been applied to many blockchain scenarios in the field of distributed coordination services and distributed databases. However, the traditional Raft algorithm has some disadvantages in the smart data market which has many transactions in real-time. In the traditional Raft algorithm, the servers work hard to keep the logs constant to keep the latest command of the client. Depending on system policy, the size of the log is often smaller than the number of transactions on the blockchain, so the size of the message is often smaller. If the Raft algorithm is applied to the existing blockchain system as it is, network loads depend on the number of messages. Network loads and leaders cannot manage all the messages, which causes fatal degradation to the system.

As the number of follower nodes increases, the number of communications increases, which limits the speed of the commit. More than half of the nodes in the blockchain network are not controlled by the leader due to the bottleneck of the leader node, which is called network division. Even if the leader does not crash, Raft resumes the leader election in the case of network division. During this period, blockchain networks stop accepting new transactions and have a significant impact on consensus efficiency, real-time response, and load balancing in high real-time environments. In conclusion, transaction throughput decreases severely, which makes real-time processing impossible [33].

Therefore, this study attempts to solve the problem that occurs in the blockchain system where one log of the Raft algorithm is defined as one transaction. Whenever a transaction occurs in the existing Raft blockchain system, the leader issues AppendEntries RPC messages and receives responses from the followers, causing severe bottlenecks from the leader. This paper proposes a CBR algorithm that collects the number of transactions, called cells and completes the commit to the state machine in cell units.

B. CELL-BASED RAFT ALGORITHM

The raft-based blockchain creates blocks on transaction requests and manages replicated logs. In most cases, the logs are generated for each transaction [34]. In order to process transactions based on the optimized cell, All servers in advance to participate in the CBR consensus algorithm cell size have to be determined. Therefore, for stable processing of whoever the leader is, the optimal cell size is obtained by learning data on the throughput of the existing log replication through federated learning. The proposed CBR also defines

TABLE 1. Description of the arguments required for the log.

Arguments	Description
node	number of the node
term	nth term number($T_(n)$)
cell	nth cell number($C_(n)$)
index	log index
timestamp	transaction creation time
generator ID	ID of transaction generator
validation	validation period of data.
payload	log information such as GPS, sensor data, etc

log format as shown in Table 1. Transaction in the smart data market is defined as a basic transaction structure since various types of transactions can be generated. The difference from recording existing logs is protected in data privacy by displaying the owner of the data and the data validity period. The arguments of the log are essential elements for transactions and all servers have to be managed and stored by securely committing to the state machine of the each server. Based on the ''Leader election'' and ''Log replication'' modes of the traditional raft algorithm, this paper defines the various elements to be included in a transaction or message and presents a method to cope with various situations to keep the transaction consistent.

1) LEADER ELECTION

Institutions or companies that are servers in the smart data market form a blockchain network, such as the one shown in Figure 1, and participate in the CBR algorithm-based blockchain system. Each of the servers participating in this way plays a role according to the server status. Figure 6 shows the flowchart of the server status for the process performed in each role. All the servers are initially in the follower state and initialize the values of the term, cell, and log index required for log arguments to zero. Because there is no leader in the CBR blockchain network, the followers cannot receive a heartbeat message from the leader. Therefore, all the followers enter the leader election mode. A follower who initially notices the absence of a leader changes to the candidate state to become the leader.

The candidate sends the RequestVote RPC messages to the other followers. The arguments of the RequestVote RPC message are shown in Table 2. When the candidate sends a RequestVote RPC, the term number is included by raising 1 from the current value. The followers who have received the RequestVote RPC decide whether to elect the candidate as a new leader under two conditions. If any condition does not meet the following conditions, the candidate will not be elected as a leader.

- term number of the candidate $>$ current term number of the follower: If the term included in the received RequestVote RPC message is greater than the current term of the follower, the candidate is eligible to be elected as the leader.
- last cell number of the candidate $=>$ last cell number of the follower: If the last cell included in the received RequestVote RPC message is same as or greater than the last cell of the follower, the candidate is eligible to be elected as the leader.

TABLE 2. Description of the arguments for RequestVote RPC message and response message.

type	Arguments	Description
Request Vote RPC message	term	term number obtained by raising 1 from the term value held when the candidate was in the follower state
	candidateID	Unique ID of the server participating in the blockchain network
	lastTerm	last term number committed to the state machine of the candidate
	lastCell	last cell number committed to the state machine of the candidate
	lastIndex	last log index number committed to the state machine of the candidate
response message	term	term number that the follower updates itself
	voteGranted	true or false(true if the candidate agrees to be the leader, false if opposed to it)
	lastCellHash	hash value of the last cell committed to the state machine of the follower
	currentCell Hash	hash value of the current cell of the follower

Only when both these conditions are satisfied can the followers send ''true'' to the value of voteGranted in the response message to the RequestVote RPC messages. There are several arguments, including voteGranted, in the response message, as shown in Table 2. The values for term and cell in the response message depend on the value of voteGranted. If the value of voteGranted is true, the follower itself updates the arguments of the response message to a value that matches the term and cell values of the candidate. In the value is false, the follower updates to the value obtained by adding 1 to the current term. Then, the follower changes to a candidate state and the previous candidate automatically changes to a follower state.

