IJARSCT

ISSN (Online) 2581-9429

International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)

Impact Factor: 7.301

Volume 3, Issue 2, March 2023

Comparative Analysis of Consensus Algorithms in Blockchain

Ahlam Shakeel Ahmed Ansari[1], **Ansari Farhin Firoz**[2], **Attarwala Murtuza Suber**[3], **Sahani Shivprasad Shankar**[4]

Assistant Professor, Department of Computer Engineering[1]

Students, Department of Computer Engineering[2,3,4]

M. H. Saboo Siddik College of Engineering, Mumbai, Maharashtra, India

Abstract: *A blockchain is a decentralized, shared, and public digital ledger that is used to log transactions across many devices so that the record cannot be altered, deleted, or destroyed retroactively without the alteration of all subsequent blocks and the consensus of the network. Blockchain is global and open to all users and is considered completely secured and verified. It is possible only because of the presence of the consensus protocol, which is a core part of any Blockchain network. The consensus algorithm is a strategy that a group of computers uses to agree with each other on what's true, and it is the foundation of all cryptocurrency blockchains. It is used to verify transactions and keep the underlying blockchain secure. There are various types of consensus algorithms in blockchain, each with its own set of benefits and losses. One consensus algorithm cannot meet the needs of all applications. Comparing the available consensus algorithms on a technical level is critical to highlight their strengths, weaknesses, and application scenarios. Major algorithms in use today are PoW, PoS, etc but they suffer from one or other problems directly affecting their performance, security, efficiency, and use. From statistics available for the performance and efficiency of various consensus algorithms it is determined that PBFT showcases various promising characteristics needed from a consensus algorithm. We proposed in this paper implementation to mitigate the scalability problem of PBFT. In this paper, we have identified parameters in various consensus algorithms and determined the best suitable consensus algorithm. This paper will serve as a resource for developers and researchers looking to evaluate and design a consensus algorithm.*

Keywords: Blockchain, Consensus Algorithm, Comparative analysis of consensus algorithms, Hierarchical PBFT

I. INTRODUCTION

Blockchainis a distributed ledger technology that has become a mainstream technology in recent years, not only among technical people but also among non-technical people. The most crucial aspects of blockchain are transparency, speed, trust, security, and scalability. The answer to achieving these aspects lies in the consensus algorithm blockchain that is necessary for blockchains to exist. A consensus protocol's main function is to facilitate node communication and offer a standard set of validated transactions that may be added to the ledger. This is done to stop unscrupulous miners from creating phony blocks and transactions. The kind of network to employ determines the kind of mechanism to utilize. It is the fuel that keeps the blockchain ecosystem running, functional, and secure. Therefore, it is important what kind of consensus model there is and how to choose the best one to ensure the smooth operation of the blockchain.

Various white papers were reviewed to determine the differences between different consensus algorithms. After deep analysis, it was determined that Practical Byzantine Fault Tolerance(PBFT) algorithms are a promising path for the future generation of consensus algorithms.

Major aspects of PBFT which improve over existing consensus algorithms are:

- Very low latency
- System delivers a high throughput
- Not very resource intensive like PoW
- Does not have the centralization problem of PoS

Still, PBFT also comes with its own set of problems, especially:

PBFT is very bad in terms of scalability which prevents it from being used in a large-scale blockchain-based system.

TABLE 1: Comparison of PBFT to the most popular consensus algorithm POW and POS

	SCALABILITY	LATENCY	FAULT TOLERANCE	THROUGHPUT	NODE IDENTITY MANAGEMENT
POW	EXCELLENT	HIGH	N/4	LIMITED(7 TX/S)	OPEN
POS	EXCELLENT	HIGH	-	LIMITED(20 - 30 TX/S)	OPEN
PBFT	EXCELLENT	GOOD	(N-1)/3	EXCELLENT	MEMBER MANAGEMENT

II. HIERARCHICAL PBFT CONCEPT

We here propose a PBFT-based system that tackles the problem of scalability seen with current PBFT algorithms.

