Consensus protocols as a model of trust in blockchains

Auqib Hamid Lone* and Roohie Naaz Mir

Department of CSE,
NIT Srinagar,
J&K, India
Email: ahl@nitsri.net
Email: naaz309@nitsri.net
*Corresponding author

Abstract: Blockchain is a decentralised, replicated, transparent and immutable data store. Blockchains are best described not as 'trustless', but on the basis of distributed trust: trusting everyone in aggregate. Consensus protocols are the heart and soul of the blockchains as they help in achieving this distributed trust. Blockchains are updated via the consensus protocols that guarantee their consistency and integrity over geographically distributed network nodes. Various algorithms can be applied to achieve a consensus based on the requirements like performance, security, scalability, consistency, and failure redundancy. Creating a global fair decentralised consensus protocol is of prime importance, in order to address above-mentioned requirements sufficiently. This paper focuses on analysing the already proposed consensus protocols adopted by popular blockchain platforms to determine their feasibility and efficiency. Parameters that are critical in evaluating blockchain consensus protocol are also discussed. This paper also analyses the hardness of achieving the fair decentralised trust with proof.

Keywords: blockchain; consensus protocol; Byzantine fault tolerance.

Reference to this paper should be made as follows: Lone, A.H. and Mir, R.N. (2019) 'Consensus protocols as a model of trust in blockchains', *Int. J. Blockchains and Cryptocurrencies*, Vol. 1, No. 1, pp.7–21.

Biographical notes: Auqib Hamid Lone holds a Bachelor's in Information Technology and Engineering with distinction. His interest and passion towards information security took him to a Master's degree and completed MTech in Information Security and Cyber Forensics from the Jamia Hamdard University, New Delhi with a university rank. He is currently a Research Scholar in NIT Srinagar J&K. His areas of interest are cryptography, network security, web application security and digital forensics.

Roohie Naaz Mir is a Professor and Head of the Department Computer Science and Engineering at the National Institute of Technology Srinagar, India. She received her BE (Hons.) in Electrical Engineering from the University of Kashmir, India, ME in Computer Science and Engineering from the IISc Bangalore, India and PhD from the University of Kashmir, India. She is a Fellow of IEI, IETE, senior member of IEEE, and member of IACSIT and IAENG. She is the author of many scientific publications in international journals and conferences. Her current research interests include network security, reconfigurable computing, mobile computing, security and routing in networks.

1 Introduction

Reliability is a critical metric in the design and development of distributed systems such as Bitcoin's blockchain. Replication is an effective and convenient method used to guarantee reliability. Unfortunately, replication brings a new problem: we must guarantee the reliability of replicas. If user updates data on one machine, changes must reflect in all the replicas otherwise it leads to inconsistency. Factors that may lead to inconsistency among replicas include node failures, network partitions, message delays, deliberately malicious nodes, etc. Consensus protocols help in keeping the state consistent in multiply connected replicas.

The concept of building consensus dates back when human beings started living together as a society. In its simplest form consensus is just a way for a diverse group to reach an agreement without conflict. From the past three decades, consensus mechanisms in the computer world have gone from abstract idea to the backbone of the blockchain technology. There is great diversity in the algorithms that can be used to achieve consensus in distributed systems based on the requirements like performance, security, scalability, consistency, and failure redundancy.

Blockchain platforms are broadly classified into two main types – permissionless and permissioned (Padmanabhan, 2018). Permissionless blockchains like Bitcoin and Ethereum are open-ended as they are publicly available for use. Any node can join or leave the network and take part in the consensus process. Permissioned blockchains like Hyperledger Fabric and Multichain are close-ended as clients can only submit transactions, participation in consensus process is restricted to a fixed set of nodes governed by consortium members. The consensus in the permissionless setup is challenging, as there is no restriction on the number of nodes that can join the network and these nodes can be anonymous and untrusted. Early blockchain platforms have been designed to be permissionless. Permissioned blockchain platforms allow only semi-trusted members which are known, verified and registered to be part of the network. There is an utmost need for a consensus protocol that meets real-world application requirements like low latencies, fast transaction finality, high performance, and good scalability. This paper focuses on analysing the already proposed consensus protocols adopted by popular blockchain platforms to determine their feasibility and efficiency. We also provided proof why is it impossible to achieve 100% fair decentralised consensus protocol.

The rest of the paper is organised as follows: Section 2 provides the brief background of consensus mechanisms, Section 3 presents two popular classical consensus mechanisms in distributed systems, Section 4 presents popular consensus mechanisms adopted by major blockchains, Section 5 provide the key features of blockchain consensus protocols. Section 6 discuss the hardness of fair decentralised consensus protocol and provide proof why it is impossible to achieve 100% decentralised consensus and finally Section 7 concludes the paper.

