Simulation Based Performance Evaluation of Consensus Algorithms in NS3 for Blockchain Network

Anita Thakur[a], Virender Ranga[b] and Ritu Agarwal[c]
Delhi Technological University, India

Keywords: Blockchain, Consensus, NS3, Raft, PBFT.

Abstract: The adoption of blockchain technology extends beyond crypto assets, encompassing diverse domains within enterprise systems. The inherent qualities of decentralization and encryption lead many to perceive blockchain as an infallible repository for data security. However, within the intricate layers of blockchain architecture, the consensus layer assumes a pivotal role in governing the network's performance and ensuring robust security measures. The performance of the consensus algorithm directly impacts the efficiency and scalability of the blockchain network. Each algorithm aims to optimize the consensus process to meet the specific requirements of a blockchain network. Through simulation-based evaluations, this research paper offers an assessment of two significant consensus algorithms, namely Raft and practical byzantine fault tolerance (PBFT). The experimentation is conducted utilizing the NS3 network simulator, enabling the calculation of key performance metrics, including average throughput, packet delivery ratio, and packet loss ratio, following the leader election time and agreement time. These evaluations provide valuable insights into the effectiveness and efficiency of the Raft and PBFT consensus algorithms in real-world scenarios, contributing to the broader understanding of their applicability and potential in distributed systems.

1 INTRODUCTION

Over the past few years, blockchain has emerged as a revolutionary technology for safeguarding and validating various forms of data transfer, commencing with transactions involving cryptocurrencies. Blockchain provides exceptional advantages such as decentralization and immutability, and its supported components such as smart contracts, consensus mechanisms, distributed data storage, asymmetric encryption, and P2P networking. By commercializing these advancements, blockchain enables decentralized peer-to-peer transactions to be seamlessly executed within a distributed system, obviating the need for mutual trust among participating nodes. This revolutionary approach effectively addresses the long-standing challenges associated with high costs, low efficiency, and insecure data storage prevalent in traditional centralized systems (Mingxiao et al., 2017). In blockchain, the consensus algorithm assumes a paramount role as the foundational mechanism, exerting a profound influence on the holistic performance of the blockchain network (Li et al., 2020). Consensus represents a fundamental principle within distributed systems, extending beyond the domain of blockchain technology. It pertains to situations where multiple processes or nodes uphold a shared data state. The crucial objective of consensus algorithms is to attain unanimous accord among these nodes, ensuring that each node agrees upon a singular, authentic value.

In the context of permissioned blockchains, the participating nodes possess identifiable identities and are acknowledged as recognized entities (Cao et al., 2020). Consensus mechanisms enable decentralization by ensuring that all participants have an opportunity to participate in the decision-making process. Each participant has an equal chance of becoming a block proposer or validator, depending on the specific consensus mechanism employed. Decentralization is achieved by distributing these roles across a diverse set of network participants, mitigating the concentration of power and reducing the risk of a single entity gaining control over the network (Xiong et al., 2022). Regarding blockchain, the consensus problem pertains to achieving agreement among non-faulty or

[a] https://orcid.org/0000-0003-0926-3045
[b] https://orcid.org/0000-0002-2046-8642
[c] https://orcid.org/0000-0002-0420-1892

Thakur, A., Ranga, V. and Agarwal, R.
Simulation Based Performance Evaluation of Consensus Algorithms in NS3 for Blockchain Network.
DOI: 10.5220/0013583300004664

In *Proceedings of the 3rd International Conference on Futuristic Technology (INCOFT 2025) - Volume 1*, pages 656-663
ISBN: 978-989-758-763-4

correct processes within a distributed system regarding a specific block of transactions at a given position within the blockchain. It can be formulated with three fundamental properties: Agreement ensures that no two correct processes decide on different blocks. Validity ensures that only valid and legitimate blocks become part of the agreed-upon chain. Termination: The consensus protocol guarantees that all correct processes will eventually reach a decision.

