applied sciences

MDPI

Article

The Modification of pBFT Algorithm to Increase Network Operations Efficiency in Private Blockchains

Youn-A Min

Department of Applied Software Engineering, Hanyang Cyber University, 220, Wangsimni-ro, Seongdong-gu, Seoul 04763, Korea; yah0612@naver.com

Abstract: The use of blockchain technology is becoming more widespread. Governments have expanded their use of the technology from online polls to business management of smaller local governments while private institutions have increased their services from financial to medical services management. This paper presents the modified pBFT blockchain consensus algorithm for a more efficient data management method in cases of applying blockchains in authorized nodes such as governmental agencies. The network communication cost was minimized while the consensus accuracy was maximized by applying a method of simplifying the request management process and electing the reliability-based consensus node during the pBFT consensus algorithm process. By applying the modified pBFT consensus algorithm, stability and speed of the consensus and verification process among various organizations can be guaranteed as well as application in efficient management and value creation of data.

Keywords: blockchain; consensus algorithm; pBFT (Practical Byzantine Fault Tolerance)

Citation: Min, Y.-A. The Modification of pBFT Algorithm to Increase Network Operations Efficiency in Private Blockchains. *Appl. Sci.* **2021**, *11*, 6313. https://doi.org/10.3390/app11146313

Academic Editor: Gianluca Lax

Received: 30 May 2021
Accepted: 7 July 2021
Published: 8 July 2021

1. Introduction

Blockchain is a decentralized data ledger management-based technology that allows direct transactions between users through the sharing and management of all transactions by all nodes connected to the decentralized network environment [1]. Through a hash function-based encryption and encryption through electronic signatures, the blockchain actively applies security technology and enables data protection and precise management based on decentralization [2].

Comprehensive technology applications are increasing, such as in inventory tracking, origin certification, and financial services, and the global financial consulting enterprise PricewaterhouseCoopers (PwC) has predicted that most global businesses will adopt blockchain, and that blockchain will boost global GDP by USD 1.76 trillion by 2030, setting the economic value of blockchain at USD 7600 by this time [3]. As per the increase in contact-free environments and development of communications, the amount of data being transacted and handled is increasing. In most cases, the data managed by an organization are transacted and handled by a third party or a centralized management form. With the size and range of data created by organizations and websites becoming larger every day, and companies seeking to utilize the value of said data, many are seeking to supplement the data breach dangers associated with the centralized management form and are looking for a new value of data application [4,5].

Blockchain is widely applied not only to government agencies but also to the Industrial Internet of Things (IIoT) and, recently, research in which blocks are arbitrarily compressed without considering latency for an optimized solution of IIoT is being conducted [4].

Since the blockchain is composed of many nodes, there is a possibility that a malicious attack may occur, which may lead to the Byzantine general problem. Various consensus algorithms suggest a way to solve the Byzantine general problem, but most are processed based on a synchronous system. pBFT is one of the blockchain consensus algorithms that

can solve the Byzantine general problem in an asynchronous network and guarantee the finality of the consensus.

The pBFT consensus algorithm has the advantage in that it can normally lead to a consensus even with the malicious behavior of less than 33% of nodes, but since it is premised that communication between all nodes is required for consensus, the network operation cost increases as the number of nodes increases.

Variants of pBFT include Tendermint and Neo, and EdgeTC (electronic toll collection) is an example of using pBFT on an inter-institutional blockchain platform [6]. Tendermint and Neo increase the speed and efficiency by applying the delegated method. Recently, as the application of blockchain by government agencies increases, a modified blockchain consensus algorithm that can increase economic efficiency and availability by considering the characteristics according to the weight of work is needed. In the case of government agencies, since the weight of the client's request is varied, it is necessary to further subdivide and transform the contents of the consensus algorithm such as Tendermint. This paper proposes a modified pBFT (mpBFT) as a method to flexibly process the consensus algorithm in consideration of the weight of tasks in a public blockchain composed of nodes based on trust, such as government agencies, and to efficiently manage network operation costs. The mpBFT proposed in the study of this paper is a trust-based consensus algorithm that can be used efficiently between organizations, and it simplifies the process and guarantees the stable selection of leader nodes. In this study, when a client requests a state change, a state flag that distinguishes the importance of Msg is requested together, and PreCommit is performed by the minimized leader nodes in the preparation and commit stages. The election process of leader nodes is also made to be voted based on the reliability of the nodes, so that both efficiency and stability are considered. Through the proposal of this paper, it is possible to operate discriminatively with respect to node requests and to manage data between nodes safely and efficiently [7,8].

