A Novel Dynamic Practical Byzantine Fault Tolerance Protocol Based on Node Grouping

Abstract—**The Practical Byzantine Fault Tolerance protocol (PBFT) has been widely deployed in the blockchain network. However, two main issues, the communication complexity and the inability of nodes to join/exit the network without restarting the system, significantly degrade consensus efficiency. To solve these problems, we propose a novel dynamic practical byzantine fault tolerance protocol based on node grouping (NG-PBFT), which is realized by grouping nodes into consensus and observation groups. The nodes in observation group preprocess nodes' joining/exiting requests, while the nodes in consensus group reach consensus about the preprocessed results. Furthermore, an adaptive three-phases or two-phases consensus mechanism that can effectively improve consensus efficiency is proposed. We provide theoretical complexity to show that our protocol reduces the communication complexity of consensus and improves consensus efficiency significantly.**

Index Terms—**PBFT, node grouping, adaptive consensus, dynamic, consortium blockchain**

I. Introduction

Blockchain is a decentralized distributed ledger technology that relies on consensus protocols to ensure data consistency among all peer nodes and to reach an agreement on proposals. The Byzantine Fault Tolerance (BFT) consensus protocols are widely used in blockchain networks to improve consistency and reliability.

In 1999, Castro and Liskov [1] introduced the Practical Byzantine Fault Tolerance (PBFT) protocol, which provides a robust solution to the Byzantine Generals Problem and is the first BFT protocol used in asynchronous networks. Although the PBFT protocol effectively provides a solution for the limitations of the original Proof of Work (POW) algorithm by augmenting throughput and reducing transaction confirmation delay, its vulnerability to malicious attacks, poor scalability, and high communication complexity issues hinder its applicability to blockchain-related projects.

In recent years, many improved algorithms for PBFT have been proposed to reduce the risk of malicious behaviors for nodes. A series of PBFT protocols based on credit mechanisms measuring the honestness have been proposed [2]–[6] to enhance the security and efficiency of the consensus protocol. Wu [3] introduced a node credit mechanism, which was used to supervise and score the behaviors of the nodes to improve security. Lei [4] proposed reputation based BFT for consortium blockchain (RBFT), in which it selected primary nodes from nodes with high reputation values to improve the consensus efficiency, thus reducing the risk of primary nodes being malicious nodes. Wang [5] proposed a credit-delegated BFT protocol to stimulate the enthusiasm of reliable nodes and reduce the participation of abnormal nodes in the consensus process. Tong [6] proposed a peer trust-based practical byzantine consensus protocol that can select the right number of participants from a large-scale blockchain network to execute the traditional PBFT consensus protocol. Although these protocols improve PBFT, they ignore the scalability of the consensus network [1].

Andrew Miller [7] proposed the Honey Badger of BFT Protocol that builds upon the strengths of PBFT to provide increased scalability, robustness, and efficiency while maintaining a high level of security. This protocol consists of n-reliable broadcast (RBC) protocols, whose inputs are the proposals of nodes, followed by n-asynchronous binary agreement (ABA) protocols to make a decision on the value of each proposal [8]. As a result, the protocol has high communication complexity.

Although these protocols improve PBFT, they often ignore the communication complexity of the consensus network. In practical scenarios, a lower communication complexity means faster transaction times. Yin [9] proposed HotStuff, which is a consensus protocol with low communication complexity. However, reliance on the primary node can lead to overload, data delays, and low consensus efficiency. Further, nodes can not dynamically join/exit the blockchain network in HotStuff. To resolve this issues, Duan [10] proposed an efficient dynamic BFT protocol, which relies on configuration discovery sub-protocol to manage requests. However, the communication complexity is also high since additional costs are needed for sub-protocol.

Now, further research is needed to address issues related to the PBFT: (1) Reducing the communication complexity of consensus for PBFT is urgent for practical scenarios. (2) PBFT works in a completely enclosed environment, and nodes in it are not allowed to join/exit the network without restarting the system, which makes it difficult to be applied in the actual system [11]. To solve these issues, we propose a novel dynamic PBFT protocol based on node grouping (NG-PBFT), which provides a strategy for nodes to join/exit the blockchain network dynamically, and reach consensus with low cost. The protocol enables nodes dynamic joining/exiting the blockchain network without restarting the system and changing views by isolating the nodes that processing the joining/exiting requests from the consensus nodes. Furthermore, we adaptively use three-phase or two phase consensus based on the reputation model [4] to implement adaptive consensus, reducing the consensus's communication complexity. Thus, the contributions in this paper are as follows.