The role of a consensus protocol in the blockchain context is crucial as it establishes a mechanism to order blocks in total order. By doing so, it prevents conflicts and inconsistencies that may arise when multiple blocks are concurrently appended to the blockchain, ensuring the integrity and reliability of the transaction history (Gramoli, 2020). There exists a multitude of consensus algorithms within the blockchain domain, encompassing both private and public blockchain systems. These algorithms can be categorized into proof-based mechanisms, wherein nodes furnish evidence of their leadership to append new blocks to the blockchain, and committee-based mechanisms, wherein nodes engage in voting to determine the subsequent block to be appended. Noteworthy algorithms falling within these classifications include Proof-of-stake (PoS), Proof-of-work (PoW), DPoS (Delegated PoS), Raft, and PBFT. This research endeavors to conduct a simulation-based evaluation of two compelling committee-based consensus protocols, namely Raft and PBFT, utilizing the NS3 network simulator. The objective is to measure various performance indicators and gain comprehensive insights into their functioning.

Section 1 presents a concise overview, elucidating the fundamental concepts of blockchain and consensus. Section 2 delves into the review of existing research conducted in consensus algorithms, encompassing a thorough examination of performance metrics and other relevant measures. Section 3 provides detailed insights regarding the two consensus algorithms selected for our study. The subsequent section 4, encompasses the simulation employed, followed by the presentation and analysis of the obtained results. Finally, section 5 summarizes the key findings and contributions of our study. This research provides valuable contributions to the field of consensus algorithms in the blockchain. The paper sheds light on performance characteristics by evaluating the Raft and PBFT protocols using the NS3 network simulator. It offers insights that can inform future advancements in distributed consensus mechanisms.

2 LITERATURE REVIEW

This section aims to undertake an extensive literature review concerning consensus algorithms in a broader context while also focusing on studies specifically dedicated to the comparative analysis of these algorithms. The objective is to identify relevant scholarly works contributing to our understanding of consensus mechanisms, their underlying principles, and the factors influencing their performance.

The work presented by (Huang et al., 2019) focuses on examining the effects of key parameters, namely network size, election timeout period, and packet loss rate, on both the probability of network splitting and network availability. The findings of this study indicate that by increasing the election timeout period, it is possible to effectively reduce the likelihood of network splitting resulting from packet loss. Numerous prominent approaches have emerged within the consensus algorithms domain, including PoS, PoA, and PoW, each with distinct advantages and drawbacks. However, it is worth noting that these algorithms also exhibit certain limitations. For instance, PoW necessitates substantial computational power, PoS addresses complexity concerns, and PoA demands additional processing time during the screening process. To alleviate these issues encountered in the algorithms, as mentioned earlier, the RSP (Rock-Scissors-Paper) algorithm (Kim et al., 2019) has been devised, offering effective mitigation strategies. The paper by (Kaur et al., 2021) presents an extensive review of mainstream consensus protocols, including PoS, PoW, PoA, and DPoS. It offers detailed explanations of these protocols and conducts performance analysis. Moreover, the paper introduces a performance matrix that evaluates these protocols based on parameters such as scalability, fault tolerance rate, latency, degree of decentralization, and other relevant factors.

The work of (Foytik et al., 2020) introduces a blockchain simulator developed to assess consensus algorithms within a configurable and realistic network environment. The simulator offers the ability to analyze the influence of both the consensus and network layers, thereby enabling practitioners to make informed decisions regarding selecting appropriate consensus algorithms. Additionally, it facilitates the evaluation of network layer events in scenarios characterized by congestion or contention in the context of the Internet of Things (IoT).

The simulator enables users to define consensus algorithm operations with greater fidelity than real-time performance, all while maintaining scalability. Simulators are integral across various industries, in-

cluding blockchain, for their role in testing, evaluation, cost and time efficiency, scalability analysis, and risk mitigation.

