Study of Blockchain Based Decentralized Consensus Algorithms

Soumyashree S. Panda[1], Bhabendu Kumar Mohanta[2], Utkalika Satapathy[3], Debasish Jena[4], Debasis Gountia[5], Tapas Kumar Patra[6]
[1,2,3,4]Department of Computer Science & Engineering, IIIT Bhubaneshwar, Odisha, India, 751003
[5]Department of Computer Science & Engineering, IIT Roorkee, Uttarakhand, India, 247667
[6]Department of Electronic Systems Engineering, IISC Bangalore, Karnataka, India, 560012
Email: C117011@iiit-bh.ac.in[1], C116004@iiit-bh.ac.in[2], A117010@iiit-bh.ac.in[3]
debasish@iiit-bh.ac.in[4], dgountia@gmail.com[5], tkpatra@gmail.com[6]

***Abstract*—Blockchain is the backbone technology behind crypto-currency and Bitcoin. By concept, Blockchain is a distributed database where transactions are recorded in an incorruptible and non-modifiable manner. Currently, Blockchain technology is envisioned as a powerful framework for open-access networks, decentralized information processing and sharing systems, etc. This review is motivated due to the lack of an extensive survey on the existing decentralized consensus mechanisms in Blockchain technology. So in this paper, an in-depth review of the distributed consensus mechanisms has been presented. In addition to this, a comparative analysis of the consensus protocols based on the type of Blockchain is also demonstrated.**

***Index Terms*—Blockchain, distributed consensus, Permissionless Blockchain, Permissioned Blockchain.**

I. Introduction

A Blockchain technology based system is a classical distributed system where all the participating entities are geographically scattered but connected through different types of networks. It was formally theorized and implemented by S. Nakamoto, in the year 2008 and 2009, respectively [1]. Traditional transaction management systems require a centralized trusted party who is responsible for confirmation and storage of transactions. This obviously have many issues like cost, privacy,efficiency and security, etc. Decentralization is the fundamental characteristic of Blockchain which can be employed to solve the above issues. Blockchain basically provides a platform where multiple entities who dont trust each other can work or share information in a common platform. Bitcoin, the first application that brought Blockchain into the global picture, is also the first cryptocurrency developed and used. But with progress and in-depth study of Blockchain technology, its application is no more limited to financial sector only. Rather it has gained much popularity in other fields like Government, Technological enterprises, Supply chain, etc [2]. Mainly Blockchain can be used in two different ways Permission-less and Permissioned. Permission-less design which is generally established on an open environment like Bitcoin, Ethereum, allows anyone to join the system as well as allows writing to the shared blocks. Permission-less design also gives equal privilege to all the nodes in case of consensus process. On the contrary, a Permissioned Blockchain design such as Hyperledger fabric is managed by known set of entities and is established in a closed environment. Though all the entities are allowed to perform transactions, only a fixed set of predetermined nodes can take part in the consensus process in a Permissioned Blockchain. Consensus algorithms hold an important part in managing the an efficient and secure Blockchain system. Some of the popular algorithms are Proof of Work (PoW), Proof of Burn (POB), Proof of Stake (PoS), Raft, Practical Byzantine Fault Tolerant (PBFT), Paxos, etc. [3]. Table 1 shows a detailed comparison among these two types of Blockchain configuration.

Until now, the focus is predominantly on the application of Blockchain in different fields. So this paper presents a detailed study of different consensus algorithms and also analyzed their advantages and disadvantages in terms of performance and fault tolerance ability. So the rest of the paper is outlined as follows: Section 2 describes why do we need consensus in a distributed system and the various requirements of consensus algorithms. Section 3 and Section 4 presents various consensus mechanisms for a permission-less and Permissioned Blockchain system consensus respectively. Finally Section 5 gives a comparative analysis of these consensus algorithms and Section 6 concludes the paper.

