EasyChair Preprint

№ 7005

Survey of Blockchain Consensus Algorithms

Mayank Jain

November 7, 2021

Survey of Blockchain Consensus Algorithms

Mayank Jain
B.Tech Computer Science
VIT Vellore (2019-2023)
jmayank2001@gmail.com

Abstract
A testament to the rapid transition of world order from the centralised means of operation to progressive decentralisation is the growing popularity of the Blockchain technology. While some sectors have undergone a complete paradigm shift courtesy of this, others have certainly felt its influence. Much of this can be accredited to its revolutionising principles of maintaining irreversible, tamper-free and distributed records. However, despite welcoming this technology with open arms most companies are still struggling to replace their conventional models with Blockchain. This problem is spawned by the lack of exhaustive research that draws boundaries between the vast amount of consensus algorithms that developers are flooded with. To cater to the dire need of laying benchmarks for consensus algorithms, this paper attempts to list down comprehensive parameters that organisations could refer to in order to adopt the Blockchain model effectively. It further does an exhaustive comparison of popular consensus protocols against these parameters.
Keywords: Blockchain - Decentralisation - Consensus Algorithm - Byzantine Fault Tolerance - Crash Fault Tolerance - Scalability - Throughput

1. Introduction

Any system that involves decision making could broadly be categorised into two categories - *centralised* and *decentralised*. To an average individual, centralisation is the more obvious way of going about things. The need for a ‘regulatory body’ in every system is so necessary that one cannot imagine an efficient and secure system without it. This approach can be classified as the *centralised approach*. Consider the simple example of storing a transaction onto a ledger. Multiple parties may agree to keep a bookkeeper or some sort of centralised authority that would modify the ledger. The parties must entrust their transactions to this authority, who would take

care of validating these and appending them onto the ledger. This is a flawless model until enthusiasts start to question the legitimacy of the authority itself! By agreeing to the terms of centralisation, the parties are not only entrusting their transactions to the centralised authorities but are permitting them to exert greater control over the system. The mere assumption that centralised authorities are genuine rips the entire centralised model apart. To not have anyone exert greater control over the system is why we thought of creating a regulatory body in the first place! Is there still a way to achieve this without making the so-called governing body almighty?

1.1. Towards Decentralisation

Decentralisation is the answer to the question raised in the previous section. It can be thought of as dividing all the power that a central authority should ideally have, into chunks distributed amongst the individual parties (nodes). The nodes are no longer connected to each other by this intermediary authority. They adopt a *peer-to-peer* method of communication, forming what's known as a *peer-to-peer network*. Figure 1 shows the fundamental difference between the two approaches.

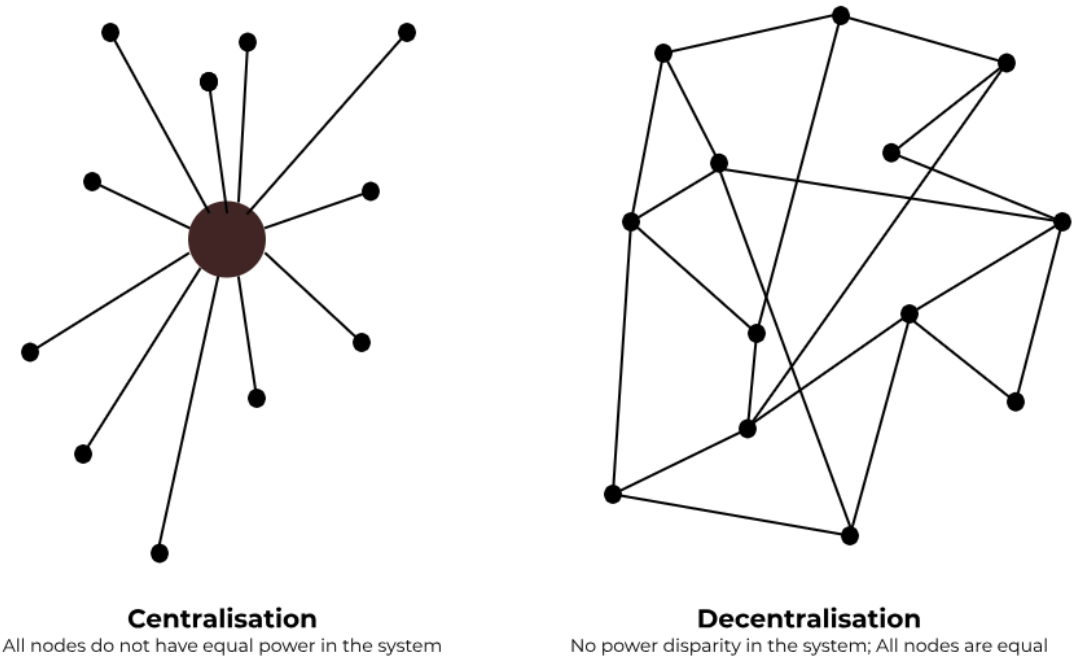

Figure 1: Depicts centralised and decentralised models

To illustrate how decentralisation really works, let us consider the ledger example from before (Figure 2). The parties (*nodes*) now maintain a copy of the ledger with them. Whenever new transactions are recorded by the system, each node updates its ledger. Any node found tampering with the ledger, could easily be recognised as illegitimate because the changes it makes to its local copy may not be validated by other 'honest' nodes in the network. This solves the *key limitation of centralisation* - maintaining tamper-free ledgers.

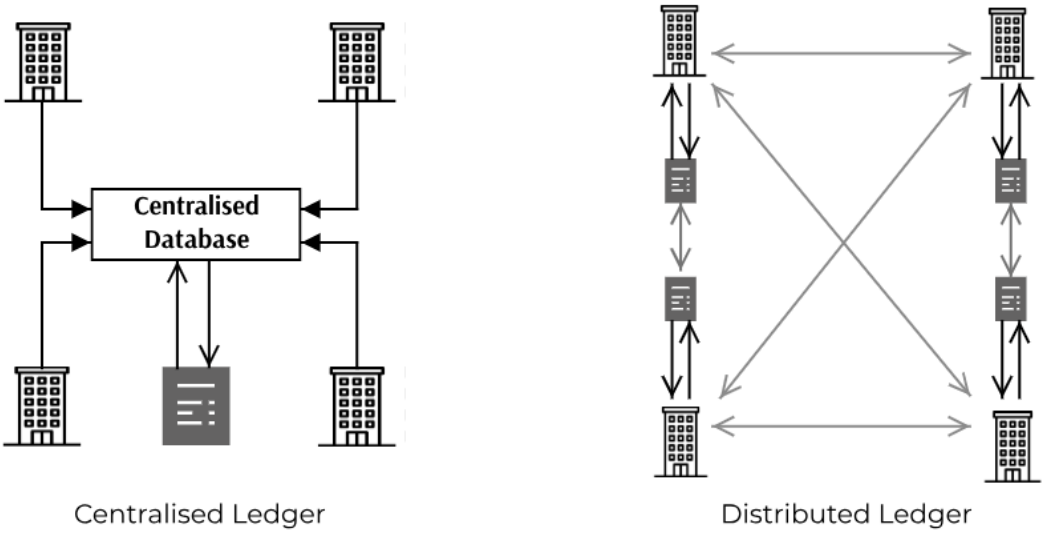

Figure 2: Notions of centralised and decentralised ledger

1.2. Blockchain

The above idea of distributed ledgers has given birth to the Blockchain Technology. Blockchain is a decentralised ledger containing transaction information across a peer-to-peer network. The structure of a blockchain ledger is a computer science marvel as it combines two fundamental data structures - linked list and hash pointers - to produce an immutable, unanimous, secure, programmable, distributed and decentralised ledger.