The study (Hanggoro and Sari, 2021) compares the performance of two simulators, NS3 and SimBlock, in CPU utilization, memory consumption, and simulation time. The findings reveal that SimBlock exhibited approximately 10% higher CPU usage than NS3, whereas SimBlock's memory utilization was approximately 14% greater than that of NS3. Notably, the simulation time for NS3 is the highest, taking 55188.3 seconds, whereas SimBlock accomplished the simulation of the same number of blocks and nodes in a significantly reduced timeframe of 68.242 seconds. In contrast, NS3 excels in providing a highly realistic simulation environment with informative output, ensuring stability and efficient resource utilization. However, one notable drawback is the significant trade-off in simulation time, which tends to be considerably longer than other simulators.

The work by Hidayat et al. (Hidayat et al., 2022) aims to assess the performance of various consensus algorithms, including Paxos, Raft, and PBFT, by utilizing the NS3 network simulator and specifically, simulating the time required to achieve consensus among participants. The results showed that the PBFT algorithm demonstrates a remarkable speed advantage, being approximately six times faster than Paxos and five times faster than Raft in reaching consensus. According to the authors, their paper is the very first to evaluate three consensus protocols such as Paxos, PBFT, and Raft on NS3. We drew inspiration from their work and conducted the study in our simulation environment; we also extensively observed the packet delivery ratio, packet loss ratio, and average. The findings show the PBFT is 10 times faster than Raft in reaching the agreement.

3 CONSENSUS ALGORITHM

A consensus algorithm can be envisioned as a consortium of machines operating as a synchronized unit, resilient enough to withstand the potential breakdown of certain constituents within the group.

3.1 Raft Consensus Algorithm

Raft is a consensus algorithm engineered with the intention of achieving optimal comprehensibility. It matches Paxos' fault tolerance and performance capabilities, making it an equivalent alternative in the blockchain domain. While designing the Raft, techniques such as decomposition (wherein Raft segregates leader election, log replication, and safety) and state space reduction (compared to Paxos, Raft minimizes the level of non-determinism and the potential for servers to exhibit inconsistencies among themselves) are employed to enhance understandability (Ongaro and Ousterhout, 2014). Following the emer-

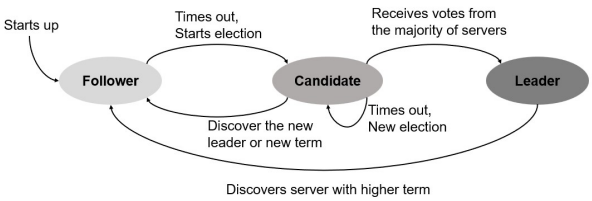

Figure 1: Raft Consensus Algorithm

gence of the Byzantine Generals Problem, Lamport presented the Paxos algorithm as a solution to the challenge of ensuring consistency under specific conditions in 1990. However, due to the intricate nature of the paper, it faced difficulty gaining acceptance. To address this, Lamport republished the paper in 1998, and subsequently, Paxos was reintroduced in 2001 (Lamport, 2001),(Lamport, 2019). To maintain the understandability of the consensus algorithm, Raft was introduced. In brief, Raft, a distributed system, is composed of multiple nodes, and one of them is elected as the leader. The leader is responsible for coordinating the operations and maintaining the consistency of the system. The other nodes are followers, which replicate the leader's actions.

The Leader in the Raft consensus algorithm plays an important role. In brief, Each node i maintains a log, denoted as log_i $=[(term_1, command_1), (term_2, command_2), ...]$, where $term_n$ represents the term number and $command_n$ represents the operation for entry n in the log. The current term of node i at time t can be represented as $current_{term_{i(t)}}$. The state of node i at time t is denoted as $state_{i(t)}$, where $state_{i(t)}$ takes values from the set follower, candidate, leader. Votes received by candidate i in term t can be denoted as $votes_{i(t)}$. Candidate i becomes the leader if it receives votes from the majority of nodes. Acknowledgment of node j for the log entries of node i is represented as ack_{ij}. When the leader receives acknowledgments from the majority of nodes, it considers the log entries committed.