The paper examines the characteristics of the blockchain consensus algorithm within a network compromised of authorized nodes in the following chapter, while the third chapter discusses the management and modification process of the pBFT consensus algorithm. The fourth chapter compares and analyzes the benefits of the application of modified pBFT, followed by the results assessed regarding the presented algorithm in the final chapter.

2. Blockchain Consensus Algorithm

As a method of sharing data between nodes existing in the network, blockchain applies an algorithm for the nodes in a network to reach a consensus to a single result for an accurate and transparent management of data [1].

Per the composition range of the nodes, blockchains can be divided into public and private blockchains [1,9]. In a public blockchain, anyone can participate in the network, and all the nodes go through consensus and verification to document the data to the block [8].

If a multitude of malicious nodes participate in the network in a public blockchain, a Byzantine fault may occur in which at least 50% of the participating nodes have reached a consensus [8]. This may lead to forgery and alteration of data. In a consensus to create a block, certain nodes are given rewards, which may lead to an extensive mining by various nodes to create a bifurcation of blocks, resulting in finality. This would increase the calculation costs of the network, and the consensus would need to be rewarded [10,11].

The representative consensus algorithm of public blockchains would be Proof of Work (PoW) and Proof of Stake (PoS) [12].

2.1. Consensus Algorithm of Public Blockchains

2.1.1. PoW (Proof of Work)

PoW is the consensus algorithm most widely used in public blockchains, in which the miner who finds the nonce of random numbers can create a block. If multiple people find the number simultaneously, multiple blocks will form, and the longest blockchain will be chosen through a fork [12].

2.1.2. PoS (Proof of Stake)

PoS is a method that sets priority of block creation depending on the share a node has. A node with a higher share will have a lower computation difficulty for mining and will have the upper hand in creating a block. PoS has a lower computing cost compared to PoW, but it still has issues regarding competitive computing and bifurcation from malicious nodes [13]. PoS can apply sharding technology. Sharding is a technology that presupposes data partitioning. The sharding technology divides the entire network, stores transactions by area, processes them in parallel, and expands them to the blockchain. Data are stored and processed by dividing the data into units called shards [14].

Through sharding, an on-chain solution that improves performance by changing the protocol of the main chain itself for blockchain scalability is possible. The sharding technology has the disadvantage that the procedure becomes complicated and slow when transmission between multiple shards occurs. In addition, the PoS consensus algorithm to which sharding technology is applied is a proof-of-stake consensus algorithm that allows all nodes to participate in consensus, and among them, block generation and verification opportunities are given differentially by stake. It is not efficient because there may be a possibility of threat, and nodes with a large stake may cause a nothing at stake problem, resulting in the possibility of a fork in the blockchain [14].

2.2. Consensus Algorithm of Private Blockchains

Only authorized nodes may participate in a network in a private blockchain, and compared to public blockchains, the consensus process is much shorter and does not need rewards in consensus [15]. Because only authorized nodes make up the network, extensive mining competition and computing exhaustion are unnecessary. The representative consensus algorithms are Paxos, Raft, and pBFT [15,16].

2.2.1. Paxos and Raft

Paxos is an algorithm that calculates in order of sequence numbers for a single value consensus among many nodes [17]. Paxos has three roles: proposer, acceptor, and learner. The proposer chooses a new proposal number n and sends a prepare message of n to numerous acceptors [17,18]. Once an acceptor receives a prepare message of a proposal number larger than the one responded to previously, the acceptor accepts by promising not to accept requests of proposal numbers smaller than n and also sends the request with the largest number accepted previously as a response. Once the proposer receives multiple responses from acceptors, the accept message is sent to the acceptors regarding the proposal number n. Once the acceptor accepts the request for n, the request is accepted only if the acceptor had never previously accepted a request larger than n. If the acceptor has responded to a prepare message larger than n, the acceptor may tell the proposer to give up on the request and help for optimization without any effect to the accuracy. However, there is the problem that each agent may become slower and may cause a shutdown and restart error, causing an overlapping or dissipation [17].

Raft is an algorithm that can process only the leader for client requests, and if there is no information about the leader, the request is accepted for random selection among all nodes (Figure 1). It has the status of Follower, Candidate, and Leader, and Raft's algorithm is modified and used for the Leader Group Election proposed in this paper.

