Enhancing Byzantine Fault Tolerance in Blockchain Networks through Dynamic Clustering

Teppei Okada*, Noriaki Kamiyama† and Akihiro Fujihara‡
*Graduate School of Information and Engineering, Ritsumeikan University, Osaka 567-8570, Japan
†College of Information and Engineering, Ritsumeikan University, Osaka 567-8570, Japan
‡Faculty of Engineering, Chiba Institute of Technology, Chiba 275-0016, Japan
Email: is0498ex@ed.ritsumei.ac.jp, kamiaki@fc.ritsumei.ac.jp, akihiro.fujihara@p.chibakoudai.jp

***Abstract*—In recent years, blockchain technology, which enables transactions to be distributed across multiple computers and managed in an immutable and secure manner, has garnered significant attention. Within blockchain networks, consensus mechanisms ensure the consistent sharing of ledger information when new blocks are added. In consortium blockchains, typically employed by a limited number of organizations, the Practical Byzantine Fault Tolerance (PBFT) protocol is widely used. PBFT is designed to tolerate Byzantine nodes—nodes that may be compromised or malfunctioning—by achieving consensus as long as fewer than one-third of the total nodes are Byzantine. However, PBFT relies on the assumption that at least two-thirds of the nodes behave correctly, making consensus challenging when the number of malicious nodes exceeds this threshold. Previous research has explored the use of clustering to enhance throughput, but these methods are static and unsuited to dynamic environments. Moreover, clustering techniques aimed at bolstering Byzantine resistance remain underexplored. This paper presents a novel method for constructing clusters within a blockchain network to resist Byzantine nodes. By employing clustering, we estimate the locations of potential attackers, thereby enhancing the system's resilience to Byzantine faults.**

I. Introduction

Blockchain is a technology that allows for the secure, tamper-resistant sharing and management of transactions across multiple computers using a distributed ledger. There are three primary types of blockchain: public, private, and consortium. Public blockchains offer high transparency without a central administrator, but their main drawback is that transaction processing and approval times can increase significantly as the number of participants and transaction volume grows. In contrast, private blockchains restrict participation to a single administrator, allowing for faster approval times and limited data disclosure. However, this comes at the expense of decentralization, as authority is concentrated in the hands of the administrator.

The consortium type falls between the public and private blockchains, with multiple administrators sharing control. While processing times are slower compared to private blockchains, the decentralized distribution of authority makes consortium blockchains more resistant to data tampering.

Additionally, blockchains use consensus mechanisms where new blocks or transactions are verified by all participants upon creation, allowing for the detection and elimination of potentially falsifiable transactions. In consortium blockchains, the Practical Byzantine Fault Tolerance (PBFT) mechanism [1] is commonly employed.

In PBFT, consensus can be correctly achieved as long as the number of Byzantine nodes is less than one-third of the total number of nodes. Conversely, PBFT requires that more than two-thirds of the nodes are functioning correctly, which complicates the agreement process if the number of attackers exceeds a certain threshold, as illustrated in Figure 1.

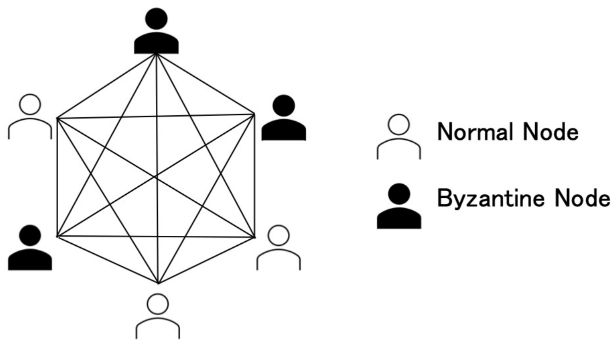

Fig. 1: PBFT

Thus, consensus in PBFT is influenced by the presence of Byzantine nodes. This paper proposes a method to enhance resilience against Byzantine attacks by partitioning the topology through clustering, estimating the locations of attackers, and concentrating them within specific clusters. While the proposed method aims to achieve correct consensus, there is a concern that this may lead to increased communication traffic. Therefore, we compare the probability of consensus formation with the amount of communication traffic generated when employing the proposed method.