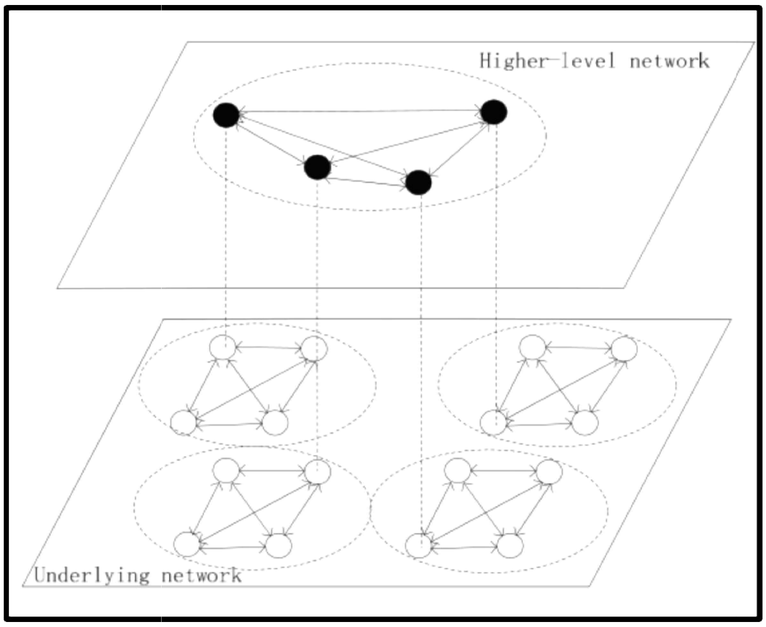

Fig. 1. Hierarchical PBFT [7]

In our system, we divide the nodes into individual clusters where PBFT can operate at excellent performance. The higher-level virtual nodes divide into actual nodes and the working of the system can be described in the following steps:

Step 1: Sending message. A client sends a request to invoke a service operation to the primary.

Step 2: Pre-prepare. The primary multicasts the request to the backups.

Step 3: Prepare. Replicas execute the request and send a reply to the other replicas.

Step 4: Commit. The primary and replicas send the executed results to other replicas.

Step 5: Reply. The client waits for f + 1 replies from different replicas with the same result; this is the result of the operation.

Algorithm:

Input: a client node, n consensus nodes, multicast algorithm, the size of the area k

Output: agreement results

```
1: begin // layer processing
2: if k ≥ n
3: run the PBFT algorithm directly;
4: else client sent request message to k agent nodes;
5: for i= 1 ton do
6: set agent to be the set of all nodes that can execute at this layer;
7: while the agent is not empty
8: do PBFT(area) for all clusters in this layer;
9: agent sends a Pre-prepare message to the next layer;
10: until an agent is empty;
11: end while
12: load balance transaction using the method of decomposing clusters, which makes nodes maintain load balance;
13: end for
```

The requests are passed down to the nodes or agents which pass the request down to their own nodes by the means of multicasting. Each node gives its consensus to the agent node which in turn passes it upward toward the central node. The final step of the algorithm is to return the execution results of all the nodes to the client.

The client determines whether the block is written on the chain according to the number of messages that are sent and the number of messages that are recovered. Therefore, the recovery mechanism and counting mechanism of the information request is very important. PBFT allows the system to have [n – 1]/3 non-faulty nodes, and its fault tolerance rate is 1/3.

In the implementation process, the recycling of the nodes is divided into two steps. First, the internal voting results of an area are returned to the agent. Second, the agent will return the votes inside the area to the client. The agent must record the amount of internal information to return. We assume that if a node agrees to accept the message, the message is sent to the agent. If the node is a non-loyal node or there is a software error or communication timeout, the node does not agree to accept.