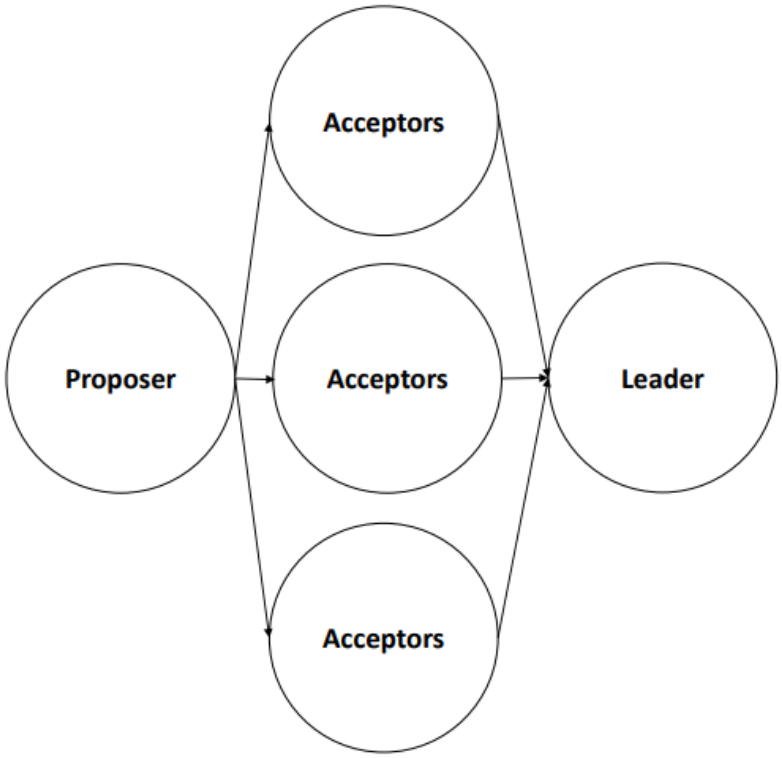

Figure 1. Raft Consensus Algorithm [16,17].

2.2.2. pBFT (Practical Byzantine Fault Tolerance)

Once a Byzantine fault occurs in a public blockchain, a 51% attack occurs. In a private blockchain, however, some of this Byzantine fault is tolerated, as only approved nodes may participate, and if a certain result reaches consensus, consensus is available. This specific case is called BFT (Byzantine Fault Tolerance), in which pBFT is the representative BFT algorithm [17–19]. pBFT is used in a system in which reliability-based nodes are participating and enables an asynchronous BFT consensus. The primary node is the representative node among all the nodes and the rest are called backups.

In a pBFT case in which there are f malicious nodes in a total *N* number of nodes, consensus can be reached if *N* is equal to 3f + 1 [18–20].

The following algorithm is applied to utilize pBFT.

As shown in Table 1, pBFT goes through five steps: request, pre-prepare, prepare, commit, and reply [18]. In the request phase, the client sends a request MM for a service operation to the primary. The primary node generates a sequential number *N* according to the request and sends a message to the backups. The pre-prepare message is structured as <Prepare, V, N, D(M)>, in which V is for view of the message, *N* is the sequential number, and D(M) is the description of the message M. The message is sent to a random node in which the relationships between D(M), V, and *N* are verified, and once the verification shows to be true, the verification message is sent to the rest of the nodes. The prepare message is structured as <Prepare, V, N, D(M), a>, in which a is the number of the verification node. Each node connected to the network collects the pre-prepare and prepare messages and if the message is at least 2f, they prepare the certificate and become a state of "prepared for the request". Requests not reaching consensus are denied [19,21].

Table 1. pBFT Process algorithm [19,20].

Node: Primary Node + Backup
Sends the client's state conversion request—request(M) —to the primary node
Request result cleanup and Pre_Prepare by Primary(Leader) Node
-Generate sequential number for request (N)
-Sending Pre_Prepare message to Backup
<Pre_Prepare, V,N,D(M)>
v: the view to which the message is sent
N: sequential number
D(M): Summary of M
Verification of whether D(M), V,N correspond to each other for the Pre_Prepare message of any backup node a
-if T: Send Prepare to all nodes on the rest of the network
-else: do not accept message
-Prepare< Pre_Prepare, V,N,D(M), a)
a: Node number for which Pre_Prepare was verified
Each node collects Pre_Prepare and Prepare messages
Check if the status of prepared certificate: If the number of collected Pre_Prepare messages is 2f + 1 and Prepare is 2f or more, it becomes the prepared the request node.
A node that satisfies the prepared certificate sends a commit message to all nodes in the network.
-commit<commit, V,N,a)
Each node collects commit message
-commit certificate: node with 2f + 1 commit message
A node that satisfies the prepared certificate and the commit certificate accepts the request: satisfies safety
Request dismissal if not satisfied: Some sacrifices to Liveness

pBFT has a fault tolerance up to 33% and may ensure accuracy of a block creation by broadcasting messages to all nodes multiple times, but this may cause the network connection to slow down if there are too many nodes participating in the network [20,21].