The main contributions of this study are as follows:

- In PBFT, which requires more than two-thirds of participants to be normal nodes for consensus to be achieved, the proposed method enables correct consensus even when fewer than two-thirds of the nodes are functioning normally.
- We conduct a theoretical analysis of the upper limit of consensus probability, demonstrating that the simulation results align with the theoretical values.
- We also analyze the upper limit of communication traffic volume theoretically, deriving analytical results that elucidate the relationship between the number of clusters and traffic volume.

In Section II, we discuss related works, while the proposed method is outlined in Section III. Section IV presents the performance evaluation, and we conclude with a summary in Section V.

II. RELATED WORK

PBFT [1] consists of a client, replica nodes, and a primary node. Clients generate transactions and send them to the network, while replica nodes represent the nodes within that network. The primary node, designated from among the replica nodes, receives transactions from clients and forwards them to the other replica nodes. To achieve precise agreement, consensus is reached through three phases—pre-prepare, prepare, and commit—following the client's submission of the transaction to the primary node, as illustrated in Figure 2.

- **pre-prepare phase**:
  The primary node forwards the transaction to the other replica nodes. Each receiving node verifies the validity of the transaction and subsequently broadcasts the verification results to the other nodes.
- **prepare phase**:
  The replica nodes confirm that the verification results match those provided by the primary node and then broadcast their results to the other nodes.
- **commit phase**:
  If a node agrees with the message received during the prepare phase, it sends a commit message to the other nodes.

Finally, when a commit message is received from more than two-thirds of the nodes, a reply message is sent to the client to indicate that consensus has been successfully achieved.

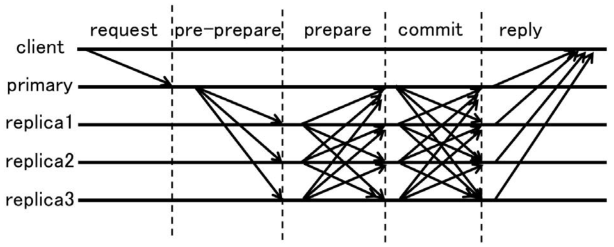

Fig. 2: Consensus in PBFT

In [2], the authors present a theoretical analysis of temporal variations in block communication among validator nodes in Ethereum 2. The study models the process of broadcasting a block from the node that generates it to the other nodes, characterizing this process as a Markov process where the probability of each node receiving the block is considered. For a total of N nodes, the resulting communication delay was confirmed to follow $O(N \log N)$.

In [3], the authors propose RC-PBFT (Random Cluster PBFT) and demonstrate its effectiveness, addressing the significant communication overhead inherent in PBFT due to the extensive interactions between nodes. Their approach involves creating clusters through clustering, performing PBFT within randomly selected clusters, and broadcasting the results to other clusters. This method reduces the time required for consensus and enhances throughput compared to conventional PBFT.

In [4], the authors address the issue of slow transaction verification in Proof-of-Work (PoW) and propose a method to reduce the verification time by utilizing parallel mining both within and between clusters.

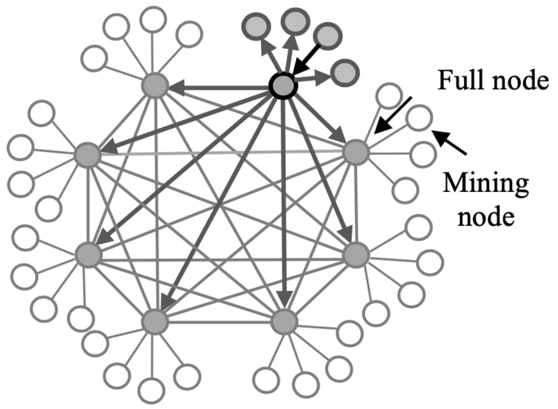

Fig. 3: Parallel Mining

As shown in Figure 3, the network comprises mining nodes that perform mining and full nodes that distribute blocks to other clusters. Mining begins simultaneously across all clusters, and the mining node that discovers a block sends it to the full node, which then forwards it to the mining nodes within its own cluster as well as to other full nodes. All full nodes that receive the block broadcast it to their respective mining nodes for verification. The experimental results indicate that this approach reduces consensus time and enhances the scalability of the network.