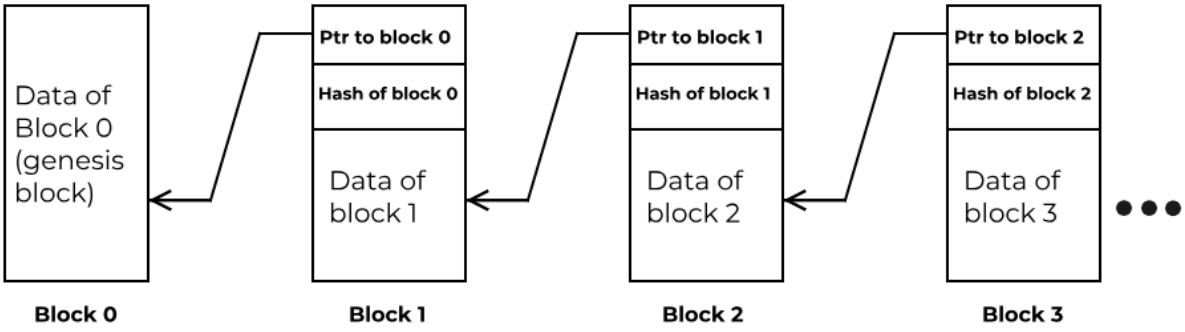

Figure 3: Fundamental structure of Blockchain

It is tamper-evident as evident from Figure 3 because each block stores a hash pointer to the previous block. So, tampering with any block results in the mismatch of its new hash value with its hash value stored in the succeeding block, thereby disrupting the entire chain.
A block contains a list of transactions in it. Any node could broadcast a transaction in the blockchain network by communicating with its peer nodes (who further communicate the transaction to the entire chain). Once the transaction information is received by all the nodes, its validation takes place. A 'consensus' is achieved on whether the transaction is valid or not. If the transaction is found valid, then it is appended onto the ledger.

This notion of a decentralised ledger provides some essential properties that the users are robbed of in the centralised means of achieving things [1]. These are:-

1. **Public Verifiability:** to allow all nodes to validate any state of the ledger.
2. **Transparency:** to ensure that all the data uploaded onto the ledger is not corrupt.
3. **Privacy:** to maintain the secure functioning of the network.
4. **Integrity:** to protect the information from unauthorised tampering.

But blockchains too are susceptible to vulnerabilities like 51% attacks (more than half of the nodes in the network are malicious), byzantine faults and more. As a thumb rule, the more tolerant the consensus mechanism is to these attacks, the more it is preferred. The following section categorically lists significant parameters to consider while choosing/developing a consensus algorithm.

2. Parameters of judging a consensus algorithm

Parameters for judging a blockchain consensus algorithm may be diverse and might certainly be more extensive than what this paper presents. However, the following is a compilation of seven parameters that prove to be effective in carrying out judgement of any decentralised consensus mechanism:-

2.1 Energy Consumption

The algorithm should either minimise the wastage of resources (like computational power) or channelise it to curb a real problem. Previously it was thought that the trade-off between energy consumption and effectiveness of the blockchain network could never be reduced. As a popular reference, Bitcoin's consensus mechanism (Proof-of-Work) is known to be very effective in maintaining decentralisation but it severely wastes computation power of the participating nodes [2]. Not only this but it was thought that this 'energy wastage' is an inevitable cost to bear decentralisation. The graph in Figure 4 is a measure of energy consumed by different transaction architectures.

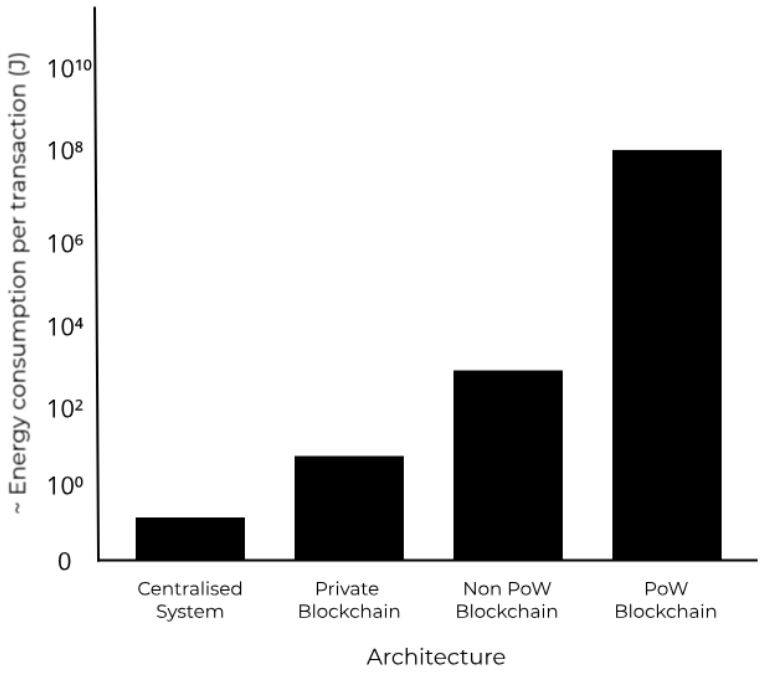

Figure 4: Energy consumed by different transaction architectures

However, newer and more targeted consensus mechanisms suggest that this gap could certainly be bridged depending on the application and its needs.

2.2. Tolerated power of adversary

This factor determines how easy it is for an adversary to disrupt or have influence over the blockchain network. While ideal protocols would account for infinite adversary tolerance, all practical protocols are not completely devoid of this factor. The term “power” could have various interpretations depending on the design of the consensus algorithm. Eg. For Proof-of-Work consensus, “power” may correspond to computation power, for Ripple it may just mean the number of faulty nodes while for BFT, it may imply voting power [3].