II. Distributed Consensus Algorithms

The concept of consensus is an engrossing topic in a decentralized or distributed network. Consensus means a procedure to arrive at a common agreement in a decentralized or distributed multi-agent platform. In a conventional distributed system, we apply consensus to ensure reliability which ensures correct execution in presence of faulty individuals and fault tolerance. In addition to this, a distributed consensus mechanism should satisfy certain properties like termination, validation, integrity and agreement. An example of consensus is the state machine replication which is a key aspect of any distributed consensus protocol. For example, if we want to run some kind of distributed protocol over a network, every individual entity runs the current protocol and they store the state of the protocol in different state machines. So the entire execution part of the protocol can be represented as a state machine. Now this state machine needs to be replicated to multiple entities so that every individual entity can reach to

978-1-7281-1895-6/19/$31.00 ©2019 IEEE

TABLE I
COMPARISON OF PERMISSION-LESS AND PERMISSION-ED BLOCKCHAIN.

	Permission-less Blockchain	**Permission-ed Blockchain**
Environment Type	Open	Closed
Participation in Consensus	All nodes	Selected nodes
Identity	Pseudo-Anonymous	Registered Participants
Consensus Type	Lottery based	Voting based
Transaction Processing Speed	Slow	Fast
Consensus Algorithms	Proof of Work, Proof of Stake, Proof of Burn, Proof of Delegated Stake, etc.	Paxos, Practical Byzantine Fault tolerance, Raft, etc.

a common output of the protocol. Achieving consensus can be easy and straight forward for certain architectures under certain scenarios like when

- The entire system is faultless or there is not be any failure in the system, so that every entity can receive the message correctly.
- The system behaves in a synchronous way, i.e., it is expected that you will receive all messages within some predefined time interval.

However achieving consensus can be non-trivial in case of a distributed environment due to the presence of multiple types of failures. Typically in a distributed system, we consider 3 different types of failures:

1) Node Failure: A node suddenly crashes or becomes unavailable in the middle of communication. So we are not expected to receive any message from that particular node. This can hardware or software fault.
2) Partitioned Faults: This type of fault occurs whenever links fail, which results in partition in the network. This can hamper reaching in consensus.
3) Byzantine Faults: This kind of fault is more difficult to handle in a distributed environment. Here the entity starts behaving maliciously. In both the above faults, we can expect the effect on the network, but in this case, it is difficult to guess because it completely depends on how maliciously the entity is behaving and what the entity is doing.

Correctness of a distributed consensus protocol can be characterized by the following two properties:

1) Safety: It ensures that one will never converge to an incorrect state.
2) Liveliness: Every correct value must be accepted eventually.

The conventional distributed consensus protocols are based on either message passing system or shared memory system. The former requires closed environment where every entity needs to know the identity of every other entity in the network [4]. But in case of shared memory approach, a common memory space is used to read and write the shared variables which everyone present in that network can access. But it is not suitable for internet grade computing as we need to put a memory which should be readable and writable by every individual entity in the network. Similarly, message passing approach is not feasible in an open environment like Bitcoin system where anyone can join the network at any time. So under this kind open environments like Bitcoin, how consensus has been achieved, are explained below. Along with that, several other algorithms which are being applied on permission-less models and Permissioned models of Blockchain are described below.

III. CONSENSUS IN PERMISSION-LESS BLOCKCHAIN SYSTEM

As mentioned above, the conventional consensus mechanisms will not be applicable in an open or permission-less Blockchain system. Hence this Section suggests how consensus has been achieved in Bitcoin like open environments along with their shortcomings.

A. Bitcoin Consensus

The main purpose of consensus in Bitcoin is to add a new block to the existing Blockchain. There can be multiple miners in the Bitcoin network and these miners can propose their new blocks based on the transactions they have heard of. It is not necessary that every miner will propose the same block. Generally the miners include those transactions in the new block that they have heard of since the last time a block has been added. The miners also need to ensure that size of the newly proposed block does not exceed certain threshold. We should focus on the following two observations before designing the algorithm:

- Any valid block (a block with all valid transactions) can be accepted even if it is proposed by only one miner.
- The entire protocol can work in rounds. Broadcast the accepted block to the peers and collect the next set of transactions. (Once a valid block has been accepted or the system has reached to a consensus, from that point a round starts. The miner starts getting the transactions and they can find out the transactions that are not already committed in the Blockchain. Based on that they can propose a new block.)