- A mechanism for nodes dynamic joining/exiting the blockchain network without restarting the system and changing views frequently is proposed. It is realized by grouping nodes into consensus and observation groups. While the observation group preprocesses requests for nodes joining/exiting the blockchain network, and the consensus group executes the consensus operation uninterruptedly.

- An adaptive three-phase or two-phase consensus is proposed based on reputation value [4] to reduce communication complexity. In some special conditions, such as the consensus nodes with high reputation values, the committing phase can be skipped. Then the traditional three-phase consensus becomes two phase consensus, and the communication complexity goes down distinctly.

II. The proposed NG-PBFT

In this paper, we propose an NG-PBFT consensus protocol, which can reach consensus with three-phase or two phase adaptively to reduce the complexity and make the nodes joining/exiting blockchain network dynamically without view changing and system restarting. As shown in Fig. 1, it is the diagram of the proposed NG-PBFT. Table I shows the definitions of the nations used in this article. In this section, we will introduce the details of the NG-PBFT.

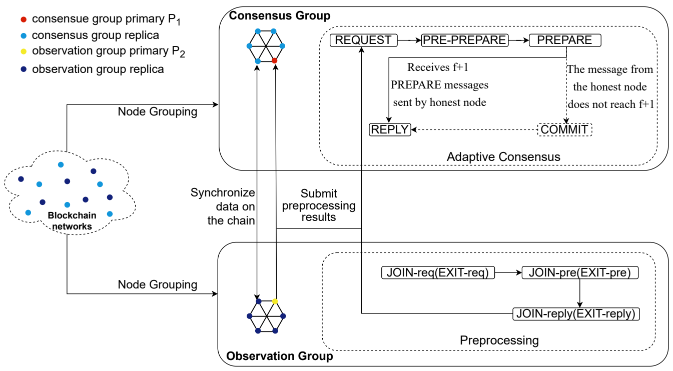
● consensue group primary P_1
● consensus group replica
● observation group primary P_2
● observation group replica
Consensus Group
REQUEST
PRE-PREPARE
PREPARE
Receives f+1 PREPARE messages sent by honest node
The message from the honest node does not reach f+1
REPLY
COMMIT
Adaptive Consensus
Node Grouping
Blockchain networks
Synchronize data on the chain
Submit preprocessing results
Node Grouping
JOIN-req(EXIT-req)
JOIN-pre(EXIT-pre)
JOIN-reply(EXIT-reply)
Preprocessing
Observation Group

Fig. 1. Principal diagram

A. Initialization

In the NG-PBFT consensus protocol, for a blockchain network with n nodes, we randomly group them into consensus and observation groups. The consensus group consists of n_1 nodes with primary node p_1, while the observation group consists of n_2 nodes with primary node p_2. The n_1 and n_2 are as follows:

$$n_1 = \left\lfloor \frac{n}{2} \right\rfloor \tag{1}$$

$$n_2 = n - \left\lfloor \frac{n}{2} \right\rfloor \tag{2}$$

where $n_1 \geq 4, n_2 \geq 4, n_1 + n_2 = n$.

The reputation value (r) is real number with value between 0 and 1 and is used to measure the honestness of nodes. Higher reputation values indicate more honestness. Initially, all nodes are given a reputation value of 0.5, including new nodes, and each node maintains a local node list (NL) including their identity number, IP address/port, public key, reputation value, and trusted state. After each round consensus, all nodes update their reputation values using the reputation model in [4].