If a candidate receives more than half of the votes, the candidate takes the role of the leader. The first activity as a leader is to compare the hash values of the last committed cell obtained and the unfilled current cell of the followers. If these two hash values do not match, the new leader makes the followers overwrite and update the information stored by the new leader. In addition, leaders and followers initialize cell numbers to zero. This is because we want to reduce any indiscriminate increase in numbering errors. For example, in the absence of a leader, a follower becomes a candidate, which increments the numbers of both the term and the cell. It is good to have a leader elected during this log selection period, but in unstable networks or blockchain systems that involve too many servers, a leader would not be elected due to a network split. The servers enter a new leader election mode, which raises the numbers of both the term and the cell again. Therefore, the cell number is initialized whenever a new leader is elected, because re-election can cause potential

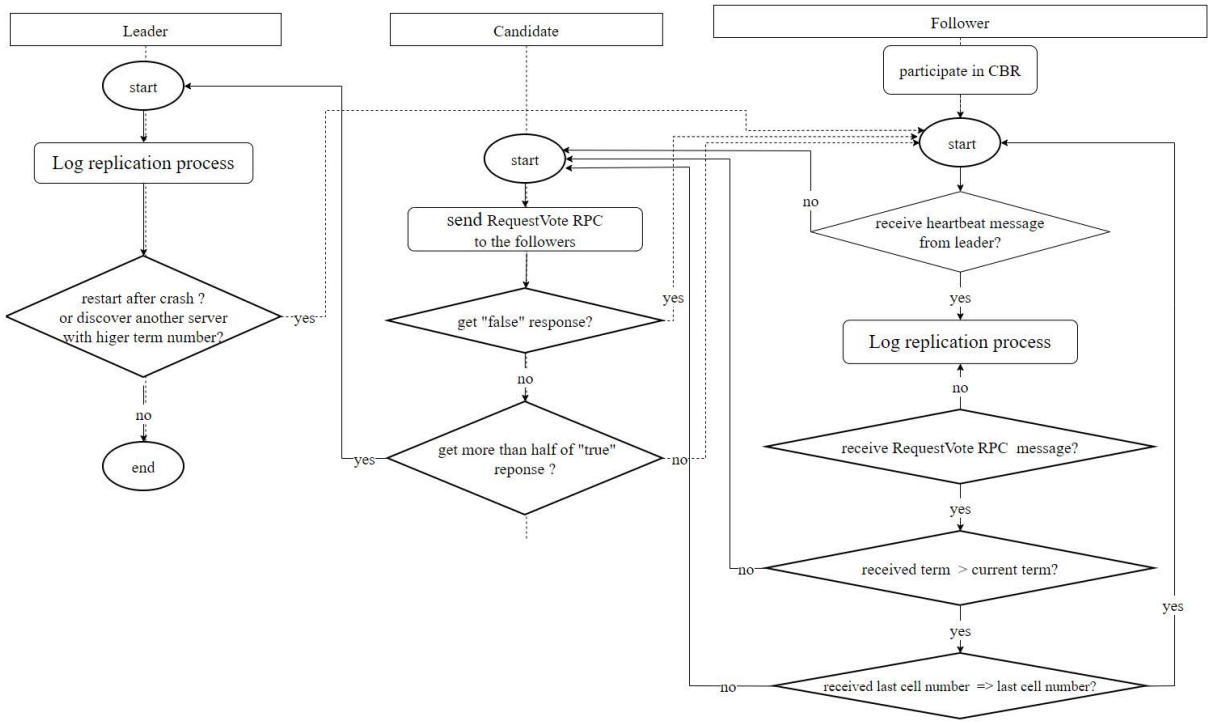

FIGURE 6. Flowchart of server status for the process performed by each role in the proposed CBR algorithm-based blockchain system.

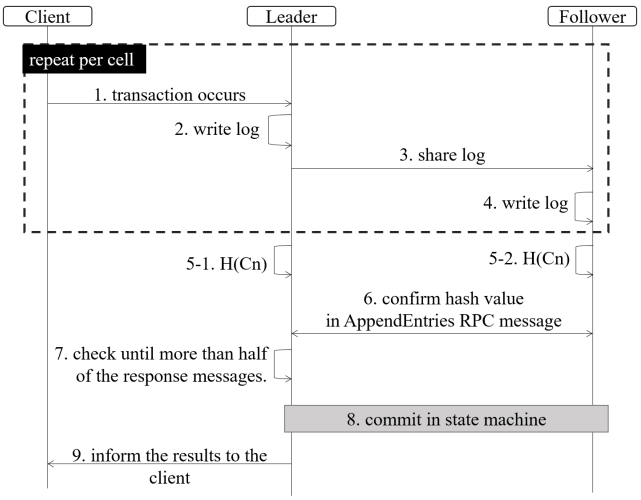

FIGURE 7. Sequence diagram of consensus process for Cell-based raft algorithm.

errors in the system with both term and cell numbers increasing meaninglessly.

2) LOG REPLICATION

After the leader election period is over, the leader and follower servers begin a log replication period in which actual transactions are processed and logs are left. The CBR algorithm proceeds with the log replication process, as shown in Figure 7, to reach a suitable and efficient consensus for the smart data environment. A description of Figure 7 is as follows:

1) When an event occurs at the client and becomes a transaction, the client immediately informs the leader.
2) The leader records the log for the transaction in local storage, not in its own state machine.
3) The leader also issues the AppendEntries RPC message (referred to in Table 3) to the followers, with the recorded log included.
4) The followers write the received log into their local storage. This process is repeated until the logs are filled to the predetermined cell size.
5) When the logs are filled to the size of the cell, the leader and followers collect all the logs in the cell and derive the hash value. $H(C_n)$ refers to the hash value of the nth cell.
6) The followers send their response message with their current $H(C_n)$. The leader compares the follower's $H(C_n)$ and prepares to commit if $H(C_n)$ is matched. If $H(C_n)$ does not match, the current logs stored in the cell are sent to the follower.
7) The leader confirms whether more than half of the responses from the followers are successful.
8) The leader first commits his state machine and informs the followers of the results. The followers also commit and inform the leader of the results, so the same logs are stored on all servers.
9) Finally, the leader sends the results to the client.

