Journal of Theoretical and Applied Information Technology

31st August 2023. Vol.101. No 16

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

OPTIMIZED PRACTICAL BYZANTINE FAULT TOLERANCE (O-PBFT) ALGORITHM USING GROUPING METHOD FOR CONSORTIUM BLOCKCHAIN

JANNAH YUSOFF[1], ZARINA MOHAMAD[2], HASNI HASSAN[3], AZNIDA HAYATI ZAKARIA@MOHAMED[4], SHADI A ALJAWARNEH[5]

[1,2,3,4]Faculty of Informatics & Computing, University Sultan Zainal Abidin, Terengganu, Malaysia

[5]CIT Faculty, Jordan University of Science and Technology, Irbid, Jordan

E-mail: [1]jannahyusoff03@gmail.com, [2]zarina@unisza.edu.my, [3]hasni@unisza.edu.my, [4]aznida@unisza.edu.my, [5]saaljawarneh@just.edu.jo

ABSTRACT

Blockchain is a distributed ledger that records every transaction that has ever occurred in a system. A consensus algorithm is required in blockchain to ensure that all transactions function properly. At present, the consensus algorithm that is commonly used in consortium blockchain is Practical Byzantine Fault Tolerance (PBFT) algorithm. PBFT requires all nodes in the network to participate in the consensus process. However, PBFT still has some problems such as communication overhead, low throughput, and high latency due to increasing the number of nodes in the network. To overcome these disadvantages, an optimized PBFT (O-PBFT) algorithm is proposed. The O-PBFT algorithm used grouping method in a mid-stage (prepare stage) to reduce communication complexity and assign random Byzantine nodes to improve consensus efficiency. The consistency protocol in O-PBFT were modify from the original PBFT so that O-PBFT can reach consensus with less communication in a stable network. Experimental results show that O-PBFT algorithm reduces number of communication times between nodes, increases transaction throughput, and improves consensus efficiency compared to the original PBFT algorithm. Experimental results prove that O-PBFT algorithm can be used when many nodes are involved.

Keywords: *Consensus Algorithm, Practical Byzantine Fault Tolerance (PBFT), Blockchain, Consortium Blockchain, Grouping Method.*

1. INTRODUCTION

A blockchain is a decentralized, tamper-resistance database that is shared and accessible to everybody, controlled by no single entity [1]. The bitcoin system's core technology, blockchain, was initially made public in Satoshi Nakamoto's 2008 paper "Bitcoin: A peer-to-peer electronic cash system". L. M Bash et al. [2] said that, according to the original Bitcoin whitepaper, this new technology was developed with the intention of enabling the development of a "peer-to-peer version of electronic cash [which] would allow online payments to be sent directly from one party to another without going through a financial institution". Blockchain can be viewed as a novel decentralized architecture [3] and a distributed computer paradigm that stores data in encrypted chained blocks. It verifies the data using distributed consensus algorithms that ensure the security and privacy of data access. Data transmission is secured with cryptography, manipulated using self-executed program scripts (also known as smart contracts).

The structure of the blockchain is illustrated in Figure 1. As new sets of "blocks" are added to the ledger, it continues to expand. Each block contains transaction information, a timestamp, and a link to the previous block, forming a continuous chain. By hashing the previous block and linking the hash value into the current block, the continuous chain is protected. This enables a trust chain of block or tamper resistant property since the genesis block. The ledger is not managed by a single identity. Instead, each user on the network receives a copy of the entire ledger. Old blocks are maintained indefinitely, and new blocks are added to the ledger irreversibly, making it nearly impossible to tamper data by fabricating transactions, and other data.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Blockchain can be divided into three categories that are public blockchain, private blockchain, and consortium blockchain. In a public blockchain or decentralized permissionless blockchain, anyone can participate in the consensus process and contribute to receive the rewards by following the rules, and also can freely join the network [4]. An example of a public blockchain is Bitcoin. Next, private blockchain can be viewed as a centralized blockchain since a single authority or organization controls and decides who can join the consensus process. Ethereum is an example of a private blockchain. Finally, consortium blockchain combines the characteristics of both public and private blockchain, that are the low trust of public blockchain and the single highly trustable entity mode of private blockchain [5]. It is suitable for semi-closed networks that are built by different enterprises such as supply chain industries.