Based on the above two observations, we can design solutions so that

- Every miner independently tries to solve a problem.
- The block is accepted for the miner who can prove first that the challenge has been solved.

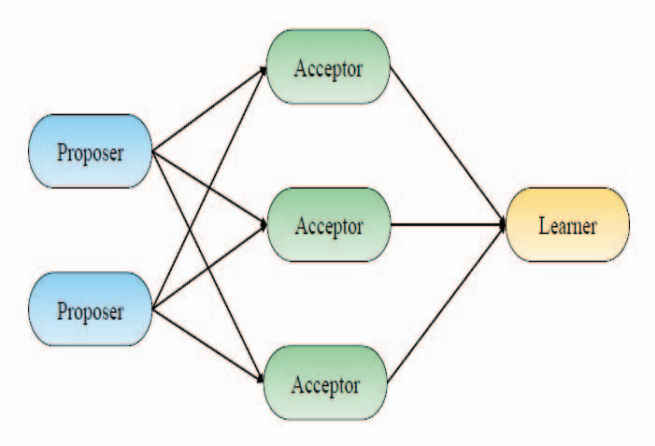

Fig. 1. Types of nodes in Paxos.

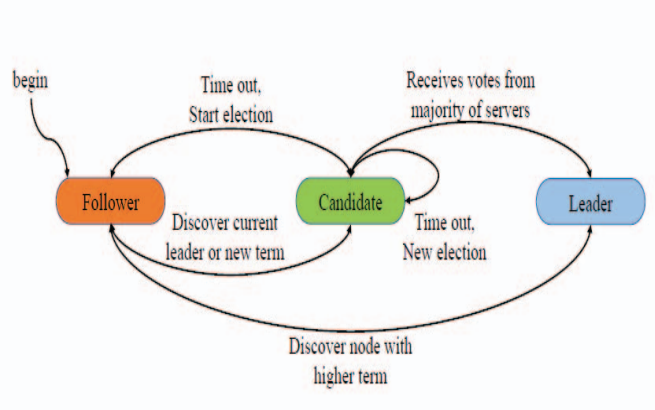

Fig. 2. State change model for Raft algorithm.

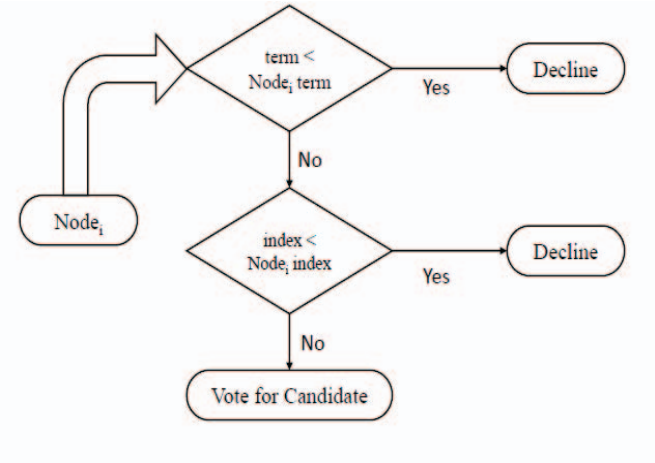

Fig. 3. Leader election process for Raft algorithm.

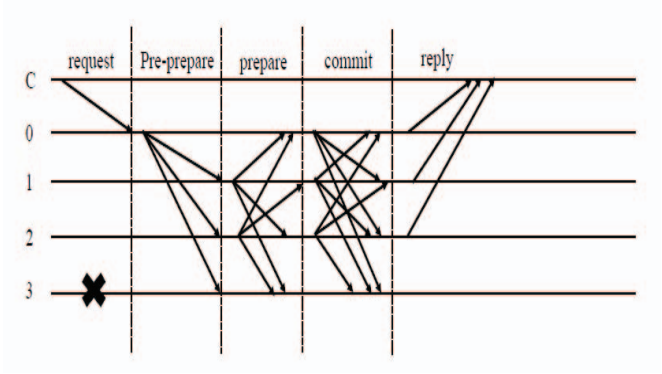

Fig. 4. Interaction process in PBFT algorithm.