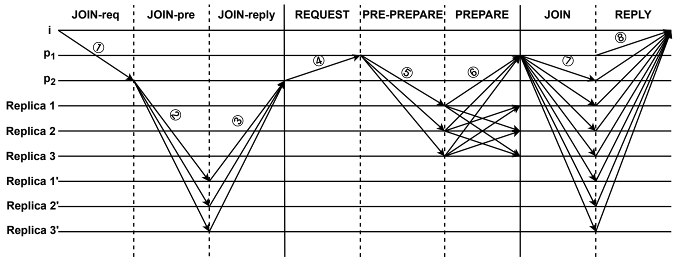

Fig. 2. The process for nodes to join the blockchain network dynamically

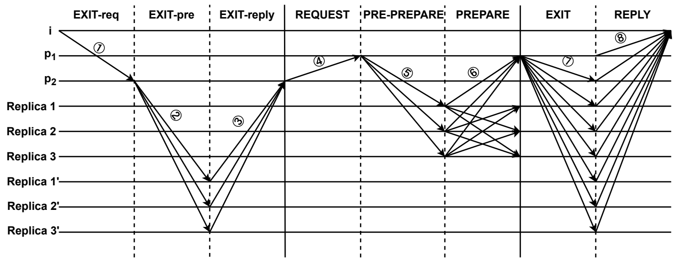

Fig. 3. The process for nodes to exit the blockchain network dynamically

B. Dynamically joining/exiting the blockchain network

Here, we provide a detailed process for a node joining and exiting the blockchain network dynamically, and the processes are shown in Fig. 2 and Fig. 3 respectively. Consensus for new nodes joining the blockchain network dynamically in Fig. 2 is as followed:

1) A new node i sends a message <JOIN-req, i, IP, pk>σ_i to p_2 (① in Fig. 2).
2) p_2 multicasts a message<<JOIN-pre, v', s', d'>, m>σ_{p_2} to replicas in the observation group (② in Fig. 2).
3) Assuming replica j' receive the multicast message from p_2, it replies message <JOIN-reply, v', s', j', D(m)>$\sigma_{j'}$ to p_2 (③ in Fig. 2). The observation group verifies the joining requests of new nodes and finishes the preprocessing.
4) ④, ⑤, ⑥ in Fig. 2 are the consensus phase, which is used to reach the consensus about preprocessing result and the more details are in the following adaptive consensus.
5) After consensus, when p_1 receives $2f$ preparing messages from high reputation values, p_1 broadcasts the message <JOIN, i, IP,>σ_{p_1} to all nodes (⑦ in Fig. 2).
6) Replicas send the message <REPLY, v, s, j, C, NL> to node i (⑧ in Fig. 2).
7) Once node i receives $f+1$ reply messages, node i synchronizes the configuration information in the network. Finally, node i successfully joins the blockchain network.

TABLE I
DEFINITION OF NOTATIONS

Nations	Descriptions	Nations	Descriptions
n	n is the number of the network blockchain	pk	pk is the public key
f	f is the number of faulty nodes	σ_i	σ_i is the signature of node i
n_1	n_1 is the number of consensus nodes	v	v is the current view number
n_2	n_2 is the number of observation nodes	m	m is the message to transmit
p_1	Primary node of the consensus group	s	s is the message sequence number
p_2	Primary node of the observation group	d	d is the message m digest
i	a new node's ID	|m|	|m| is the size of the client request message
j	a replica	C	C is the stable checkpoint

The dynamically exiting a blockchain network for a node is similar to the joining process, and is shown in Fig. 3.

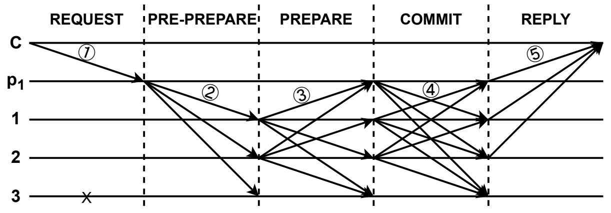

Fig. 4. Consensus process for the PBFT protocol

C. Adaptive consensus

Our adaptive consensus is realized by implementing three-phase consensus shown in Fig. 4 or two-phase consensus shown in Fig. 5 based on the reputation model [4].