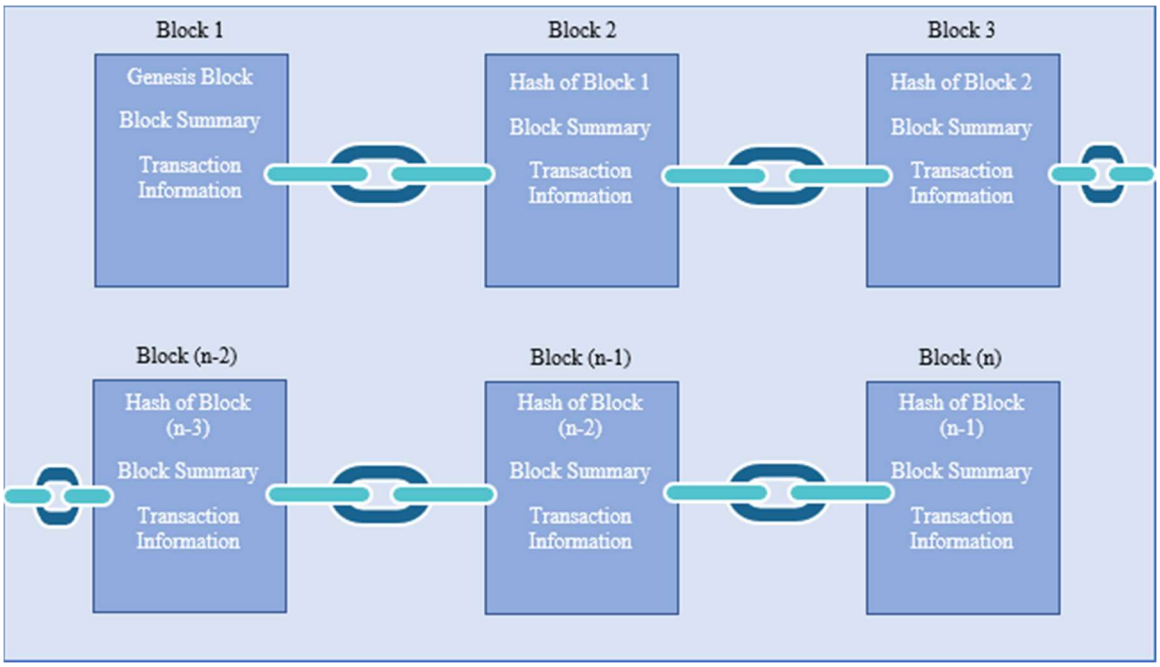

Figure 1: Blockchain Structure

Performance of a blockchain system is defined by the consensus algorithms used [6]. Consensus algorithm is the key factor in maintaining the safety [7] and efficiency of blockchain. Consensus algorithms [8] originated from the famous Byzantine generals' problems, which Lamport initially introduced in his paper "The Byzantine generals' problems" in 1982. The Byzantine generals' problems are described in [9]. Byzantine is the capital of the ancient eastern Roman Empire. Throughout Byzantine, there are several fiefs, each guarded by a general and his troops to defend against foreign enemies. Each general has two options when encountering enemies: attack or retreat. They can minimize casualties and win a war only when all honest generals agree on an order to attack or retreat. However, Byzantine is so vast that these generals cannot discuss the order together because they must guard their fiefs separately. Hence, signalmen are used to deliver the generals' commands. The generals eventually make their final decision (attack or retreat) by sending their orders to the other generals and receiving orders back from them. In this scenario, we assume that the signalmen are honest. However, some of these generals are traitors, who may send wrong orders or send different orders to different generals, undermining the overall decision of the honest generals. In conclusion, the Byzantine generals' problem can be defined as a problem of getting honest generals to reach a consensus in the presence of several traitors. Adopting this concept, consensus algorithm is used to solve the issue of maintaining data consistency in distributed systems with the presence of several failure nodes. These are some consensus algorithms introduced in the early days of blockchain systems such as Proof of Work (PoW), Proof of Stake (PoS), Delegated Proof of Stake (DPoS), and Practical Byzantine Fault Tolerance (PBFT).

Despite the fact that there are many consensus algorithms introduced and used in the blockchain, the existing consensus algorithms are mostly concerned with the public blockchain while the consortium blockchain is given the least attention [32]. This can be supported by [33] and [34] that said, at present, most of the researches on consensus algorithms are focused on the public blockchain and there are few consensus algorithms that have been developed for consortium blockchain.

Consortium blockchain is a permissioned blockchain, in which the primary nodes are pre-specified by the participants which are composed of many parties. Thus, whoever wants to access to the ledger needs to be a member of any organization. PBFT algorithm is widely used in consortium blockchain. However, PBFT has poor and low scalability because of the communication complexity [9]. The communication complexity issue in the PBFT algorithm is due to increased communication with a greater number of nodes. Therefore, this paper proposes an optimized PBFT (O-PBFT) that improved from the original PBFT algorithm. To overcome the original PBFT algorithm disadvantages, we are using the grouping method (the nodes will achieve consensus within their group only) in the prepare stage that can improve scalability and performance while ensuring fault tolerance.

The rest of the paper is organized as follows. Section 2 presents an overview of the PBFT algorithm, its disadvantages, and improvement methods in PBFT by other research papers. Section 3 introduces our O-PBFT consensus algorithm while performance analysis, and experimental results are presented, and discussed in section 4, Finally, in section 5; we provide the conclusions together with recommendations for future work.

2. PRACTICAL BYZANTINE FAULT TOLERANCE (PBFT) ALGORITHM

2.1 Overview of the PBFT Algorithm

Practical Byzantine Fault Tolerance (PBFT) algorithm was proposed by Castro and Liskov to solve the General Byzantine problem in a paper that was published in 1999. The PBFT's main goal is to replicate Byzantine state machines in a useful way such that the Byzantine fault can be tolerated [10] and the algorithm can be used in real-world system applications [11]. The process of PBFT algorithm is divided into three phases that are pre-prepare, prepare, and commit [12]. In each phase, a node moves on to the following phase if it receives votes from more than two-thirds of all nodes [8]. PBFT algorithm provides a fault tolerance of (n—1)/3 under the premise of ensuring activity and safety. The maximum number of malicious nodes, f must not exceed 1/3 of the total number of nodes, n so that

$$n > 3f + 1 \quad (1)$$

PBFT algorithm flow is illustrated in Figure 2 [13], where Node 3 is considered to be the Byzantine node, while Nodes 0 (primary node), 1, and 2 are the normal nodes. The three phases in PBFT are described as follows :

1) Request: The client sends a request to the primary node.
2) Pre-prepare: The primary node checks, and processes the client's request and broadcasts a pre-prepare message to the nodes. Each node receives and verifies the validity of the pre-prepare message. Once it is authenticated, a node will accept the request and start the prepare phase.
3) Prepare: The nodes broadcast prepared messages to each other and also received prepared messages from others and checked its validity. When a node gets 2f valid prepared messages from different nodes, the prepare phase is finished, where f is the number of Byzantine nodes in the system.
4) Commit: Each node broadcasts a commit message to others for validation. Once the number of the commit message including itself that is received by a node in the prepare phase is greater than 2f+1, it will send a reply message to the client.
5) Reply: When the client has received f+1 of the same reply message, the consensus is achieved.