2.3. Byzantine fault tolerance

The concept of Byzantine fault tolerance arises from the popular ‘Byzantine’s Generals Problem’, which has remained unsolved for thousands of years. To understand this factor, we need to take a look at the Byzantine Generals’ Problem first.

The Byzantine Generals’ Problem

The Byzantine empire has its army divided into a few groups, each led by one general. These groups surround a city on which an attack is intended. The attack would be successful only if all the generals command their groups to attack at the same time. Instead of attacking, the generals may also choose to retreat. Ideally, all generals must arrive at a decision in unison; which requires effective communication between them. Figure 5 provides a diagrammatic representation of this scenario.

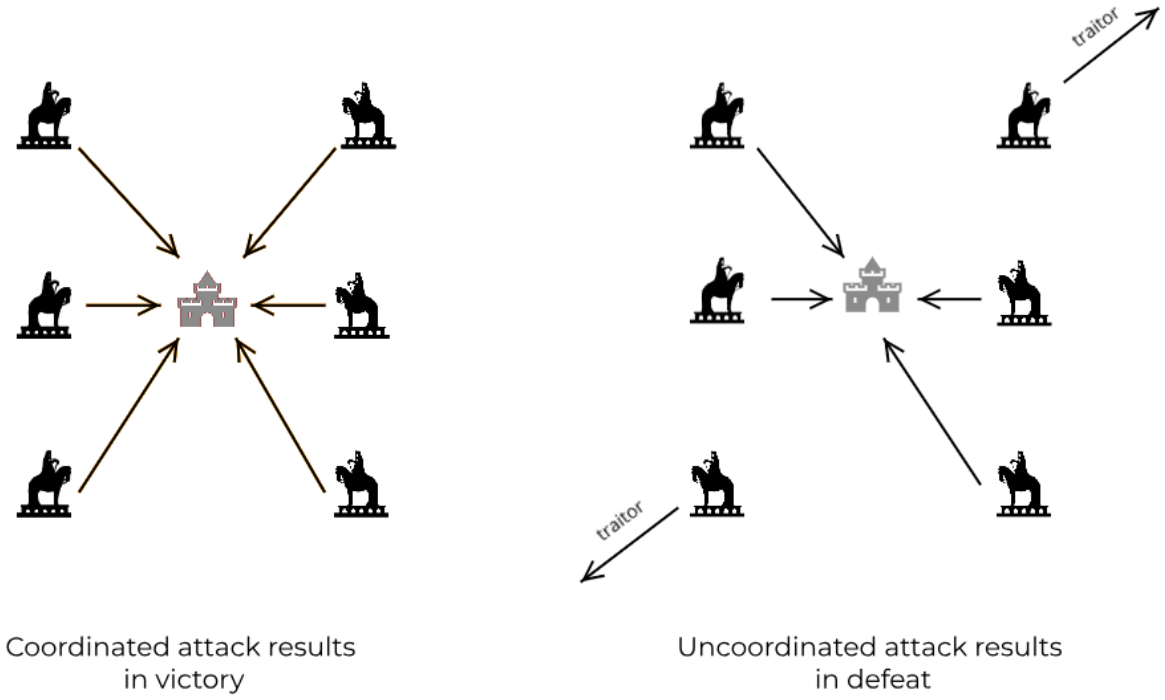

Figure 5: Display of coordinated and uncoordinated attacks by the Byzantine army on the city

They may ask their messengers to send each other messages but the problem is capturing of the messenger, tampering of the original message by the opponent, or worse, dishonesty of the generals. Reaching consensus on whether to attack or retreat becomes a problem in this unfaithful environment.

This problem can also be extended in blockchain where the generals can be thought of as nodes and the messages as transactions. The problem of tampering of messages could be eliminated as blockchain assumes a distributed ledger, enabling all nodes to validate transactions.

While the problem of dishonesty of nodes cannot be completely curbed, different consensus algorithms may develop a degree of resistance towards it. This degree of resistance is called the algorithm's Byzantine fault tolerance [4].

2.4. Crash Fault Tolerance

Crash fault is a name given to the condition when a number of nodes in the blockchain network crash, that is, halt or disconnect from the network. A decentralised system's crash fault tolerance is its ability to maintain the same state of truth on the system even if some nodes crash.

2.5. Verification Speed

The speed at which transactions could be validated by different nodes in the blockchain network is its verification speed. The faster the verification speed, the more desirable the consensus algorithm.

2.6. Throughput

Throughput is defined as the number of valid transactions per second. The higher the throughput, the better the consensus algorithm [5].

2.7. Scalability

Scalability is the ability of the blockchain network to cater to increasing load as well as accommodate increasing number of nodes. The higher the scalability, the better the consensus algorithm [6].

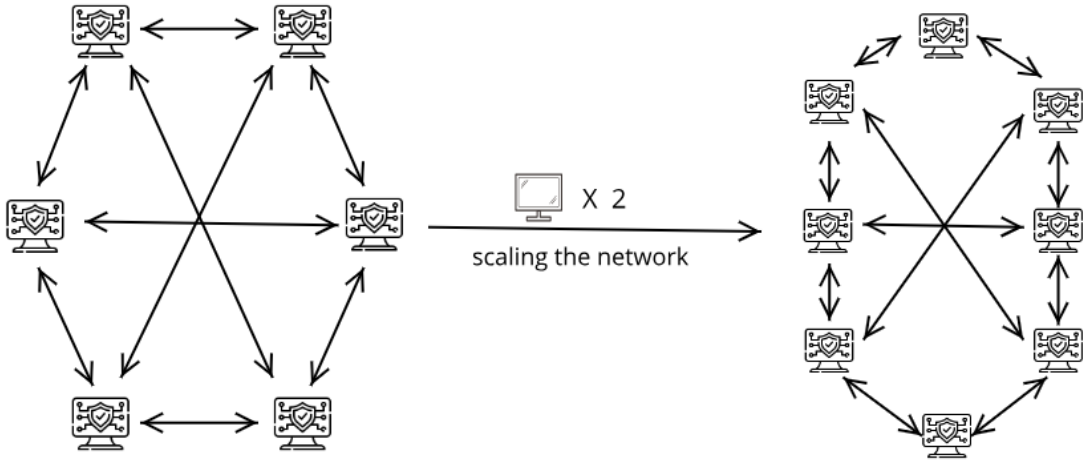

Figure 6: Scalability of blockchain

3. Blockchain Consensus Mechanisms

Transactions proposed in the blockchain network are deemed valid or invalid by achieving a consensus. While there have been ideas like randomly proposing nodes to 'mine' the next block, a practical implementation of such systems would make it vulnerable to malicious nodes, as those might get randomly picked, which would be highly undesirable for the network. There are numerous mechanisms of achieving consensus in the blockchain network but some of the most prominent ones make up for the scope of this study.