In the Raft consensus Algorithm (Fig.1.), a Raft cluster comprises multiple servers, typically five, designed to withstand up to two failures while ensuring system availability. Initially, all nodes within the cluster assume the follower state. However, if a follower does not receive communication from the

leader within a specified time frame, it transitions to the candidate state. In the Leader Election process, During the candidate state, the node seeks votes from other cluster nodes to secure leadership. The candidate sends out vote requests, and the remaining nodes respond accordingly. If the candidate receives votes from the majority of the nodes, it attains the leader position.

3.2 Practical Byzantine Fault Tolerance

The initial purpose of the PBFT consensus algorithm (Castro et al., 1999) was to establish a mechanism that guarantees the integrity of a distributed network. In a distributed system composed of $3f+1$ nodes, where f denotes the count of Byzantine nodes, consensus can be achieved when a minimum of $2f+1$ non-Byzantine nodes operate without disruptions. PBFT offers assurances of safety and liveness properties, enabling the system to reach a consensus on the correct ordering of blocks and progress even in the presence of Byzantine faults. By accommodating a maximum of F faulty nodes within a network of $N = 3f+1$, $f = (n-1)/3$ validator, PBFT achieves resilience against malicious or faulty behavior (Wu et al., 2020).

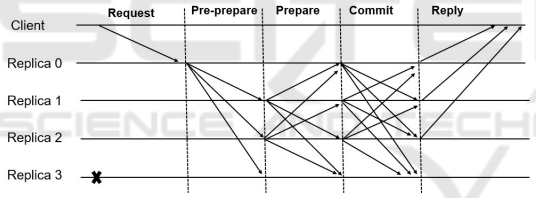

Figure 2: PBFT Consensus Algorithm

The phases of PBFT are shown in Figure 2. In the request phase, the client initiates a request to the primary node, which assigns a timestamp to the request. The client nodes, entrusted with transmitting transaction requests, proceed with their designated tasks. Later in the pre-prepare phase, the primary node (replica 0) logs the request message and assigns it a sequential order number. Additionally, the replica 0 node transmits pre-prepare messages $\ll PRE\_PREPARE, v, n, d >, m >$ to the remaining replica nodes (replica 1, replica 2, and replica 3). Here, v denotes the view number, n represents the serial number, d signifies the message digest and m denotes the original message content. Subsequently, the primary node broadcasts the message to the subsequent consensus nodes/ server nodes. These server nodes then undertake an initial evaluation to determine whether they accept or reject the received request (shown in the algorithm below) (Castro et al., 1999).

Algorithm: PBFT

1. $< REQUEST, o, t, c >$ σ_c = TRUE
   $broadcast \ll PRE\_PREPARE, v, n, d > \sigma_p$, m >

2. // Pre_prepare Phase
   $if < PRE\_PREPARE >= TRUE$ // replica accept the pre_prepare message.
   {
   brodcast $< PREPARE, v, n, d, i > \sigma_i$; // to all replica nodes.
   }
   else do nothing

3. $if\ prepared(m, v, n, i) = TRUE$
   {
   broadcast $< COMMIT, v, n, D(m), i > \sigma_i$ }

4. in Commit Phase
   $if\ committed\_local(m, v, n, i) = TRUE$
   {
   // replica i executes client requested operation and sends reply to client
   $< REPLY, v, t, c, i, r > \sigma_i$
   }
   else do nothing

Upon receiving $2f$ prepare messages from fellow server nodes (including its own, resulting in a total of $2f+1$ messages), server node i meticulously verifies the consistency of the received v, n, and d parameters with those it had initially sent out. Subsequently, when server node i gathers $2f+1$ messages, and if a majority of nodes opt to accept the request, it proceeds to transmit a commit message $< COMMIT, v, n, d, i >$ and transitions into the commit state. In the commit state, every node sends a commit message to all other nodes. server node i, upon receiving $2f$ commit messages from other server nodes (including its own, totaling $2f+1$ messages), carefully verifies the consistency of the v, n, and d parameters across these messages. Once client D receives $f+1$ identical commit messages, it confirms the attainment of consensus regarding its request. Subsequently, the server nodes respond to the client. If the client does not reply due to network delays, the request is re-transmitted to the server nodes (Wu et al., 2020).