2.2 Disadvantages of the PBFT Algorithm

Generally, PBFT algorithm is widely used in a consortium blockchain because it significantly improves the performance of the blockchain consensus. This algorithm has a better overall performance and is more suitable for most customized application scenarios due to its advantages of low time delay, low energy consumption, no bifurcation, and resolution of Byzantine fault tolerance. However, according to [9, 14-15], PBFT has poor and low scalability. To reach a consensus in the PBFT algorithm, all nodes must communicate with one another. This results in increased communication with a greater number of

nodes, resulting in increased communication overhead. Because of this, PBFT does not have good scalability. This is also supported by M. Vukolic [16] iterating that the PBFT algorithm processes transaction requests at high speed, but the overhead resulting from communications limits their scalability. Wei Liu et al. [17] concurred that PBFT has the advantages of high efficiency, and fast feedback as a consensus algorithm in consortium blockchain. However, when the distributed system is scaled up, PBFT is inefficient due to its high resource consumption and latency. As the number of nodes increases in the PBFT system, the communication costs required to reach a consensus in the mesh network increase exponentially, resulting in a scalability problem. The authors of [18] said that the blockchain can only scale to a few tens of nodes due to the high communication complexity. When the number of nodes in the network exceeds this threshold, the transaction confirmation delay of the PBFT algorithm increases significantly, and the throughput will be greatly reduced.

2.3 Improvement Method in the PBFT Algorithm by Other Research Papers

The PBFT algorithm has undergone extensive study and improvements by many scholars [19] along with the development of today's consortium blockchain. The three-phases consensus process between the nodes of the PBFT algorithm produces most of the communication overhead. Currently, the main improvement ideas are as follows. X. F. Liu [20] proposed a simplified PBFT three-stage protocol that simplifies the three-phase consensus of PBFT to two-phase consensus, which reduces the communication overhead, and improves the consensus efficiency. However, reducing the consensus phase will compromise the security and consistency of the algorithm. In another method, a reputation model is used to assess the trust of nodes based on their behavior, and nodes with high trust of nodes are elected as primary nodes or nodes with high trust value of nodes are chosen to form a consensus group as opposed to all nodes participating in consensus. To improve Byzantine fault tolerance, [21] proposed an optimized PBFT based on the feature trust model. In [22], a consensus group is created from nodes with a high trust degree. Still, the credibility model requires multiple iterations of credibility assessment and at the end of each consensus, the nodes must be assessed for trust. This aggregates the node voting results using a digital signature approach before sending the statistical data to each node individually. SBFT proposed in [23] assigns collectors in the blockchain network to gather and relay each node's votes throughout the three-phase consensus to minimize direct communication between nodes. Based on this technique, direct contact between nodes can be avoided but as the number of nodes in the network increases, the aggregated signature grows too large, necessitating high network performance. In another method, a grouping approach is used, based on node attributes (node hardware, performance, network communication capabilities, geographic location, etc). The nodes are divided into multiple groups by clustering algorithms, with intragroup consensus coming first and subsequently intergroup consensus. To perform intragroup and intergroup consensus by clustering and hierarchical dividing the properties of large-scale network nodes, an improved PBFT consensus mechanism based on K-medoids is proposed in [24]. In [25], a network self-clustering-based approach is proposed to group nodes based on their communication capabilities, and nodes are grouped based on their geographical locations in [26]. Based on these papers, we can conclude that grouping approach can reduce communication overhead and improve consensus efficiency, hence throughput of a system will be significantly improved.

3. OPTIMIZED PRACTICAL BYZANTINE FAULT TOLERANCE (O-PBFT) ALGORITHM

3.1 Parameters

The main parameters involved in optimized PBFT (O-PBFT) consensus algorithm in this paper are shown in Table 1.

Table 1: Parameters

Parameters	Description
n	total nodes
f	total faulty nodes (expected)
d	total groups
k	nodes in groups
group_prepare	total nodes for the first layer (prepare phase)
total_faulty_nodes_first_layer	total faulty nodes for the first layer (prepare phase)
group_commit	total nodes for the second layer (commit phase)
total_faulty_nodes_second_layer	total faulty nodes for the second

	layer (commit phase)
group_agree_commit	total groups agreeing to update new value into blockchain (commit phase)
block	blockchain

3.2 Flow of O-PBFT Algorithm

The PBFT consensus algorithm is a state machine replication algorithm [27]. The proposed algorithm is designed to use less resources to achieve the same outcome as the original PBFT algorithm. According to the PBFT algorithm [28], the algorithm model is modified based on the organization and properties of the nodes. First, we assume that the entire blockchain network is available where each node can send any messages to any other nodes in the blockchain network in real time. Next, the assumptions are there are an unknown number of Byzantine nodes in the network and the entire network system works in a permissioned environment, requiring identity verification of the node before it can participate in the consensus protocol. The Byzantine nodes, f is at most

$$f = (n - 1) / 3 \quad (2)$$

O-PBFT algorithm flow is illustrated in Figure 3 [29] with the following processes :