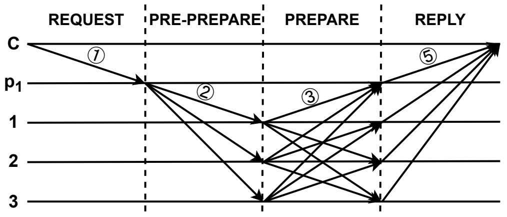

Fig. 5. The two-stage consensus process

1) In the pre-preparing phase: Once one client sends a request to the primary node p_1 (① in Fig. 4 and Fig. 5), the pre-preparing phase starts. p_1 assigns a sequence number s to the request and multicasts a pre-prepare message <<PRE-PREPARE, v, s, d>σ_{p_1}, m> to all replicas (② in Fig. 4 and Fig. 5).
2) In the preparing phase: When the replicas receive the pre-prepare message, the preparing phase is activated and executed by multicasting a message <PREPARE, v, s, d, j>σ_j to all the replicas (③ in Fig. 4 and Fig. 5).
3) Adaptive consensus: If consensus nodes receive $2f$ prepare messages from high reputation value nodes, the consensus nodes skip the committing phase and execute the client requested, followed by a reply to the client (⑤ in Fig. 5). Otherwise, the committing phase is executed. In the committing phase: replica j multicasts message <COMMIT, v, s, d, j>σ_j to all the replicas including primary (④ in Fig. 4). Upon the replica receiving $2f+1$ committing messages from different replicas, the replica executes the requested operation and sends a reply to the client (⑤ in Fig. 4). After one round consensus, each node uses the reputation model in [4] to calculate the reputation value and updates its local NL.

III. Complexity calculation

Communication complexity and message complexity are two metrics of efficiency for a consensus protocol. Communication complexity $CommCplx$ denoted as measures the total communication time for executing a consensus, while message complexity $MsgCplx$ indicates the total number of messages exchanged. It is noted that in this paper, we calculate the communication complexity by only considering the size of the message denoted as $|m|$. The communication complexity of a node joining the blockchain network denoted as $CommCplx$ consists of preprocessing communication complexity $CommCplx_{preprocessing}$ and the communication complexity of consensus $CommCplx_{consensus}$ as follows:

$$CommCplx = |\mathrm{m}| MsgCplx \tag{3}$$

$$CommCplx = CommCplx_{preprocessing} + CommCplx_{consensus} \tag{4}$$

1) Calculating communication complexity of the preprocessing phase.
   a) In the JOIN-req phase, node i sends $\mathbf{n}$ messages.
   b) In the JOIN-pre phase, p_2 sends $\mathbf{n_2}$ messages.
   c) In the JOIN-reply phase, n_2 replicas of the observation group reply to p_2, then the communication times are $\mathbf{n_2}$.

   The communication complexity of the preprocessing phase is as follows:

$$CommCplx_{preprocessing} = |m|\,(n + 2n_2) \tag{5}$$

2) Calculating the communication complexity for consensus phase.
   a) In the REQUEST phase, p_2 sends a request message to n_1 nodes in the consensus group, then, the communication time is $\mathbf{n_1}$.
   b) In the PRE-PREPARE phase, p_1 sends a message to n_1 the consensus replicas, then, the communication time is $\mathbf{n_1}$.

TABLE II
THE COMPLEXITY COMPARISON OF THE FIVE ALGORITHMS

	NG-PBFT	PBFT (1999)	The Honey Badger of BFT Protocols(2016)	HotStuff (2018)	Foundation of Dynamic BFT (2022)
Deployment Scenario	Consortium blockchain	Consortium blockchain	Consortium blockchain	Consortium blockchain	-
Msg Cplx.	$\frac{1}{4}n^2+5n$	$2n^2+3n$	$6n^3+2n^2$	$8n$	$2n^2+7n$
Comm Cplx.	$\|m\|\left(\frac{1}{4}n^2+5n\right)$	$\|m\|\left(2n^2+3n\right)$	$\|m\|\left(6n^3+2n^2\right)$	$\|m\|\left(8n\right)$	$\|m\|\left(2n^2+7n\right)$
Time Cplx.	4	5	logn	8	5
Fault Tolerant	$3f+1\leq n$	$3f+1\leq n$	$3f+1\leq n$	$3f+1\leq n$	$3f+1\leq n$

c) In the PREPARE phase, n_1 consensus nodes multicast messages to n_1 nodes, and the communication time is $\mathbf{n_1}^\mathbf{2}$.

d) In the JOIN phase, p_1 broadcasts message to all nodes in the group, and the communication time is $\mathbf{n}$.

e) In the REPLY phase, all nodes in the blockchain network reply message to node i, then the communication time is $\mathbf{n}$.