2 Consensus background

Blockchain driven systems are a classic example of a distributed system with shared state (i.e., blockchain) where nodes are geographically distributed and connected via peer to peer network. Achieving consensus in a distributed environment is challenging and has

been an active topic of the research from the past three decades. One common technique for achieving fault tolerance in distributed systems is replication wherein shared state is replicated across all nodes. Shared state is updated according to predefined state transition rules defined by the state machine executed on all the replicas. This technique is called state machine replication. Replication of state ensures the integrity of the state even if one or more nodes fail. In blockchain-based systems shared state is blockchain itself and state transition rules are the rules of blockchain protocol. The consensus in blockchain ensures that all nodes in the network agree upon a consistent global state of the blockchain.

Consensus mechanisms are the backbone of blockchain-based systems. Poor choice of consensus mechanisms can make blockchain vulnerable to data modification recorded on it. Some of the issues that could result due to the failure of consensus mechanisms are listed below (Padmanabhan, 2018):

- Failure or misconfiguration of consensus mechanisms can lead to blockchain fork wherein different nodes converge on different blocks as being part of the blockchain. A blockchain fork leads to inconsistent view on data recorded in blockchain thereby forcing applications to behave in an unpredictable manner.
- Some consensus algorithms may not guarantee the convergence. An example is if the consensus algorithm requires a super-majority vote from a certain number of nodes, failing to reach this number due to various reasons like network failures, etc. may lead to consensus failure.
- Unoptimised design of the consensus algorithm could result in poor performance. Some consensus algorithms require more time under certain conditions for consensus to converge, accordingly, they need to be optimised for better performance.
- The output of consensus algorithms can be manipulated by a single or group of entities if it is not designed to be resilient against Sybil attacks, where one or more nodes can generate millions of identities that they can control.

The applicability and efficiency of consensus protocols are governed by three key properties (Baliga, 2017).

1 *Safety:* A consensus protocol is said to be safe if all nodes produce the same output and the output produced by the nodes is valid according to the rules of the protocol.

2 *Liveness:* A consensus protocol is said to be live if all non-faulty nodes participating in consensus eventually produce a value.

3 *Fault tolerance:* A consensus protocol is said to be fault tolerant if it can recover from the failure of node participating in consensus.

In simpler terms *liveness* guarantees ‘something good eventually happens’ and *safety* guarantees ‘something bad does not happen’. Thus safety and liveness are inseparable properties of fault-tolerant protocols. According to Fischer et al. (1982), in deterministic asynchronous consensus system, it is only possible to have at most two properties among safety, liveness, and fault-tolerance. Distributed consensus protocols compromise on one of the properties to drive blockchain-based systems for achieving desired goals.

Collecting literature to have a longitudinal and representative view of consensus mechanisms is challenging because of its rich diversity. In order to have clear-cut,

unbiased, complete and broader perspective many sources have been explored including major online databases and industry whitepapers. The reason behind exploring major databases is their rich library of journals with high impact factors. The review also takes into account industry whitepapers because most of the new consensus mechanisms are proposed by industries and presented as whitepapers. Collected literature is broadly classified into two categories: One consensus in blockchains and another consensus in systems other than blockchains called classical consensus mechanisms. We consider consensus in classical systems and blockchains separately because there is a significant difference in the working of these two fields. This chapter focuses more on blockchain consensus mechanisms and a sufficient background of classical consensus mechanisms is provided to contextualise their subsequent applications to blockchain.

3 Popular classical consensus protocols

3.1 Paxos

Paxos (Lamport, 2001) is a popular fault-tolerant, asynchronous distributed consensus algorithm. It was first described in 1989 by Lamport and was the first consensus algorithm that had a formal proof of correctness. The algorithm compromises of processes that can propose values. The objective of the algorithm is to ensure only single value from the set of proposed values is chosen and must be learned by everyone. Nodes in Paxos are of three types: proposers, acceptors, and learners. Any system process in the network can take any of these roles at any instant of time. *Proposers* propose values that should eventually be chosen by consensus. *Acceptors* form the consensus and accept the values. *Learners* are a valuable source of information because they learn which value was chosen by each acceptor and therefore the consensus. Acceptors either reject a proposal or agree to it and make promises on what proposals they will accept in the future. This ensures that only the latest set of proposals will be accepted. A system process can take multiple roles in any given implementation of Paxos (Cs.rutgers.edu, 2018).

