International Journal of
INTELLIGENT SYSTEMS AND APPLICATIONS IN ENGINEERING
ISSN:2147-6799 www.ijisae.org Original Research Paper

Choosing a Suitable Consensus Algorithm for Blockchain Applications: A Review of Factors and Challenges

[1]Rajat Jain, [2]Dr. Pradnya Borkar, [3]Priyanshu Deshmukh, [4]Dr. Sagarkumar Badhiye, [5]Kritika Nimje, [6]Dr. Kapil Gupta

Submitted: 25/10/2023 **Revised**: 15/12/2023 **Accepted**: 25/12/2023

Abstract: Consensus algorithms are essential for ensuring the security, reliability, and performance of blockchain systems, which are distributed ledgers that store and process transactions among multiple nodes. However, choosing a suitable consensus algorithm for a blockchain application is a challenging task, as different algorithms have different trade-offs and limitations in terms of scalability, performance, and security. This paper reviews and compares various consensus algorithms, such as Byzantine fault tolerance (BFT), practical Byzantine fault tolerance (PBFT), and their improved variants, based on multiple criteria, such as goals, power consumption, cost, CAP theorem, application scenarios, and research directions. The paper provides a comprehensive overview of the current state and future challenges of consensus algorithms for blockchain technology and offers some guidelines and recommendations for selecting the best algorithm for different blockchain applications.

Index Terms: *consensus algorithms, blockchain, Byzantine fault tolerance, scalability, performance, security, CAP theorem.*

1. Introduction

Blockchain is a new way of storing and sharing data that is very secure and transparent. It works by creating a chain of blocks, where each block contains some information and a link to the previous block. This way, anyone can see the history and verify the accuracy of the data. However, to make sure that everyone agrees on what data is in each block, there is a need for consensus algorithms. These are rules that help the different parts of the network, called nodes, to decide on the same data and avoid conflicts or errors. Consensus algorithms are very important for blockchain, and they are the focus of many research projects that try to make them better and faster. Distributed systems, which form the backbone of blockchain technology, consist of intricate components that require careful consideration. One of the challenges is to ensure the concurrent processing of information by individual nodes, each maintaining its own clock without a global time reference, while safeguarding against potential component failures. Another challenge is to achieve consensus—a shared decision-making process wherein nodes unanimously converge on a definitive output given a specific input. The pioneering work of Leslie Lamport in designing the underlying scheme for proving the correctness of distributed systems elucidates two essential criteria—safety and liveness. A robust distributed system must neither engage in erroneous behaviors (safety) nor fail to make progress towards the intended outcome (liveness). Consensus algorithms play a pivotal role in achieving this critical objective. They must satisfy three fundamental principles: validity, agreement, and termination. Validity necessitates that any value agreed upon must be proposed by one of the processes involved, underlining the importance of a transparent and accountable decision-making process. Agreement man-dates that all non-faulty nodes reach a unanimous consensus, reinforcing the significance of reliability and uniformity in the decision-making mechanism. Termination ensures that all non-faulty nodes ultimately arrive at a definitive resolution, emphasizing the need for prompt and efficient decision-making without undue delays or bottlenecks.

The paper will discuss the following parameters.
• Goals: The main objectives and design choices of each algorithm
• Power consumption: The amount of energy required to run each algorithm

[1] *Symbiosis Institute of Technology Nagpur campus, Symbiosis International (Deemed University), Pune, India. Nagpur, India*
rajat.jain.btech2022@sitnagpur.siu.edu.in
[2]*Symbiosis Institute of Technology Nagpur campus, Symbiosis International (Deemed University), Pune, India. Nagpur, India*
pradnyaborkar2@gmail.com
[3]*Symbiosis Institute of Technology Nagpur campus, Symbiosis International (Deemed University), Pune, India. Nagpur, India*
priyanshu.deshmukh.btech2022@sitnagpur.siu.edu.in
[4]*Symbiosis Institute of Technology Nagpur campus, Symbiosis International (Deemed University), Pune, India. Nagpur, India*
sagarbadhiye@gmail.com
[5]*Symbiosis Institute of Technology Nagpur campus, Symbiosis International (Deemed University), Pune, India. Nagpur, India*
kritika.nimje.btech2022@sitnagpur.siu.edu.in
[6]*St. Vincent Pallotti College of Engineering and Technology, Nagpur Nagpur, India*
kaps04gupta@gmail.com

• Potential attacks and security: The types and severity of threats that each algorithm faces and how they cope with them
• Cost: The monetary and computational expenses involved in implementing each algorithm

• CAP theorem: The trade-offs between consistency, availability, and partition tolerance that each algorithm makes
• Real world performance, adoption and enterprise use:

The practical outcomes and challenges of deploying each algorithm in various scenarios and the suitability and applicability of each algorithm for business purposes

• Scalability: The ability of each algorithm to handle increasing workloads and network sizes

• Research: The current state and future directions of research on each algorithm.