4 EXPERIMENTAL RESULTS AND ANALYSIS

Our study employs the discrete-event network simulator NS3, which simulates the Raft and PBFT consensus algorithms and evaluates the performance. This simulation-based approach allows us to analyze and assess the efficiency and effectiveness of these algorithms within a controlled network environment. NS3 is an advanced, open-source discrete-event network simulation tool that enables researchers and developers to model and analyze complex communication networks. Developed in C++ and optimized for performance, NS3 offers an extensive range of networking protocols, device models, and simulation scenarios.

The Raft consensus algorithm, implemented (Zhayujie, 2023), consists of the important task of electing the leader node since only the leader is responsible for coordinating the operations and maintaining the consistency of the system. In this experiment, we have observed the leader selection time among a group of nodes. We have also analyzed the agreement time and calculated the performance metric packet delivery ratio and packet loss ratio, etc.

Algorithm 1: Leader Election

Input: $total\_number\_of\_nodes$, $node\_has\_voted$, $vote\_pass$, $vote\_fail$,

Output: Node $< node\_id >$ has elected as leader at time $< seconds >$,

1. Defining the value to $total\_number\_of\_nodes$ that are going to be participating in the consensus process.
2. In the case of $VOTE\_REQUEST$
   $if(node\_has\_voted == 0)$
   {
   //successfully process the vote request
   set the $(node\_has\_voted = 1)$
   }
   otherwise
   $if(node\_has\_voted == 1)$
   // node has already voted and the request is not processed
3. In the case of $VOTE\_RESPONSE$
   //if more than half of the nodes give the response to vote request,the node is elected as the Leader
   $if(vote\_pass + 1 > total\_number\_of\_nodes/2)$
   {
   // node $< node\_id >$ is elected as Leader
   // stop the next election process
   $vote\_pass = 0$;
   $vote\_fail = 0$;
   }
   else
   $if(vote\_fail >= total\_number\_of\_votes/2)$
   // more than half of the node has opposed
   // node $< node_id >$does not become Leader
   // re_initiate the voting process
   $vote\_pass = 0$;
   $vote\_fail = 0$;

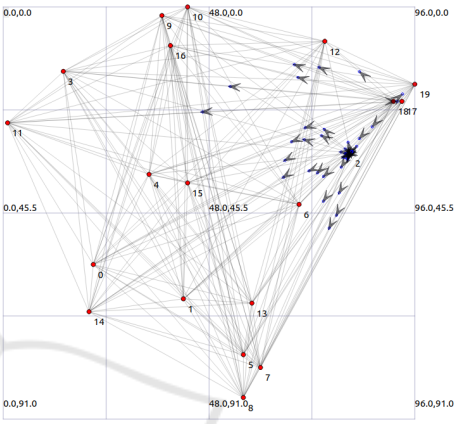

Figure 3: Topology of 20 Nodes with 4 Nodes Broadcasting Votes to Neighboring Nodes

In the pursuit of attaining leadership status, candidate nodes initiate the election process by sending vote requests to other nodes. Each server is allowed to cast a vote for a single candidate during the designated term, adhering to a first-come, first-served principle. A simulation of the leader selection algorithm, depicted in Figure 3, reveals that two candidate nodes commenced the election process with slight temporal variations. Once a node has cast its vote for a candidate, it sets the value of the variable $node\_has\_voted$ to 1 and refrains from voting for any other candidate node.