1) Proof-of-Work (PoW): The idea of PoW came in the year 1992 by Dwork and Naor to combat junk emails where we have to do some work to send a valid email [5]. The attacker this way will be discouraged to send junk emails because in that case they have to do some work of some complexity to forward the junk emails that not prove beneficial for them. Blockchain based PoW system must have certain features like:

- Asymmetry: The task must be relatively hard but feasible for the service requester.
- The task must be easy to verify for the service provider. In this way the service requester will get discouraged to forge the work but service provider can easily check the validity of the work because of the asymmetry nature of the work.

Bitcoin based PoW system extends the hashcash based PoW system and develop a methodology to protect the Blockchain by applying the distributed consensus mechanism. Hashcash system was proposed by Adam Back which uses the puzzle friendliness property of the cryptographic hash function [6].

Now coming to Bitcoin based PoW, the miners who take part in the consensus process need to give a proof that they have done some work before proposing a new block. The attacker will be discouraged to propose a new block or make a change in the existing blocks. Because in that case, they have to do the entire work of the Blockchain which is computationally difficult in a generic environment. Every miner will try to find out a nonce value which will satisfy certain hash equation.

For example, BH = Hash (PH: MR: N)
N = ?
where BH = Block Hash
PH= Previous Block Hash
MR = Merkle Root
and N = Nonce

This BH has a given challenge which is to ensure zeros at the beginning. So this is termed as the difficulty of the system. The miners will try with different values of nonce (N) to find out difficulty. The miner who will first find out nonce value for his/her own block, that miner will able to include the block as a part of the Blockchain. Most implementation of Bitcoin

PoW use double SHA256 hash function. In the default set up, all miners wait for around 10 minutes and look for all the transactions which are taken place within that duration. Then they start the mining process.

As we have seen the probability of getting a PoW is very low, hence it will never happen that any miner will be able to control the Bitcoin network exclusively. In case any attacker wants to make some changes in one block, they have to do the more work compared to the collective work of all the blocks in the longest chain which is computationally difficult, though not impossible. Thus making an attack difficult with current hardware and hence Bitcoin system is tamper proof. This tamper proof characteristics of PoW also ensures that double spending does not happen in case of a Blockchain network. Apart from this, PoW based systems suffer from a number of security attacks like sybil attack, DOS attack, etc. In case of sybil attack, the attacker tries to fill the network with clients under its control. This helps the attacker to control the entire network. These clients work according to the instructions of the attacker which compromises the PoW mechanism. In DOS, the malicious node will send a lot of data to a particular node or nodes which will hamper the normal functioning of the Bitcoin transactions. Another major flaw in a Bitcoin system is monopoly problem. It happens when a particular miner gains the control of the network by deploying huge servers for mining process. So it may happen that this particular miner will gradually generate a huge number of blocks and hence the miner can control the entire flow of transactions. As a result, other nodes will get discouraged to join as miners and only few miners with large computing resources will control the network.

B. Proof of Stake (PoS)

PoS mechanism is an improved version of PoW to reduce the excessive electricity consumption of PoW based systems. As a substitute to the computationally expensive PoW mechanism, PoS aims to stake nodes' economic share in the network. Like PoW, a node (miner) is selected to add a block to the Blockchain [3]. But here the miner selection procedure is different from PoW. In PoS, the selection of a miner is proportional to the amount of Bitcoin it holds. Block finality in PoS based system is faster compared to PoW Blockchain, as computationally difficult puzzle solving is not there in PoS.The PoS based algorithm developed by Ethereum is known as Casper.