In [5], the authors express concern that while a distributed network structure is anticipated to enhance scalability and eliminate single points of failure in networks comprising multiple robots, PBFT requires extensive communication to achieve consensus. To address this, they propose a method that groups robot nodes into clusters using the k-means algorithm, aiming to reduce consensus delays and energy consumption between groups. Additionally, the behavior of nodes in each cluster during the consensus process is monitored and scored, with the node having the highest score randomly selected as the primary node to enhance the reliability of the primary node.

In [6], the authors provide a systematic and comprehensive review of blockchain sharding techniques, identifying the key elements and challenges associated with sharding. Sharding is a technique that divides nodes into multiple groups (shards) to enhance the scalability of a blockchain network, enabling each shard to process transactions independently and thereby increasing the overall processing capacity. This study analyzes in detail the essential components of sharding implementation and the challenges that may arise.

In [7], the study employs network coding, a technique for efficient data transmission, in PBFT. The proposed method aims to enhance scalability by reducing the maximum bandwidth requirement, demonstrating improved communication efficiency and scalability compared to conventional methods.

In [8], a solution is presented for enhancing the scalability of blockchains through a three-layer architecture. This approach aims to increase overall efficiency by utilizing sharding and distributing processing tasks across different layers. Meanwhile, [9] categorizes existing sharding schemes based on blockchain type and sharding technology, analyzing their respective advantages and disadvantages. The study also establishes criteria for the applicability of various sharding techniques.

III. Proposed Method

A. Overview

Following is the flow of the proposed method, as illustrated in Figure 4.

1) Conduct PBFT across the entire network.
2) If consensus cannot be reached, apply the k-means method to partition the network.
3) Conduct PBFT within each cluster.
4) Perform PBFT between clusters based on the results obtained within each cluster.
5) If consensus is not achieved, it is determined that there are many attackers in the cluster that sent the minority opinion, leading to the merging of the minority clusters.
6) If consensus is successfully formed, the process is terminated.

Thus, the proposed method can estimate clusters with a high concentration of attackers through clustering and merge them to reduce the overall proportion of attacker clusters. Consequently, the percentage of consensus reached also increases. Additionally, the clustering using the k-means method and the merging process must be executed autonomously by the nodes, with results shared among them. Therefore, there is a phase dedicated to broadcasting geographical and other relevant information.

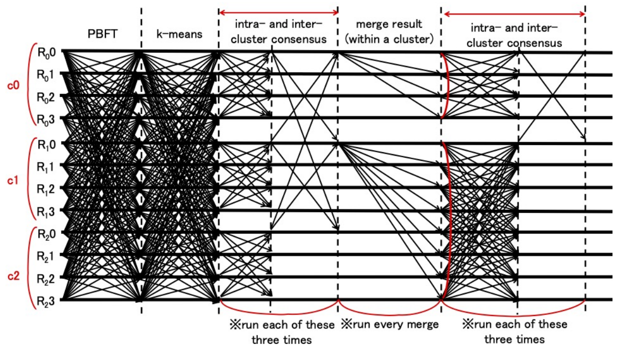
PBFT
k-means
intra- and inter-cluster consensus
merge result (within a cluster)
intra- and inter-cluster consensus
c0
c1
c2
$R_0$0
$R_0$1
$R_0$2
$R_0$3
$R_1$0
$R_1$1
$R_1$2
$R_1$3
$R_2$0
$R_2$1
$R_2$2
$R_2$3
※run each of these three times
※run every merge
※run each of these three times