TABLE 3. Description of the arguments for AppendEntries RPC message and response message.

type	Arguments	Description
Append Entries RPC message	term	current term number of the leader
	cell	current cell number of the leader.
	lastCellHash	hash value of the last cell committed to the state machine of the leader
	leaderID	Unique ID of the server participating in the blockchain network
	logEntries	log entries that need to be added to the state machine(In the case of a heartbeat message, this arrangement is excluded)
	prevTerm	term number of the previous log entry immediately preceding new log
	prevCell	cell number of the previous log entry immediately preceding new log
	prevIndex	index number of the previous log entry immediately preceding new log
response message	term	current term number of the follower
	cell	current cell number of the follower
	logGranted	true or false(true if the log is successfully stored, false if opposed to it.)
	lastCellHash	hash value of the last cell committed to the state machine of the follower
	currentCell Hash	hash value of the current cell of the follower

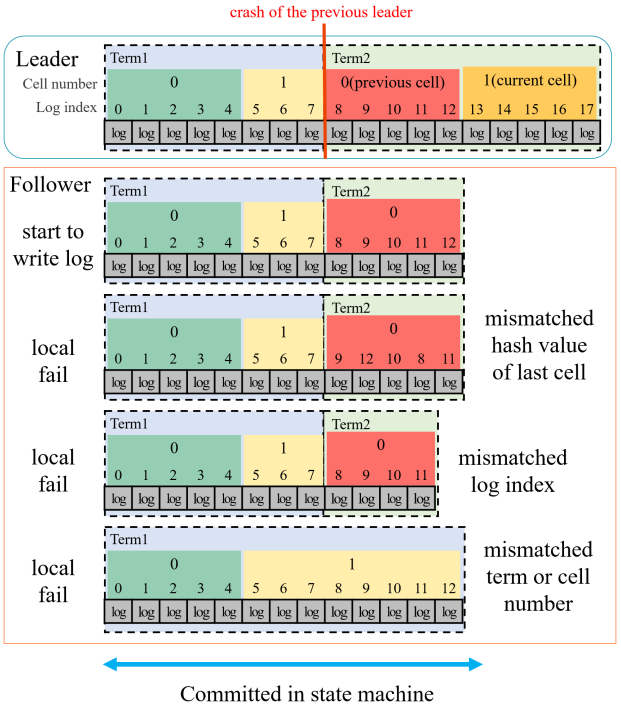

FIGURE 8. Standard logs of the leader and the various logs the follower, before a new log is started.

During the log replication period, the leader and followers maintain the latest log state in the form of the messages shown in Table 3. Through "lastCellHash," it is possible to check whether the contents and order of the stored log are accurate. The server notices whether it receives the latest logs through the remaining arguments of the AppendEntries RPC message. The arguments other than "logEntries" in the AppendEntries RPC message become the arguments of the "Heartbeat" message periodically sent within a timeout.

Finally, the logs must be stored in each server in the log sequence of the leader shown in Figure 8. The size of the cell is designated by the server when the CBR-based blockchain system starts. Due to the crash of the previous leader, the last cell of term1 commits based on the log sequence of the current leader. Then, all followers, centering on the leader, perform the log replication process shown in Figure 7. However, it is rare for all servers to communicate stably for every term and cell and to successfully perform log replication without exception. This paper confirms the previous logs before adding new logs or cells to cope with the exceptions, which are called "local fails." For example, the "lastCellHash" argument is an element that can confirm whether the order of the previous logs stored in the state machine of the follower is correct. If the log index does not match, the error is known as a local failure. Even if the log indices match, in the case of the mismatched term or the mismatched cell number, the errors are considered a local failure and the follower does not receive a new log from the leader. These cases are identified by the arguments of

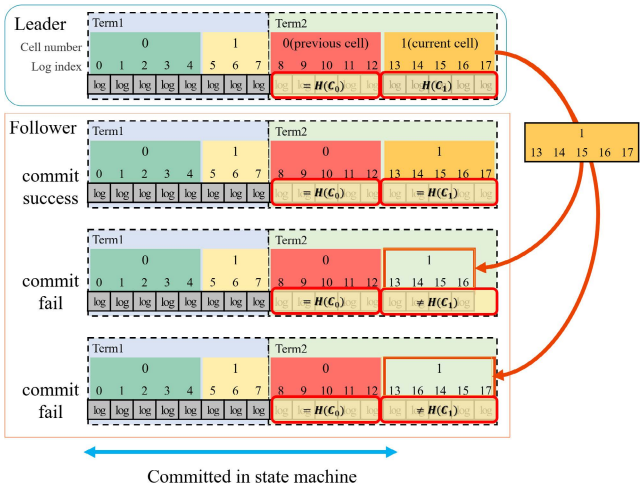

FIGURE 9. Standard logs of the leader and various logs of the followers before a new cell commits.

the heartbeat message and are solved by the leader sending the logs of the previous cell to the follower experiencing local failure. The CBR mechanism periodically checks the logs of the previous cell through the heartbeat message and prepares to save the next logs.

When the logs of the previous cells completely match the content and order, the value of "lastCellHash" becomes the value of $H(C_{n-1})$,(the value of $H(C_0)$ shown in Figure 9). The leader then sends the logs to the followers until the log is filled by the size of the cell. When the current cell are filled based on the leader, the leader changes "lastCellHash" to

$H(C_1)$ and raises the values of ''cell'' and ''prevCell'' in the AppendEnries RPC message. If the logs of the current cell stored in the follower match the leader perfectly, the follower responds the true as value of ''logGranted'' and $H(C_1)$ as ''lastCellHash''. However, if any of the logs in the current cell are empty or out of order, the value of $H(C_1)$ is different from $H(C_1)$ of the leader. Since a follower is in a fail state that cannot be committed, the value of ''logGranted'' is answered with false. The leader who receives false as the ''logGranted'' in the response message sends all the logs of its own cell and makes the follower overwrite the logs.

Like the concept of the existing Raft algorithm, this study focuses on all the logs of the leader. This paper also periodically confirms the state of the previous log for the heartbeat message and ensures the stability of the log. The leader is responsible for all logs and processes many logs because the leader appends and checks the logs on a cell basis. In order not to interfere with state machines in the event of unexpected errors, the CBR scheme maintains uniformity by overwriting the log sequence of the leader before adding or committing the new logs.

The Raft algorithm is basically a published protocol intended for an environment with a small number of distributed servers. However, many of the current environments that require blockchain are not centralized environments composed of a few nodes and need to deal with mass data. If the traditional Raft algorithm is applied to the real environment, the system is paralyzed when the throughput exceeds what a leader can handle. Therefore, even if the node increases in an actual environment, a CBR mechanism that stably processes mass data is needed.