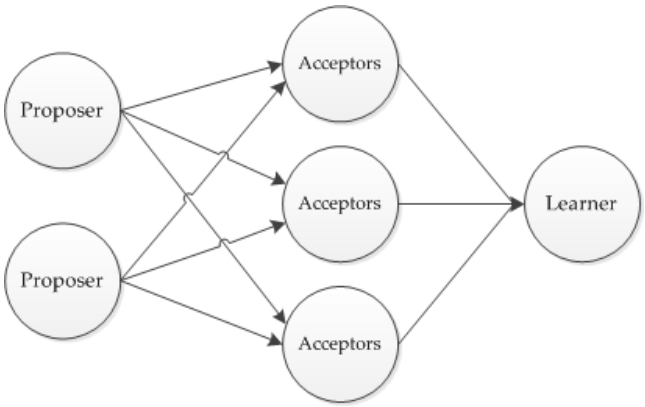
3.1.1 Paxos protocol phases

1 Prepare phase
   - If a node decides to become the leader, it selects a sequence number x and value v_1 to create a proposal $P_1(x, v_1)$. It multicasts this proposal to the acceptors and waits till a majority of nodes respond.
   - Acceptor after receiving the proposal $P_1(x, v_1)$ does the following:
     A Acceptor replies with agree if this was the first proposal that acceptor is going to agree – this is actually a promise now that every future proposal request with a value $< x$ will be rejected.
     B If acceptor has agreed on already made proposals:
       a compare x to the highest sequence number proposal it has already agreed to, say $P_2(y, v_2)$
       b if $x < y$, reply 'reject' along with y
       c if $x > y$, reply 'agree' along with $P_2(y, v_2)$.

2 Accept phase
   - If leader fails to get a majority of acceptors reply or gets majority replies as 'reject', the leader abandons the proposal and may start again.
   - If leader gets the majority of 'agree' as acceptors reply, the leader will also receive the values of proposals they have already accepted. The leader picks any of these values (uses its own, if no values have been accepted) and sends a 'accept request' message with the proposal number and value.
   - On receiving the 'accept request' message acceptors send 'accept' message only if:
     A value is same as any of the previously accepted proposals
     B sequence number is the highest proposal number the acceptor has agreed to.
   - If the leader fails to receive an 'accept' message from a majority of acceptors, leader is abandoned the proposal and starts again. However if the leader does receive an 'accept' from a majority, the protocol can be considered terminated.

After getting proposed value, acceptors notify learners about the majority voted value by forwarding the accepted value to the learners.

Figure 1 Nodes in Paxos



Source: Adapted from Lamport (2001)

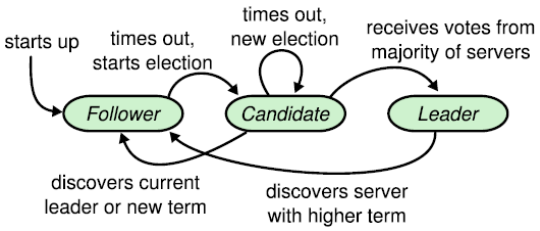
3.2 Raft

The raft (Lamport et al., 2014) is a distributed consensus algorithm. It was designed as an alternative to Paxos and is fairly easy to understand. The objective of raft is to ensure that every server agrees upon the same series of state transitions. It is equivalent to Paxos in fault tolerance and performance. The difference is that it is relatively easier because of its decomposition into independent modules that work together to clearly addresses all major requirements for practical systems. The basic idea behind raft is that: nodes collectively selects a leader, others become followers. The leader is responsible for accepting client requests, state transition and managing the replication of the log across followers. The data flow is from leader to other servers. Raft decomposes consensus into three sub-problems:

- Leader election: On failure of present leader new one is to be elected for consensus to proceed.
- Log replication: The leader keeps the logs of all followers in sync with its own through replication.
- Safety: Once the leader has committed a log entry at a particular index, no other leader can apply a different log entry for that index.