So, the communication complexity of the consensus phase is as follows:

$$CommCplx_{consensus} = |\mathrm{m}|\left(2n_1 + n_1^2 + 2n\right) \tag{6}$$

According to Equations (1)-(6) and Fig. 2, the communication complexity and the message complexity of a node joining the blockchain network are as follows:

$$MsgCplx = \frac{1}{4}n^2 + 5n \tag{7}$$

$$CommCplx = |\mathrm{m}|\left(\frac{1}{4}n^2 + 5n\right) \tag{8}$$

To compare the communication complexity, we show the communication complexity of the proposed consensus protocol and the existing consensus protocols in Table II. Obviously, by using NG-PBFT, the communication complexity is reduced from $|m|(2n^2+3n)$ to $|m|(\frac{1}{4}n^2+5n)$. In other words, if $n > 1$, the communication complexity is really reduced. Compared with PBFT [1], the Honey Badger [7], HotStuff [9], and the Foundation of Dynamic BFT [10], the complexity is decreased significantly. That is, when $n \geq 2$, the NG-PBFT protocol effectively reduces the complexity and improves the consensus efficiency.

IV. CONCLUSION

In this paper, we have proposed a novel dynamic PBFT based on node grouping, which optimizes the communication complexity and consensus efficiency of PBFT. The protocol enables nodes to join/exit the blockchain network dynamically while executing consensus. Furthermore, an adaptive consensus based on reputation values can skip the committing phase when the consensus nodes are honest enough, then, the communication complexity and consensus efficiency are improved naturally. The complexity demonstrates that NG-PBFT performs superior to PBFT. Future work will focus on implementing the proposed adaptive consensus in blockchain systems.

REFERENCES

[1] M. Castro and B. Liskov, "Practical byzantine fault tolerance," in *Proceedings of the Third Symposium on Operating Systems Design and Implementation*, USA, Feb. 1999, pp. 173–186.

[2] S. Tang, Z. Wang, J. Jiang, S. Ge, and G. Tan, "Improved pbft algorithm for high-frequency trading scenarios of alliance blockchain."

[3] X. Wu, W. Jiang, M. Song, Z. Jia, and J. Qin, "An efficient sharding consensus algorithm for consortium chains."

[4] K. Lei, Q. Zhang, L. Xu, and Z. Qi, "Reputation-based byzantine fault-tolerance for consortium blockchain," in *2018 IEEE 24th International Conference on Parallel and Distributed Systems (ICPADS)*, Singapore, Dec. 2018, p. 036106.

[5] Y. Wang, S. Cai, C. Lin, Z. Chen, T. Wang, Z. Gao, and C. Zhou, "Study of blockchains's consensus mechanism based on credit," *IEEE Access*, vol. 7, pp. 10 224–10 231, Jan. 2019.

[6] W. Tong, X. Dong, and J. Zheng, "Trust-pbft: A peertrust-based practical byzantine consensus algorithm," in *2019 International Conference on Networking and Network Applications (NaNA)*, Daegu, Korea (South), Oct. 2019, pp. 344–349.

[7] A. Miller, Y. Xia, K. Croman, E. Shi, and D. Song, "The honey badger of bft protocols," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, New York, NY, USA, Oct. 2016, pp. 31–42.

[8] B. Guo, Z. Lu, Q. Tang, J. Xu, and Z. Zhang, "Dumbo: Faster asynchronous bft protocols," in *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, New York, NY, USA, Oct. 2020, pp. 803–818.

[9] M. Yin, D. Malkhi, M. K. Reiter, G. G. Gueta, and I. Abraham, "Hotstuff: Bftconsensus with linearity and responsiveness," in *Proceedings of the 2019 ACM Symposium on Principles of Distributed Computing*, New York, NY, USA, Jul. 2019, pp. 347–356.

[10] S. Duan and H. Zhang, "Foundations of dynamic bft," in *2022 IEEE Symposium on Security and Privacy (SP)*, San Francisco, CA, USA, May 2022, pp. 1317–1334.

[11] Y. Jiang and Z. Lian, "High performance and scalable byzantine fault tolerance," in *2019 IEEE 3rd Information Technology, Networking, Electronic and Automation Control Conference (ITNEC)*, Chengdu, China, Mar. 2019, pp. 1195–1202.