1) Request: The client sends a request to the primary node.
2) Pre-prepare: The primary node checks, and processes the client's request, and then broadcasts a pre-prepare message to the nodes. Each node receives and verifies the validity of the pre-prepare message. Once it is authenticated, a node will accept the request, and start the prepare phase.
3) Prepare: In this phase, the node does not need to communicate with all other nodes, but within their groups only. This happens if the total number of nodes in the blockchain, n is more than or equal to x, and x is the preset number of nodes. The primary node will send a client's request to d representative nodes where d represents the number of groups in the network. If the total number of nodes in the blockchain is less than to the number of x, there is no need to form groups. In this case, PBFT can be directly used in the blockchain. When each node receives 2f valid prepared messages consistent with itself and get acknowledgement within their groups, it means that all nodes in the groups have reached consensus, and they will move to the commit phase.
4) Commit: The group (node) broadcasts a commit message to others for validation. Once the number of the commit messages including itself received by a group (node) in the prepare phase is greater than 2f+1, it will send a reply message to the client. The decision is made based on majority groups which are 2/3 from all groups. The concept to achieve the majority node in this proposed algorithm (two-third) is still the same with the original algorithm, PBFT.
5) Reply: When the client has received f+1 of the same reply message, the consensus is achieved.

3.3 Grouping in Prepare and Commit Stage

The communication complexity problems in the original PBFT algorithm are due to the large communication among the nodes to achieve a consensus. To reduce the communication cost, intuitively, one can construct an optimized PBFT by refraining the communication within their groups only. A sub-consensus can be reached for each group, and the consensus is reached when more groups reach their sub-consensus. Based on Figure 4 [30], grouping method starts in the prepare stage. The nodes will be grouped into a few groups, sequentially where the first node will be grouped in group 1, followed by group 2, group 3, and so on. These nodes in the groups will communicate and achieve the consensus within their groups. In the commit stage, the group (node) will broadcast a commit message to another group (node) for validation, get the decision between them. When the decision is made based on majority that are 2/3 from all groups, the consensus is achieved.

4. EXPERIMENT

A simulation platform was proposed to analyze the performance of O-PBFT. The code is written in the Python language (version 3.9.10). This code is improved from [31]. For the accuracy of the experiment, the network environment is excluded and all the consensus nodes are running on the same host. The software and hardware configuration are shown in Table 2.

Table 2: Software & Hardware Configuration

Software & Hardware Configuration	Configuration
Processor	Intel Core i5-7200U @ 2.50GHz
Memory	4 GB
Operating System	Windows 10
Python IDE	PyCharm Community Edition 2021.2.1

4.1 Latency

Latency is one of the important indexes of consensus protocols. Latency represents the time difference from transaction submission to transaction confirmation. Figure 5 shows the relationship between the number of nodes and the consensus latency. The latency of the PBFT algorithm increases rapidly as the number of nodes grows. In contrast, the variation of latency based on the proposed algorithm is more stable as the number of nodes increases.