The pBFT may be used in an asynchronous network and goes through two consensus processes as shown in Table 2, which increases network communication for pBFT consensus if the number of participating nodes increase [21,22].

This paper takes the fact that pBFT is comprised of authorized nodes based on reliability into consideration and uses safety, stating that if a consensus is reached among nodes, any node should have the same value once approached, and liveness, that if there is no problem within the block, there should definitely be a consensus within the network, as an index of evaluation to modify the consensus process. In order to satisfy the two conditions for a blockchain consensus, safety and liveness, pBFT considered $N = 3f + 1$ for the conditions, resulting in a 33.3% loss of liveness and If the number of nodes increases in a pBFT consensus algorithm, the network connection costs also increases, resulting in a burden of expenses [21,22].

3. Modification of the pBFT Algorithm

Since pBFT is a BFT consensus algorithm, final agreement is possible by consensus of nodes corresponding to two thirds of all nodes. The consensus of pBFT requires duplicate verification and confirmation by all nodes, which increases network operation cost and causes duplicate processing.

In this paper, we propose mpBFT in order to increase network operation cost, which is a disadvantage of pBFT, and to efficiently operate duplicate verification for all nodes.

In mpBFT, for the purpose of operating an efficient BFT consensus algorithm in an asynchronous network, flexible consensus algorithm selection through status bits for client requests and the consensus process of the leader node group are applied. The difference from the previously announced Tendermint is the application of a flexible algorithm and the method of selecting a representative node.

First, to apply a flexible algorithm, apply the status bit to the client's request and determine whether the requested content is based on processing or future queries. If the request is for simple transaction processing, the mpBFT algorithm can quickly reach consensus without a duplicate agreement, and verification is applied through leader nodes. If transaction sharing and verification by all nodes such as a query is required, the existing pBFT method and mpBFT should be properly mixed and utilized.

The difference between mpBFT and pBFT is that leader node groups can be selectively used in response to client requests. Considering that it is a network in which only trusted nodes participate, verification of the leader node group is possible in order to reduce the duplicate verification process. In case of failure, it is necessary to reply to the client that agreement is not possible. When selecting a leader node, a node weight is assigned based on the node's reliability, frequency of use, and previous history. For the leader node election, the method of RAFT, the existing consensus algorithm, is modified and applied.

3.1. The Performance of the Consensus Algorithm

The number of nodes necessary for pBFT consensus and the calculation cost for the network connection are examined to compare the consensus algorithm performance.

3.1.1. Number of Nodes for Consensus

Regarding the node f that has the failed message error, consensus is reachable if N-f is met. The total number of nodes N and the malicious node f should satisfy (N-f)-f > f. Therefore, pBFT can reach a consensus if $N > 3f$.

3.1.2. The Network Communication Cost within a Consensus Process

While processing pBFT, all nodes are broadcasted to in the prepare phase. Another broadcast is given out to all the nodes in the commit phase. With the total number of nodes N, the network connection cost within these processes can be calculated as $2\ N^2$.

3.1.3. Drawbacks to the pBFT Algorithm Handling

The features of processing the pBFT algorithm discussed above are as follows.

3.2. mpBFT (Modified pBFT)

The mpBFT presented in this paper is designed to be useful in inter-organizational consensus algorithms. By actively using the characteristics of the private network that has reliability-based nodes, the algorithm proposes a simplification of safety and process by assuming that only honest nodes would be present in a blockchain network. Based on reliability among nodes, the number of broadcasts were minimized during the prepare and commit stages in order to manage the network connection cost more efficiently, in which the process is handled based on reliability to have a node elected via vote to be ID_α (pBFT Leader ID). When electing a leader node, an opportunity for prioritization is given in consideration of the frequency of the nodes and the reliability of the institution, and voting should be taken into account when the leader node is re-elected.

The election process of the ID_α and the primary node is shown in Figure 2.