Fig. 4: Proposed Method

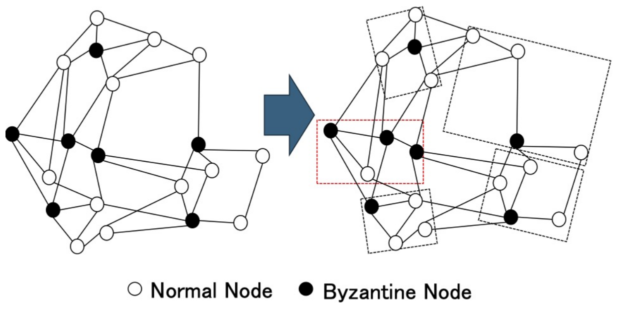

Fig. 5: Clustering

The specific flow is illustrated in Figure 5. The circles represent the nodes in the blockchain network. In this figure, 6 out of 20 nodes are Byzantine nodes. Since Byzantine nodes comprise more than one-third of the total, it is impossible to achieve consensus even if PBFT is executed in this scenario. Consequently, clustering is performed, resulting in one cluster containing more than one-third of Byzantine nodes, as indicated by the red dotted line. In this case, when PBFT is conducted within each cluster and between clusters, consensus is reached in four out of the five clusters, indicating that consensus can be correctly formed.

On the other hand, as shown in the left part of Figure 6, even when clustering is performed, consensus may not be achievable if two clusters contain the majority of Byzantine nodes, resulting in a situation where more than one-third cannot reach agreement. However, by merging the minority clusters (indicated by the red dotted line), three normal clusters can be formed out of the four clusters, allowing for successful consensus formation.

In the proposed method, during the merging process, clusters with a smaller average number of nodes are combined until the minimum required number of clusters for PBFT, which is four, is achieved. Table I illustrates an example of how the number of nodes within clusters changes, with the average number of nodes per cluster denoted as n and the initial number of clusters represented by m.

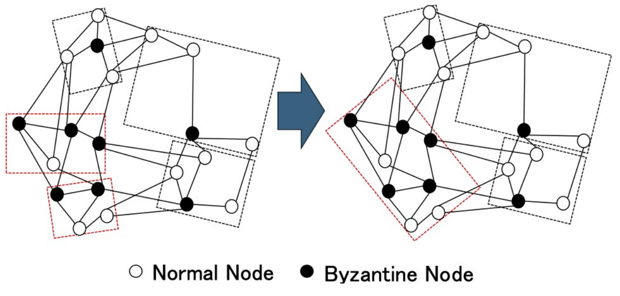

Fig. 6: Merging clusters

No. of Merges	No. of Nodes in Each Cluster	No. of Clusters
0	$n, n, \ldots, n, n$	m
1	$2n, \ldots, n, n$	$m-1$
2	$2n, \ldots, 2n$	$m-2$
...	...	...

TABLE I: Number of Nodes within Clusters

B. Performance Limit

In this section, we present a theoretical analysis of the probability of consensus formation and the upper bounds on communication traffic when the proposed method is applied.

variable name	meaning
N	Total number of nodes
m	Number of clusters
n	Number of nodes in cluster

TABLE II: variable

1) Probability of Consensus Formation: We analyze the performance limits of the probability of consensus formation. Assuming equal divisions, we use the variables defined in Table II to represent the total number of nodes as follows:

$$N = mn \tag{1}$$

The upper bound on the number of Byzantine nodes f in a normal PBFT is given by:

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

Therefore,

$$f = \frac{mn-1}{3} \tag{3}$$

Also, the upper bound on the number of Byzantine nodes f^m in an m-equally divided two-stage PBFT is as follows:

1) From $m = 3f' + 1$, all nodes can be Byzantine nodes in f' clusters. Hence,

$$f' = \frac{m-1}{3} \tag{4}$$

2) For the remaining $(2f'+1)$ clusters, from $n = 3f'' + 1$,

$$f'' = \frac{n-1}{3} \tag{5}$$

nodes can be Byzantine nodes.