The PoS algorithm pseudo-randomly chooses miners to create blocks of the Blockchain, so that no miner can predict its turn in advance.It also solves the monopoly problem of PoW based consensus. Naive PoS algorithms are prone to an attack known as Nothing-at-Stake, thus require some improvements in order to provide safety.

IV. Consensus in Permission-ed Blockchain System

In Permissioned Blockchain, consensus is achieved through the help of a smart contract, which is basically an extension of Bitcoin scripts. Smart contracts are self-executing software programs in which the terms and conditions among the negotiating parties are penned down. Generally, the concept of state machine replication mechanism is used to ensure consensus in Permissioned Blockchain environment because of the following reasons:

- The network is closed and the nodes know each other, so state replication is Possible among the known nodes.
- It avoids the overhead of mining (in terms of time, Power, Bitcoin, etc.).

A. Paxos

The first ever consensus algorithm, Paxos, propoed by Lamport [7] with the objective to choose a single value under crash or network fault. The idea behind Paxos is very simple. All the nodes in the network have been categorized into three types namely the proposers, the acceptors and the learners as shown in Fig 1. Every one is a learner in the network that learns the consensus value. Let us look into the procedure in detail:

- The proposer initially prepares a proposal with a proposal number known as prepare message and send it to the acceptors. This proposal number forms a time-line and the biggest number is considered up to date. For example, between proposal number x and z = x+y, where y ¿= 1, proposal number z will be considered as latest and will be accepted.
  $Prepare\_message$: (proposal number)
- Each acceptor compares received proposal number with the current known values for all proposer's proposal message. If it gets a higher number, then it accepts the proposal or else declines it. Then the acceptor prepares the response message of the following form
  $Prepare\_response$: (accept/reject, proposal number, accepted values)
  where, proposal number is the biggest number the acceptor has seen.
  and accepted values are the already accepted values from other proposer.
- Next a vote is being taken based on the majority decision. The proposer checks whether the majority of the acceptors have rejected the proposal. If yes, then the proposer updates it with the latest proposal number. If no, then the proposer further checks whether the majority of the acceptors have already accepted values. If yes then the proposers value can not be selected or else it sends accept message.
- Finally, the proposer sends the accept message having the following format to all the acceptors.
  $Accept\_message$: (proposal number, value)
  where proposal number: same as prepare phase value
  value: single value proposed by proposer
- whenever the acceptor accepts a value, it informs the learner nodes about it so that everyone will learn about the accepted value.

As mentioned, if more than (N/2 - 1) acceptors fail, then no proposer will get enough reply messages to form a conclusion and we can not reach consensus. Even though the concept behind Paxos is very simple to understand but the theoretical proof is very complicated. The real life application of the same may need to go for sequence of selections which is known as multi-Paxos system. Multi-Paxos systems need a repeated application of Paxos which increases the system complexity. This is because the number of messages exchanged between the nodes also increases. There is also a chance of livelock or starvation in Paxos based systems.

B. Raft

Raft algorithm is designed as an easy alternative to Paxos. The basic idea behind Raft is that the nodes collectively selects a leader and rest of the nodes become followers. The leader is responsible for state transition log replication across the followers [8]. Record entries flow in only one direction from the leader to the followers. Unlike Paxos, here each node can be at any of the three states namely leader, candidate and follower at any particular time. Raft algorithm runs in rounds which is known as term. Each term starts with an election where one or more candidates strive to become a leader. The state change is shown in the Fig 2.

Coming to a detailed view of the algorithm:

- Initially, we have a set of follower nodes, who look out for a leader. If within certain time interval they do not find one, then the leader election process starts. In this election phase, some of the followers volunteer to become a leader and request votes from all other nodes. Then these candidate nodes send request messages to other followers of the system for vote.
  $Request\_vote$:(term, index),
  where term: last calculated number known to candidate + 1 and index: committed transaction available to the candidate. This is the first part of Raft algorithm. This $Request\_vote$ message will be forwarded to all the nodes in the network.
- When a node receives the request, it compares the term and index in the received message with corresponding current known values. Fig. 3 shows the voting procedure in detail.
- Like Paxos, the followers vote for one of the candidates and based on the majority of votes, a leader is selected.
- Then in the next part, the elected leader will propose values which the followers will choose so that the system can reach consensus.