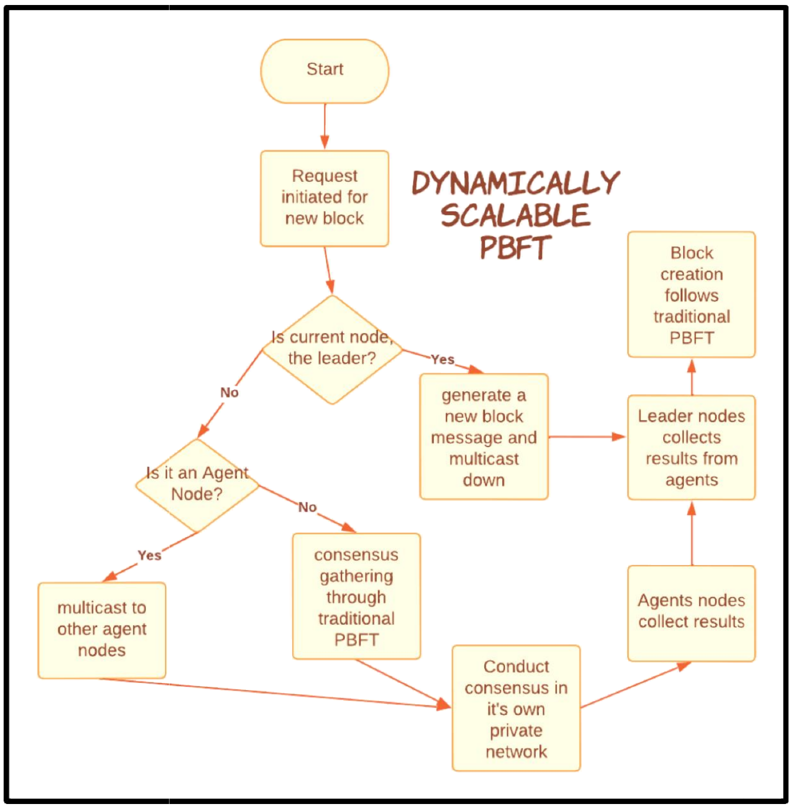
Start
Request initiated for new block
DYNAMICALLY SCALABLE PBFT
Is current node, the leader?
Yes
No
generate a new block message and multicast down
Block creation follows traditional PBFT
Leader nodes collects results from agents
Is it an Agent Node?
No
Yes
consensus gathering through traditional PBFT
multicast to other agent nodes
Agents nodes collect results
Conduct consensus in it's own private network

Fig. 2.Flowchart

III. CORRECTNESS CONDITIONSAND PROOF

The consensus nodes in a blockchain system are susceptible to arbitrary failures, such as remaining silent, delaying messages, or altering messages. As a result, the proposed method is valid only when the system possesses the following three qualities, according to the strong consistency principle:

- **Termination:** Eventually, a value is chosen by each non-faulty node.
- **Proof:** Assume the tolerance time is T and that the system has a tolerance limit. The system will forcibly end the execution of the request if the client receives no more than half of the votes gathered by the agent within time T; otherwise, the block is added to the chain.
- The global spanning tree algorithm is used by the agent to multicast the request information to each node during the agent multicast process. The system should make sure that every consensus node is a part of it. The agent is in charge of gathering and transmitting to the higher level the results of the present area's voting. The agent is eliminated if it is a flawed node or a Byzantine node. The criteria for the system to reach the termination state will eventually be met.
- **Agreement:** In a distributed system, all non-faulty nodes select the same value.
- **Proof:** All consensus nodes begin in the same condition before the client node submits a request message to the consensus system. In a consensus algorithm, a round consists of the following three steps: sending, receiving, and local computation. All nodes conduct the same local computation, send and receive identical messages, and as a result, arrive at the same state. A node that is malfunctioning will send out a mistake message, which the agent does not count. In our blockchain system, all non-faulty nodes choose the same value as a result.
- **Validity:** The non-faulty nodes must choose the same consensus value, and at least one node must enter the consensus value.
- **Proof:** The validity requirement indicates that the nodes must choose x if every node has the same input. The input message for our SDMA-PBFT method is sent by the client and conveyed by agents. The input is the same because agents merely added the counting function and left the view, message digest, and other properties alone. Numerous aspects of distributed blockchain technology affect consensus. Although there are malfunctioning nodes, the system's operation leads to agreement or disagreement in each circumstance. Consequently, our algorithm is reliable.