Within our simulated environment comprising 20 nodes, the initiation of an election is observed at 4 distinct nodes, specifically nodes 18, 2, 7, and 10, at timestamps of 0.172 seconds, 0.177 seconds, 0.192s, and 0.212s, respectively. Ultimately, node 18 won the election, securing the majority of votes and subsequently assuming the role of the leader at time 0.204s. Similarly, In 50 nodes, the election process is initiated by nodes 20, 42, 18, 2, 30,24,45,7, and 36 at timestamps of 0.161s, 0.171s, 0.172s, 0.177s, 0.179s, 0.182s, 0.187s, 0.192s and 0.193s respectively, where node 20 becomes the leader node. The agreement time, which refers to the time taken to achieve the consensus, is evaluated by varying the number of

nodes and message size while maintaining a constant delay of 15ms.

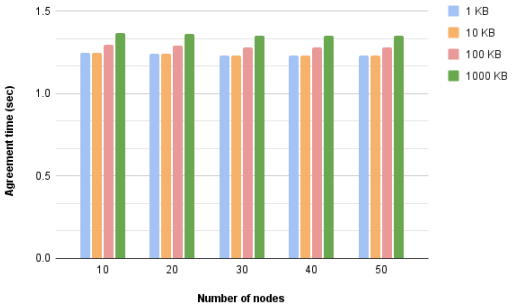

Figure 4: Time to reach the consensus in Raft consensus algorithm

On the other hand, PBFT encompasses three crucial elements: Replica, Primary, and View. The primary entity serves as the initiator of the voting mechanism, while the replica node ensures the efficacy of the voting process. In the event of a primary node failure, the view rotation function is invoked to replace the existing primary node. The PBFT algorithm implemented (Zhayujie, 2023) within the simulated environment follows a phased approach consisting mainly of Pre-prepare, Prepare, Commit, and View Change. Within this network, the client initiates a request, which is then sent to the primary node. The primary node, selected from a cluster of nodes, proceeds to execute the requested operation on behalf of the client and broadcasts a prepared message to the replica nodes. During the prepare response phase, if more than half of the responses indicate a PASS (i.e., $tx[index].prepare\_vote >= (2*N)/3$), the nodes proceed to broadcast a COMMIT message. Subsequently, if the client receives responses from more than half of the nodes indicating a commit message (i.e., $tx[index].commit\_vote > (2*N)/3$), it can be concluded that the consensus has been successfully achieved. Similarly, for the next round, a new primary node (Leader) is selected.

4.1 Observation

Based on the experimental findings, several performance metrics are analyzed, including agreement time, packet send, packet loss ratio, and packet delivery ratio. In Figure 4, the tx_size is varied from 1 KB to 1000 KB, while the number of nodes ranged from 10 to 50, with a constant delay of 15ms. Notably, both the PBFT and Raft algorithms exhibited minimal, nearly negligible impact on agreement time in response to changes in the number of nodes and tx_size. Specifically, for message sizes of 1 KB, 10 KB, 100 KB, and 1000 KB, the agreement times in 10 nodes are observed to be 1.24461 s, 1.24776 s, 1.29646 s, and 1.36801 s, respectively. Similarly, within the PBFT consensus algorithm Figure 6, agreement times for message sizes of 1 KB, 10 KB, 100 KB, and 1000 KB are recorded as 0.127485 s, 0.128685 s, 0.1408615 s, and 0.262621 s, respectively. Comparative analysis reveals that PBFT achieved agreement significantly faster, approximately 10x faster, than the Raft consensus algorithm Figure 7. To investigate the