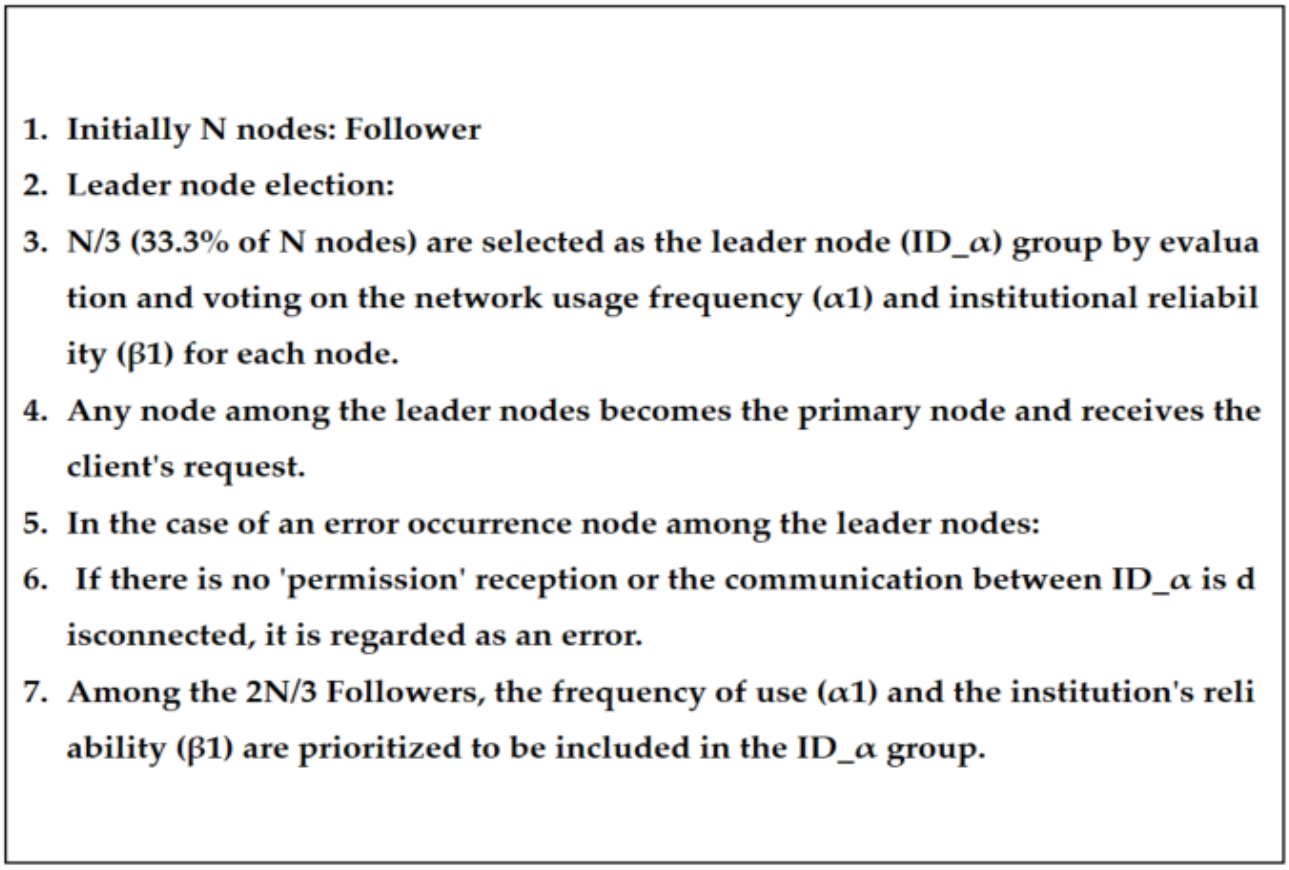

Figure 2. ID_α election algorithm.

The primary node delivers a message to the leader node and, at the same time, communicates to all nodes that consensus is in progress by the leader node. The leader nodes go through the consensus process through two methods depending on the importance of the message. When the importance of the message is high, a consensus process is performed through mutual confirmation between the leader nodes, and the computation cost of $(N/3) * (N/3) + N/3 = N^2/9$ is consumed. In general, it is delivered to the leader node for consensus, and the leader nodes pass through the process of delivering it to the primary node, which consumes the computational cost of $N/3 + N/3 = 2N/3$.

It can be seen that the efficiency in terms of computational cost can be improved, considering that $2N^2$ of computational cost is consumed in the case of conventional pBFT. Schematic of the mpBFT process is as follows.

A leader node is selected at the start of processing. ID_α (PBFT_Leader) is selected through voting and node status and, when an error occurs, the insufficient leader node is re-elected in the form shown in Figure 3.

When Msg arrives, it sends the client's state change request to the start node, which forwards the client's request to the leader nodes and the leader nodes start processing the message. At this time, the content to be transmitted to the leader node is Msg1 (PreStep, V N, D (M)), ID_α) and the ID and Msg summary of the selected leader nodes (ID_α) are transmitted. The leader nodes transmit the contents of Msg2 (V, N, ID_α, ID_αT) to all nodes, along with the ID (ID_α) of the elected node, the reliability of the ID (ID_αT), and the request status flag. The PreCommit processed by the leader nodes is delivered and PreCommit is carried out, and the contents of the PreCommit (Msg (Commit, V, ID_α)) are sent to the start node. In each process, cases with high message importance and cases without messages are selected and processed, and for the PreCommit result, the initiating node transmits the consensus decision to all nodes. The pseudocode that schematically shows the consensus process is shown in Figure 4.