The leader sends out heartbeats (signals) to all followers at regular intervals in order to maintain its authority. Compared with Paxos and PBFT, this algorithm has high efficiency and clarity. Hence it has been extensively employed in distributed systems.Raft algorithm realizes the same safety performance as Paxos and is better suited in real life implementation and comprehension. As mentioned, Raft algorithm cannot support byzantine nodes and can stand up to failure of 50 % of nodes . As in case of Permission-ed Blockchain, nodes are verified members. Hence, it is more essential to resolve crash faults rather than Byzantine faults for private Blockchain.

C. Byzantine Fault Tolerance and and it's Variant

In relation to distributed systems, Byzantine Fault Tolerance is the capability of a distributed network to execute as required and correctly reach to a consensus despite malicious nodes. Derived from the Byzantine General Problem, where the General sends an attack message to one group of lieutenants where as send a retreat message to another group of lieutenants. Hence it becomes difficult for the system to find out what action to take. The byzantine fault tolerance model has the following assumptions:

- Total N number of nodes with at most f faulty nodes
- Closed Environment (Receiver always knows the identity of the sender)
- Fully connected
- Reliable communication medium
- Synchronous system

Byzantine fault-tolerant systems are typically built using replication. For this, the state machine approach is used which helps to implement fault toerant services. The variant of BFT that has been designed for synchronous distributed systems is called "Lamport-Shostak-Pease" algorithm [9]. It was one of the first algorithm for handling byzantine faults [Byzantine general problem, lamoprt, acm]. This ensures consensus in presence of f number of faulty nodes, provided we have $(2f + 1)$ number of lieutenants apart from the commander. But our real systems behave in an asynchronous way as there is no guarantee that a message will be received within a certain time interval. For this reason, a variant of BFT known as Practical Byzantine Fault Tolerance (PBFT), has been developed for real life asynchronous systems [10]. As in case of a pure asynchronous system achieving consensus is impossible even in the presence of a single faulty node. So to ensure liveness property, instead of pure asynchronous system, a weak asynchronous system has been considered.

Coming to the algorithm, the byzantine model consists of three types of nodes: the clients, a commander (leader) and the lieutenants (followers).

The entire algorithm runs in three phases: pre-prepare, prepare and commit phase.

- Pre-prepare Phase: The commander assigns a sequence number to the request submitted by a client and multicast it to the network. Among other data, the request message also contains the digital signature and message digest for verification. The lieutenants of the network confirm the block by verifying the digital signature and message digest.
- Once the validating lieutenants accepts the pre-prepare message, they enter to the prepare phase by multicasting the message to the rest of the network. Once again the replicas (commander and lieutenants) verify the prepare message before accepting them.
- The messages commits, when $(2f)$ prepare message from different backups matches with the corresponding pre-

TABLE II
COMPARISON BASED ON CHARACTERISTICS.

Characteristics	**PoW**	**PoS**	**PoB**	**Paxos**	**Raft**	**BFT and its variants**
Trust Model	Un-trusted	Un-trusted	Un-trusted	Semi-trusted	Semi-trusted	Semi-trusted
Blockchain Type	Permission-less	Both	Both	Permission-ed	Permission-ed	Permission-ed
Transaction Finality	Probabilistic	Probabilistic	Probabilistic	Immediate	Immediate	Immediate
Degree of Decentralization	High(Entirely Decentralized)	High	High	Low	Low	Low
Scalability	High	High	High	Low	Moderate	Low
Compliance to Distributed Consensus Properties	Probabilistic	Probabilistic	Probabilistic	Deterministic	Deterministic	Deterministic
Reward	Yes	Yes	Yes	Mostly No	Mostly No	Mostly No
Network Realization	Bitcoin,Ethereum	Peercoin, 808Coin	Slimcoin			Hyperldeger Fabric, Byzcoin, Tendermint

TABLE III
COMPARISON BASED ON PERFORMANCE.