Analysing the given parameters can help one to fully understand the nature of the specific blockchain and if the person happens to work on it this analysis can help to decide which type of consensus algorithm to choose based on the system requirements. We also need to learn the CAP theorem before starting the discussion which is given in the next section.

2. Cap Theorem

[11] Brewer's theorem, or the CAP theorem, states that a distributed system cannot ensure consistency, availability, and partition tolerance simultaneously. Consistency means that all nodes have the same data. Availability means that all nodes can respond to requests. Partition tolerance means that
the system can function despite network failures. The CAP theorem implies that a system must sacrifice one of these properties and choose two.

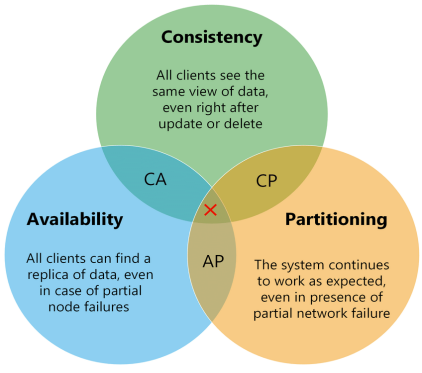

Fig. 1. CAP diagram

[11] The CAP theorem is a helpful tool to check the accuracy of a consensus algorithm, which is a protocol that helps nodes concur on the same data. A consensus algorithm must meet validity, agreement, and termination. Validity means that the agreed value is proposed by a node. Agreement means that all non-faulty nodes reach the same value. Termination means that all non-faulty nodes finally decide on a value.

The CAP theorem inspired the NoSQL movement and ignited a debate about trade-offs in data systems. It questioned the assumption that strong consistency was essential for databases. However, the CAP theorem also has some drawbacks and criticisms. It has vague and conflicting interpretations, and it does not offer a comprehensive framework to describe trade-offs. Therefore, a better framework is required to reason about systems, which should be clear, intuitive, and formal.

The CAP theorem is still a fundamental constraint for any distributed system, and ignoring it may have hidden consequences.

3. Consensus Algorithms

A. Proof of Work (PoW)

[15][10] Proof of Work (PoW) is the consensus algorithm that is mostly used by cryptocurrencies, including Bitcoin. Adding any transaction to the network requires participants, known as miners. New transactions are added only after being validated by the miners.

Miners in a Proof of Work (PoW) consensus algorithm compete to find a hash value that meets certain criteria by solving complex mathematical puzzles to validate and add transactions in the blockchain network. The computation process requires extensive energy. The work is declared as done when the first miner to solve the puzzle broadcasts the solution to the network. Other existing nodes verify the solution, and a new block is added to the blockchain.

Miners achieve this by repeatedly attempting to hash the contents of the new block while adding a random number, known as a nonce, until they find a hash that meets the required criteria. As a reward for their efforts, miners receive cryptocurrencies and transaction fees from users. By maintaining the integrity of the network, miners are incentivized to continue participating in it.

The PoW consensus algorithm is known for its high-security mechanism due to its high computational energy requirements, making it challenging for hackers to manipulate the system.

The implementation of the Proof of Work (PoW) consensus algorithm in cryptocurrencies primarily addresses Availability and Partition Tolerance in the context of the CAP theorem, while it may have an impact on Consistency. Cryptocurrencies like Bitcoin prioritize availability. Therefore, miners compete constantly to validate and add transactions to the blockchain, ensuring that the network remains responsive and that transactions continue to be processed. It ensures that the network can function even when there are network partitions or isolated nodes. The

network will continue to function as long as some miners can communicate and maintain the blockchain.

PoW does not ensure that all nodes in the network see the same data at the same time, showcasing a lack of consistency. Bitcoin, the first cryptocurrency, is based on PoW, with miners competing to solve puzzles and secure transactions. Ethereum initially used Proof of Work (PoW), but with Ethereum 2.0, it is switching to Proof of Stake (PoS). The "silver" to Bitcoin's "gold," Litecoin, employs Scrypt-based PoW. Bitcoin Cash, a fork of Bitcoin, keeps PoW while adjusting block sizes.