IV. IMPLEMENTATION

This study implements the CBR mechanism based on the existing Raft algorithm and constructs a blockchain system. A performance evaluation is conducted by comparing the basic Raft and the CBR scheme. Unlike Raft in existing distributed systems which store the changed values in the log, Raft algorithms in blockchain systems proceed with the consensus process by storing one log on a transaction line or block. In fact, we implement CBR and measure performance by constructing a virtual node provided through an open-source, as there is no environment in which more than five devices can be obtained and agreed upon.The Raft algorithm is an already well-known algorithm, and we use the already implemented algorithm project by downloading Github (https://raft.github.io). Most of the downloaded projects were developed to achieve consensus in accordance with CBR, but improved AppendEntries RPC messages and RequestVote RPC messages, as well as the main parts where each node stores the logs in cell size in logging to State Machines. In particular, the number of messages and transactions per second (TPS) are compared for each node, and the TPS is measured for each cell size. Even if the node increases, these simulations on the most optimized cell size prove the excellence of the CBR mechanism that reliably processes consensus on transactions.

TABLE 4. Simulation environment.

Criteria	
Device Type	Desktop
CPU	i5-8250
Memory	8GB
OS	Window 10
Language	java(JDK12)
Encoding	UTF-8
Tool	Spring Tool Suit(STS)
Main library	java.io.*
	java.net.*
	java.util.*
	java.security.MessageDigest
	https://raft.github.io/

The implementation was developed in a simulation environment, as shown in Table 4. The basic Raft algorithm was developed using various languages and tools by many researchers. This study used a Java-based project that included a stable environment and many functions among the official Raft sites. Because no additional protocols were required, no other libraries were added, and default libraries were made available. The hash value of the logs stored in the cell is the result of the SHA-256 function.

Figure 10 shows part of the console screen in the blockchain system implemented with the proposed CBR algorithm. The proposed CBR algorithm-based blockchain system sets five servers, and cell 10 sizes and runs this system. In first running the proposed blockchain system, all nodes numbering from 1 to 5 start with a follower, as shown in Figure 10. During the first leader election period, the fifth server, which quickly recognizes the absence of a leader, becomes a candidate and sends RequestVote RPC messages to the other servers. The fifth server, which receives more than half of the RequestVote RPC messages, becomes the first leader, and the log replication period is started. The last red square line shown in Figure 10 shows the current term number, the latest committed cell number, the latest committed index number, and part of the latest committed hash value every 500 milliseconds.

V. EVALUATION

A. PERFORMANCE EVALUATION

This study performs consensus on a cell and simulates the CBR algorithm in various ways to evaluate performance. In particular, by comparing the existing Raft algorithm with CBR, this study proves stable and high transaction processing.

The traditional Raft algorithm requires that one leader deal with many messages received from the followers because the more nodes increase, the more followers the leader must manage. Figure 11 shows the number of messages that leaders must handle as nodes increase in the transitional Raft algorithm-based blockchain system. This paper measures the

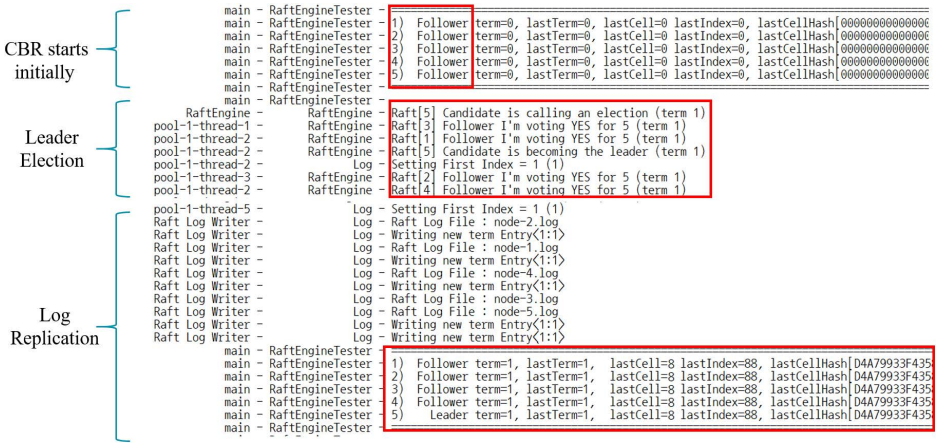

FIGURE 10. Implementation capture image of the proposed CBR algorithm-based blockchain system.

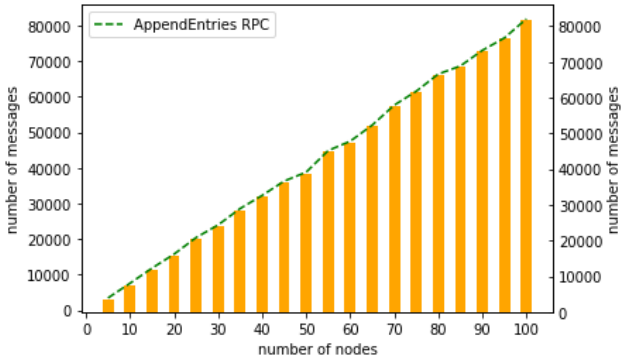

FIGURE 11. Number of AppendEntries RPC messages according to the number of nodes in the transitional Raft algorithm-based blockchain system.

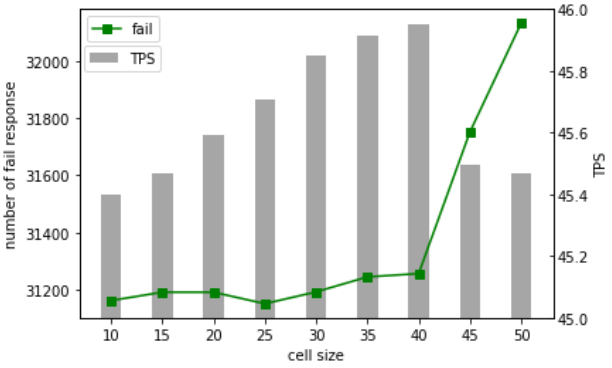

FIGURE 13. Relationship between TPS and fail response messages according to the cell sizes in the proposed CBR-based blockchain system.