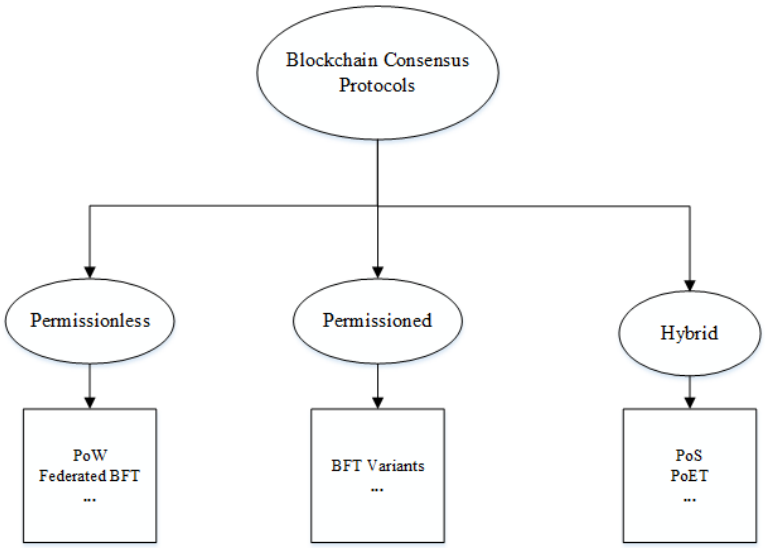
Each node in raft exists in one of the three states: leader, follower, or candidate.

Figure 2 State changes of the server in raft (see online version for colours)



Source: Adapted from Lamport et al. (2014)

Figure 3 Blockchain consensus protocol classification



4 Popular blockchain consensus protocols

4.1 Proof of work

The key innovation in Bitcoin is using proof of work (PoW) to achieve the consensus. As the name suggests it is the proof that enough computational resources have been spent to

build the valid block. PoW involves searching for a value that when hashed, such as with SHA256, the resulting hash begins with a number of zero bits or is less than a specific target. The following equation sums up the PoW requirement in Bitcoin:

$$H\left(N \| P_{hash} \| Tx \| Tx \| \ldots Tx\right) < Target$$

where N is a nonce, P_{hash} is a hash of the previous block, Tx represents transactions in the block, and *Target* is the target network difficulty value. This means that the resulting hash should be less than the target hash value. The difficulty is adjustable, smaller the target, more the number of guesses required to generate the valid hash. The only way to find this nonce is the brute force method. The nodes that generate hashes are called as *miners* and the process is referred to as *mining*.

PoW is used by Bitcoin (Nakamoto, 2008), Bitcoin-NG (Eyal et al., 2016) and many more.

4.2 Proof of stake

One of the common alternatives to PoW is proof of stake (PoS). In this consensus algorithm instead of investing in high computational power in a race to mine the blocks, validators invest in the stake of virtual currency in the blockchain system. The PoS algorithm is designed to select pseudo-randomly validators for block creation, thereby ensuring that no validator can guess its turn in advance. In PoS there is no coin creation (mining), instead, all coins exist from day one and validators (stakeholders) are paid strictly in transaction fees. Naïve PoS algorithms suffer from a problem called nothing-at-stake. Where nodes can vote on multiple blocks therefore, supporting multiple forks to maximise their chances of winning a reward as they have nothing to lose in doing so. One simple way overcome this problem is to require a validator to lock their currency in a virtual wallet. If the validator tries to double sign or fork the system, these coins are slashed. Ethereum's PoS algorithm, called Casper (Zamfir, 2015) is the most advanced PoS algorithm currently available. Peercoin (King and Nadal, 2012) was the first coin to implement PoS. Different variations of PoS are also used by BitShares (Schuh and Larimer, 2015), NXT (Nxt Wiki, 2018) and Tendermint (Kwon, 2014).

4.2.1 Delegated proof of stake

Delegated proof of stake (DPoS) (Larimer, 2014) is a distributed consensus algorithm proposed by Daniel Larimer, build on the top of original PoS algorithm. DPoS speeds up block and transaction creation while maintaining the true decentralised incentive structure. DPoS simplifies the process of achieving consensus as follows:

- *Voting:* In DPoS stakeholders vote to select any number of witnesses for transaction verification and block creation. Voters have the right to delegate their voting power to other users. The weight of votes depends on the stake the user holds.
- *Witnesses:* Users elected as witnesses in the *voting* phase are responsible for validating transactions and creating blocks and in return are awarded associated fee. Thus witnesses in DPoS are like miners of PoW consensus.
- *Delegates:* Users in DPoS can also vote to select group of 'delegates', responsible for governing and maintain overall blockchain.

EOS (Grigg, 2017) is popular implementations of DPoS.

4.3 Proof of elapsed time

Intel has proposed its own alternative consensus protocol called proof of elapsed time (PoET) (Intelledger.github.io, 2018). This protocol works very similar to PoW but consumes far less electricity. At a broader level PoET works in the following way:

- participants in the blockchain network wait a random amount of time
- the first participant to finish waiting gets to be selected as leader for proposing a new block.