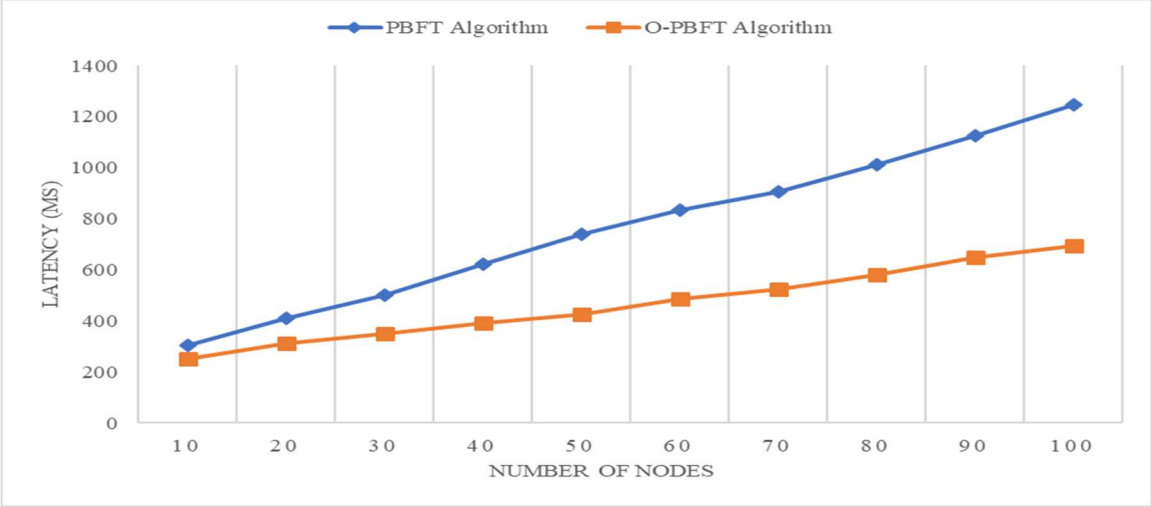

Figure 5: Latency PBFT and O-PBFT

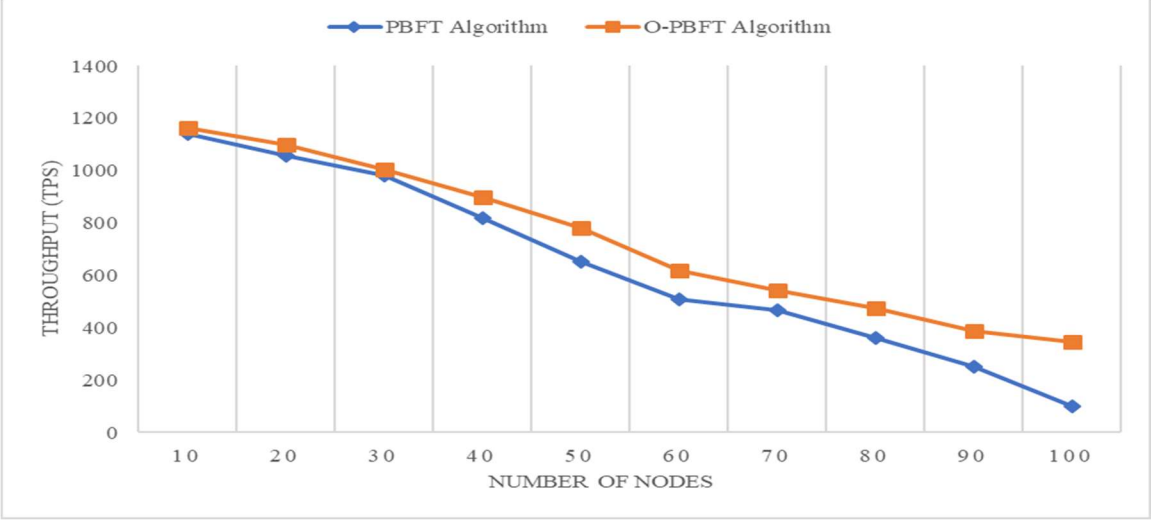

Figure 6: Throughput PBFT and O-PBFT

4.2 Throughput

Throughput is also another important indexes for measuring consensus algorithms. Throughput represents the number of transactions that the network can process per second, which is, the number of transactions generated over time

during the test. Throughput performance is usually expressed as transactions per second (TPS) when presenting performance test results. Figure 6 shows the comparison of throughput of PBFT and O-PBFT with the same number of nodes. It can be observed that, as the number of nodes in the network increases, the throughput of PBFT and O-PBFT will decrease. However, the throughput of the O-PBFT algorithm is much higher than the original PBFT algorithm.

4.3 Communication Complexity

Communication complexity is an important index of the consensus protocols [15]. It can be reflected by the number of communications times. Table 3 presents a comparison of communication complexity of each phase (pre-prepare, prepare, commit) of the original PBFT algorithm and the optimized PBFT algorithm, O-PBFT. To show the differences more clearly between them, the communication complexity is compared at different numbers of nodes. For example, there are 80 nodes in the network, where n is 80. If each group has 4 nodes, there are 26 groups; d is 26, and k is 4. According to Table 3, the complexity of PBFT is 12719 while the complexity of O-PBFT is only 985.

Table 3: Comparison of Communication Times PBFT & O-PBFT

Phases	PBFT	O-PBFT
Pre-prepare	n – 1	n – 1
Prepare	n * (n – 1)	$(k-1)^2 * (n / 3)$
Commit	n * (n – 1)	(d – 1) * (n/ 3)
Total	$2n^2 – n – 1$	$(n-1) + ((k^2n – 2kn + n) / 3) + ((dn – n) / 3)$

When the number of nodes in the blockchain network varies between 10 and 100, the communication complexity and communication times of PBFT and O-PBFT are shown in Table 4 and Figure 7. It can be observed that the communication complexity of the PBFT algorithm sharply increases with the number of nodes, reflecting the algorithm is 2n2–n–1 message complexity. Meanwhile, the communication complexity of the O-PBFT algorithm does not increase much as the number of nodes increases. IN particular, when there are 100 nodes in the blockchain network, the communication complexity of the PBFT algorithm is almost 20 times more than the communication complexity of the O-PBFT algorithm. It demonstrates that our proposed algorithm significantly reduces the number of communication times between nodes compared to the original PBFT algorithm.

Table 4: Communication Times PBFT & O-PBFT

No. of Nodes	PBFT	O-PBFT
10	189	45
20	779	112
30	1769	209
40	3159	319
50	4949	449
60	7139	619
70	9729	792
80	12719	985
90	16109	1229
100	19899	1465