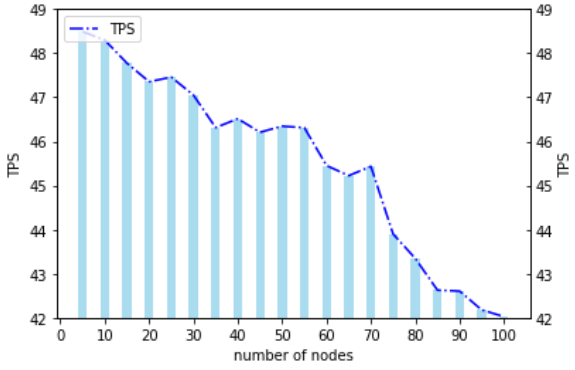

FIGURE 12. TPS according to the number of nodes in the transitional Raft algorithm-based blockchain system.

number of each AppendEntries RPC message when the number of nodes is increased by 5 and increased to 100 for consensus in 1000 transactions. As the number of nodes is increased to a constant size, Figure 11 represents gradual growth in the AppendEntries RPC messages. However, the traditional Raft algorithm shows that the TPS decreases as nodes increase, as shown in Figure 12. In the traditional Raft blockchain, which has 75 nodes, the TPS dropped sharply from 45.43 to 43.90. This result means that the traditional Raft-based blockchain can affect the entire system due to the load balancing problem of the leader. Because this suggests that this problem can lead to serious overhead in the blockchain system, the proposed Raft algorithm should determine the cell size that is acceptable in the blockchain consensus system.

This study measured TPS and the number of fail messages while increasing the cell size in a CBR-based blockchain with 70 nodes that stably accommodate the largest number of nodes, as shown in Figure 13. The fail message means that the follower has abnormally stored the log and notifies this situation to the leader. The cell size begins at 10, and the TPS and the fail messages are measured while the cell size is gradually increased by 5. The TPS increases in proportion to the cell size, but when the cell size reaches 45, the TPS decreases rapidly. The reason is that, as the cell size increases, the follower logs are not stored like the log sequence of the leader. Before committing the cell, the followers send

TABLE 5. Optimal cell size and average accuracy according to node size derived through federated learning.

node	optimized cell size	average accuracy
5	80	99.82%
10	75	98.49%
15	75	99.50%
20	65	98.66%
25	70	96.22%
30	55	95.59%
35	60	98.76%
40	50	97.80%
45	50	97.89%
50	55	98.75%
55	50	99.44%
60	40	96.67%
65	45	98.27%
70	40	99.59%
75	40	99.68%
80	35	98.87%
85	30	96.57%
90	35	98.11%
95	20	97.68%
100	25	99.19%

the leader a fail response to the AppendEntries RPC messages. The consensus process is delayed because the follower receives and overwrites all the logs of the cell. As shown in Figure 13, the number of fail response messages in 45 cells increases rapidly. Therefore, the cell size should not increase unconditionally. The optimal cell size that can be accommodated by both the leader and follower should be obtained.

We simulated various cell size-based Raft consensus processes for each of the 5 to 100 nodes and observed that the TPS was different for each situation. Therefore, the proposed Raft algorithm calculates the optimized cell size using federated learning. Because the logs shown in Table 1 contain a lot of implicit information, the logs are suitable for use as learning data, and communication resource data such as network bandwidth is also included in the learning data. Each node constituting the blockchain calculates local weight with learning data including TPS and receives global weight from trust organization to update the modeling. Figure 14 shows that the more the local model of the node is updated through federated learning, the more accurately the optimized cell can be obtained. In the early epoch of 10 or less, the accuracy dropped to around 0.5, but after that, the exact cell size can be obtained by converging to 1. Furthermore, as the learning of each node increases, the training loss rate value also gradually decreases and converges to zero, proving that optimized cells can be accurately obtained. In conclusion, our simulation proves that the accuracy is high and derives the optimized cell size for number of nodes, shown in Table 5. The result shows that the smaller the node, the higher the TPS is when the Raft consensus is processed with a larger cell size. This result also shows an average accuracy of 98.28%.

The number of AppendEntries RPC messages and TPS were measured again by performing the Raft algorithm with

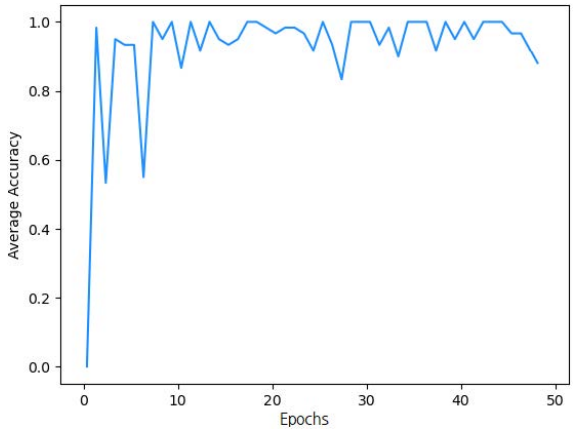

FIGURE 14. Average accuracy according to epochs used for federate learning.

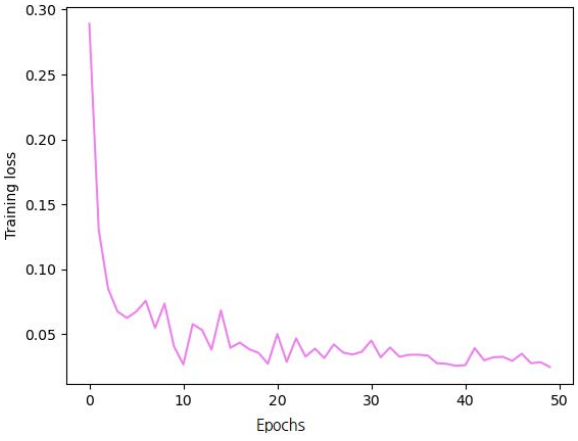

FIGURE 15. Training loss according to epochs by federated learning.