PoW's competitive structure encourages equitable cryptocurrency distribution, thwarts attacks, and prohibits centralized control. While PoW has been shown to be effective, criticism of its high energy consumption has prompted ongoing research into more environmentally friendly alternatives that balance security. Noteworthy drawbacks include excessive power consumption and susceptibility to 51 percent attacks.

One suggested solution involves adding multiple proof rounds to improve PoW and lower energy consumption. The results show notable improvements, with energy gain rates of 15.63 percent for five rounds and 19.91 percent for ten rounds.

In distributed systems, where every computer has a partial and constrained view of the system as a whole, diversity is essential. These distributed systems, often located in different geographic locations, use different hardware and software architectures. Decentralized peer-to-peer (P2P) networks contribute significantly to this diversity.

Introduced to address trust-related problems in distributed systems, the Byzantine Generals Problem (BGP) focuses on issues caused by malicious or untrustworthy nodes sending false information. In distributed systems, "fault tolerance" is emphasized, particularly in asynchronous environments where there is no global clock for synchronization. This is crucial in distributed computing and multi-agent systems, reaching consensus when dealing with malfunctioning processes is essential. Applications such as blockchain, clock synchronization, and cloud computing require this coordination of processes to reach a consensus on a common value. Blockchain has applications outside of finance, including smartcontracts, goods and service exchanges, predictive systems, and product traceability in sectors like food. It also provides safe, decentralized exchanges and unfalsifiable traceability.

The following are some current and potential paths for pow algorithm research:

• **Utilizing Renewable Energy**: Research into the integration of renewable energy sources and PoW algorithms. By creating systems that use clean energy, the environmental impact of conventional PoW implementations will be lessened.

• Optimizing Incentive Mechanisms: Future studies are probably going to concentrate on making PoW-based systems' incentive mechanisms more efficient. The goal of the research is to persuade miners to voluntarily adopt more energy-efficient practices by better aligning incentives.

• Strengthening Security and Decentralization: Persistent efforts to improve PoW's decentralization and security features. Researchers will look into ways to keep these important characteristics while also lowering the total number of resources used by the algorithm.

• Multidisciplinary Teamwork: Future directions call for more interdisciplinary cooperation between specialists in computer science, energy engineering, and environmental science. The goal of this cooperative strategy is to create more sustainable alternatives while addressing the problems caused by PoW from all angles

*B. **Proof of Stake (PoS)***

[10][15] Proof of Stake (PoS) is an alternative consensus algorithm used by cryptocurrencies like Ethereum 2.0 and Cardano. This algorithm saves computational work by creating stakes and having validators validate transactions, contributing to increased scalability. Participants, known as validators, put forward stakes of a certain amount of cryptocurrencies. This stake serves as collateral to participate in block validation. Validators are selected based on stake age and the quantity of tokens staked, and they are motivated to act honestly as their staked tokens may be lost if they act maliciously.

In the PoS system, consistency is maintained as all validators agree on the state of the blockchain. When a block is added to the blockchain, it is assumed that all validators will reach an agreement on the ledger's state, ensuring consistency. The processing of transactions continues even in the presence of network partitions or isolated nodes, demonstrating system responsiveness and availability. Validators are expected to be available to validate transactions and create new blocks, ensuring that the blockchain continues to function. The blockchain is designed to operate even in the event of network partitions or connectivity issues, and partition tolerance is achieved as long as some validators can communicate and validate transactions.

Proof of Stake (PoS) is a popular consensus algorithm in cryptocurrencies and blockchain networks. Ethereum 2.0 is transitioning from Proof of Work (PoW) to PoS to improve scalability and reduce energy consumption. Cardano's PoS model prioritizes security and interoperability, Tezos uses on-chain governance, Polkadot employs Nominated Proof of Stake, Avalanche uses PoS via the "Snowman" protocol, and Algorand uses a variant of PoS for speed and

decentralization. Solana utilizes "Proof of History," Cosmos and EOS use PoS and Delegated Proof of Stake, respectively, for interconnectivity and transaction validation. PoS is favored in blockchain projects for its energy efficiency and scalability.

Researchers have addressed challenges such as long-range attacks and nothing-at-stake attacks resulting from forks in enhanced PoS algorithms. Notable algorithms include Ouroboros, Sleepy Consensus, Snow White, and Delegated Proof of Stake (DPoS). These algorithms advance consensus mechanisms in areas such as encouraging voting, ensuring security in dynamic situations, and improving verification efficiency.