3.1. Proof-of-Work (PoW)

Proof of Work protocol achieves consensus among nodes using '*hash puzzles*'. In order to add a block to the blockchain, the node that proposes it is required to find a "*nonce*", such that when it is concatenated with the previous hash, the hash of the whole block results in a particular type of output, which has a very large output space (Figure 7). If the hash function is *puzzle-friendly*, meaning there isn't an easily distinguishable relationship between the input and the output, then the only viable way to solve it is to find the golden nonce by hit-and-trial method. The target space of the golden nonce is just a tiny fraction of the output space of the hash which makes it extremely challenging to solve [7].

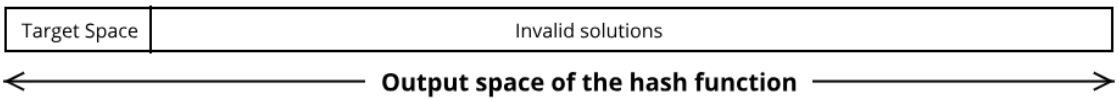

Figure 7: Output space of a challenging hash function

This notion of hash puzzles completely does away with the requirement to pick a random node to propose a block. Instead, all these nodes (called *miners*) compete individually to solve these hash puzzles at the same time. This process is called *mining*. Once in a while, one of these nodes would get lucky and find the golden nonce. That node gets to propose the next block. This mechanism makes the PoW protocol completely decentralised.

But the PoW mechanism comes at a huge computational cost, so much so that the miners require

exceptionally expensive hardware to participate in it. So, theoretically, anyone can take part in the mining process but practically the mining power remains concentrated in a mining ecosystem. The PoW consensus mechanism is also susceptible to 51% attackers.
Applications: Bitcoin

3.2 Proof-of-Stake (PoS)

Proof-of-Stake (PoS) is the most popular consensus algorithm after Proof-of-Work. Instead of having the users find the nonce in an infinite space, Proof-of-Stake functions by making the users prove their ownership of the currency in question (Figure 8). This accounts for Proof-of-Stake functioning on significantly less energy than Proof-of-Work. This protocol links the honesty of the node to the amount of currency it holds. The only criticism of this method is linked to the '*Matthew effect*' or the *rich-get-richer* syndrome.

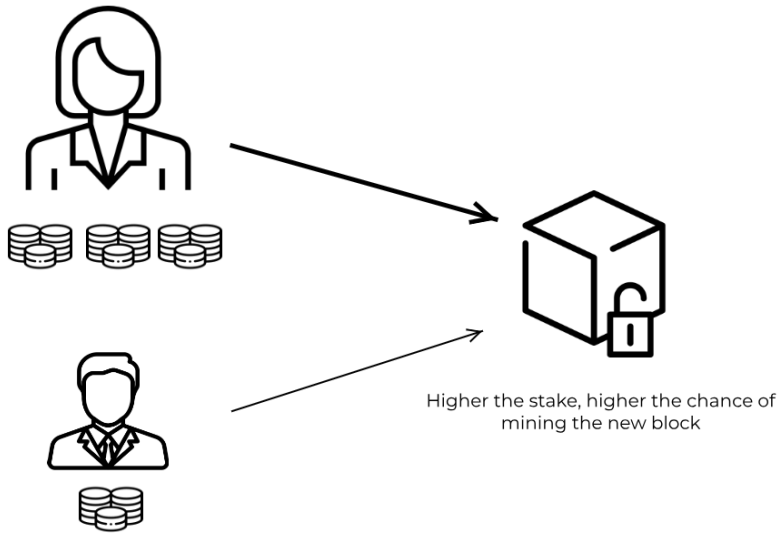

Figure 8: Higher the stake, higher the chance to become the miner

PoS participants having a greater coin age, or the product of network tokens and holding time, have a better chance of being chosen. For attributing the mining power, it employs a formula that considers the lowest hash value and the *stake* size of the miner. A PoS miner can only mine a certain percentage of transactions which is based on their ownership share. A miner who owns 7% of the coins available, for example, may theoretically only mine 7% of the blocks [8].

Applications: Cardano, Algorand

3.3. Practical Byzantine Fault Tolerance (PBFT)

PBFT provides a Byzantine State Machine Replication which tolerates Byzantine faults by assuming there are independent node failures and manipulated messages sent through specific nodes [9]. However, it only works if the number of malicious nodes is not greater than or equal to *one-third of all nodes* in the system in a given vulnerability window. Nodes in PBFT are ordered sequentially with one node being the *primary node* and others being the *backups*. There are majorly 3 phases in a PBFT consensus round - *Pre-prepare*, *Prepare* and *Commit*. The algorithm begins with the client first requesting a primary node (leader) to invoke some service operation. Following this, the commencement of the pre-prepare phase takes place. The primary node sends pre-prepare message to nodes, who can only accept it so long as they are not faulty. After validating this message, the nodes move on to the prepare phase. Here, the nodes send prepare messages to other nodes, who again accept it so long as they are not faulty. After the nodes are prepared, they send out commit messages. Upon receiving x+1 valid commit messages, nodes execute the request and reply to the client. The client waits for x+1 replies with the same result, where x denotes the maximum number of potentially faulty nodes (or byzantine faults). Figure 9 illustrates this**.**

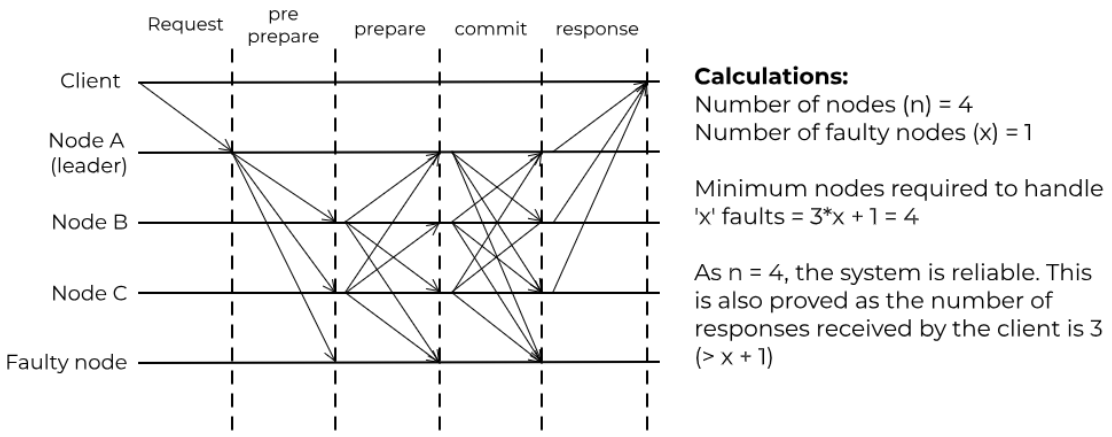

Figure 9: Phases of PBFT algorithm

The leader node could be changed using the *View Change protocol* during every view (consensus round). A supermajority of honest nodes may replace the leader if they find it faulty. PBFT

doesn't tend to take as much toll on energy as a mechanism like Proof of Work. So, it conserves computational power. Every node in PBFT gets the chance to be incentivised (unlike mechanisms like PoW). The scalability of PBFT though, is not very encouraging as the peer to peer reliance may incur huge costs. It is also more vulnerable to *Sybil attacks*.
Applications: Hyperledger Fabric, Zilliqa