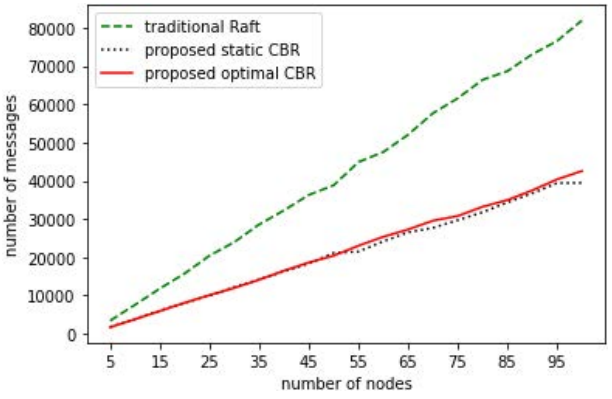

FIGURE 16. Number of AppendEntries RPC messages according to the number of nodes in the proposed CBR-based blockchain system.

the cell size shown in Table 5. The green dotted line in Figure 16 is the same as the line in Figure 11 and refers to the AppendEntries RPC messages measured by the traditional Raft algorithm. The red line in Figure 16 refers to the number of AppendEntries RPC messages when consensus is achieved with the optimized CBR algorithm through

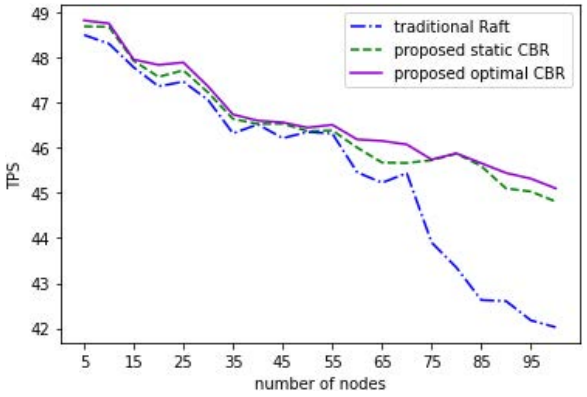

FIGURE 17. TPS according to the number of nodes in the proposed CBR-based blockchain system.

Federated Learning, and the black dotted line refers to a value obtained by fixing the cell size to 40 and measuring the number of messages regardless of the number of nodes. The reason why the cell size is fixed at 40 is that it is the cell size value that maintains the highest TPS in the correlation graph between the fail message and the TPS. For both Raft algorithms, as the number of nodes increases, the number of AppendEntries RPC messages increases. The Figure 16 can be seen that there is little difference in the number of messages between the proposed static CBR and the proposed optical CBR. Basically, when the number of nodes is the same, if the cell size is large, the number of messages exchanged will inevitably decrease. For example, when the number of nodes is 5, the optimized cell size is 80 larger than the CBR fixed at 40. However, the difference in the actual number of messages is not double. The reason is that as large as that size, an empty log occurs in storing the log, or the order of the log changes, so a large amount of fail messages are also generated. Finally, we measure how much the number of messages burdening the blockchain network overhead has decreased according to the number of nodes, showing that on average it has decreased by 45.03%.

The blue dotted line shown in Figure 17 is the same as the dotted line in Figure 12, and it refers to the TPS measured by the traditional Raft algorithm. The purple line and the green dotted line are the TPS value measured by performing the proposed Raft algorithm. As the number of nodes increases, TPS decreases for both algorithms. In particular, if the blockchain system is out of the resource tolerance range, the TPS decreases rapidly, making the system impossible to process almost logs. In the case of fixing the cell size or obtaining an optimized cell value through federated learning, the TPS value is measured to be higher than the traditional Raft. However, this paper proves the CBR algorithm that relieves the burden of the blockchain system and increases performance by increasing the TPS by 2.4% on average without intervals of rapid decrease. In addition, the optimal CBR is with the cell size obtained through deep learning, and a higher TPS value is obtained than the static CBR. The conclusion proves that even if the number of nodes increases, the number of messages decreases, which reduces the load balancing of the reader and allows more transactions with stable communication. Raft algorithms that collect and process existing schedule logs often perform better than Raft algorithms when comparing TPS or the number of messages. However, CBR proves a significant reduction in the number of messages and that TPS also ensures a more stable system than the traditional Raft. In addition, blockchain systems consisting of lightweight devices that cannot perform both federated learning and consensus algorithms show performance comparable to optimal CBR even when they are forced to perform consensus with a static cell size.

B. SAFETY EVALUATION

The most important element in the Raft algorithm is that all nodes store the same logs while maintaining a stable sequence of logs. For all nodes to maintain the log sequence, the proposed CBR algorithm includes the following properties:

- Leader Append-Only: All nodes cannot modify the log themselves, and if there is an error, the log is appended only by overwriting.
- Log Matching: this contains elements with the same index, term, and cell, and the logs are the same for all nodes up to the committed cell.
- State Machine Safety: the safety of the log sequence is ensured by separating and coping with the failure time point adding logs and committing them.

In addition, this paper ensures the safety of the system by reducing the load balancing of the leader even if the number of nodes increases.

VI. CONCLUSION

Based on various resources in the social sector, many companies are choosing blockchain systems for the purpose of introducing smart services to help human activities more broadly in fields such as health. Since the Raft algorithm adopted as the main consensus algorithm in these systems is a process in an environment composed of a small number of servers, it is difficult to apply it to smart services that require the participation of many nodes in various environments.

Therefore, this study first focuses on reducing the number of messages to reduce the burden on the leader. In particular, this paper proposes a method for processing collected transactions by dividing transactions into cells without additional protocols. The proposed CBR consensus algorithm is a protocol to reduce the number of messages without interfering with the concept of the traditional Raft algorithm to maintain stable throughput in the smart data market. In addition, the proposed algorithm obtains a suitable cell size according to the number of nodes through federated learning with accumulated logs. This implies that as a greater number of transactions are processed in the real environment, better

processing performance can be achieved by obtaining a more suitable cell size. Through various simulations, this study demonstrates that the burden of the leader is reduced and the TPS is increased by reducing the number of messages. In particular, the proposed system communicates stably without a rapid decline while maintaining the completeness of the log sequence.