For above to work, two requirements need to be verified. First, did the participants really choose a random wait time? Second, did the participant really finish waiting the specified amount of time? PoET relies on a special CPU instruction set called Intel Software Guard Extensions (SGX). SGX allows applications to run trusted code in a protected environment. For PoET, the trusted code is what ensures that these two requirements are satisfied, keeping the lottery fair. PoET approach is based on a guaranteed wait time provide through the trusted execution environment (TEE). PoET algorithm scales to thousands of nodes and will run efficiently on any Intel processor that supports SGX. The only drawback with this approach is the dependency on specialised hardware.

4.4 Byzantine fault tolerant and variants

In distributed systems, Byzantine fault tolerance is the property that system tolerates a certain class of failures known as Byzantine Generals Problem, for which there is an unsolvability proof. Byzantine failures are most dangerous and difficult to deal class of failures because they do not restrict or make any assumption about the behaviour that nodes take at any point in time. Hyperledger fabric is the most popular permissioned blockchain platform of Hyperledger project developed by the Linux Foundation. It provides a flexible architecture with a plug and consensus model. Hyperledger fabric is a consortium blockchain where the group of participants in the consortiums is known and their identities are registered and verified with a central registry service running within the system. It also supports smart contracts on the blockchain, also known as chaincode. Hyperledger fabric currently supports two consensus models – the popular practical Byzantine fault tolerance (PBFT) algorithm and its variation SIEVE.

4.4.1 PBFT

PBFT algorithm proposed by Castro and Liskov (1999) was the first practical solution to the problem of achieving consensus in the presence of Byzantine nodes. PBFT works in asynchronous environments like the internet. PBFT is the first practical algorithm for replicated state machine where state changes are governed with voting by replicas. PBFT algorithm provides both safety and liveliness provided at most $\left\lfloor \frac{n-1}{3} \right\rfloor$ out of on n replicas are simultaneously faulty. PBFT comes with important optimisations, such as signing and encryption of messages exchanged between replicas and clients, reduction in

the size and number of messages exchanged, for the system to be practical in the face of Byzantine faults. PBFT algorithm requires '3*f* + 1' replicas to be able to tolerate *f* failing nodes. PBFT, however, has one disadvantage that it can be scaled to only 20 replicas. As the number replicas goes beyond 20 messaging overhead increases significantly. Three examples of blockchains that rely on the PBFT or its variation for consensus are Hyperledger, Stellar, and Ripple.

4.4.2 SIEVE

SIEVE (GitHub, 2018) is a consensus algorithm inspired by the classic PBFT algorithm by Castro and Liskov (1999). SIEVE increases the value of the original PBFT algorithm by including speculative execution and verification phases for:

1 detecting and filtering out possible non-deterministic requests and establishing the determinism of transactions entering the PBFT three-phase agreement protocol

2 allowing consensus to be run output state of validators, in addition to the consensus on their input state offered by classic PBFT.

SIEVE consensus mechanism is being used by Hyperledger fabric and is designed to handle non-determinism in chaincode execution. Non-determinism in chaincode leads to a different output on execution by different replicas in a distributed network. SIEVE handles non-determinism by initially allowing all operations to run speculatively and then compares the outputs across replicas. If divergence among a small number of replicas is detected by protocol then the diverging values itself are sieved out, else offending operation itself is sieved out.

4.4.3 Cross fault tolerance

Cross fault tolerance (XFT) (Liu et al., 2016) is a novel approach to building reliable and secure distributed systems. XFT is Paxos with Byzantine fault tolerance built-in. The XFT protocol is a new protocol that simplifies the attack model and makes Byzantine fault tolerance feasible and efficient for practical scenarios and is also applicable to the classical state machine replication problem. XFT tolerates both Byzantine as well as non-Byzantine faults as long as the majority of the replicas are correct and can communicate with each other synchronously.

4.5 Federated Byzantine agreement

In federated Byzantine agreement (FBA) systems, each node does not have to be known and verified ahead of time, membership is open, and control is decentralised. Nodes can choose whom they trust. System-wide quorums emerge from decisions made by individual nodes. Ripple and Stellar are two blockchain-based platforms and payment protocols that use variations of the Byzantine fault tolerance consensus models by making them open-ended with respect to node participation the Ripple blockchain pioneered the FBA consensus mechanism. The Stellar blockchain refined this approach even further, adopting the first provably safe FBA protocol.

Ripple and Stellar use their own consensus models that are a derived form of Byzantine fault tolerance modified to include open-ended participation from users.