3.4. Delegated Proof of Stake (DPoS)

DPoS aims to reach a consensus on what data should be added to the blockchain, through an election process [10]. Instead of all the nodes taking part in the consensus process, DPoS relies on some *delegate nodes* for validating transactions. These delegate nodes can be imagined as 'representatives of all the nodes' and are voted into power by other nodes (called *token holders*). The delegates may require a certain amount of funds (as a stake) to show that they are indeed worthy of being representatives. They are incentivised to act honestly by the risk of loss of income.
DPoS algorithm reduces energy consumption as well. As fewer nodes are needed to validate the transactions, the DPoS consensus mechanism accounts for higher throughput. The greater efficiency produced by the DPoS algorithm indeed takes a toll on decentralisation. It could lead to the concentration of voting power. The DPoS algorithm essentially works as good as a real-world democracy, so it is better to have more voters (token holders).
Applications: BitShares, EOS

3.5. Ripple

Unlike most cryptocurrencies that aim to serve the individual, Ripple tends to serve banks aiding them in lowering transaction costs. The Ripple algorithm is based on a network of validators called *RippleNet*. It works on the protocol *RTXP (Ripple Transaction Protocol)*.

Every node in the blockchain keeps a list of nodes called a *UNL (Unique Node List)*, which is the only subnetwork that the node must trust. It is purely up to the validator to decide which nodes should they include in their UNL. In a consensus cycle, each node creates a *candidate set* of all valid transactions it has seen, which can include new transactions from clients as well as old transactions from the previous consensus period [11].

Every node combines its candidate set with that of its UNL peers, votes for the validity of each transaction in the combined set, and sends votes to its UNL peers. After collecting votes from its peers, each node discards a transaction if its rating falls below a certain threshold from its candidate collection. The transactions that are discarded cannot be replicated in the next consensus cycle. After continuing this process multiple times, every validator pushes the lasting transactions onto the ledger.
Application: XRP, xRapid, xVia, xCurrent

3.6. Tendermint

In the blockchain model, Tendermint is a secure *state-machine replication* algorithm. It starts with a group of validators who are identified by their public keys and are responsible for keeping a complete copy of the replicated state as well as proposing and voting on new blocks [12].

Each block is given an increasing index, or *height*, such that there is only one valid block at each height in a valid blockchain. Validators at each height take turns in proposing new blocks, with no more than one valid proposer per round. For choosing the proposer for the next round, Tendermint employs a round-robin mechanism with the probability of selection of a validator being proportional to its voting power.

Once the proposer is chosen, they propose a new block. In the first round, also called the "prevote" round, the fellow validators may choose to prevote the block or prevote nill. If the block acquires positive votes from more than two-thirds of the validators, the algorithm goes to the next phase called "pre-commit". The prevoted validators may pre-commit the block if they see that the block has gained more than two-thirds of the prevotes. The validators who had prevoted nill may not participate in this step. If the block acquires more than two-thirds of pre commits then the block is committed, otherwise not. Figure 10 shows the State Transition Network (STN) of the algorithm.

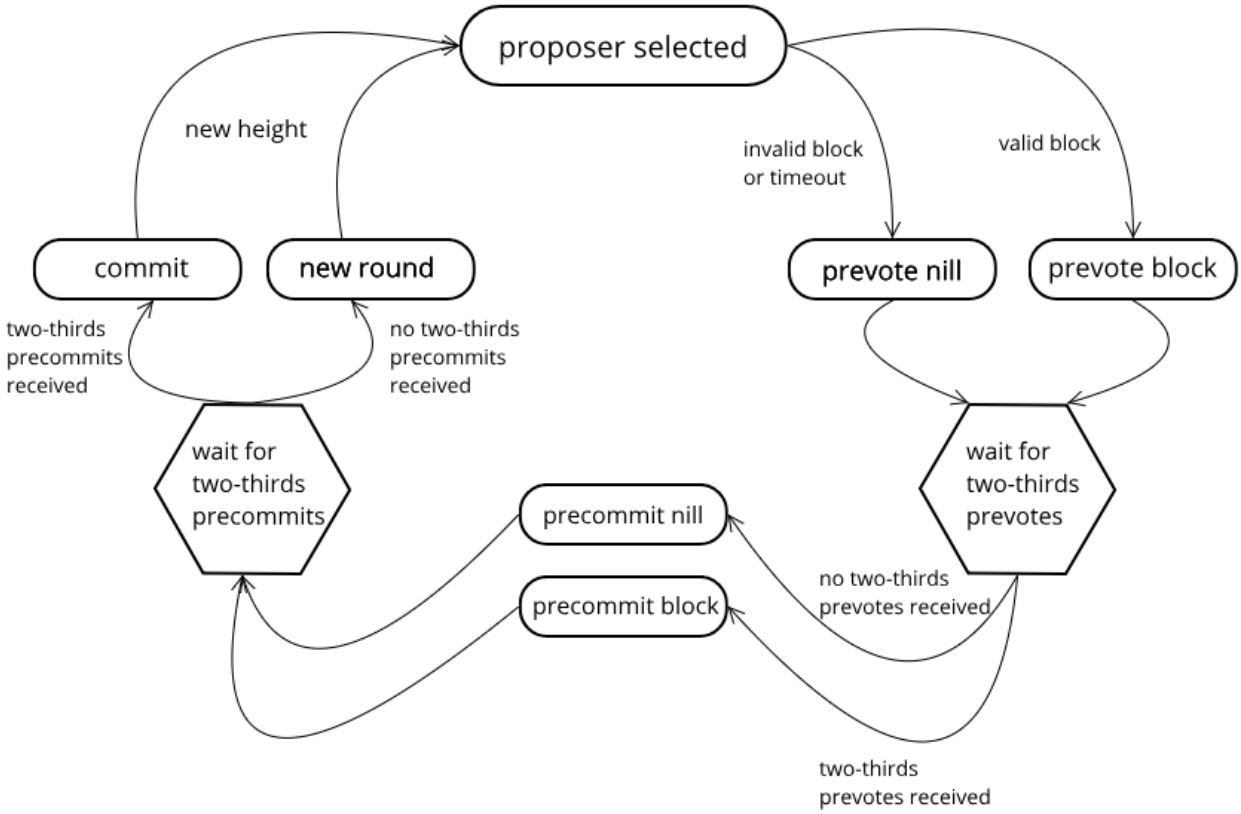

Figure 10: State Transition Network for Tendermint's consensus

Owing to network asynchrony, it may take several rounds to commit a block at a given height, and the network may come to a complete halt if one-third or more validators are partitioned or offline. Until committing a proposed block, validators vote twice on it, and a clear locking procedure prohibits every malicious alliance with less than one-third of the validators from jeopardising security.