Sleepy Consensus (2017), created by researchers at Cornell University, questions conventional consensus algorithms and addresses their shortcomings in maintaining security in dynamic situations. Ouroboros Praos (2017) offers security against fully adaptive corruption in a semi-synchronous setting, improving the identity verification process for block producers and preventing bribery and Distributed Denial of Service (DDoS) attacks.

The dynamic availability of Ouroboros Genesis (2018) is intended to handle long-term PoS attacks and accommodate

newly added nodes to the network. In order to choose block producers, it uses a verifiable random function (VRF), and participants are free to join or exit the system whenever they choose. The protocol ensures flexibility in the face of fluctuating network conditions by handling parties going offline for protracted periods of time. The topic of privacy protection in PoS-based blockchain protocols is the focus of Ouroboros Crypsinous (2019). Because it guarantees privacy consistency and activity, it operates independently of other protocols. By fending off adaptive attacks, this design seeks to improve PoS-based consensus's overall privacy features.

A reconfigurable consensus algorithm appropriate for proof-
of-work blockchains is proposed by Snow White (2019). With a brief reconfiguration time, it permits nodes to join and leave the network at random, thwarting adversarial posterior corruption attacks. Snow White's added flexibility guarantees resilience to shifting network dynamics. These developments mark a substantial step forward for blockchain consensus algorithms, tackling important problems with security, privacy protection, and dynamic network conditions. The main objective is to increase the resilience and adaptability of blockchain technology to real-world situations.

The paragraph goes into more detail about Delegated Proof of Stake (DPoS) and how it's used in EOS and the Bit shares project. DPoS is defined as a "accounting method of democratic centralism," addressing concerns expressed by Proof of Stake (PoS) participants about not wanting to participate in bookkeeping, as well as issues related to energy waste and cooperative mining related to PoW.

C. BFT and PBFT

BFT is a characteristic of a distributed system that enables it to achieve consensus among nodes, even if some of the nodes are defective or malicious. Consensus algorithms are the principles that determine how nodes in a distributed system concur on the same value or state. BFT is crucial for ensuring the dependability and security of distributed systems, especially in applications such as blockchain, where nodes may have opposing or dishonest interests.[12]

The Byzantine Generals Problem is a well-known problem in distributed computing, where a group of generals has to concur on a common action, such as attacking or retreating from an enemy city, despite the existence of traitors and unreliable communication channels. Leslie Lamport and his co-authors introduced and solved this problem in 1982 by presenting two solutions: oral message and digital signature.

The oral message solution utilizes a verified channel and a recursive voting scheme. It works if less than one-third of the generals are traitors. On the other hand, the digital signature solution employs a cryptographic technique and a simple majority rule. It works if there are at most m traitors, where m is any number less than the total number of generals.

The choice of solution depends on the trade-off between performance and reliability, as well as the number of traitors.

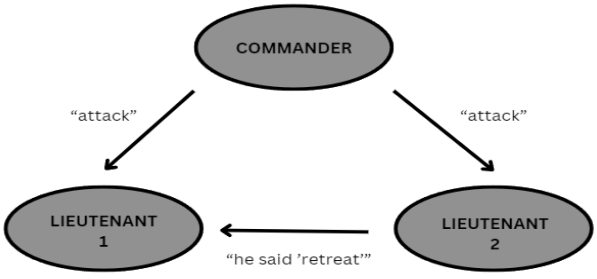

Fig. 2. Byzantine Generals Problem

Byzantine Generals	Blockchain
Geographic distance	Distributed Network
Generals	Nodes
Attack or retreat	Transactions
Traitor Generals	Faulty or malicious nodes
Unreliable Messengers	Unreliable network

TABLE 1. Comparison of Byzantine and Blockchain

[9] PBFT (Practical Byzantine Fault Tolerance) is an algorithm that replicates a state machine and consists of three main parts: the consistency protocol, the view change protocol, and the garbage collection protocol. The consistency protocol is the key component of the PBFT algorithm, ensuring that messages in the blockchain system are accurate and coherent.

The view change protocol is triggered by other nodes when the primary node fails, utilizing a timeout mechanism to maintain consensus activity. The garbage collection protocol is employed to remove old logs and update the valid range of request sequence numbers.