IV. PROPERTIESOFHIERARCHICAL PBFT EXPECTED

1. **Scalability:** A consensus node can access the blockchain consensus system safely and fast and execute consensus algorithms with other nodes without affecting the current system performance. The consensus time meets the system tolerance conditions, and the output characteristics of the entire system are enhanced as the number of nodes increases.
2. **Safety:** The ultimate goal of a consensus algorithm is to achieve consistent results. It encounters Byzantine errors, software errors, and other unpredictable circumstances in the course of its work, and thus it must be fault tolerant. The fault-tolerance performance of the PBFT algorithm is $[n - 1]/3$, which means that it can tolerate $n/3$ error nodes in the system. As our proposed system is based on PBFT, its fault-tolerance performance also satisfies $[n - 1]/3$. However, if the agent node is chosen incorrectly, the system fault tolerance is reduced.
3. **Fault Tolerance:** Here there are two cases

   a) Faulty Nodes are not Agent nodes: They can be evenly distributed or skewed on some agent nodes but in both cases, the theoretical fault tolerance remains

$$\text{Fault Tolerance} = \frac{\frac{n}{k} - 1}{3}$$

   b) Agent Nodes are the Faulty nodes: This could affect the fault tolerance of the system but the use of extreme care while selecting agent nodes could mitigate the risk to a minimum acceptable level.

IJARSCT

ISSN (Online) 2581-9429

International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)

IJARSCT

Impact Factor: 7.301

Volume 3, Issue 2, March 2023

V. CONCLUSION

While comparing the consensus algorithms it is seen that the major algorithms in use today like PoW, PoS, etc suffer from one or other problem directly affecting their performance, security, efficiency and use. Newer algorithms have being introduced throughout the years which mitigate one or many problems of the older algorithms. From statistics available for the performance and efficiency of various consensus algorithms it is determined that PBFT showcases various promising characteristics needed from a consensus algorithm. Traditional PBFT faces the problem of being very terrible in terms of scalability hence the paper references recent research works on PBFT especially Dynamic Hierarchical Practical Byzantine Fault Tolerance, a system which was proposed in this paper could be implemented to mitigate the scalability problem of PBFT.

REFERENCES

[1]. Fu, X., Wang, H. & Shi, P. A survey of Blockchain consensus algorithms: mechanism, design and applications. Sci. China Inf. Sci. 64, 121101 (2021). https://doi.org/10.1007/s11432-019-2790-1

[2]. Xiong, Huanliang, Muxi Chen, Canghai Wu, Yingding Zhao, and Wenlong Yi. 2022. "Research on Progress of Blockchain Consensus Algorithm: A Review on Recent Progress of Blockchain Consensus Algorithms" Future Internet 14, no. 2: 47. https://doi.org/10.3390/fi14020047

[3]. Md Sadek Ferdous, Mohammad Jabed Morshed Chowdhury, Mohammad A. Hoque. A survey of consensus algorithms in public blockchain systems for crypto-currencies. Journal of Network and Computer Applications, Volume 182. 2021. 103035. ISSN 1084-8045.https://doi.org/10.1016/j.jnca.2021.103035.

[4]. IEEE 15th International Conference on Smart City; IEEE 3rd International Conference on Data Science and Systems (HPCC/SmartCity/DSS), 2017, pp. 466-473, doi:10.1109/HPCC-SmartCity-DSS.2017.61.https://ieeexplore.ieee.org/abstract/document/8291964

[5]. N. Chaudhry and M. M. Yousaf, "Consensus Algorithms in Blockchain: Comparative Analysis, Challenges and Opportunities," 2018 12th International Conference on Open Source Systems and Technologies (ICOSST), 2018, pp. 54-63, doi: 10.1109/ICOSST.2018.8632190.https://ieeexplore.ieee.org /abstract/document/8632190

[6]. Castro, M., and B. Liskov. "Practical byzantine fault tolerance miguel." Proceedings of the Third Symposium on Operating Systems Design and Implementation. 2002.https://pmg.csail.mit.edu/papers/osdi99.pdf

[7]. Feng L, Zhang H, Chen Y, Lou L. Scalable Dynamic Multi-Agent Practical Byzantine Fault-Tolerant Consensus in Permissioned Blockchain. Applied Sciences. 2018; 8(10):1919.https://doi.org/10.3390/ app8101919