Tendermint aspires to be an open and high-performance Byzantine Fault Tolerant infrastructure that can handle anyone from the private sector to global currencies as the world continues to tread towards decentralisation but as of now due to limited scalability, it happens to be majorly operational in private blockchains.

Application: Tendermint

3.7. Delegated Byzantine Fault Tolerance (DBFT)

The DBFT algorithm is based on a voting procedure and is applied similar to the DPoS algorithm. The consensus nodes are broadly categorised as - Speaker and Delegate nodes. Delegate nodes are voted into power by the network nodes to represent them. Ordinary nodes are all the members of the network that keep the tokens. These nodes are in charge of executing transactions in the protocol and voting in real-time for consensus nodes. A randomly drawn speaker from the group of delegates proposes the next block. For a transaction to be successful, 66% of all delegates need to approve the transactions [13].

The ordinary nodes of the network overlook the procedure by identifying ingenuine delegates, pushing them out of power in the next voting round.

Applications: NEO

3.8. Stellar Consensus Protocol

The Stellar consensus mechanism is a "federated Byzantine agreement system" that enables decentralised computing networks to reach a consensus on a decision in a timely manner. It aims to achieve consensus in a blockchain network with the help of two sub-protocols - Nomination protocol and Ballot Protocol. Rounds of federated voting on "nominees" are conducted by nodes. A federated voting round entails a node voting process for a slot. This is followed by the node listening to its peers' votes until it chooses the one it will "accept"; then the node searches out a "*quorum*" of peers who agree with the argument, which "confirms" the assertion. A quorum is just an overlapping of "*quorum slices*", which is essentially a list of trusted nodes maintained by each node. If a node has confirmed one or more candidates, it attempts to prepare a ballot through additional rounds of federated voting. As soon as a node confirms that a ballot has been loaded, it attempts to "commit" the ballot via additional rounds of federated voting. If a node has confirmed that a ballot has been committed, it will externalise the value present in that ballot and use it as the consensus result.

3.9. Proof of Importance (PoI)

The consensus algorithm of PoI decides which nodes are entitled to append a block onto the ledger based on their value scale. The nodes are aggregated using this algorithm, which is built on a transaction graph analysis. Priority is allocated to the most significant nodes using this form,

which is dependent on the hash computation power of those nodes [14]. This algorithm is similar to the Proof-of-Stake consensus except it does not calculate the stake only based on the amount of cryptocurrency the validator holds. The stake is decided on a more rigorous importance score. To understand this, it might be helpful to consider the example of XEM cryptocurrency which is based on the NEM network employing PoI consensus. An account must have at least 10,000 vested XEM to be considered for an "Importance Calculation". Importance is measured depending on the sum of vested XEM owned, the account's rank throughout the network, a weighing factor based on the account's topological position, and two other constants decided by the NEM network, given eligibility. Each NEM account has a XEM balance that is divided into two parts: *vested* and *unvested*. Whenever a new XEM is received, it is added to the account's unvested balance. Per 1440 blocks, one-tenth of each account's unvested balance is transferred to the vested portion. Furthermore, when an account sends XEM, it is deducted from both vested and unvested accounts to maintain the same vested to unvested ratio [15].

Applications: XEM (NEM)

3.10. Proof of Luck (PoL)

In this algorithm, the validator node is chosen based on the highest luck value. Each node starts adding a new block to its current chain and then assigns a purely random value between 0 and 1 to the block header [16]. Due to this, the algorithm wastes a lot of computing power. On the contrary, however, it makes it extremely resistant to double-spending attacks as the attacker would have to be very lucky in order to pull it off. Another issue pertaining to this protocol is that if the node's clock is not aligned with the network clock, it may lose its chance of being lucky.

3.11. Raft

Raft is a leader-based algorithm that relies on leader election as its key consensus protocol component. In a Raft-based scheme, ledger entries only flow from the leader to the other servers in one direction. When more than half of the nodes are no longer under the jurisdiction of the current leader, the network is broken. If the network splits, the blockchain network will begin the process of electing a new leader.

A node could be in one of these states at any time: *leader*, *follower*, or *candidate*. The Raft

algorithm divides time into finite-duration terms. Consecutive integers are used to number the terms. Each term starts with a leader election, in which one or more candidates compete for the position of the leader. If a nominee is elected, it becomes the leader for the remainder of the term. When a follower hasn't heard from the leader for a certain amount of time, it becomes a candidate. To become a leader, the candidate then solicits votes from other nodes. The vote request is responded to by other nodes [17]. The candidate will become the leader if a majority of the nodes vote for them. This is known as the *leader election method*. To retain authority, the leader sends out heartbeats to all of its supporters on a regular basis. During this, all transactions must go through the leader. Each transaction is recorded in the ledger of the node. The leader, in particular, replicates the obtained transaction to the followers first. The submission is already uncommitted and in a volatile state at this moment. Figure 11 shows the State Transition Network of this protocol.

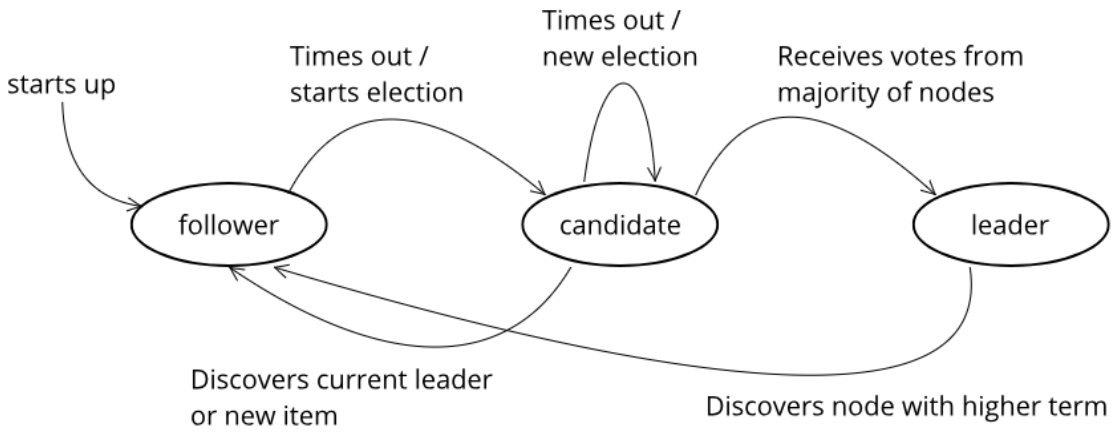

Figure 11: State transition diagram for Raft consensus

4. Parameterised Comparison

Table: *Comparison of the above-listed consensus algorithms against the mentioned parameters*