The algorithm operates under the assumption that the total number of nodes in the consensus system is $n = 3f + 1$, where f is the maximum number of Byzantine nodes that the system can withstand.[9]

The PBFT algorithm has five phases for achieving consensus: The request phase, the pre-prepare phase, the prepare phase, the commit phase, and the reply phase. The algorithm process can be described as follows:

1) **Request phase:** The client sends a request to the primary
node.
2) **Pre-prepare phase**: The primary node receives the request from the client, then packages the transactions into blocks in a sequential order and broadcasts them to all replicas.
3) **Prepare phase:** All replicas verify the validity of the preprepared message by broadcasting each other.

4) **Commit phase**: Similar to the prepare phase, each node verifies the validity of the message again by broadcasting each other.

5) **Reply phase**: The node that has committed the message sends the consensus result to the client. The system reaches a consensus if the client receives more than f valid reply messages from all nodes.

To explain the Goal of the BFT lets compare it with Simple
fault tolerant system or crash fault tolerant (CFT).

Type of System	Definition	Environment	Faults Handled
Crash Fault Tolerant (CFT)	Protects the system from node(s) crashes or failures in a controlled environment	Controlled	Node crashes or failures
Byzantine Fault Tolerant (BFT)	Ensures system operation despite malicious or failing node(s) in uncontrolled, open. permission-less environments	Uncontrolled, open, and permission-less	Node crashes, failures, or malicious behavior

TABLE 2. CFT VS BFT

[17] The BFT algorithm aims to ensure the continuation of system operations even in scenarios where nodes within an open and uncontrolled environment might behave maliciously or fail unpredictably. It emphasizes not only the resilience against node crashes or failures, as seen in Crash Fault Tolerant (CFT) systems, but also guards against intentional misbehavior or Byzantine faults, hence the name “Byzantine Fault Toler- ant.” Therefore, while CFT primarily addresses node crashes, BFT expands its scope to include protection against both node-related faults and malicious activities that might compromise the integrity or consensus within the system. The power consumption of BFT depends on various factors, such as the number of nodes, the communication over- head, the cryptographic operations, and the network latency. One of the advantages of BFT is that it does not require intensive computational efforts, unlike proof-of-work (PoW) consensus algorithms, which involve solving complex hashing problems. Therefore, BFT can reduce the electrical energy consumption and the environmental impact of distributed systems, especially in applications such as blockchain.[6] In Bitcoin, each block requires a Proof-of-Work round, which leads to a rise in miners’ electricity consumption. This can surpass the amount of electricity used by some small countries annually.[3] India’s per capita electricity consumption in 2021-22 was 1255 kWh, which is approximately one-third of the global average. This implies that India has a low energy consumption compared to other countries, and may benefit from using BFT consensus algorithms to run distributed systems in a more energy-efficient and reliable way. The common attacks and security challenges for BFT algorithms are discussed below. BFT algorithms are designed
to achieve consensus among distributed nodes, even in the presence of faults or attacks. Some of the common threats to BFT algorithms are timing attacks, DoS attacks, sharding
attacks, and ARP attacks, which can affect the correctness, availability, security, and efficiency of the consensus process.
Different BFT algorithms cope with these attacks and security challenges in different ways. Some of the possible solutions or countermeasures are:

• Using cryptographic techniques, such as digital signatures, message authentication codes, or hash-based proofs, to verify the authenticity and integrity of messages and nodes.

• Using trusted hardware or software components, such as trusted platform modules (TPMs) or secure enclaves, to enhance the security and trustworthiness of nodes and messages.
• Using dynamic or adaptive mechanisms, such as leader rotation, node replacement, or parameter tuning, to cope with changing network conditions or fault scenarios.
• Using advanced or novel protocols, such as hybrid or randomized consensus, to improve the resilience and performance of BFT algorithms.

The Cost of implementing BFT algorithm can be estimated by using the following formula:

$$C = N (E + L + S + T) \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad (1)$$

where C is the total cost, N is the number of nodes, E is the energy consumption per node, L is the network latency per node, S is the storage space per node, and T is the time complexity per node. The values of E, L, S, and T can vary depending on the specific BFT algorithm and the system configuration.
PBFT, Practical Byzantine Fault Tolerance, is one of the most common and efficient Byzantine Fault Tolerance (BFT) algorithms. It operates in three phases of communication and can tolerate up to one-third of the nodes being Byzantine. The cost of PBFT can be estimated using the formula:

$$C = N (E + 3L + S + O(N^2)) \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad (2)$$

where:

- E is the energy consumption of cryptographic operations,

- L is the network latency of message transmission,

- S is the storage space of message logs, and

- $O(N^2)$ is the time complexity of message verification.