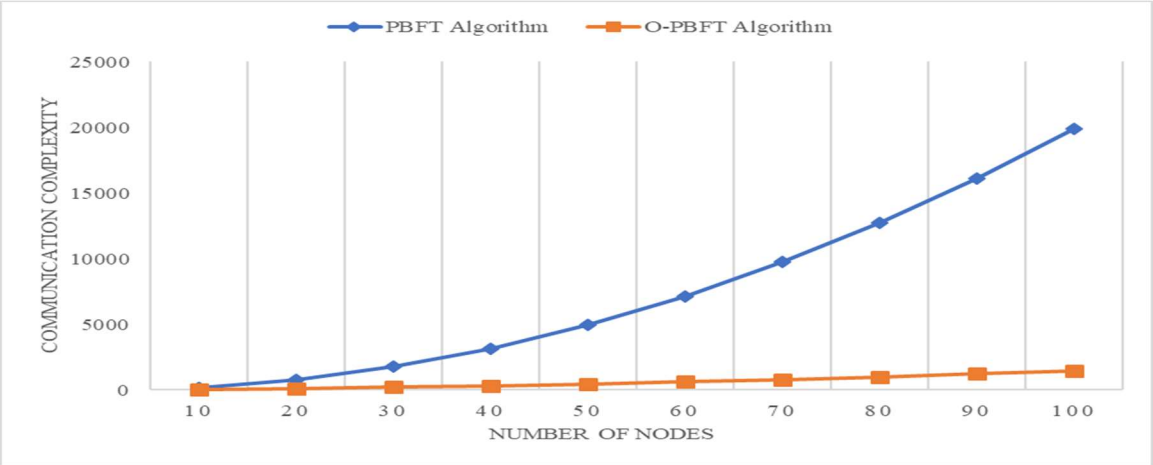

Figure 7: Communication PBFT and O-PBFT

5. CONCLUSIONS AND FUTURE WORK

In this paper, an optimized PBFT (O-PBFT) algorithm is proposed to optimize the problem of the original Practical Byzantine Fault Tolerance (PBFT) algorithm when greater nodes are involved**.** O-PBFT used a grouping method in the prepare stage to reduce communication times between nodes and assign random byzantine nodes to regulate the process fairly. This paper demonstrates that the proposed O-PBFT shows better performance in terms of latency, throughput and communication complexity compared with the original PBFT algorithm. However, the cons of this proposed algorithm are still available but we still can try to improve it because the consensus algorithm that is specially designed for each scenario is still very rare. For future work, we will focus on developing a more efficient and effective optimization model that can decide exactly on the number of nodes needed in a group to optimize the consensus.

ACKNOWLEDGEMENT

This work is supported by Matching Grant (UniSZA/2023/JORDAN (JUST)/001) Universiti Sultan Zainal Abidin (UniSZA), Malaysia and Jordan University of Science and Technology (JUST), Jordan.

REFERENCES:

[1] Salimitari M, Chatterjee M. "An overview of blockchain and consensus protocols for IoT networks". arXiv preprint arXiv:1809.05613. 2018 Sep:1-2.

[2] Bach LM, Mihaljevic B, and Zagar M. "Comparative analysis of blockchain consensus algorithms". In2018 41st International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO) 2018 May 21 (pp. 1545-1550). Ieee.

[3] Yuan Y, Wang FY. "Blockchain and cryptocurrencies: Model, techniques, and applications". IEEE Transactions on Systems, Man, and Cybernetics: Systems. 2018 Jul 25;48(9):1421-8.

[4] Wang W, Hoang DT, Hu P, Xiong Z, Niyato D, Wang P, Wen Y,and Kim DI. "A survey on consensus mechanisms and mining strategy management in blockchain networks". Ieee Access. 2019 Jan 30;7:22328-70.

[5] Joshi AP, Han M, and Wang Y. "A survey on security and privacy issues of blockchain technology". Mathematical foundations of computing. 2018;1(2):121.

[6] Kombe C, Dida M, and Sam A. "A review on healthcare information systems and consensus protocols in blockchain technology".

[7] Prisscilya, Sheren, and Togar Alam Napitupulu. "valuation of Blockchain Technology Acceptance Factors in the Tokocrypto Application". Journal of Theoretical and Applied Information Technology 101.4 (2023).

[8] Yusoff J, Mohamad Z, and Anuar M. "A Review: Consensus Algorithms on Blockchain". Journal of Computer and Communications. 2022 Sep 7;10(9):37-50.

[9] Fu X, Wang H, and Shi P. "A survey of Blockchain consensus algorithms: mechanism, design and applications". Science China Information Sciences. 2021 Feb;64:1-5.

[10] Alsunaidi SJ, Alhaidari FA. "A survey of consensus algorithms for blockchain technology". In2019 International Conference on Computer and Information Sciences (ICCIS) 2019 Apr 3 (pp. 1-6). IEEE.

[11] Zhang C, Wang R, Tsai WT, He J, Liu C, and Li Q. "Actor-based Model for Concurrent Byzantine Fault-tolerant Algorithm". In2019 International Conference on Computer, Network, Communication and Information Systems (CNCI 2019) 2019 May (pp. 552-558). Atlantis Press.

[12] Castro M, Liskov B. "Practical byzantine fault tolerance". InOsDI 1999 Feb 22 (Vol. 99, No. 1999, pp. 173-186).

[14] Zhu S, Zhang Z, Chen L, Chen H, and Wang Y. "A PBFT consensus scheme with reputation value voting based on dynamic clustering". InSecurity and Privacy in Digital Economy: First International Conference, SPDE 2020, Quzhou, China, October 30–November 1, 2020, Proceedings 1 2020 (pp. 336-354). Springer Singapore.

[15] Wang X, Guan Y. "A Hierarchy Byzantine Fault Tolerance Consensus Protocol Based on Node Reputation". Sensors. 2022 Aug 6;22(15):5887.

[16] Vukolić M. "The quest for scalable blockchain fabric: Proof-of-work vs. BFT replication". InOpen Problems in Network Security: IFIP WG 11.4 International Workshop, iNetSec 2015, Zurich, Switzerland, October 29, 2015, Revised Selected Papers 2016 (pp. 112-125). Springer International Publishing.

[17] Liu W, Zhang X, Feng W, Huang M, and Xu Y. "Optimization of PBFT algorithm based on QoS-

aware trust service evaluation". Sensors. 2022 Jun 17;22(12):4590.