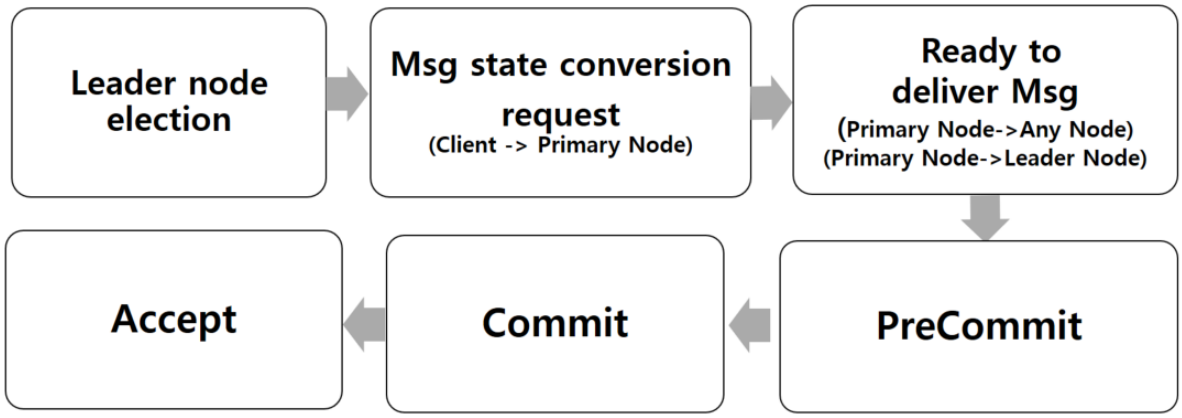

Figure 3. Schematic of the mpBFT processing.

```
Start processing
Step 1: Select Leader Nodes
Step 2: Msg arrival and processing
if (Msg arrives)
State transformation request from client to start node: network communication cost: 1 (constant)
do : Prepare for message delivery
     Generate sequence number N as many as the number of participating nodes
     Start PreStep Process for Msg Consensus with Leader Nodes
     Delivers consensus of leader nodes to all nodes
     Network communication cost: N-1 + N = 2N

Step 3: PreCommit
do : PreCommit process of leader nodes:
     Send Msg (Commit, V, ID_α) to the leader node
     case 1:Msg of high importance
               do : After mutual review among all leader nodes,
                    each leader node forwards PreCommit to the initiating node
                     Network communication cos: (N / 3) * (N / 3) + N / 3 = N2 / 9
     Case 2:Msg of low high importance
               do: Passes whether or not to commit to the starting node after
                     reviewing each leader node
                    Network communication cost: 2N / 3 + ε (additional overload)

Step 4: Commit and Accept
Same as case 1,2:
do : Delivers the PreCommit result of the leader node to all nodes
     MSg (Commit, V, N, D (M), ID_α)
     Network communication cost: N
    Accept: Send request result to customer
    Network communication cost: 1 (constant)
```

Figure 4. mpBFT process.

A blockchain network consisting of trust-based nodes (Private Network) and Process Singularity is modified and processed by the existing pBFT processing process.

Requirements of the elected node is $N/3$; nodes are selected in consideration of the node's frequency of use and the node's reliability variable among all nodes.

All nodes share the ID of the elected node and have the qualification of Followers who can become elected nodes in the future. A start node is any node among the elected nodes. mpBFT processing is available through the process shown in Figures 3 and 4, and the

Byzantine fault tolerance ratio and the network communications according to an increase in the number of nodes can be expressed more efficiently.

3.3. Performance Evaluation of mpBFT

Applying mpBFT can give the calculation costs as shown in Equation (1). The consensus node ratio had sacrificed 33% liveness in mpBFT, assuming malicious nodes, but this research only assumes honest nodes within the system. However, faults caused within the node would be taken into consideration.

3.3.1. The Superiority of the Consensus Node Ratio

Equation (1): consensus node ratio.
Total number of nodes: N, Error Node: f

$$2/N = 3\mathrm{f} + 1 \tag{1}$$

Fault Tolerance $\geq 16.66\ldots$ %
As shown in Table 2, while pBFT sacrifices 33.3% of nodes, the revised new pBFT would only sacrifice 16.7%.

Table 2. Fault Tolerance Comparison.