In the context of the CAP theorem, BFT algorithms, including PBFT, make trade-offs between consistency, availability, and partition tolerance depending on the design and purpose of the system. PBFT prioritizes consistency and availability over partition tolerance. This means that it can ensure all nodes have the same view of the data and respond to every request as long as the network is reliable, and the number of faulty nodes is below the threshold.

However, in cases where the network is partitioned or the number of faulty nodes exceeds the threshold, PBFT may struggle to reach consensus or provide service. The CAP theorem highlights the inherent trade-offs in distributed systems, and PBFT's design reflects its emphasis on maintaining consistency and availability under normal operating conditions.[6]

BFT and PBFT have been deployed in various applications, such as blockchain, cloud computing, and IoT, to achieve different practical outcomes. In blockchain, they ensure the security and performance of the distributed ledger. In cloud computing, they ensure the availability and consistency of the cloud services. For IoT, they ensure the privacy and authenticity of IoT data and devices. Byzantine Fault Tolerance (BFT) and Practical Byzantine Fault Tolerance (PBFT) are consensus algorithms suitable and applicable for business purposes, especially in scenarios requiring high security, reliability, and performance.

The text mentions scalability issues with BFT and PBFT algorithms, which are used to achieve consensus among distributed nodes in the presence of faults or attacks. BFT algorithms have a high communication complexity of O(N^2) or O(N^3), where N is the number of nodes, limiting their scalability. PBFT, a specific BFT algorithm, has a lower communication complexity of O(N^2) but still cannot support very large networks. Techniques and methods to improve the scalability of BFT and PBFT algorithms include reducing the number of communication phases or messages, as seen in Zyzzyva, which reduces communication phases from three to two in the best case. Other methods involve using sharding or grouping techniques to divide nodes into smaller subsets or clusters, performing consensus within or across the subsets or clusters, and utilizing a mesh-and-spoke network to group nodes into different layers. Additionally, hybrid or randomized consensus protocols, such as Istanbul BFT, combine the advantages of different consensus algorithms by merging PBFT with a proof-of-authority algorithm.

Research on Byzantine Fault Tolerance (BFT) and Practical Byzantine Fault Tolerance (PBFT) algorithms is an active and important area of study in distributed systems, particularly in blockchain applications. BFT and PBFT algorithms aim to achieve consensus among nodes, even if some nodes are faulty or malicious, ensuring the security and performance of the system. However, these algorithms face various challenges and limitations, such as scalability, performance, and security, which require further improvement and optimization.[7][8]

Some of the current state and future directions of research on BFT and PBFT algorithms are:

- Improving the scalability and throughput of BFT and PBFT algorithms by reducing the communication and computational complexity, using sharding or grouping techniques, or combining different consensus algorithms.

- Improving the performance and efficiency of BFT and PBFT algorithms by using speculative execution, leader rotation, or randomization techniques, or leveraging trusted hardware or software components.

- Enhancing the security and resilience of BFT and PBFT algorithms by employing cryptographic techniques, reputation models, or voting mechanisms, or addressing various attacks, such as timing attacks, denial of service attacks, sharding attacks, and address resolution protocol attacks.

- Applying BFT and PBFT algorithms to various scenarios and applications, such as cloud computing, Internet of Things, smart contracts, and data analytics, and evaluating their feasibility and effectiveness.

D) Paxos Algorithm

Paxos is a family of protocols designed to achieve consensus in a network of unreliable or error-prone processors. It has been a dominant topic in discussions on consensus algorithms over the past decade and is frequently used to teach students about consensus. However, Paxos can be challenging to grasp, despite various attempts to make it more accessible. Furthermore, implementing Paxos often involves complex changes in system architecture, making it a daunting task for both system builders and students.

Paxos operates within a network of nodes, with each node providing input as a proposal related to a topic, allowing each node to participate in the decision-making and agreement process. The Paxos algorithm consists of three phases:

1**) Phase 1 (Prepare and Promise):** In this phase, a proposer node suggests a value for the system to agree upon. Subsequently, the proposer node requests other nodes to propose their views on the value. Once all nodes submit their input, acceptor nodes accept the proposal with the highest majority and communicate the result.