Many methods to reduce the burden on the leader in processing transactions in the Raft algorithm-based blockchain have been studied. If a leader is not properly elected during the leader election period, a re-election occurs, and the TPS decreases. Re-election occurs when there is a network split with two leaders. In future work, we will research how to prevent network splits to further enhance the performance of the Raft algorithm.

REFERENCES

[1] S. Nižetić, P. Šolić, D. López-de-Ipiña González-de-Artaza, and L. Patrono, "Internet of Things (IoT): Opportunities, issues and challenges towards a smart and sustainable future," *J. Cleaner Prod.*, vol. 274, Nov. 2020, Art. no. 122877.

[2] A. Wang, A. Zhang, E. H. W. Chan, W. Shi, X. Zhou, and Z. Liu, "A review of human mobility research based on big data and its implication for smart city development," *ISPRS Int. J. Geo-Inf.*, vol. 10, no. 1, p. 13, Dec. 2020.

[3] M. Yechezkel, A. Weiss, I. Rejwan, E. Shahmoon, S. Ben-Gal, and D. Yamin, "Human mobility and poverty as key drivers of COVID-19 transmission and control," *BMC Public Health*, vol. 21, no. 1, pp. 1–13, Dec. 2021.

[4] P. Voigt and A. Von dem Bussche, "The EU general data protection regulation (GDPR)," in *A Practical Guide*, 1st ed. Cham, Switzerland: Springer, 2017.

[5] Z. Zheng, S. Xie, H.-N. Dai, X. Chen, and H. Wang, "Blockchain challenges and opportunities: A survey," *Int. J. Web Grid Services*, vol. 14, no. 4, pp. 352–375, 2018.

[6] D. Berdik, S. Otoum, N. Schmidt, D. Porter, and Y. Jararweh, "A survey on blockchain for information systems management and security," *Inf. Process. Manage.*, vol. 58, no. 1, Jan. 2021, Art. no. 102397.

[7] W. Fu, X. Wei, and S. Tong, "An improved blockchain consensus algorithm based on raft," *Arabian J. Sci. Eng.*, vol. 46, pp. 8137–8149, Feb. 2021.

[8] B. Bhushan, P. Sinha, K. M. Sagayam, and A. J, "Untangling blockchain technology: A survey on state of the art, security threats, privacy services, applications and future research directions," *Comput. Electr. Eng.*, vol. 90, Mar. 2021, Art. no. 106897.

[9] Y. Li, L. Qiao, and Z. Lv, "An optimized byzantine fault tolerance algorithm for consortium blockchain," *Peer Peer Netw. Appl.*, vol. 14, no. 5, pp. 2826–2839, Sep. 2021.

[10] M. E. Beqqal and M. Azizi, "Taxonomy on IoT technologies for designing smart systems," *Int. J. Interact. Mob. Technol.*, vol. 12, no. 5, pp. 182–191, 2018.

[11] *Bundesminis-Terium für Wirtschaft und Energie (BMWi)*, Die Nationale Wasserstoffstrategie, Berlin, Germany, 2020.

[12] P. K. Sharma and J. H. Park, "Blockchain based hybrid network architecture for the smart city," *Future Gener. Comput. Syst.* vol. 86, pp. 650–655, Sep. 2018.

[13] R. Yang, R. Wakefield, S. Lyu, S. Jayasuriya, F. Han, X. Yi, X. Yang, G. Amarasinghe, and S. Chen, "Public and private blockchain in construction business process and information integration," *Autom. Construct.*, vol. 118, Oct. 2020, Art. no. 103276.

[14] A. Akram and P. Bross, "Trust, privacy and transparency with blockhain technology in logistics," in *Proc. MCIS*, 2018, pp. 1–16.

[15] O. Dib, K. L. Brousmiche, A. Durand, E. Thea, and E. B. Hamida, "Consortium blockchains: Overview, applications and challenges," *Int. J. Adv. Telecommun.*, vol. 11, nos. 1–2, pp. 51–64, 2018.

[16] G. Vizier and V. Gramoli, "ComChain: Bridging the gap between public and consortium blockchains," in *Proc. IEEE Int. Conf. Internet Things (iThings) IEEE Green Comput. Commun. (GreenCom) IEEE Cyber, Phys. Social Comput. (CPSCom) IEEE Smart Data (SmartData)*, Jul. 2018, pp. 1469–1474.

[17] S. Hakak, W. Z. Khan, G. A. Gilkar, M. Imran, and N. Guizani, "Securing smart cities through blockchain technology: Architecture, requirements, and challenges," *IEEE Netw.*, vol. 34, no. 1, pp. 8–14, Jan. 2020.

[18] S. Singh, P. K. Sharma, B. Yoon, M. Shojafar, G. H. Cho, and I.-H. Ra, "Convergence of blockchain and artificial intelligence in IoT network for the sustainable smart city," *Sustain. Cities Soc.*, vol. 63, Dec. 2020, Art. no. 102364.

[19] H. Sukhwani, N. Wang, K. S. Trivedi, and A. Rindos, "Performance modeling of hyperledger fabric (permissioned blockchain network)," in *Proc. IEEE 17th Int. Symp. Netw. Comput. Appl. (NCA)*, Nov. 2018, pp. 1–8.

[20] V. Yodaiken, "Understanding Paxos and other distributed consensus algorithms," 2022, *arXiv:2202.06348*.

[21] D. Ongaro and J. Ousterhout, "In search of an understandable consensus algorithm," in *Proc. USENIX Annu. Tech. Conf. (Usenix ATC)*, 2014, pp. 305–319.

[22] T. S. L. Nguyen, G. Jourjon, M. Potop-Butucaru, and K. L. Thai, "Impact of network delays on hyperledger fabric," in *Proc. IEEE INFOCOM Conf. Comput. Commun. Workshops (INFOCOM WKSHPS)*, Apr. 2019, pp. 222–227.

[23] H. Howard and R. Mortier, "Paxos vs raft: Have we reached consensus on distributed consensus?" in *Proc. 7thWorkshop Princ. Pract. Consistency Distrib. Data*, Apr. 2020, pp. 1–9.