Performance Attributes	**PoW**	**PoS**	**PoB**	**Paxos**	**Raft**	**BFT and its variants**
Crash fault tolerance	50%	50%	50%	50%	50%	33%
Byzantine fault tolerance	50%	50%	50%	NA	NA	33%
Response Time	10 Minutes	1 Minute	1 Minute			1 second
Rate of Energy consumption	High	Better than PoW	Better than PoW	High	High	Moderate
Transaction Throughput	Very Low	Low	Low	Very High	Very High	Average
Transaction Latency	Very High	High	High			Very Low

prepare messages. Hence, the total $(2f + 1)$ votes (one from primary) from the non-faulty replica helps the system to reach to a consensus. Fig. 4 shows the interaction process.

PBFT based consensus algorithms, we need $(3f + 1)$ number of replicas in order to reach to the consensus. It provides high throughput, low latency and less energy usage in verifying transactions as compared to PoW based consensus mechanism. It also provides privacy over the system by ensuring tamper proof message transmission. Authenticity of nodes is also verified through signatures. Despite its promising advantages, there are some significant limitations to the PBFT consensus algorithm. The overhead incurred by multicasting messages again and again makes it difficult to scale beyond a fixed number of replicas. PBFT is also susceptible to sybil attack. PBFT has well adopted in consensus for Permissioned Blockchain environments like Hyperledger, Tendermint Core, etc.

V. COMPARATIVE ANALYSIS

This Section presents a comparison of the different consensus algorithms that we have discussed so far in this paper. Table 2 and Table 3 below present a detailed comparison between the above mentioned consensus algorithms in terms of characteristics and performance.

VI. CONCLUSION

With the evolution of ICT, the Blockchain technology has attracted interest from various dimensions. Consensus algorithm is the main technology of Blockchain, but on-going research of the consensus algorithms is still in its initial stage. Hence, a detailed review of the existing consensus algorithms has been presented in the paper. In case of permission-less systems, it is easy to achieve robust consensus among large number of untrusted nodes using complex computations though transaction finality remains non-deterministic. On the contrary, permission-ed Blockchain provides high throughput in less time while sacrificing the degree of decentralization.

REFERENCES

[1] S. Nakamoto *et al.*, "Bitcoin: A peer-to-peer electronic cash system," 2008.

[2] C. Shen and F. Pena-Mora, "Blockchain for citiesa systematic literature review," *IEEE Access*, vol. 6, pp. 76 787–76 819, 2018.

[3] M. S. Ali, M. Vecchio, M. Pincheira, K. Dolui, F. Antonelli, and M. H. Rehmani, "Applications of Blockchains in the Internet of Things: A comprehensive survey," *IEEE Communications Surveys & Tutorials*, 2018.

[4] Q. He, N. Guan, M. Lv, and W. Yi, "On the consensus mechanisms of blockchain/dlt for internet of things," in *2018 IEEE 13th International Symposium on Industrial Embedded Systems (SIES)*. IEEE, 2018, pp. 1–10.

[5] C. Dwork and M. Naor, "Pricing via processing or combatting junk mail," in *Proc. of the Annual International Cryptology Conference*. Springer, pp. 139–147, 1992.

[6] A. Back *et al.*, "Hashcash-a denial of service counter-measure," 2002.

[7] L. Lamport *et al.*, "Paxos made simple," *ACM Sigact News*, vol. 32, no. 4, pp. 18–25, 2001.

[8] D. Ongaro and J. Ousterhout, "In search of an understandable consensus algorithm," in *Proc. of the USENIX Annual Technical Conference (ATC)*, pp. 305–319, 2014.

[9] L. Lamport, R. Shostak, and M. Pease, "The byzantine generals problem," *ACM Transactions on Programming Languages and Systems (TOPLAS)*, vol. 4, no. 3, pp. 382–401, 1982.

[10] M. Castro, B. Liskov *et al.*, "Practical byzantine fault tolerance," in *OSDI*, vol. 99, pp. 173–186, 1999.