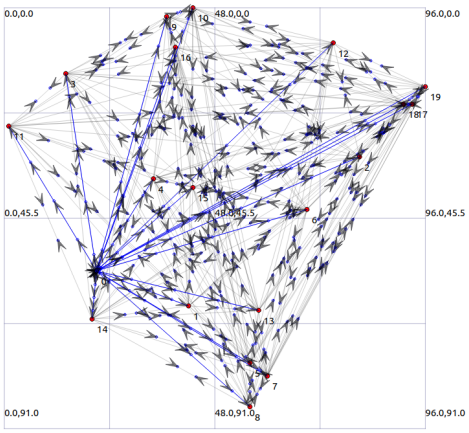

Figure 5: Topology of 20 Nodes, where nodes broadcasting message

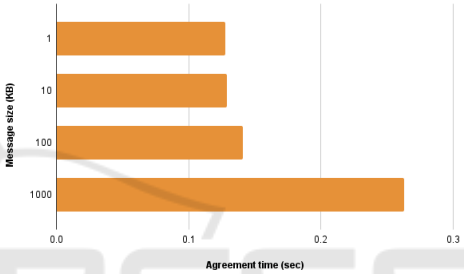

Figure 6: Time to reach the consensus in PBFT consensus algorithm

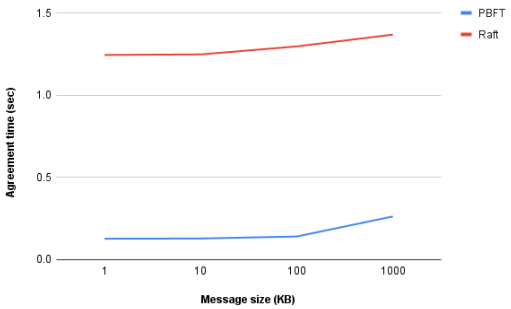

Figure 7: Agreement time comparison between Raft and PBFT

total number of packets sent in the Raft and PBFT, the number of nodes is systematically varied while maintaining a constant delay of 1ms and a fixed tx_size of 11 KB. It has been observed that PBFT sent a massive number in comparison to Raft. The PBFT algorithm presents a notable challenge due to its high message complexity. In common situations, PBFT necessitates N^2 messages for each consensus round, which can potentially hinder scalability when applied to large-scale networks.

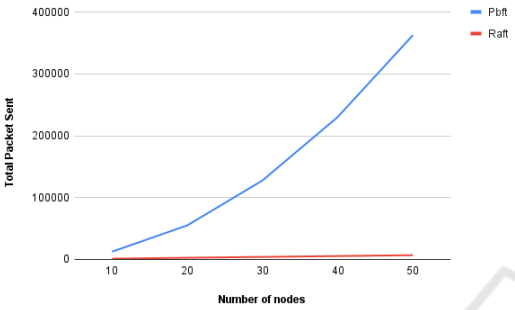

Figure 8: Total number of packets sent

Furthermore, within the PBFT algorithm, an increase in message size under low delay and limited bandwidth results in packet loss. Conversely, in the Raft algorithm, a packet delivery ratio of 100% is observed across a range of nodes from 10 to 50 and message sizes from 1 to 1000KB. In the simulated environment with a 1 ms delay and 11 KB message size, the average throughput in Raft is found to be higher compared to PBFT. Specifically, for Raft, the average throughput values are measured 66.77 Kbps, 66.79 Kbps, and 66.80 Kbps for 10, 30, and 50 nodes, respectively. In contrast, PBFT exhibited lower average throughput, with values of 22.89 Kbps, 17.80 Kbps, and 16.76 Kbps for the corresponding node configurations.

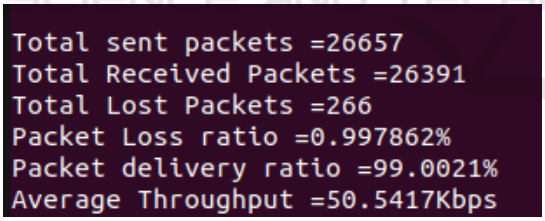

Figure 9: Packet loss in PBFT algorithm

5 CONCLUSION

Over the years, there has been an exponential surge in the advancement of blockchain technology and its utilization across various fields. Within these decentralized and distributed systems, a significant challenge lies in attaining consensus among all participants on a single data value, which is crucial for maintaining the system's reliability. The primary aim of this study is to analyze two widely adopted algorithms within a controlled environment and examine their consensus mechanisms, throughput, packet loss ratio, and other related factors. It has been observed that the PBFT algorithm is approximately ten times faster than the Raft algorithm, as evidenced by the agreement time. However, as previously discussed, Raft demonstrates better throughput performance. In future research endeavors, our focus will be on enhancing the PBFT consensus algorithm based on the observed results. We will investigate and explore potential modifications or optimizations that can be implemented to further improve the performance and efficiency of PBFT in achieving consensus in distributed systems.