[24] A. Alexandridis, G. Al-Sumaidaee, R. Alkhudary, and Z. Zilic, "Making case for using RAFT in healthcare through hyperledger fabric," in *Proc. IEEE Int. Conf. Big Data (Big Data)*, Dec. 2021, pp. 2185–2191.

[25] D. Wang, N. Tai, and Y. An, "Byzantine fault tolerant raft," Dept. Comput. Sci., Stanford Univ., Stanford, CA, USA, 2018. [Online]. Available: http://www.scs.stanford.edu/17au-cs244b/labs/projects/wang_tai_an.pdf

[26] J. Hu and K. Liu, "Raft consensus mechanism and the applications," *J. Phys., Conf. Ser.*, vol. 1544, no. 1, May 2020, Art. no. 012079.

[27] A. Barger, Y. Manevich, H. Meir, and Y. Tock, "A byzantine fault-tolerant consensus library for hyperledger fabric," in *Proc. IEEE Int. Conf. Blockchain Cryptocurrency (ICBC)*, May 2021, pp. 1–9.

[28] A. Barger, Y. Manevich, H. Meir, and Y. Tock, "A byzantine fault-tolerant consensus library for hyperledger fabric," in *Proc. IEEE Int. Conf. Blockchain Cryptocurrency (ICBC)*, May 2021, pp. 1–9.

[29] D. Huang, Q. Liu, Q. Cui, Z. Fang, X. Ma, F. Xu, and X. Tang, "TiDB: A raft-based HTAP database," *Proc. VLDB Endowment*, vol. 13, no. 12, pp. 3072–3084, Aug. 2020.

[30] R. Wang, L. Zhang, Q. Xu, and H. Zhou, "K-bucket based raft-like consensus algorithm for permissioned blockchain," in *Proc. IEEE 25th Int. Conf. Parallel Distrib. Syst. (ICPADS)*, Dec. 2019, pp. 996–999.

[31] H. Howard, M. Schwarzkopf, A. Madhavapeddy, and J. Crowcroft, "Raft refloated: Do we have consensus?" *ACM SIGOPS Operating Syst. Rev.*, vol. 49, no. 1, pp. 12–21, Jan. 2015.

[32] W. Fu, X. Wei, and S. Tong, "An improved blockchain consensus algorithm based on raft," *Arabian J. Sci. Eng.*, vol. 46, pp. 8137–8149, Feb. 2021.

[33] Y. Wang, S. Li, L. Xu, and L. Xu, "Improved raft consensus algorithm in high real-time and highly adversarial environment," in *Proc. Int. Conf. Web Inf. Syst. Appl.* Cham, Switzerland: Springer, Sep. 2021, pp. 718–726.

[34] S. Pahlajani, A. Kshirsagar, and V. Pachghare, "Survey on private blockchain consensus algorithms," in *Proc. 1st Int. Conf. Innov. Inf. Commun. Technol. (ICIICT)*, Apr. 2019, pp. 1–6.

[35] Y. Chen, A. Huang, Z. Wang, I. Antonoglou, J. Schrittwieser, D. Silver, and N. de Freitas, "Bayesian optimization in AlphaGo," 2018, *arXiv:1812.06855*.

[36] A. Kamilaris and F. X. Prenafeta-Boldú, "Deep learning in agriculture: A survey," *Comput. Electron. Agricult.*, vol. 147, pp. 70–90, Aug. 2018.

[37] S. Pouyanfar, S. Sadiq, Y. Yan, H. Tian, Y. Tao, M. P. Reyes, and S. S. Iyengar, "A survey on deep learning: Algorithms, techniques, and applications," *ACM Comput. Surv.*, vol. 51, no. 5, pp. 1–36, Sep. 2018.

[38] D. C. Nguyen, M. Ding, P. N. Pathirana, A. Seneviratne, J. Li, and H. V. Poor, "Federated learning for Internet of Things: A comprehensive survey," *IEEE Commun. Surveys Tuts.*, vol. 23, no. 3, pp. 1622–1658, 3rd Quart., 2021.

[39] Q. Yang, Y. Liu, T. Chen, and Y. Tong, "Federated machine learning: Concept and applications," *ACM Trans. Intell. Syst. Technol. (TIST)*, vol. 10, no. 2, pp. 1–19, 2019.

[40] Q. Zhang, L. T. Yang, Z. Chen, and P. Li, "A survey on deep learning for big data," *Inf. Fusion*, vol. 42, pp. 146–157, Jul. 2018.

[41] S. Dong, P. Wang, and K. Abbas, "A survey on deep learning and its applications," *Comput. Sci. Rev.*, vol. 40, May 2021, Art. no. 100379.

DANA YANG received the B.S. degree from the Department of Computer Software, Korean Bible University, in 2013. She is currently pursuing the Ph.D. degree with the Department of Computer Science and Engineering, Ewha Womans University, Seoul, South Korea.

Her research interests include authentication, blockchain, steganography, flying *ad-hoc* network (FANET) security, and D2D network security. She has received the Outstanding Paper Award from the 20th International Conference on Advanced Communication Technology (ICACT), in 2018.

INSHIL DOH received the B.S., M.S., and Ph.D. degrees in computer science and engineering from Ewha Womans University, Seoul, South Korea, in 1993, 1995, and 2007, respectively.

From 1995 to 1998, she has worked at Samsung SDS, South Korea. She was a Research Professor at Ewha Womans University, in 2009 and 2010, and Sungkyunkwan University, in 2011. She is currently an Associate Professor of the Department of Cyber Security, Ewha Womans University. Her research interests include wired and wireless network security, sensor network security, and the IoT network security.

KIJOON CHAE received the B.S. degree in mathematics from Yonsei University, in 1982, the M.S. degree in computer science from Syracuse University, in 1984, and the Ph.D. degree in electrical and computer engineering from North Carolina State University, in 1990. He is currently a Professor with the Department of Computer Science and Engineering, Ewha Womans University, Seoul, South Korea. His research interests include blockchain, security in communication and information networks, network protocol design, and performance evaluation.