2) **Phase 2 (Accept):** After the proposal gains approval from the majority of acceptors, the proposer sends an acceptance request containing the proposed value, and the acceptors confirm their agreement.

3) **Phase 3 (Learn):** When the proposer receives acknowledgments from the majority of nodes, consensus is achieved. The value that has been agreed upon is considered finalized, and the proposer can inform learners of this agreement.

Several systems have implemented the Paxos algorithm:

1) **Google Chubby**: Chubby by Google is a distributed lock service that employs Paxos to maintain consistency among replicas in case of failures.

2) **Google Spanner and Megastore:** Both of these Google systems use Paxos internally to achieve consistency.

3) **Open Replica:** Open Replica utilizes Paxos to maintain its open-access storage system.

4) **Apache Cassandra:** Apache Cassandra employs a 'Paxos-like protocol,' which is a variant of the Paxos algorithm.

The Paxos consensus algorithm, while being a fundamental contribution to distributed systems, does face certain challenges:

a) **Complexity**: The complexity of Paxos can lead to errors during implementation, making it challenging to ensure the correctness of the system. To improve this, it may be beneficial to simplify the process or break it down into smaller components that can be tested and verified independently. Additionally, implementing thorough testing protocols and regularly reviewing and updating the code can help catch and prevent errors before they become major issues.

b) **Performance**: Paxos can experience performance issues due to message delays, participant activity, and message volume. To improve the performance of Paxos, you may want to consider implementing message prioritization, optimizing participant activity, and employing message compression techniques to reduce message volume. Additionally, investigating the possibility of using a more powerful server infrastructure or upgrading your network capabilities to handle higher message volumes may be beneficial.

c) **Non-Byzantine Fault Tolerance**: When it comes to the Paxos algorithm, Byzantine faults can be a major concern due to the potential for faulty nodes to engage in malicious behaviour. This is particularly worrisome in scenarios where there may be malicious actors seeking to disrupt the system. It's worth noting that the Paxos consensus algorithm is not specifically designed to handle Byzantine faults, whereas the Raft consensus algorithm is. This is an important consideration to keep in mind when deciding which algorithm to use in a given scenario.

E) Raft Algorithm

The Raft consensus algorithm is a protocol that enables members of a distributed system to agree on a sequence of values even when there are failures in the system. It was designed as an alternative to the Paxos family of algorithms, to be simple to understand. In a Raft system, there are two parties- The Leader and the Follower or candidate, in case the leader is not available or during an election. The leader has the duty of log replication to the followers and regularly keeps the followers informed about the log replication in the form of a heartbeat. In case there

is no heartbeat signal, the follower changes its status from follower to candidate to start a new election of leader.[20]

Raft implements consensus with the means of a leader approach. One cluster has a single leader who is completely responsible for log replication on the cluster's other servers.
The Raft algorithm was designed to make it easier for developers to understand and implement compared to the more complex Paxos algorithm. However, like any algorithm, Raft is not without its challenges.[16]

1) One of the main challenges of the Raft algorithm is its strong dependency on the leader replica. This means that the system's availability can be compromised, especially when dealing with grey failures.[15] Therefore, it is crucial to have a reliable leader replica to ensure system stability.

2) Another challenge of the Raft algorithm is its performance. Since only the leader server interacts with clients, it can become a bottleneck in case of a spike in client activity.[19] This can lead to a slower system response time and a decrease in overall performance.
3) Despite being relatively simpler than Paxos, implementing the Raft algorithm in a real-world system can still be
complex. There might be several issues to consider, such as network partitions, node failures, and message loss. Therefore, it is essential to have a thorough understanding of the algorithm and how it works to implement it effectively.[16]

Here are some of the systems that have Raft implementation:
1) **CockroachDB**: They use Raft in the replication layer.

2) **MongoDB**: Uses a variant of raft in their replication set.
3) **RabbitMQ**: Uses Raft to implement FIFO queues.

4. Conclusion

Consensus algorithms are essential for ensuring the security,
reliability, and performance of blockchain systems, which are
distributed ledgers that store and process transactions among
multiple nodes. However, choosing a suitable consensus algorithm for a blockchain application is a challenging task,
as different algorithms have different trade-offs and limitations in terms of scalability, performance, and security. This paper reviewed and compared various consensus algorithms, such as PoW, PoS, BFT, PBFT, Paxos, Raft and their improved variants, based on multiple criteria, such as goals, power consumption, cost, CAP theorem, application scenarios, and research directions. The paper provided a comprehensive overview of the current state and future challenges of consensus algorithms for blockchain technology, and offered some guidelines and recommendations for selecting the best algorithm for different blockchain applications.