4.5.1 Ripple consensus protocol algorithm

Ripple consensus protocol algorithm (RCPA) (Schwartz et al., 2014) proposed by Ripple Labs, is a novel approach for building consensus with minimal latency and robustness against Byzantine failures. Ripple protocol requires each node *n* to define a unique node list (UNL). The UNL is a set of other Ripple nodes that are trusted by the given node *n*. For building, consensus *s* only considers votes from the nodes in its UNL, instead of from entire nodes on the network. RCPA works in rounds, each round has the following steps:

- Each node collects transactions in a data structure called 'candidate set' and broadcasts its candidate sets to other nodes in its UNL.
- Each node then amalgamates the candidate sets of all the nodes on its UNL, and votes on the veracity of all transactions.
- Transactions receiving more than minimum percentage of 'yes' are passed to the next round, while other transactions are either discarded or included in the candidate set for the next ledger.
- Last round of consensus requires a minimum of 80% of nodes in UNL agreeing on a transaction. All transactions that pass this requirement are applied to the ledger and the ledger is closed, becoming new-closed ledger.

4.5.2 Stellar consensus protocol

Stellar consensus protocol (SCP) proposed by Maziers (2015) is an open Byzantine agreement protocol resistant to Sybil attacks. SCP solves the problem of building consensus in the presence of Byzantine nodes, without relying on any computational inefficient methods like PoW. SCP, unlike prior Byzantine agreement protocols, does not presuppose any accepted membership list, however, it supports an open membership model that eventually leads to substantial growth like that of the internet. SCP uses the concept of quorums and quorum slices. Quorum is defined as a set of nodes required to reach the agreement. A quorum slice is a subset of a quorum, which can convince another specific node to agree. An individual node can appear on multiple quorum slices. The quorum slices and quorums in SCP are inspired by real-life business relationships existing between various entities thereby leveraging trust that already exists in business models. For building global consensus in the entire systems, quorums have to intersect. Decisions made by the individual nodes lying on the same page leads to global consensus. The consensus protocol works as follows: the first step is a federated voting process, where each node performs its selection of statements (transactions) and will never vote for another statement contradicting its selection. But can accept a different statement if its quorum slices has accepted a different one. The second step is the acceptance step where nodes accept a statement if they have never accepted any statement contradicting the current statement and each node in its v-blocking set (set of nodes one each from a quorum slice to which the current node belongs to) has accepted that statement. The third step is ratification where all members of a quorum agree on a statement. Quorum slices influence one another leading to quorums that agree on the

certain statement and gradually results in the third step of the consensus. Confirmation is the final step where nodes send each other confirmation messages so that all agree upon the final value of the state in the system for system level agreement.

4.5.3 Advantages of FBA

- Open membership and decentralised control.
- No gatekeeper or central authority – individual nodes can decide whom they trust for information.
- The ability to choose whom each node trusts decentralises the network.
- Nodes can have multiple slices.
- Low barrier to entry (anyone can join).
- Robust in the face of failure (one node can go down and the rest of the system will stay intact).
- Nodes can be programmed to trust quorum slices or external sources depending on its performance over time.

Table 1 Comparative analysis of various blockchain projects based on consensus protocol characteristics

Project	*Consensus mechanism*	*Strong consistency*	*DoS resistance*	*Max. achievable throughput in tps*	*Adversary model*	*Transaction confirmation in seconds*
Bitcoin (Nakamoto, 2008)	PoW	No	Full	7	< 25% of computing power	3600
Ethereum (Wood, 2014)	PoW	No	Full	15	< 25% of computing power	360
Bitcoin-NG (Eyal et al., 2016)	PoW	No	Partial	7	< 25% of computing power	< 1 min
Hyperledger (Androulaki et al., 2018)	PBFT	Yes	Full	110 k	< 33.3% of replicas	Immediate
Ripple (XRP) (Schwartz et al., 2014)	BFT variant (RCPA)	*	*	1,500	< 20% of faulty nodes	4
Stellar (Mazieres, 2015)	BFT variant (SCP)	*	*	1,000	variable	2–5

Notes: *Value could not be extracted.
Table data was sourced from community discussions between members of each cryptocurrency's respective community, whitepapers or third-party sites.

5 Key features of blockchain consensus protocols

There is diversity in blockchain consensus mechanisms however following key features are considered for evaluating the overall feasibility of consensus mechanisms (Seibold and Samman, 2016).

1 Overall consensus methodology

Key aspects under this include:

- underlying methodology-permissioned or permissionless
- ownership of nodes
- validator nodes – type and number
- transaction validation by nodes.