Therefore, from equations (4) and (5), we have:

$$\begin{aligned} f^m &= f' \cdot n + (2f'+1) \cdot f'' \\ &= \frac{mn-1}{3} + \frac{2}{9}(m-1)(n-1) \\ &= f + \frac{2}{9}(m-1)(n-1). \end{aligned} \tag{6}$$

This means that the proposed method increases the upper limit of Byzantine nodes by $\frac{2}{9}(m-1)(n-1)$ compared to the conventional method.

Given that it increases by $m \approx n$,

$$\frac{f^m}{N} = \frac{5}{9} - \frac{4}{9} \cdot \frac{1}{n} - \frac{1}{9} \cdot \frac{1}{n^2} \tag{7}$$

As $N \to \infty$,

$$\frac{f^m}{N} \to \frac{5}{9} \tag{8}$$

Consequently, from equation (8), if the number of Byzantine nodes is less than $\frac{5}{9}$ of the total number of nodes, the proposed method may be used to reach a consensus.

2) Amount of Communication Traffic: We also calculate the communication complexity based on [10] for communication traffic. Using the variables in Table II, the amount of traffic associated with location information sharing using the k-means method and consensus building using PBFT is as follows:

$$N \log N + 3(m \log m + n \log n) \tag{9}$$

When the merge is performed once, the number of clusters is $m-1$, with the number of nodes in one cluster being $2n$ and n for the remaining $m-2$ clusters. In this case, the traffic generated will be:

$$3\left\{(2n)\log(2n) + (m-1)\log(m-1)\right\} \tag{10}$$

When the number of merges is $\frac{m}{2}$ times, the number of clusters is $\frac{m}{2}$, and the number of nodes in each cluster is all $2n$. Therefore, the amount of communication at this time is:

$$3\left\{(2n)\log(2n) + \frac{m}{2}\log\frac{m}{2}\right\} \tag{11}$$

Item	Condition
Number of nodes	90
Number of Byzantine nodes	0∼90
Clustering method	k-means
Number of clusters	7, 10, 15
Network topology	Barabasi-Albert (BA) Erdos-Renyi (ER) Watts-Strogatz (WS)

TABLE III: Simulation Conditions

Therefore, from equations (9), (10) and (11),

$$
\begin{aligned}
&N\log N + 3(m\log m + n\log n)\\
&+3\cdot\frac{m}{2}\cdot 2n\log 2n + \sum\nolimits_{i=\frac{m}{2}}^{m-1} i\log i\\
&= nm\log nm + 3n\log n + 3nm\log 2n + \sum\nolimits_{i=\frac{m}{2}}^{m-1} i\log i
\end{aligned} \tag{12}
$$

Here, in (12),

$$
\begin{aligned}
\sum\nolimits_{i=\frac{m}{2}}^{m-1} i\log i &\simeq \int_{\frac{m}{2}}^{m} x\log x\,dx\\
&= O(m^2\log m)
\end{aligned} \tag{13}
$$

Therefore, from (13), as m increases, the upper limit of the communication volume increases by $m^2\log m$.

IV. Performance Evaluation

In this section, we evaluate the effectiveness of the proposed method through computer simulations. Based on the following evaluation conditions, we compare the proportion of consensus achieved and the amount of communication traffic generated. For the probability of consensus, we count the number of times consensus is reached out of 100 executions.

A. Evaluation Conditions

Under the conditions outlined in Table III, we compare the proposed method (Proposed Method $k = 7, 10, 15$) with conventional PBFT (Existing PBFT) and PBFT with the k-means method applied only once (Existing Method $k = 7, 10, 15$).