The value of unreferenced fields are either obvious or derived

Algorithm	Property						
	Energy Saving	**Tolerated Power of Adversary**	**Byzantine fault tolerance**	**Crash fault tolerance**	**Verification Speed(s)**	**Through-put**	**Scalability**
PoW	No [15]	< 25% computing power [7]	50% [18]	50% [18]	>100s [18]	<100 [18]	Strong [18]
PoS	Partial [15]	<51% stake [7]	50% [18]	50% [18]	<100s [18]	<1000 [18]	Strong [18]
PBFT	Yes [15]	< 33.3% replicas [7]	33% [18]	33% [18]	<10s [18]	<2000 [18]	Weak [18]
DPoS	Partial [15]	< 51% validators [7]	50% [18]	50% [18]	<100s [18]	<1000 [18]	Partial [18]
Ripple	Yes [15]	<20% faulty nodes [7]	<20% [19]	N/A	<10 [5]	<1500 [21]	strong [22]
Tendermint	Yes [7]	<33.3% stake value [11]	stake value	33.33% [23]	<20s [12]	>1000 [12]	strong [24]
DBFT	Yes [15]	<33.3% replicas [15]	33.33%	uses a two phase protocol [25,26]	<=20 [20]	<1000 [13]	Strong [27]
SCP	Yes [15]	Variable [15]	N/A	N/A	5-6 [20]	4000 [20]	Strong
PoI	Yes [14]	<50% importance [15]	N/A [14]	N/A [14]	N/A [14]	N/A [14]	Strong [14]
PoL	Yes [14]	<50% of processing power [14]	N/A [14]	N/A [14]	>15 [14]	N/A [14]	Strong [14]
Raft	Yes [14]	50% of nodes [14]	Intolerant [17]	50% [18]	<10s [18]	>10k [18]	Weak [14]

Explanation and Deductions from the table

In public blockchains - where anyone could join the network and there is no trust between different nodes, ***safety***, ***fault tolerance*** and ***scalability*** of the network are of utmost importance. Without it, the blockchain would not sustain itself, let alone thrive. So, algorithms like Proof-of-Work and Proof-of-Stake remain good choices because they offer the maximum decentralisation and support for the above mentioned factors, even if it comes at the cost of high computational power. PoW and PoS have extremely slow speeds of transaction which make them unideal for applications demanding faster consensus.

In private blockchains however, where participants are particularly permitted to take part in the blockchain, the nodes are bound by strict obligations of their organisation. So, here safety and fault tolerance and to some extent scalability could be compromised. As an equal trade-off, the nodes could then enjoy ***greater transaction speeds***. Protocols like **PBFT** and **Raft** are ideal for such purposes. This tradeoff has looked more appealing to the hyperledger fabric [28] as well as ethereum [29], both of which are looking to develop consortium frameworks to build such blockchains. Decentralisation is the main idea behind blockchain and achieving it as mentioned earlier comes with its own costs. However, sacrificing some amount of decentralisation can lead to other benefits like increased transaction speeds, as in the case of **PoI** or **DPoS.**
If one is looking for some ironclad methods in terms of fault tolerance and even wishes to enjoy great transaction speeds, protocols like **Ripple**, **SCP**, **Tendermint** could come in handy.

The ***FLP (Fischer Lynch Paterson) impossibility*** in Figure 12 even suggests that a *purely asynchronous* consensus algorithm may only have 2 out of the 3 characteristics - *fault tolerance, liveness, or safety*.

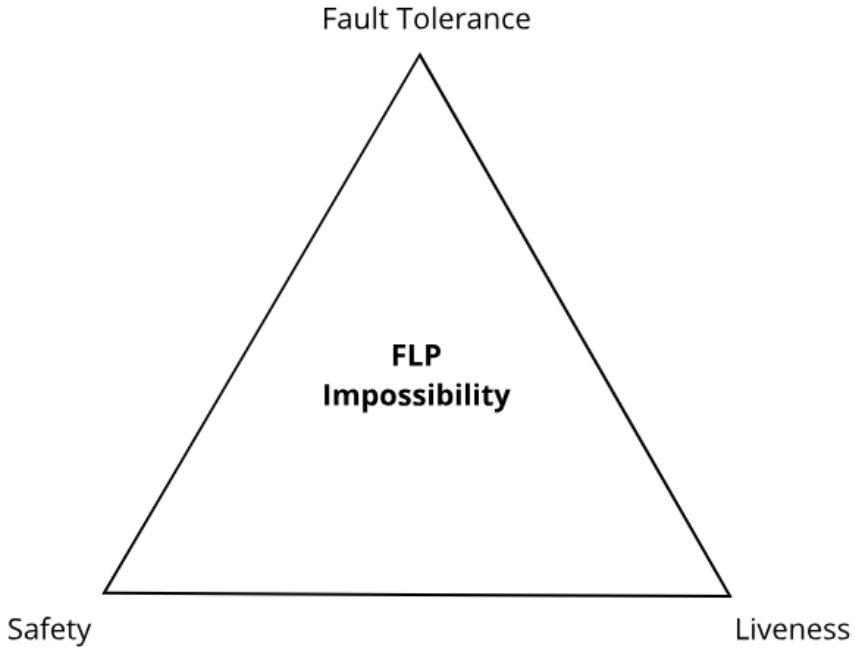

Figure 12: Fischer Lynch Paterson impossibility for asynchronous distributed systems

Fault Tolerance: Ability to withstand honest node failures in the system
Liveness: Guarantee that the ledger will continue to accept future transactions.
Safety: The network must continue to be reliable even if validators do not agree with each other

Algorithms like **PoW** compromise safety to achieve a higher level of *liveness and fault tolerance*. On the contrary algorithms like **PBFT**, **Stellar** and **Tendermint** look to ensure a higher level of *safety and fault tolerance* by sacrificing liveness.

Practically, the idea of using purely asynchronous consensus models is mitigated to ensure that the system repeatedly checks for all the parameters mentioned in FLP impossibility.

To facilitate better design of blockchain networks, relying on FLP impossibility is indeed very helpful in deciding the perfect consensus algorithm, however, it is equally important for the designer to categorise the application's needs based on the factors listed above.