REFERENCES

Cao, B., Zhang, Z., Feng, D., Zhang, S., Zhang, L., Peng, M., and Li, Y. (2020). Performance analysis and comparison of pow, pos and dag based blockchains. *Digital Communications and Networks*, 6(4):480–485.

Castro, M., Liskov, B., et al. (1999). Practical byzantine fault tolerance. In *OsDI*, volume 99, pages 173–186.

Foytik, P., Shetty, S., Gochhayat, S. P., Herath, E., Tosh, D., and Njilla, L. (2020). A blockchain simulator for evaluating consensus algorithms in diverse networking environments. In *2020 Spring Simulation Conference (SpringSim)*, pages 1–12. IEEE.

Gramoli, V. (2020). From blockchain consensus back to byzantine consensus. *Future Generation Computer Systems*, 107:760–769.

Hanggoro, D. and Sari, R. F. (2021). Performance comparison of simblock to ns-3 blockchain simulators. In *2021 4th International Conference on Circuits, Systems and Simulation (ICCSS)*, pages 45–50. IEEE.

Hidayat, S. A., Juniardi, W., Khatami, A. A., and Sari, R. F. (2022). Performance comparison and analysis of paxos, raft and pbft using ns3. In *2022 IEEE International Conference on Internet of Things and Intelligence Systems (IoTaIS)*, pages 304–310. IEEE.

Huang, D., Ma, X., and Zhang, S. (2019). Performance analysis of the raft consensus algorithm for private blockchains. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 50(1):172–181.

Kaur, M., Khan, M. Z., Gupta, S., Noorwali, A., Chakraborty, C., and Pani, S. K. (2021). Mbcp: Performance analysis of large scale mainstream blockchain consensus protocols. *Ieee Access*, 9:80931–80944.

Kim, D.-H., Ullah, R., and Kim, B.-S. (2019). Rsp consensus algorithm for blockchain. In *2019 20th Asia-*

Pacific Network Operations and Management Symposium (APNOMS), pages 1–4. IEEE.

Lamport, L. (2001). Paxos made simple. *ACM SIGACT News (Distributed Computing Column) 32, 4 (Whole Number 121, December 2001)*, pages 51–58.

Lamport, L. (2019). The part-time parliament. In *Concurrency: the Works of Leslie Lamport*, pages 277–317.

Li, X., Jiang, P., Chen, T., Luo, X., and Wen, Q. (2020). A survey on the security of blockchain systems. *Future generation computer systems*, 107:841–853.

Mingxiao, D., Xiaofeng, M., Zhe, Z., Xiangwei, W., and Qijun, C. (2017). A review on consensus algorithm of blockchain. In *2017 IEEE international conference on systems, man, and cybernetics (SMC)*, pages 2567–2572. IEEE.

Ongaro, D. and Ousterhout, J. (2014). In search of an understandable consensus algorithm. In *2014 USENIX annual technical conference (USENIX ATC 14)*, pages 305–319.

Wu, Y., Song, P., and Wang, F. (2020). Hybrid consensus algorithm optimization: A mathematical method based on pos and pbft and its application in blockchain. *Mathematical Problems in Engineering*, 2020(1):7270624.

Xiong, H., Chen, M., Wu, C., Zhao, Y., and Yi, W. (2022). Research on progress of blockchain consensus algorithm: A review on recent progress of blockchain consensus algorithms. *Future Internet*, 14(2):47.

Zhayujie (2023). Blockchain-simulator. https://github.com/zhayujie/blockchain-simulator. Accessed: 2023-05-20.