Node	pBFT	mpBFT Case1	mpBFT Case2
4	1.452	0.7981	0.7981
5	1.783	0.8991	0.8991
6	1.902	0.9919	0.9919
7	2.411	1.1891	1.1891
8	2.684	1.4523	1.4523

3.3.2. NCC: Network Communication Cost

The network communication cost can be written as Equation (2) based on the node communication process in the mpBFT process.
Equation (2): network communication cost.
γ = computational cost for frequency of use ($\alpha 1$) and the institution's reliability ($\beta 1$)

$$\mathrm{NCC} - \mathrm{case1}: \left(\frac{N}{3}\right) \times \left(\frac{n}{3}\right) + \gamma \tag{2}$$

$$\mathrm{NCC} - \mathrm{case2}: \left(\frac{N}{3}\right) + \left(\frac{N}{3}\right) + \gamma \tag{3}$$

While pBFT requires $2N^2$, mpBFT NCC(NCC–case1, NCC–case2) requires $\frac{N^2}{3} + \gamma$ and $\frac{2N}{3} + \gamma$. γ is computational cost for frequency of use ($\alpha 1$) and the institution's reliability ($\beta 1$).
As shown in Table 3, as the number of nodes increases, a larger difference in network communication costs can be seen.

Table 3. Network Communication Cost Comparison.

Node	pBFT	mpBFT Case1	mpBFT Case2
1	2.1	0.3	1.2
2	7.9	0.9	2.1
3	18.1	2.5	3.2
4	32.1	2.9	4.5
5	51.9	3.7	5.8

3.3.3. TPS (Transaction Per Second)

The TPS (transaction per second) of mpBFT was measured in the research. The results through the use of JMeter are shown in Table 4.

Table 4. TPS Comparison.

Node	pBFT	mpBFT Case1	mpBFT Case2
1	14,210	13,592	13,820
2	10,012	11,010	12,200
3	11,201	11,193	11,900
4	10,009	11,812	11,009
5	6253	10,901	9899

It can be seen that the difference in the number of transactions processed per second increases as the number of nodes increases.

Through the performance evaluation Table 4, there is an evident decrease in the number of fault tolerance nodes and network costs, while there is an increase in TPS.

4. Conclusions

With the increase in data usage through a contact-free society and the development of communications, blockchain technology applications are becoming more widespread to manage the mass of data accurately and transparently. Since pBFT, where BFT consensus is possible, requires redundant consensus and verification processes by all nodes, the network operation cost for consensus increases as the number of nodes increases. In this paper, the purpose of the request is identified through the status bit of the client's request, and the mpBFT algorithm is applied flexibly. In addition, the mpBFT algorithm selects a leader node group to speed up the consensus process, and when consensus fails, a new leader node group is selected to give an opportunity for consensus once again. The selection algorithm for the new leader node group election and leader node failure was explained through the paper, and the RAFT algorithm was modified and applied.

The paper presents the mBFT blockchain consensus algorithm for a safer and quicker method of managing internodal data. Through mpBFT, the request process is simplified, and the network communications cost is minimized through the application of reliability-based consensus nodes. While mpBFT requires 33.3% of node sacrifice, mpBFT is shown to only need 16.7%. In terms of network costs, pBFT costs $2N^2$ while mpBFT requires $\frac{N^2}{3} + \gamma$ and $\frac{2N}{3} + \gamma$. γ is computational cost for frequency of use ($\alpha 1$) and the institution's reliability ($\beta 1$). mpBFT shows to also have superior TPS compared to pBFT.

In order to further solidify the research content proposed in this paper, more in-depth studies on the number of nodes and accurate evaluation of institutional reliability are needed.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Bitcoin: A Peer-to-Peer Electronic Cash System. Available online: http://nakamotoinstitute.org/static/docs/bitcoin.pdf (accessed on 24 January 2020).
2. Košťál, K.; Helebrandt, P.; Belluš, M.; Ries, M.; Kotuliak, I. Management and Monitoring of IoT Devices Using Blockchain. *Sensors* **2019**, *19*, 856. [CrossRef] [PubMed]