[18] Sukhwani H, Martínez JM, Chang X, Trivedi KS, and Rindos A. "Performance modeling of PBFT consensus process for permissioned blockchain network (hyperledger fabric)". In2017 IEEE 36th symposium on reliable distributed systems (SRDS) 2017 Sep 26 (pp. 253-255). IEEE.

[19] Zhong W, Zheng X, Feng W, Huang M, and Feng S. "Improve PBFT Based on Hash Ring. Wireless Communications and Mobile Computing". 2021 Nov 9;2021:1-9.

[20] Liu XF. "Research on Performance Improvement of Blockchain Based on Dynamic Fault Authorization of Byzantine Tolerance Consensus Algorithm". Zhejiang University, Zhejiang. 2017.

[21] Gao S, Yu T, Zhu J, and Cai W. "T-PBFT: An EigenTrust-based practical Byzantine fault tolerance consensus algorithm". China Communications. 2019 Dec;16(12):111-23.

[22] Zheng X, Feng W, Huang M, and Feng S. "Optimization of PBFT algorithm based on improved C4.5". Mathematical Problems in Engineering. 2021 Mar 3;2021:1-7.

[23] Gueta GG, Abraham I, Grossman S, Malkhi D, Pinkas B, Reiter M, Seredinschi DA, Tamir O, and Tomescu A. "Sbft: a scalable and decentralized trust infrastructure". In2019 49th Annual IEEE/IFIP international conference on dependable systems and networks (DSN) 2019 Jun 24 (pp. 568-580). IEEE.

[24] CHENZH, LIQ. "Improved PBFT Consensus Mechanism Based on KGmedoids". ComputerScience 46.12 (2019): 101G107.

[25] Gao N, Chuangming Z, Yang C, Lina S, and He W. "Improvement of PBFT algorithm based on network self-clustering". Computer Application Research. 2021;38(11):1-8.

[26] Yoo H, Yim J, and Kim S. "The blockchain for domain based static sharding". In2018 17th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/12th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE) 2018 Aug 1 (pp. 1689-1692). IEEE.

[27] Cowling J, Myers D, Liskov B, Rodrigues R, and Shrira L. "HQ replication: A hybrid quorum protocol for Byzantine fault tolerance". InProceedings of the 7th symposium on Operating systems design and implementation 2006 Nov 6 (pp. 177-190).

[28] Kotla R, Dahlin M. "High throughput Byzantine fault tolerance". InInternational Conference on Dependable Systems and Networks, 2004 2004 Jun 28 (pp. 575-584). IEEE.

[31] https://github.com/lakki0704/Blockchain---PBFT-algorithm-in-Python, February 2023.

[32] Jiang Y, Ding S. A high performance consensus algorithm for consortium blockchain. In2018 IEEE 4th International Conference on Computer and Communications (ICCC) 2018 Dec 7 (pp. 2379-2386). IEEE.

[33] Li K, Li H, Wang H, An H, Lu P, Yi P, Zhu F. PoV: an efficient voting-based consensus algorithm for consortium blockchains. Frontiers in Blockchain. 2020 Mar 18;3:11.

[34] Li Y, Qiao L, Lv Z. An optimized byzantine fault tolerance algorithm for consortium blockchain. Peer-to-Peer Networking and Applications. 2021 Sep;14:2826-39.

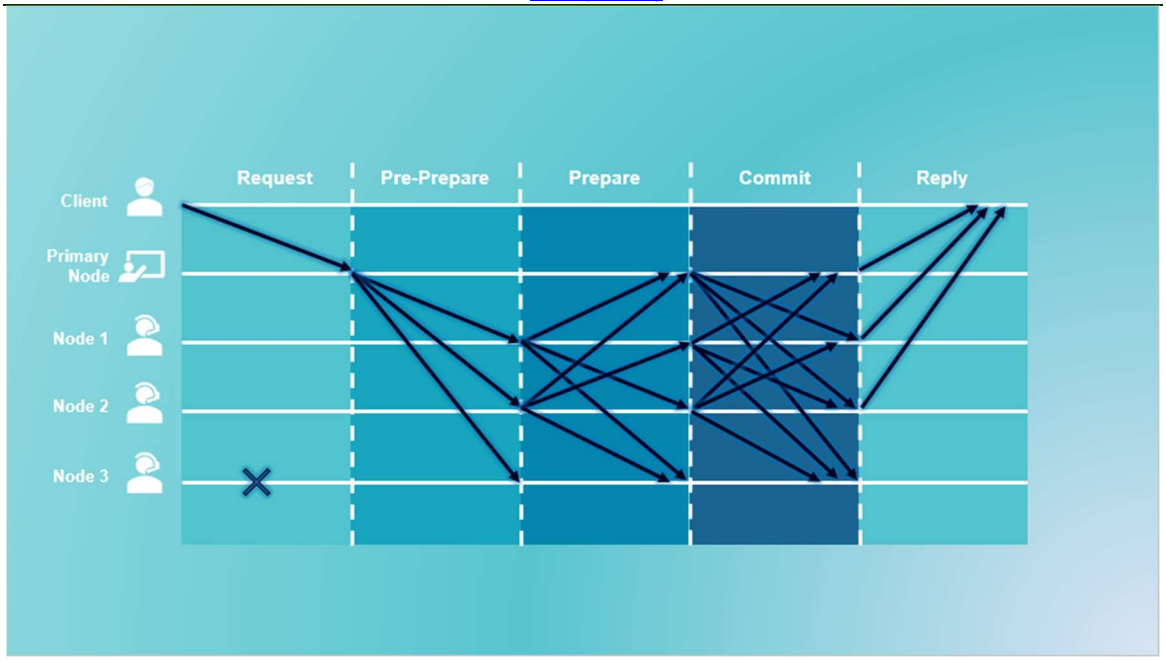

Figure 2: PBFT Algorithm Flow [13]