B. Consensus Probability

Figure 7 illustrates the probability of reaching consensus in the BA, ER, and WS models as the number of attackers varies from 0 to 90. In all models, the proposed method successfully achieved consensus even when the number of attackers exceeded one-third of the total number of nodes. It is evident that the consensus probability was higher in the proposed method compared to the conventional method. This improvement can be attributed to the aggregation of clusters with a high number of attackers, which increased the proportion of clusters that produced correct results. Furthermore,

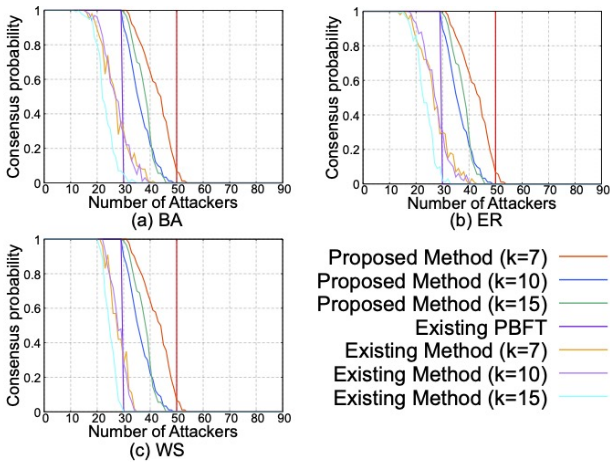

Fig. 7: Consensus Probability

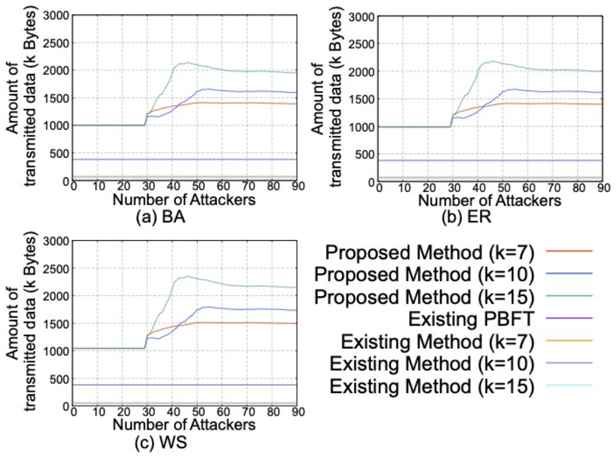

Fig. 8: Amount of Traffic

correct consensus was still attainable even with up to approximately 50 attackers, representing more than 5/9 of the total participants. This outcome aligns with the performance limits derived from the theoretical analysis, confirming that the theoretical values are consistent with the simulation results. However, it is noteworthy that when the number of clusters is small (i.e., $k = 7$), the performance limit is exceeded. This discrepancy arises because the theoretical values derived in Section III-B1 are based on a single instance of clustering, while the simulations involve multiple rounds of clustering and merging, leading to surpassing the expected performance limits.

C. Amount of Traffic

Figure 8 presents the amount of communication traffic. In the proposed method, the traffic volume increased compared to the conventional method due to the additional traffic generated by PBFT after clustering and the necessity to broadcast data among nodes for executing the k-means method. Moreover, when the number of attackers exceeded one-third of the total nodes, the traffic

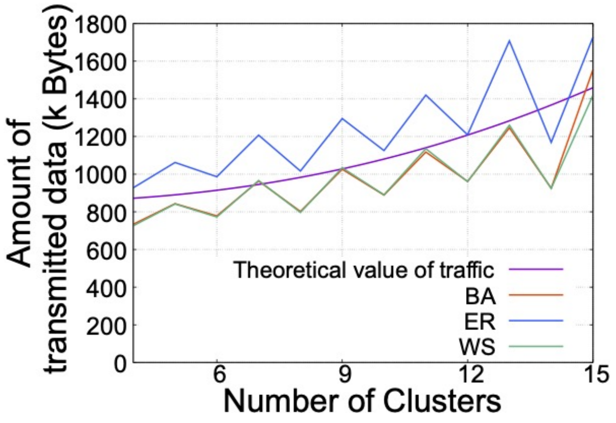

Fig. 9: Traffic Volume against the Number of Clusters

volume surged significantly as a result of clustering, merging, and PBFT traffic within and between clusters. Furthermore, as the number of clusters increased, the frequency of merging also increased if consensus was not reached, leading to a further rise in traffic volume as the number of attackers escalated.