References

[1] Arati Baliga. Understanding blockchain consensus models.

[2] Ethan Buchman, Jae Kwon, and Zarko Milosevic. The latest gossip on bft consensus. arXiv preprint arXiv:1807.04938, 2018.

[3] Press Information Bureau. Raksha mantri us secretary of defence to co-chair india-us 2+2 ministerial dialogue hold bilateral talks on november10, 2023. Ministry of Defence, 2023.

[4] Miguel Castro and Barbara Liskov. Practical byzantine fault tolerance.In Proceedings of the Third Symposium on Operating SystemsDesign-and Implementation, page 173–186. USENIX Association, 2001.

[5] Stefano De Angelis, Leonardo Aniello, Roberto Baldoni, Federico Lombardi, Andrea Margheri, and Vladimiro Sassone. Pbft vs proof- of-authority: Applying the cap theorem to permissioned blockchain. In Italian Conference on Cyber Security, 2018.

[6] Diego Geroni. Byzantine fault tolerance - a complete guide. 101 Blockchains, 2021.

[7] Suyash Gupta, Jelle Hellings, Sajjad Rahnama, and Mohammad
Sadoghi. An in-depth look of bft consensus in blockchain: Challenges
and opportunities. In 2019 IEEE International Conference on Blockchain and Cryptocurrency (ICBC), page 370–377. IEEE, 2019.

[8] Zainab Hussein, Mohamed A Salama, and Sherif A El-Rahman. Evo-lution of blockchain consensus algorithms: a review on the latest milestones of blockchain consensus algorithms. Cybersecurity, 6(30),2023.

[9] Wangxi Jiang, Xiang Wu, Ming Song, Jie Qin, and Zhen Jia. Improved pbft algorithm based on comprehensive evaluation model. Applied Sciences, 13(2):1117, 2023.

[10] Mohammad Ayoub Khan, Lina Ge, Jie Wang, and Guifen Zhang. Survey of consensus algorithms for proof of stake in blockchain. Security and Communication Networks, 2022:2812526, 2022.

[11] Martin Kleppmann. A critique of the cap theorem. arXiv preprint arXiv:1509.05393, 2017.

[12] Leslie Lamport, Robert Shostak, and Marshall Pease. The byzantine generals problem. ACM Transactions on Programming Languages and Systems (TOPLAS), 4(3):382–401, 1982.

[13] Leslie Lamport, Robert Shostak, and Marshall Pease. The part-time parliament. ACM Transactions on Computer Systems (TOCS), 16(2):133–169, 1998.

[14] Caitie McCaffrey. The verification of a distributed system: A practi-tioner's guide to increasing confidence in system correctness.

[15] Milvus. Raft or not? the best solution to data consistency in cloud. https://milvus.io/blog/raft-or-not.md, 2023.

[16] Kara Mostefa, Abdelkader Laouid, Muath Alshaikh, Mohammad Ham- moudeh, Ahc`ene Bounceur, Abdelfattah ammamra, and Brahim Laouid. A compute and wait in pow (cw-pow) consensus algorithm for preserving energy consumption. Applied Sciences, 11, 07 2021.

[17] Diego Ongaro and John Ousterhout. In search of an understandable consensus algorithm. In 2014 USENIX annual technical conference (USENIX ATC 14), pages 305–319, 2014.

[18] Ranjeet Patel. Byzantine fault tolerance (bft) and its significance in blockchain world.

[19] Deepak Puthal, Saraju P Mohanty, Priyadarsi Nanda, Elias Kougianos, and Gautam Das. Proof-of-authentication for scalable blockchain in resource-constrained distributed systems. IEEE Transactions on Indus- trial Informatics, 16(9):6083–6091, 2020.

[20] Raft. Raft consensus algorithm. https://raft.github.io/, 2013.

[21] Robbert Van Renesse and Deniz Altinbuken. Paxos made moderately complex. ACM Computing Surveys (CSUR), 47(3):1–36, 2015.

[22] Wenbing Zhao. Building dependable distributed systems. John Wiley & Sons, 2014.

[23] Zibin Zheng, Shaoan Xie, Hongning Dai, Xiangping Chen, and Huaimin Wang. An overview of blockchain technology: Architecture, consensus, and future trends. In 2017 IEEE International Congress on Big Data (BigData Congress), pages 557–564, 2017.