5. Conclusion

Gone are the days when Blockchain networks just had a handful of protocols to achieve consensus. Though Proof of Work and Proof of Stake still remain the most popular consensus mechanisms, it is essential that we question their accountability in diverse applications. As of now, the world seems to be advancing towards different consensus algorithms to tailor the needs of different applications. By the means of this survey, we deduce that although no algorithm could be deemed as the most superior algorithm, it depends on the nature of the application to find the single best consensus protocol - like whether the blockchain is public or private, performance and throughput needs, project scale etc. Nonetheless, this survey shows how much of an impact blockchain-based consensus algorithms are creating to empower decentralisation and to root out the centralised means of going about things.

6. References

[1] Wüst, K., & Gervais, A. (2018, June). Do you need a blockchain?. In 2018 Crypto Valley Conference on Blockchain Technology (CVCBT) (pp. 45-54). IEEE.

[2] Sedlmeir, J., Buhl, H. U., Fridgen, G., & Keller, R. (2020). The energy consumption of blockchain technology: beyond myth. Business & Information Systems Engineering, 62(6), 599-608.

[3] Vukolić, M. (2015, October). The quest for scalable blockchain fabric: Proof-of-work vs. BFT replication. In International workshop on open problems in network security (pp. 112-125). Springer, Cham.

[4] Driscoll, K., Hall, B., Sivencrona, H., & Zumsteg, P. (2003, September). Byzantine fault tolerance, from theory to reality. In International Conference on Computer Safety, Reliability, and Security (pp. 235-248). Springer, Berlin, Heidelberg.

[5] Hao, Y., Li, Y., Dong, X., Fang, L., & Chen, P. (2018, June). Performance analysis of consensus algorithm in private blockchain. In 2018 IEEE Intelligent Vehicles Symposium (IV) (pp. 280-285). IEEE.

[6] Improving performance & scalability of blockchain networks - Wipro. (2019). Wipro.com. https://www.wipro.com/blogs/hitarshi-buch/improving-performance-and-scalability-of-blockchain-networks/#:~:text=Scalability%20of%20blockchain%20networks%20is,of%20nodes%20in%20the%20network.

[7] Zheng, Z., Xie, S., Dai, H. N., Chen, X., & Wang, H. (2018). Blockchain challenges and opportunities: A survey. International Journal of Web and Grid Services, 14(4), 352-375.

[8] Nguyen, C. T., Hoang, D. T., Nguyen, D. N., Niyato, D., Nguyen, H. T., & Dutkiewicz, E. (2019). Proof-of-stake consensus mechanisms for future blockchain networks: fundamentals, applications and opportunities. IEEE Access, 7, 85727-85745.

[9] Castro, M., & Liskov, B. (1999, February). Practical byzantine fault tolerance. In OSDI (Vol. 99, No. 1999, pp. 173-186).

[10] Yang, F., Zhou, W., Wu, Q., Long, R., Xiong, N. N., & Zhou, M. (2019). Delegated proof of stake with downgrade: A secure and efficient blockchain consensus algorithm with downgrade mechanism. IEEE Access, 7, 118541-118555.

[11] Christodoulou, K., Iosif, E., Inglezakis, A., & Themistocleous, M. (2020). Consensus crash testing: exploring Ripple's decentralization degree in adversarial environments. Future Internet, 12(3), 53.

[12] Buchman, E. (2016). Tendermint: Byzantine fault tolerance in the age of blockchains (Doctoral dissertation).

[13] Christofi, G. (2019, October). Study of consensus protocols and improvement of the Delegated Byzantine Fault Tolerance (DBFT) algorithm (Projecte Final de Màster Oficial). UPC, Escola Tècnica Superior d'Enginyeria de Telecomunicació de Barcelona, Departament d'Enginyeria Telemàtica. Retrieved from http://hdl.handle.net/2117/171243

[14] Alsunaidi, S. J., & Alhaidari, F. A. (2019, April). A survey of consensus algorithms for blockchain technology. In 2019 International Conference on Computer and Information Sciences (ICCIS) (pp. 1-6). IEEE.

[15] Bach, L. M., Mihaljevic, B., & Zagar, M. (2018, May). Comparative analysis of blockchain consensus algorithms. In 2018 41st International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO) (pp. 1545-1550). IEEE.

[16] Milutinovic, M., He, W., Wu, H., & Kanwal, M. (2016, December). Proof of luck: An efficient blockchain consensus protocol. In proceedings of the 1st Workshop on System Software for Trusted Execution (pp. 1-6).

[17] Huang, D., Ma, X., & 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.

[18] Mingxiao, D., Xiaofeng, M., Zhe, Z., Xiangwei, W., & Qijun, C. (2017, October). A review on consensus algorithm of blockchain. In 2017 IEEE international conference on systems, man, and cybernetics (SMC) (pp. 2567-2572). IEEE.

[19] Schwartz, D., Youngs, N., & Britto, A. (2014). The ripple protocol consensus algorithm. Ripple Labs Inc White Paper, 5(8), 151.

[20] Görkey, I., El Moussaoui, C., Wijdeveld, V., & Sennema, E. (2020). Comparative Study of Byzantine Fault Tolerant Consensus Algorithms on Permissioned Blockchains.

[21] Han, R., Gramoli, V., & Xu, X. (2018, February). Evaluating blockchains for IoT. In 2018 9Th IFIP international conference on new technologies, mobility and security (NTMS) (pp. 1-5). IEEE.

[22] The Most (Demonstrably) Scalable Blockchain | Ripple. (2017, October 2). Ripple. https://ripple.com/insights/demonstrably-scalable-blockchain/#:~:text=The%20XRP%20Ledger%20has%20been,per%20second%20on%20commodity%20hardware.

[23] What is Tendermint | Tendermint Core. (2021). Tendermint.com. https://docs.tendermint.com/master/introduction/what-is-tendermint.html

[24] Kajpust, D. (2018, September 24). Blockchain Scaling Solutions: Cosmos and Plasma - Tendermint Blog - Medium. Medium; Tendermint Blog. https://medium.com/tendermint/blockchain-scaling-solutions-cosmos-and-plasma-b5ee09456f80

[25] Wang, Q., Yu, J., Peng, Z., Bui, V. C., Chen, S., Ding, Y., & Xiang, Y. (2020, February). Security Analysis on dBFT protocol of NEO. In International Conference on Financial Cryptography and Data Security (pp. 20-31). Springer, Cham.

[26] Tanenbaum, A. S., & van Steen, M. (2002). Fault tolerance. Distributed systems: principles and paradigms.

[27] Crain, T., Gramoli, V., Larrea, M., & Raynal, M. (2018, November). Dbft: Efficient leaderless byzantine consensus and its application to blockchains. In 2018 IEEE 17th International Symposium on Network Computing and Applications (NCA) (pp. 1-8). IEEE.

[28] Hyperledger – Open Source Blockchain Technologies. (2021, May 19). Hyperledger. https://www.hyperledger.org/

[29] ethereum. (2018, August 22). ethereum/wiki. GitHub. https://github.com/ethereum/wiki/wiki/Consortium-Chain-Development