Figure 9 illustrates the amount of traffic for the BA, ER, and WS models, with the number of nodes and attackers set to 90 and 40, respectively. The theoretical value $O(m^2 \log m)$ for the traffic volume, derived in Section III-B, is also plotted for comparison. To observe the trends between the simulation results and theoretical values, we added a constant to the theoretical values. Notably, the lines for the BA and ER models overlap in the graph. Furthermore, the jagged nature of the curves can be attributed to situations where, in cases of an even number of clusters, the number of supporting clusters may equal the number of opposing ones. This can lead to instances where consensus is not achieved, resulting in a lower probability of consensus formation compared to cases with an odd number of clusters.

In all models, the amount of traffic increased with the number of clusters. Additionally, the observed increase in traffic volume aligned closely with the theoretical value, confirming the robustness of the theoretical analysis presented in Section III-B. Since the theoretical values serve as approximations in terms of order of magnitude, the trends observed in both the theoretical and simulation results were found to be consistent.

V. Conclusion

In this paper, we proposed a method to enhance resistance against Byzantine attacks by utilizing clustering and merging techniques. We compared the percentage of consensus formation and the amount of communication traffic between the proposed method and conventional approaches. Through simulation evaluations, we confirmed the following key findings:

- By applying the proposed method, it became possible to reach correct consensus even when more than one-third of the participants were attackers.
- The proposed method enabled correct consensus to be reached even with up to 5/9 of the participants being attackers, which aligns with the performance limit of consensus probability derived through theoretical analysis.
- The amount of traffic increased considerably compared to conventional methods due to the rise in the number of communications and the volume of data exchanged. Additionally, as the number of attackers increased, the interactions with other clusters also escalated, leading to higher traffic.
- As the number of clusters increased, the traffic volume also grew. We derived the theoretical upper limit of the communication overhead generated by the proposed method, and by comparing it to the simulation results—after adding a constant—we confirmed that both exhibit a similar increasing trend.

In the future, we plan to devise a method to reduce the amount of traffic while maintaining the probability of consensus.

Acknowledgement

This work was supported by JSPS KAKENHI Grant Number 23K11086, 23K21664, and 23K28078.

References

[1] M. Castro, et al., "Practical byzantine fault tolerance," in Proc. of third symposium on Operating systems design and implementation (OSDI '99), 1999, pp. 173-186.
[2] A. Fujihara, "Explaining temporal fluctuations of broadcast communications between validator nodes in a proof-of-stake blockchain," in Proc. of the Proceedings of Blockchain Kaigi (BCK23), 2023, pp. 011004-1–011004-11.
[3] R. M. Othmen, et al., "Simulation of Optimized Cluster Based PBFT Blockchain Validation Process," in Proc. of the IEEE Symposium on Computers and Communications (ISCC), 2023, pp. 1317-1322.
[4] A. J. Al-Musharaf, et al., "Improving Blockchain Consensus Mechanism via Network Clusters," in Proc. of the 2021 1st Babylon International Conference on Information Technology and Science (BICITS), 2021, pp. 293-298.
[5] Y. Sun, Y. Fun, "Improved PBFT Algorithm Based on K-means clustering for Emergency Scenario Swarm Robotic Systems," IEEE Access, 2023, pp. 121753-121765.
[6] G. Wang, et al., "SoK: Sharding on Blockchain," in Proc. of the 1st ACM Conference on Advances in Financial Technologies (AFT '19). Association for Computing Machinery, 2019, pp. 41-61.
[7] B. Choi, et al., "Scalable Network-Coded PBFT Consensus Algorithm," in Proc. of the 2019 IEEE International Symposium on Information Theory (ISIT), 2019, pp. 857-861.
[8] J. Xi, et al., "A Comprehensive Survey on Sharding in Blockchains," Mobile Information Systems, 2021.
[9] X. Liu, et al., "A survey on blockchain sharding," ISA Transactions, 2023, pp. 30-43.
[10] A. Fujihara, "Theoretical Analysis on Block Time Distributions in Byzantine Fault-Tolerant Consensus Blockchains," in Proc. of the IEEE International Conference on Blockchain, 2024, pp. 378-385.