2 Performance

Key aspects under this include:

- transaction validation time
- transaction format
- transaction speed
- transaction traffic
- synchronisation-synchronous/asynchronous.

3 Security

Key aspects under this include:

- monitoring of transactional activities
- digital signatures
- key generation and key life-cycle
- preventing signature fraud.

4 Privacy

Key aspects under this include:

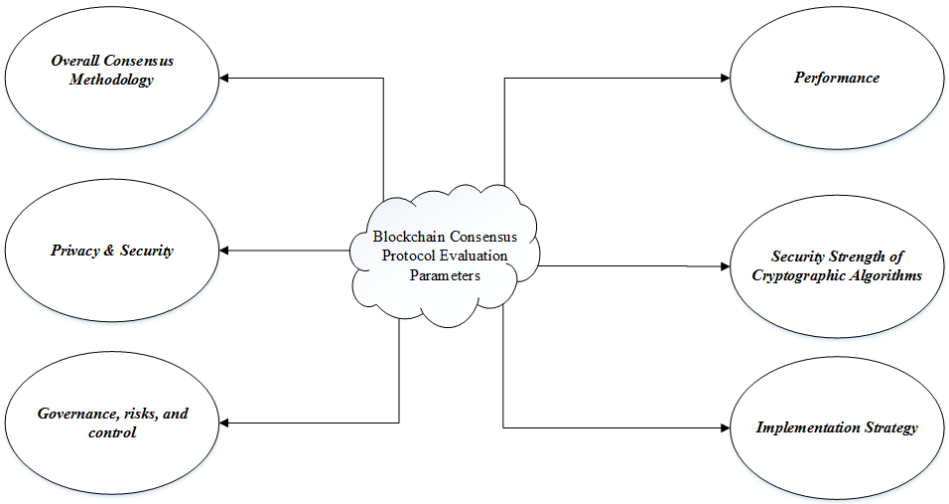
- authentication and authorisation
- transparency and transaction visibility
- confidentiality through encryption.

5 Governance, risks, and control

Key aspects under this include:

- access control and admin privileges
- enforcement of governance/controls
- types of nodes – read or write
- risk mitigation measures.

Figure 4 Blockchain consensus evaluation parameters



6 The hardness of fair decentralised consensus

Achieving full decentralised consensus is hard. Laurie (2017) showed that it is impossible to achieve fair decentralised consensus. Below are the assumptions and simplified proof used by Laurie (2017) to justify the arguments.

6.1 Assumptions and definitions

In simpler terms decentralised means without central controlling authority. In more formal sense decentralised systems consists of a set of participants P of unknown size. Where no member of P can enumerate P (no single participant can control the decision making of P). Also, the members of P are not special in any way.

By decentralised consensus we mean a deterministic algorithm, A with set of possible outcomes (not special any way), O and a vote by every member p of P for some outcome $o_p \in O$, $A(Q) \in O$ where $Q \in P$ and $\ni P' \subset P$ such that $A(P') = A(P)$ (which means it is possible to determine consensus without enumerating P). A is also allowed to fail, i.e., there is no consensus.

Consensus algorithm A is said to be fair if $A(Q) \in \{O_q: q \in Q\}$ where $Q \subseteq P$, i.e., consensus for any subset is voted for by at least one member of that subset. Obviously, that is a weak definition of 'fair' but sufficient for our proof.

Proof: Let us take consideration of one particular member say $t \in$ Q where $Q \subseteq P$. t must assume that there is another subset of participants R of P, i.e., $R \subset P$, and Q and R are disjoint subsets of P, i.e., $Q \cap R = \varnothing$. Because of fairness and the unknowability of P: t must assume R exists where all members have voted for an outcome other than $A(Q)$, which means that $A(R) \neq A(Q)$, because of fairness.

As no one is special among all the participants thus R could also meet the consensus rules, whatever they are. Since no participant is special, t must assume that either $A(Q)$ or $A(R)$ could be the same as $A(P)$ [note that $A(P)$ could be neither], because choosing one would make t special, and hence, t cannot know what $A(P)$ is. This argument applies to all members of P, which implies that no participant can ever know $A(P)$. Thus it is impossible to achieve 100% decentralised consensus algorithm. Researchers are striving hard to achieve the one.