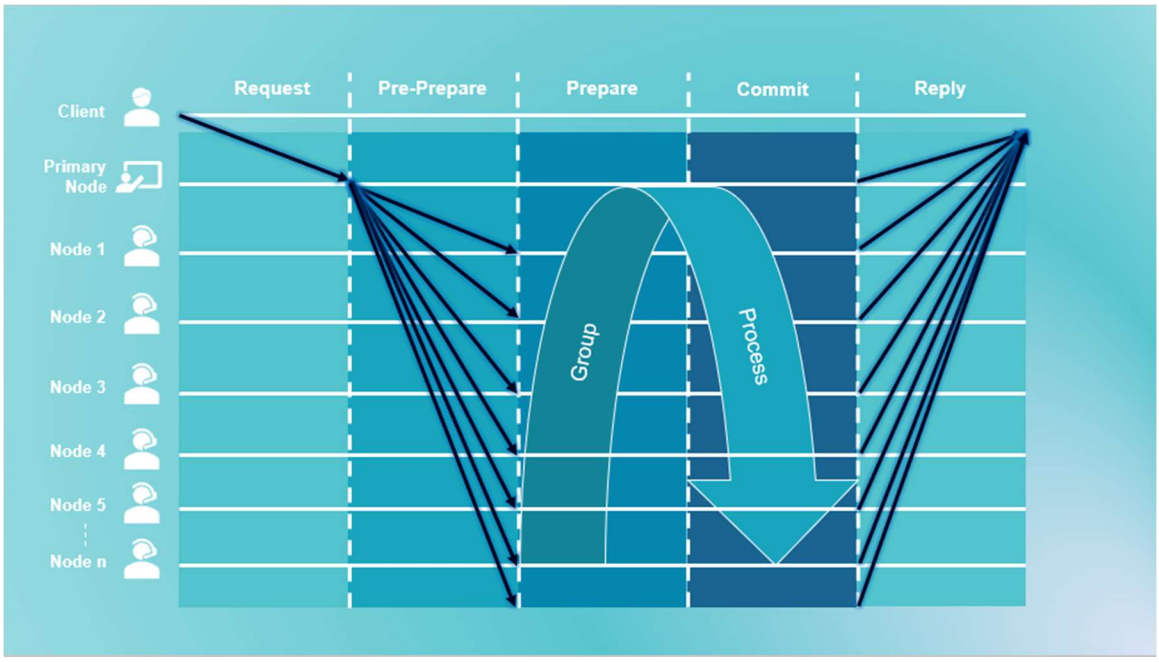

Figure 3: O-PBFT Algorithm Flow [29]

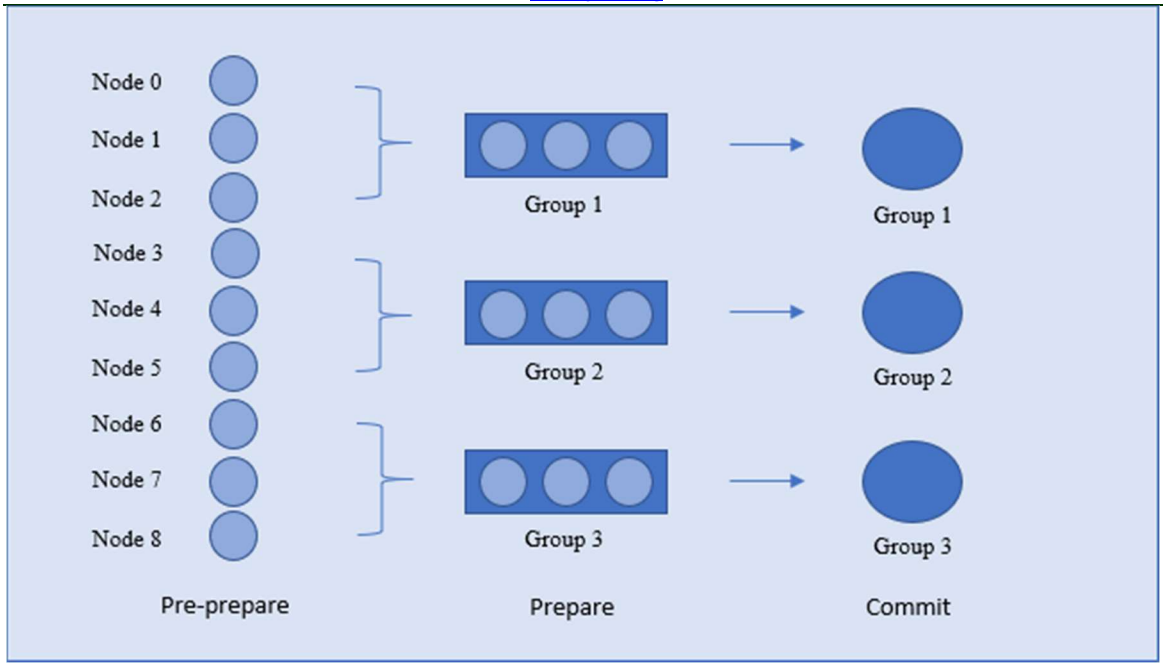

Figure 4: O-PBFT Algorithm Grouping [30]