3. Plato. Available online: https://zephyrnet.com/ko/pwc-%EB%B3%B4%EA%B3%A0%EC%84%9C-%EB%B8%94%EB%A1%9D-%EC%B2%B4%EC%9D%B8,-1-%EB%85%84%EA%B9%8C%EC%A7%80-%EA%B8%80%EB%A1%9C%EB%B2%8C-GDP-76-%EC%A1%B0-2030-%EC%A1%B0-%EC%A6%9D%EA%B0%80/ (accessed on 20 March 2021).
4. Jiang, S.; Cao, J.; Wu, H.; Yang, Y. Fairness-based Packing of Industrial IoT Data in Permissioned Blockchains. *IEEE Trans. Ind. Inform.* **2020**, 1–2. [CrossRef]
5. IEEE. *2144.1-2020—IEEE Standard for Framework of Blockchain-based Internet of Things (IoT) Data Management*; Institute of Electrical and Electronics Engineers Incorporated Report; IEEE: Piscataway, NJ, USA, 2020; ISBN 978-1-5044-7255-5.
6. Chiu, W.-Y.; Meng, W. EdgeTC—A PBFT blockchain-based ETC scheme for smart cities. *Peer-to-Peer Netw. Appl.* **2021**, 1–13. [CrossRef]
7. BFT Consensus. Available online: https://docs.google.com/presentation/d/10W7gKEvk_6XRIlSdiKwnwP9gVzo5Re5m_24QzLGaqvk/edit (accessed on 25 January 2020).
8. Yi, Z. Distributed Energy Intelligent Transaction Model and Credit Risk Management Based on Energy Blockchain. *J. Inf. Sci. Eng.* **2021**, *37*, 55–66. [CrossRef]
9. Abraham, I.; Gueta, G.G.; Malkhi, D.; Yin, M.; Reiter, M.K. HotStuff: BFT consensus in the lens of blockchain. *arXiv* **2019**, arXiv:1803.05069.
10. Amir, Y.; Coan, B.; Kirsch, J.; Lane, J. Prime: Byzantine Replication under Attack. *IEEE Trans. Dependable Secur. Comput.* **2010**, *8*, 564–577. [CrossRef]
11. Fischer, M.J.; Lynch, N.A.; Paterson, M.S. Impossibility of distributed consensus with one faulty process. *J. ACM* **1985**, *32*, 374–382. [CrossRef]
12. Reddy, B.; Sharma, G.V.V. Scalable Consensus Protocols for PoW based Blockchain and blockDAG. In *Computer Science—Distributed, Parallel, and Cluster Computing*; 2020. Available online: http://arxiv.org/abs/2010.05447 (accessed on 8 July 2021).
13. Liu, D.; Alahmadi, A.; Ni, J.; Lin, X.; Shen, X. Anonymous Reputation System for IIoT-Enabled Retail Marketing Atop PoS Blockchain. *IEEE Trans. Ind. Inform.* **2019**, *15*, 3527–3537. [CrossRef]
14. Paper Trail. Available online: https://cosmos.network/resources/whitepaper (accessed on 30 May 2021).
15. Gueta, G.G.; Abraham, I.; Grossman, S.; Malkhi, D.; Pinkas, B.; Reiter, M.; Seredinschi, D.-A.; Tamir, O.; Tomescu, A. SBFT: A Scalable and Decentralized Trust Infrastructure. In Proceedings of the 2019 49th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN), Portland, OR, USA, 24–27 June 2019; pp. 568–580.
16. Li, B.; Jiang, J. Security Analysis of Paxos Mechanism Design Based on Game Theory. In Proceedings of the 2020 IEEE International Conference on Information Technology, Big Data and Artificial Intelligence (ICIBA), Chongqing, China, 6–8 November 2020; pp. 59–66. [CrossRef]
17. Kotla, R.; Alvisi, L.; Dahlin, M.; Clement, A.; Wong, E. ZYZZYVA: Speculative Byzantine fault tolerance. *ACM Trans. Comput. Syst.* **2010**, *27*, 1–39. [CrossRef]
18. Huang, D.; Ma, X.; Zhang, S. Performance Analysis of the Raft Consensus Algorithm for Private Blockchains. *IEEE Trans. Syst. Man Cybern. Syst.* **2019**, *50*, 172–181. [CrossRef]
19. Castro, M.; Liskov, B. Practical Byzantine Fault Tolerance and Proactive Recovery. *ACM Trans. Comput. Syst.* **2002**, *20*, 398–461. [CrossRef]
20. Veronese, G.S.; Correia, M.; Bessani, A.N.; Lung, L.C. Spin One's Wheels? Byzantine Fault Tolerance with a Spinning Primary. In Proceedings of the 2009 28th IEEE International Symposium on Reliable Distributed Systems, Niagara Falls, NY, USA, 27–30 September 2009; pp. 135–144.
21. Paper Trail. Available online: https://www.the-paper-trail.org/post/2008-08-13-a-brief-tour-of-flp-impossibility/ (accessed on 16 March 2021).
22. Yim, J.C.; Yoo, H.K.; Kwak, J.Y.; Kim, S.M. Blockchain and Consensus Algorithm. *Electron. Telecommun. Trends* **2018**, *33*, 45–56.