7 Conclusions

From past few years, there has been dramatic growth in blockchain consensus protocols, as a result of which the field has bought diversity in blockchain applications. Liveness and safety are striking properties of consensus protocols which in turn form the heart of blockchains. Permissionless blockchains achieve very strong consensus among a very large number of untrusted participants using computational or memory complexity while compromising on transaction finality and throughput. On the other hand, permissioned blockchains are less scalable but have much higher throughput and transaction finality. As future work, we aim at doing an extended review on blockchain simulators for measuring the performance of various consensus protocols. In this paper, we discussed the background and current scenario of popular consensus protocols. We also provide key characteristic features of blockchain consensus protocols that are extremely important for determining the feasibility of consensus algorithms and aiding in decision making.

References

Androulaki, E. et al. (2018) 'Hyperledger fabric: a distributed operating system for permissioned blockchains', *Proceedings of the Thirteenth EuroSys Conference*, ACM.

Baliga, D.A. (2017) *Understanding Blockchain Consensus Models*.

Castro, M. and Liskov, B. (1999) 'Practical Byzantine fault tolerance', *OSDI*, Vol. 99.

Cs.rutgers.edu (2018) *Consensus* [online] https://www.cs.rutgers.edu/~pxk/417/notes/content/consensus.html (accessed 14 May 2018).

Eyal, I. et al. (2016) 'Bitcoin-NG: a scalable blockchain protocol', *NSDI*.

Fischer, M.J., Lynch, N.A. and Paterson, M.S. (1982) *Impossibility of Distributed Consensus with One Faulty Process*, No. MIT/LCS/TR-282, Massachusetts Inst of Tech Cambridge Lab for Computer Science.

GitHub (2018) [online] https://github.com/diegomasini/hyperledger-fabric/blob/master/docs/FAQ/consensus_FAQ.md (accessed 14 May 2018).

Grigg, I. (2017) *EOS, An Introduction*, Whitepaper [online] http://iang.org/papers/EOS_An_Introduction.pdf (accessed 14 May 2018).

Intelledger.github.io (2018) [online] http://intelledger.github.io/ (accessed 14 May 2018).

King, S. and Nadal, S. (2012) *Peercoin–Secure & Sustainable Cryptocoin*, August [online] https://peercoin.net/whitepaper (accessed 14 May 2018).

Kwon, J. (2014) *Tendermint: Consensus without Mining* [online] https://www.weusecoins.com/assets/pdf/library/Tendermint%20Consensus%20without%20Mining.pdf (accessed 18 May 2017).

Lamport, L. (2001) 'Paxos made simple', *ACM Sigact News*, Vol. 32, No. 4, pp.18–25.

Lamport, L. et al. (2014) 'In search of an understandable consensus algorithm', *ATC '14*, Vol. 22, No. 2, pp.305–320.

Larimer, D. (2014) *Delegated Proof-of-stake (DPoS)*, Bitshare Whitepaper.

Laurie, B. (2017) *Fair Decentralized Consensus is Impossible*, 19 November [online] https://www.links.org/files/dpoi.pdf (accessed 18 May 2018).

Liu, S. et al. (2016) 'XFT: practical fault tolerance beyond crashes', *OSDI*.

Mazieres, D. (2015) *The Stellar Consensus Protocol: A Federated Model for Internet-level Consensus*, Stellar Development Foundation.

Nakamoto, S. (2008) *Bitcoin: A Peer-to-Peer Electronic Cash System* [online] https://bitcoin.org/bitcoin.pdf.

Nxt Wiki (2018) *Whitepaper:Nxt*, Nxtwiki.org [online] https://nxtwiki.org/mediawiki/index.php?title=Whitepaper:Nxt (accessed 14 May 2018).

Padmanabhan, A. (2018) *Types of Blockchains*, Devopedia.org [online] https://devopedia.org/types-of-blockchains (accessed 14 May 2018).

Schuh, F. and Larimer, D. (2015) *BitShares 2.0: Financial Smart Contract Platform*.

Schwartz, D., Youngs, N. and Britto, A. (2014) *The Ripple Protocol Consensus Algorithm*, Ripple Labs Inc. White Paper 5.

Seibold, S. and Samman, G. (2016) *Consensus: Immutable Agreement for the Internet of Value*, KPMG [online] https://assets.kpmg.com/content/dam/kpmg/pdf/2016/06/kpmgblockchain-consensus-mechanism.pdf (accessed 12 May 2018).

Wood, G. (2014) 'Ethereum: a secure decentralised generalised transaction ledger', *Ethereum Project Yellow Paper* [online] http://gavwood.com/paper.pdf.

Zamfir, V. (2015) 'Introducing Casper 'the friendly ghost'', *Ethereum Blog* [online] https://blog.ethereum.org/2015/08/01/introducing-casper-friendly-ghost (accessed 14 May 2018).